Lecture Notes in Computer Science 3329

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Pil Joong Lee (Ed.)

Advances in Cryptology – ASIACRYPT 2004

10th International Conference on the Theory
and Application of Cryptology and Information Security
Jeju Island, Korea, December 5-9, 2004
Proceedings

Volume Editor

Pil Joong Lee
Pohang University of Science and Technology
San 31, Hyoja-dong, Nam-gu, Pohang, Kyungbuk 790-784, Korea
On leave at KT Research Center, Seoul 137-792, Korea
E-mail: pjl@postech.ac.kr

Library of Congress Control Number: 2004115992

CR Subject Classification (1998): E.3, D.4.6, F.2.1-2, K.6.5, C.2, J.1, G.2.2

ISSN 0302-9743
ISBN 3-540-23975-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11363248 06/3142 5 4 3 2 1 0

Preface

The 10th Annual ASIACRYPT 2004 was held in Jeju Island, Korea, during December 5–9, 2004. This conference was organized by the International Association for Cryptologic Research (IACR) in cooperation with KIISC (Korean Institute of Information Security and Cryptology) and IRIS (International Research center for Information Security) at ICU (Information and Communications University), and was financially supported by MIC (Ministry of Information and Communication) in Korea.

The conference received, from 30 countries, 208 submissions that represent the current state of work in the cryptographic community worldwide, covering all areas of cryptologic research. Each paper, without the authors' information, was reviewed by at least three members of the program committee, and the papers (co-)authored by members of the program committee were reviewed by at least six members. We also blinded the reviewers' names among the reviewers until the final decision, by using pseudonyms. The reviews were then followed by deep discussions on the papers, which greatly contributed to the quality of the final selection. In most cases, extensive comments were sent to the authors.

Among 208 submissions, the program committee selected 36 papers. Two submissions were merged into a single paper, yielding the total of 35 papers accepted for presentation in the technical program of the conference. Many high-quality works could not be accepted because of the competitive nature of the conference and the challenging task of selecting a program. These proceedings contain revised versions of the accepted papers. These revisions have not been checked for correctness, and the authors bear full responsibility for the contents of their papers.

This was the first year in which the program committee selected a recipient for the Best Paper Award for the ASIACRYPT conference after lengthy discussion on its procedure and voting among program committee members. The winner of the prize for the Best Paper was Claus Diem from the University of Essen for his paper "The XL-algorithm and a Conjecture from Commutative Algebra."

The conference program included two invited lectures. Adi Shamir, from the Weizmann Institute of Science, Israel, talked on "Stream Ciphers: Dead or Alive?," and Ho-Ick Suk, the Deputy Minister for Informatization Planning at MIC of Korea, talked on "Information Security in Korea IT839 Strategy." In addition, the conference also included one rump session, chaired by Moti Yung, which featured short informal talks.

I wish to thank the program committee, whose members worked very hard over several months. I am also very grateful to the external referees who contributed with their special expertise to the selection process. Their work is highly appreciated.

The submission of all papers was received electronically using Web-based submission software which was provided by Chanathip Namprempre with modi-

fications by Andre Adelsbach. During the review process, the program committee was mainly communicated using the Web-based review software developed by Bart Preneel, Wim Moreau, and Joris Claessens.

It is my pleasure to thank the General Chair, Prof. Kwangjo Kim, for all his work in organizing the conference, and for the pleasant collaboration and various pieces of advice. In addition, I would like to extend my gratitude to the members of the local organizing committee. For financial support of the conference, the organizing committee and I gratefully acknowledge the Ministry of Information and Communication (MIC) of Korea.

I am also grateful to the secretariat of the program committee. Special thanks to Sung Ho Yoo and Young Tae Youn for maintaining both the submission server and the review server, and to Yong Ho Hwang and Yeon Hyeong Yang who served as technical assistants to the chairs and helped me with the various technical aspects of running the committee and preparing the conference proceedings, and to others for miscellaneous jobs.

Finally, we would like to thank all the other people who provided any assistance, and all the authors who submitted their papers to ASIACRYPT 2004, as well as all the participants from all over the world.

December 2004 Pil Joong Lee

ASIACRYPT 2004

December 5–9, 2004, Jeju Island, Korea

Sponsored by
International Association for Cryptologic Research (IACR)

in cooperation with
Korean Institute of Information Security and Cryptology (KIISC)
International Research Center for Information Security (IRIS) at Information and Communications University (ICU)

financially supported by
Ministry of Information and Communication (MIC) in Korea.

General Chair

Kwangjo Kim, Information and Communications University, Korea

Program Chair

Pil Joong Lee, Pohang University of Science and Technology, Korea
(on leave at KT Research Center, Korea)

Organizing Committee

Program Committee

Jee Hea An ..... SoftMax, USA
Michael Backes ..... IBM Zurich Research Lab., Switzerland
Feng Bao ..... Institute for Infocomm Research, Singapore
Colin Boyd ..... Queensland University of Tech., Australia
Liqun Chen ..... Hewlett-Packard Labs, UK
Don Coppersmith ..... IBM T.J. Watson Research Center, USA
Marc Joye ..... Gemplus, France
Jonathan Katz ..... University of Maryland, USA
Yongdae Kim ..... University of Minnesota, USA
Dong Hoon Lee ..... Korea University, Korea
Jaeil Lee ..... KISA, Korea
Arjen K. Lenstra ..... Lucent Technologies, USA and TU Eindhoven, The Netherlands
Atsuko Miyaji ..... JAIST, Japan
Jesper Buus Nielsen ..... ETH Zurich, Switzerland
Choonsik Park ..... NSRI, Korea
Dingyi Pei ..... Chinese Academy of Sciences, China
Erez Petrank ..... Technion, Israel
David Pointcheval ..... CNRS-ENS, Paris, France
Bart Preneel ..... Katholieke Universiteit Leuven, Belgium
Vincent Rijmen ..... Graz University of Technology, Austria
Bimal Roy ..... Indian Statistical Institute, India
Rei Safavi-Naini ..... University of Wollongong, Australia
Kazue Sako ..... NEC Corporation, Japan
Kouichi Sakurai ..... Kyushu University, Japan
Nigel Smart ..... University of Bristol, UK
Serge Vaudenay ..... EPFL, Switzerland
Sung-Ming Yen ..... National Central University, Taiwan
Yiqun Lisa Yin ..... Princeton University, USA
Moti Yung ..... Columbia University, USA
Yuliang Zheng ..... University of North Carolina at Charlotte, USA

Organizing Committee

Khi-Jung Ahn ..... Cheju National University, Korea
Jae Choon Cha ..... ICU, Korea
Byoungcheon Lee ..... Joongbu University, Korea
Im-Yeong Lee ..... Soonchunhyang University, Korea
Kyung-Hyune Rhee ..... Pukyong National University, Korea
Dae-Hyun Ryu ..... Hansei University, Korea

External Reviewers

Michel Abdalla
Roberto Maria Avanzi
Gildas Avoine
Joonsang Baek
Thomas Baignères
Endre Bangerter
Rana Barua
Lejla Batina
Amos Beimel
Yolanta Beres
John Black
Emmanuel Bresson
Jin Wook Byun
Christian Cachin
Qingjun Cai
Chris M. Calabro
Jan Camenisch
Ran Canetti
Christophe De Cannière
Claude Carlet
Dario Catalano
Donghoon Chang
Sanjit Chatterjee
Chien-Ning Chen
Lily Chen
Yongxi Cheng
Donghyeon Cheon
Jung Hee Cheon
Eun Young Choi
Kyu Young Choi
Kilsoo Chun
Scott Contini
Jean-Sébastien Coron
Ivan Damgård
Alex Dent
Anand Desai
Orr Dunkelman
Glenn Durfee
Matthieu Finiasz
Marc Fischlin
Caroline Fontaine
Pierre-Alain Fouque
Eiichiro Fujisaki
Jun Furukawa
Yuichi Futa
Pierrick Gaudry
Henri Gilbert
Juan González
Louis Granboulan
Robert Granger
Kishan Chand Gupta
Kil-Chan Ha
Stuart Haber
Shai Halevi
Dong-Guk Han
Helena Handschuh
Keith Harrison
Florian Hess
Yvonne Hitchcock
Christina Hoelzer
Dennis Hofheinz
Jung Yeon Hwang
Kenji Imamoto
Yuval Ishai
Ik Rae Jeong
Antoine Joux
Seok Won Jung
Pascal Junod
Markus Kaiser
Masayuki Kanda
Sungwoo Kang
Joe Kilian
Jeeyeon Kim
Jinhae Kim
Jongsung Kim
Seungjoo Kim
Yong Ho Kim
Lars Knudsen
Chiu-Yuen Koo
Jaehyung Koo
Caroline Kudla
Eyal Kushilevitz
Hidenori Kuwakado
Soonhak Kwon
Taekyoung Kwon
Mario Lamberger
Tanja Lange
Joseph Lano
Byoungcheon Lee
Changhoon Lee
Eonkyung Lee
Hoonjae Lee
Su Mi Lee
Wonil Lee
Minming Li
Yong Li
Benoît Libert
Seongan Lim
Hsi-Chung Lin
Yehuda Lindell
Yi Lu
Subhamoy Maitra
John Malone-Lee
Wenbo Mao
Keith Martin
Mitsuru Matsui
Willi Meier
Nele Mentens
Kazuhiko Minematsu
Pradeep Kumar Mishra
Chris Mitchell
Brian Monahan
Jean Monnerat
Shiho Moriai
Yi Mu
Joern Mueller-Quade
Sourav Mukhopadhyay
Hirofumi Muratani
Toru Nakanishi
Mridul Nandi
Lan Nguyen
Phong Nguyen
Masao Nonaka
Daehun Nyang
Luke O'Connor
Satoshi Obana
Wakaha Ogata
Kazuo Ohta
Takeshi Okamoto
Ivan Osipkov
Elisabeth Oswald
Dan Page

Adriana Palacio
Haeryong Park
Rafael Pass
Kenny Paterson
Olivier Pereira
Hieu Duong Phan
Benny Pinkas
Bartosz Przydatek
Michaël Quisquater
François Recher
Renato Renner
Martin Roetteler
Alon Rosen
Palash Sarkar
Katja Schmidt-Samoa
Berry Schoenmakers
Jaechul Sung
Hovav Shacham
Nicholas Sheppard
Haixia Shi
Junji Shikata
Atsushi Shimbo
Alice Silverberg
Johan Sjödin
Jung Hwan Song
Masakazu Soshi
Hirose Souichi
Martijn Stam
Ron Steinfeld
Rainer Steinwandt
Reto Strobl
Makoto Sugita
Hung-Min Sun
Soo Hak Sung
Willy Susilo
Tsuyoshi Takagi
Gelareh Taban
Mitsuru Tada
Yuko Tamura
Isamu Teranishi
Emmanuel Thomé
Eran Tromer
Shiang-Feng Tzeng
Frederik Vercauteren
Eric Verheul
Ivan Visconti
Guilin Wang
Huaxiong Wang
Shawn Wang
Bogdan Warinschi
Larry Washington
Benne de Weger
William Whyte
Christopher Wolf
Duncan Wong
Chi-Dian Wu
Hongjun Wu
Yongdong Wu
Guohua Xiong
Bo-Yin Yang
Akihiro Yamamura
Youngjin Yeom
Takuya Yoshida
Chong Zhang
Yunlei Zhao
Feng Zhu
Huafei Zhu

Table of Contents

Block Ciphers

Public Key Encryption

Invited Talk I

Number Theory and Algebra

Secure Computation

Hash Functions

Key Management

Identification

XL-Algorithms

Digital Signatures

Public Key Cryptanalysis

Invited Talk II

Symmetric Key Cryptanalysis

Protocols

On Feistel Ciphers Using Optimal Diffusion Mappings Across Multiple Rounds

Taizo Shirai[1,⋆] and Bart Preneel[2]

[1] Sony Corporation, Tokyo, Japan
taizo.shirai@jp.sony.com
[2] ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Belgium
bart.preneel@esat.kuleuven.ac.be

Abstract. We study a recently proposed design approach of Feistel ciphers which employs optimal diffusion mappings across multiple rounds. This idea was proposed by Shirai and Shibutani at FSE2004, and the technique enables to improve the immunity against either differential *or* linear cryptanalysis (but not both). In this paper, we present a theoretical explanation why the new design using three different matrices achieves the better immunity. In addition, we are able to prove conditions to improve the immunity against both differential *and* linear cryptanalysis. As a result, we show that this design approach guarantees at least $R(m+1)$ active S-boxes in $3R$ consecutive rounds ($R \geq 2$) where m is the number of S-boxes in a round. By using the guaranteed number of active S-boxes, we compare this design approach to other well-known designs employed in SHARK, Rijndael, and MDS-Feistel ciphers. Moreover, we show interesting additional properties of the new design approach.

Keywords: optimal diffusion mapping, Feistel cipher, active S-boxes, MDS.

1 Introduction

A Feistel structure is one of the most widely used and best studied structures for the design of block ciphers. It was first proposed by H. Feistel in 1973; subsequently the structure was adopted in the well-known block cipher DES [5,6]. The main advantage of the Feistel structure is its involution property, which provides flexible designs of the underlying F-functions. During the 30 year history of modern block ciphers, extensive studies have been made on Feistel structure [8, 11, 14]. Currently, many well-known block ciphers, e.g. Camellia [1], Misty [10], RC6 [13], Twofish [15], employed the design of Feistel structures.

Recently, Shirai and Shibutani proposed a novel design approach of Feistel ciphers based on the concept of optimal diffusion mappings [18]. An optimal diffusion is a linear function with maximum branch number; the concept of optimal diffusion is used in the design of AES and many other cryptographic primitives [2,

⋆ A guest researcher at ESAT/SCD-COSIC, K.U.Leuven from 2003 to 2004.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 1–15, 2004.

15, 12, 4]. From their empirical analytic results, the immunity against either differential and linear cryptanalysis (but not both) would be strengthened significantly if the linear diffusion layer of the Feistel structure satisfies special optimal diffusion conditions across multiple rounds. In this way difference cancellation in the Feistel structure caused by a small number of active S-boxes will not occur. This result opened a new line of research on the Feistel structure. A theoretical proof of the effectiveness of the proposed design and a solution to improve the immunity against both differential *and* linear cryptanalysis remained unsolved.

In this paper, we will call the *"Optimal Diffusion Mappings across Multiple Rounds"* design approach of Feistel ciphers the *ODM-MR design*. Our contribution is that we first give a theoretical explanation of the effectiveness of the *ODM-MR design* implied by Shirai and Shibutani. Second, we found new conditions and proofs to improve the immunity of both differential *and* linear cryptanalysis. Let m be the number of S-boxes in an F-function. As a result, by combining previous and novel conditions, we finally show that Feistel ciphers with the *ODM-MR design* guarantees $R(m+1)$ active S-boxes in $3R$ consecutive rounds for $R \geq 2$.

In order to further investigate the properties of the *ODM-MR design*, we compare the ratio of guaranteed active S-boxes to all employed S-boxes of the *ODM-MR design* to other design approaches employed in MDS-Feistel, SHARK and AES/Rijndael. All of them use optimal diffusion mappings in their linear diffusion. Consequently, in 128-bit block and 8-bit S-box settings, we obtain a limit of 0.371 for the active S-box ratio of *ODM-MR design* when the number r of rounds goes to infinity, which means that we can guarantee 37.1% active S-boxes with this design strategy. This result is apparently better than MDS-Feistel's ratio of 0.313. Moreover we show that for the number of S-boxes in an F-function and the round number go to infinity, the converged ratio of *ODM-MR* is 0.333. This is better than Rijndael-type diffusion layer's ratio 0.250. From these limit values, we can conclude that *ODM-MR* performs better than the other approaches in certain settings.

This paper is organized as follows: in Sect. 2, we introduce some definitions used in this paper. Previous works including *ODM-MR design* approach are shown in Sect. 3. We prove in Sect. 4 the theorems regarding *ODM-MR* as our main contribution. In Sect. 5, we discuss the new design by presenting some numerical values. Finally Sect. 6 concludes the paper. The method to construct the concrete Feistel ciphers with *ODM-MR design* is proposed in Appendix A.

2 Preliminaries

In this paper, we treat the typical Feistel structure, which is called a balanced Feistel structure. It is defined as follows [14].

Definition 1. *(Balanced Feistel Structure)*
Let $E : \{0,1\}^b \times \{0,1\}^k \rightarrow \{0,1\}^b$ *be a* b*-bit block cipher (for* b *even) with a* k*-bit key. Let* r *be the number of rounds,* $k_i \in \{0,1\}^{k'}$ *be the* k'*-bit round key provided*

by a key scheduling algorithm and $x_i \in \{0,1\}^{b/2}$ *be intermediate data, and let* $F_i : \{0,1\}^{k'} \times \{0,1\}^{b/2} \rightarrow \{0,1\}^{b/2}$ *be the F-function of the i-th round. The encryption and decryption algorithm of a balanced Feistel Cipher E are defined as follows*

Algorithm $Feistel.Encrypt_K(P)$
input $P \in \{0,1\}^b$, $K \in \{0,1\}^k$
$x_0 \leftarrow msb_{b/2}(P)$, $x_1 \leftarrow lsb_{b/2}(P)$
for $i = 1$ **to** r **do**
$x_{i+1} = F_i(k_i, x_i) \oplus x_{i-1}$
$msb_{b/2}(C) \leftarrow x_{r+1}$, $lsb_{b/2}(C) \leftarrow x_r$
return $C \in \{0,1\}^b$

Algorithm $Feistel.Decrypt_K(C)$
input $C \in \{0,1\}^b$, $K \in \{0,1\}^k$
$x_0 \leftarrow msb_{b/2}(C)$, $x_1 \leftarrow lsb_{b/2}(C)$
for $i = 1$ **to** r **do**
$x_{i+1} = F_i(k_{r-i+1}, x_i) \oplus x_{i-1}$
$msb_{b/2}(P) \leftarrow x_{r+1}$, $lsb_{b/2}(P) \leftarrow x_r$
return $P \in \{0,1\}^b$

where $msb_x(y)$ *(*$lsb_x(y)$*) represents the most (least) significant x-bit of y.*

Then we define SP-type F-functions which are a special type of F-function constructions [17, 7].

Definition 2. *(SP-Type F-Function)*
Let the length of round key $k' = b/2$*. Let m be the number of S-boxes in a round, and n be the size of the S-boxes, with* $mn = b/2$*. Let* $s_{i,j} : \{0,1\}^n \rightarrow \{0,1\}^n$ *be the j-th S-box in the i-th round, and let* $S_i : \{0,1\}^{b/2} \rightarrow \{0,1\}^{b/2}$ *be the function generated by concatenating m S-boxes in the i-th round. Let* $P_i : \{0,1\}^{b/2} \rightarrow \{0,1\}^{b/2}$ *be the linear Boolean function.*

Then SP-type F-functions are defined as $F_i(x_i, k_i) = P_i(S_i(x_i \oplus k_i))$*. Note that we define the intermediate variables* $z_i = S_i(x_i \oplus k_i)$*.*

Definition 3. *((m,n,r)-SPMFC)*
An (m, n, r)*-SPMFC is defined as an r-round Feistel cipher with SP-type round function using m n-bit S-boxes, and for which all* $s_{i,j}$*,* P_i *are bijective. An* $mn \times mn$ *matrix* M_i $(1 \leq i \leq r)$ *over* $GF(2)$ *denotes a matrix representation of a linear Boolean function* P_i *of* (m, n, r)*-SPMFC.*

We also give definitions of bundle weight and branch number [4].

Definition 4. *(Bundle Weight)*
Let $x \in \{0,1\}^{kn}$*, where x is represented as a concatenation of n-bit values as* $x = [x_0 x_1 \ldots x_{k-1}]$*,* $x_i \in \{0,1\}^n$*, then the bundle weight* $w_n(x)$ *of x is defined as*

$$w_n(x) = \sharp\{x_i | x_i \neq 0\} \ .$$

Definition 5. *(Branch Number)*
Let $\theta : \{0,1\}^{kn} \rightarrow \{0,1\}^{ln}$*. The branch number of* θ *is defined as*

$$\mathcal{B}(\theta) = \min_{a \neq 0} \{w_n(a) + w_n(\theta(a))\} \ .$$

Remark 1. The maximum branch number is $\mathcal{B}(\theta) = l+1$. If a linear function has a maximum branch number, it is called an **optimal diffusion mapping** [2]. It is known that an optimal diffusion mapping can be obtained from maximum distance separable codes known as MDS codes [4].

3 Previous Work

The precise estimation of the lower bound of the number of active S-boxes of block ciphers has been known as one of the practical means to evaluate ciphers, because this lower bound can be used to calculate the upper bound of the differential characteristic probability or the linear characteristic probability [9, 3, 1, 17, 7, 4]. Shimizu has shown a conjectured lower bound of the number of differentially and linearly active S-boxes for certain (m, n, r)-SPMFC block ciphers, in which a unique optimal diffusion mapping is repeatedly used in all F-functions [16]. Since such optimal diffusion mappings can be obtained from a generation matrix of an MDS code, we call the design MDS-Feistel [4, 2, 18].

Shimizu showed the following formula.

Conjecture 1. Let A be an $mn \times mn$ matrix over $GF(2)$ of an optimal diffusion mapping with maximum branch number $m+1$. Let E be an (m, n, r)-SPMFC block cipher and all matrices of diffusion layers are represented by the unique matrix $M_i = A$ $(1 \leq i \leq r)$. Then a lower bound of the differentially and linearly active S-boxes of E is conjectured as

$$L(r) = \lfloor r/4 \rfloor (m+2) + (r \mod 4) - 1 \ . \tag{1}$$

In Table 1, the columns indicated by 'M1' show the conjectured lower bounds of number of active S-boxes, and the data of the conjectured values are plotted on the left side of Fig. 3. This simple relation between the round number and the guaranteed number of active S-boxes is considered as a useful tool for evaluating similar kinds of block cipher designs. While this conjecture has not been proved, empirically, it has been partially confirmed [18].

Recently, at FSE 2004, Shirai and Shibutani proposed a novel design approach to improve the minimum number of active S-boxes of Feistel ciphers by employing optimal diffusion mappings across multiple round functions, the *ODM-MR design* approach [18]. By carefully analyzing the difference cancellations, they found the following properties:

Property 1. Let E be an (m, n, r)-SPMFC block cipher.

- For matrices M_i $(1 \leq i \leq r)$, if every concatenation of two matrices M_j and M_{j+2} for all possible j, denoted by $[M_j | M_{j+2}]$, is an optimal diffusion mapping, the minimum number of differentially active S-boxes is increased from an MDS-Feistel cipher.
- Additionally, if each concatenation of three matrices M_j, M_{j+2} and M_{j+4} for all possible j, denoted by $[M_j | M_{j+2} | M_{j+4}]$, is an optimal diffusion mapping, the minimum number of differentially active S-boxes is increased further than when only satisfying the above conditions on two matrices.

– Even if the number of concatenated matrices is larger than 3, no explicit gain of the number of active S-boxes has been observed in their simulations.

These results imply that by avoiding a linear correlation between F-functions in round $(i, i+2)$ or rounds $(i, i+2, i+4)$, the *ODM-MR* construction guarantees more active S-boxes.

In Table 1, the columns indicated by 'D' show the result of the improved minimum number of differentially active S-boxes when every concatenated matrix of three matrices $[M_i|M_{i+2}|M_{i+4}]$ is an optimal diffusion mapping. The graph of the corresponding values are shown on the left side of Fig. 2.

This result opened a new line of research on developing more efficient Feistel ciphers. On the other hand a theoretical justification of the gain of the proposed construction and an explicit method to improve the immunity against both differential *and* linear cryptanalysis remained unsolved.

4 Proofs of Effectiveness of the *ODM-MR Design*

In this section, we provide the first proofs for the effectiveness of the *ODM-MR design* using three different matrices. We also show an additional condition and some proofs in order to improve the lower bound of linearly active S-boxes by using two different matrices. Our main contribution is to show the following corollary which presents a simple relation between the number of rounds and the guaranteed numbers of active S-boxes in the *ODM-MR design*. In the corollary, note that ${}^tM$ denotes the transpose matrix of a matrix M.

Corollary 1. *Let E be a (m,n,r)-SPMFC block cipher where $r \geq 6$.*
If $[M_i|M_{i+2}|M_{i+4}]$ and $[{}^tM_j^{-1}|{}^tM_{j+2}^{-1}]$ are optimal diffusion mappings for any i, j $(1 \leq i \leq r-4, 1 \leq j \leq r-2)$, respectively, any $3R$ consecutive rounds $(R \geq 2)$ in E guarantee at least $R(m+1)$ differentially and linearly active S-boxes.

Fig. 1 illustrates the statement of the corollary. By using the Corollary 1, we can guarantee theoretically arbitrary number of active S-boxes by increasing the number of rounds. Since the corollary can be immediately obtained from two theorems, i.e. Theorem 1 and Theorem 2, the following two subsections are devoted to the proofs of these theorems. To ease the proofs, we first introduce an additional definition.

Definition 6. *Consider a differential characteristic or linear approximation. Let D_i and L_i denote the number of differentially and linearly active S-boxes in the i-th round, respectively. These values are determined by the difference $\Delta x_i, \Delta z_i$ or by the linear mask $\Gamma x_i, \Gamma z_i$ as follows.*

$$D_i = w_n(\Delta x_i) = w_n(\Delta z_i) \ , \qquad L_i = w_n(\Gamma x_i) = w_n(\Gamma z_i) \ ,$$

where $w_n(\cdot)$ is the bundle weight as defined in Definition 4.

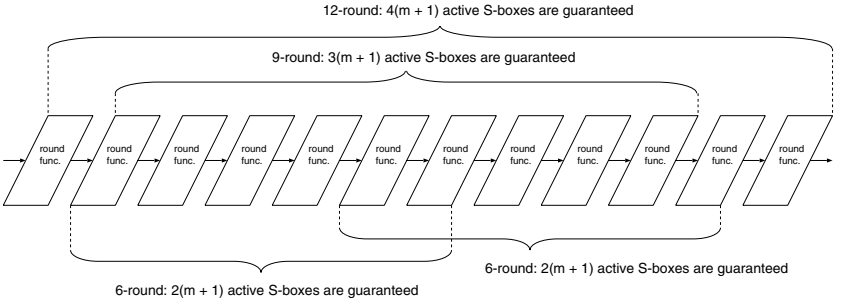


Fig. 1. Guaranteed Active S-boxes by *ODM-MR design*

Remark 2. Note that a given difference characteristic always contains a nonzero input difference, since any (m, n, r)-SPMFC's F-functions are bijective. Hence we obtain the following conditions:

$$(d0) \quad D_i = 0 \Rightarrow D_{i-2} \not= 0, D_{i-1} \not= 0, D_{i+1} \not= 0, D_{i+2} \not= 0,$$
$$(d1) \quad D_i = 0 \Rightarrow D_{i-1} = D_{i+1} \ .$$

Since a linear approximation always contains a nonzero input mask, we obtain

$$(l0) \quad L_i = 0 \Rightarrow L_{i-2} \not= 0, L_{i-1} \not= 0, L_{i+1} \not= 0, L_{i+2} \not= 0,$$
$$(l1) \quad L_i = 0 \Rightarrow L_{i-1} = L_{i+1} \ .$$

4.1 Proofs for the Lower Bound of Differentially Active S-Boxes

In this section we prove Theorem 1; the proof is based on five lemmata.

Lemma 1 shows relations between D_i of (m, n, r)-SPMFC when every M_i is an optimal diffusion mapping.

Lemma 1. *Let E be an (m, n, r)-SPMFC block cipher. If every M_i has maximum branch number $m + 1$, then E satisfies the following condition $(d2)$.*

$$(d2) \quad D_{i+1} \not= 0 \Rightarrow D_i + D_{i+1} + D_{i+2} \geq m + 1 \ .$$

Proof. From the relation between the differences Δz_{i+1},Δx_i and Δx_{i+2} in a 3 consecutive rounds, we obtain the following equation.

$$M_{i+1}\Delta z_{i+1} = \Delta x_i \oplus \Delta x_{i+2} \ .$$

Since M_i has maximum branch number $m + 1$ we have

$$w_n(\Delta z_{i+1}) \not= 0 \Rightarrow w_n(\Delta z_{i+1}) + w_n(\Delta x_i \oplus \Delta x_{i+2}) \geq m + 1 \ . \qquad (2)$$

Eq. (2) and the inequality $w_n(\Delta x_i) + w_n(\Delta x_{i+2}) \geq w_n(\Delta x_i \oplus \Delta x_{i+2})$ yield (d2). □

Remark 3. By combining Remark 2 and ($d2$), we can obtain additional underlying conditions ($d3$) and ($d4$).

$$\begin{array}{ll} (d3) & D_i = 0, \Rightarrow D_{i+1} + D_{i+2} \geq m+1 \ , \\ (d4) & D_{i+2} = 0, \Rightarrow D_i + D_{i+1} \geq m+1 \ . \end{array}$$

Equations ($d3$) and ($d4$) mean that if round k has no active S-boxes, any 2 consecutive rounds next to round k always contain more than $m+1$ active S-boxes.

Next, we show the property of (m, n, r)-SPMFC in which every $[M_i|M_{i+2}]$ is an optimal diffusion mapping. This is true for the *ODM-MR*.

Lemma 2. *Let E be a (m,n,r)-SPMFC block cipher. If every $[M_i|M_{i+2}]$ has maximum branch number $m+1$, E satisfies the following conditions* ($d5$), ($d6$).

$$\begin{array}{ll} (d5) & D_{i+4} = 0 \Rightarrow D_i + D_{i+1} + D_{i+3} \geq m+1 \ , \\ (d6) & D_i = 0 \Rightarrow D_{i+1} + D_{i+3} + D_{i+4} \geq m+1 \ . \end{array}$$

Proof. From the relation between 5-round differences,

$$M_{i+1}\Delta z_{i+1} \oplus M_{i+3}\Delta z_{i+3} = \Delta x_i \oplus \Delta x_{i+4} \ .$$

Then,

$$[M_{i+1}|M_{i+3}] \begin{pmatrix} \Delta z_{i+1} \\ \Delta z_{i+3} \end{pmatrix} = \Delta x_i \oplus \Delta x_{i+4} \ .$$

Since $[M_{i+1}|M_{i+3}]$ has maximum branch number $m+1$, and from Remark 2, we see that $w_n(\Delta z_{i+1}) = 0$ and $w_n(\Delta z_{i+3}) = 0$ will never occur simultaneously, we obtain

$$w_n(\Delta z_{i+1}) + w_n(\Delta z_{i+3}) + w_n(\Delta x_i \oplus \Delta x_{i+4}) \geq m+1 \ .$$

By assuming the cases $\Delta x_i = 0$ or $\Delta x_{i+4} = 0$, we directly obtain ($d5$) and ($d6$). □

By using the previously obtained conditions ($d0$)−($d6$), we show the following lemma for the guaranteed number of differentially active S-boxes of (m, n, r)-SPMFC.

Lemma 3. *Let E be a (m,n,r)-SPMFC block cipher where $r \geq 6$. If every $[M_i|M_{i+2}]$ is an optimal diffusion mapping, then any 6 consecutive rounds in E guarantee at least $2(m+1)$ differentially active S-boxes.*

Proof. Consider the total number of active S-boxes in 6 consecutive rounds from the i-th round,

$$\sum_{k=i}^{i+5} D_k = D_i + D_{i+1} + D_{i+2} + D_{i+3} + D_{i+4} + D_{i+5} \ .$$

If $D_{i+1} \neq 0$ and $D_{i+4} \neq 0$, the condition (d2) guarantees that $D_i + D_{i+1} + D_{i+2} \geq m+1$ and $D_{i+3}+D_{i+4}+D_{i+5} \geq m+1$. Therefore we obtain $\sum_{k=i}^{i+5} D_k \geq 2(m+1)$.
If $D_{i+1} = 0$,

$$\sum_{k=i}^{i+5} D_k = D_i + D_{i+2} + D_{i+3} + D_{i+4} + D_{i+5} \ .$$

From (d1),

$$\begin{aligned}\sum_{k=i}^{i+5} D_k &= 2 \cdot D_{i+2} + D_{i+3} + D_{i+4} + D_{i+5} \\ &= (D_{i+2} + D_{i+3}) + (D_{i+2} + D_{i+4} + D_{i+5}) \ .\end{aligned}$$

From (d3) and (d6),

$$\sum_{k=i}^{i+5} D_k \geq (m+1) + (m+1) = 2(m+1) \ .$$

The case of $D_{i+4} = 0$ is proven similarly from $(d1)$, $(d4)$ and $(d5)$. □

Next, we show the property of an (m, n, r)-SPMFC in which every $[M_i|M_{i+2}|M_{i+4}]$ has maximum branch number. This coincides with one of the *ODM-MR design*.

Lemma 4. *Let E be a (m,n,r)-SPMFC block cipher. If every $[M_i|M_{i+2}|M_{i+4}]$ is an optimal diffusion mapping, then E satisfies the following condition $(d7)$.*

$$(d7) \quad D_i = D_{i+6} = 0 \Rightarrow D_{i+1} + D_{i+3} + D_{i+5} \geq m+1 \ .$$

Proof. First, from the difference relation in 7 consecutive rounds, we obtain

$$M_{i+1}\Delta z_{i+1} \oplus M_{i+3}\Delta z_{i+3} \oplus M_{i+5}\Delta z_{i+5} = \Delta x_i \oplus \Delta x_{i+6} \ .$$

Then,

$$[M_{i+1}|M_{i+3}|M_{i+5}] \begin{pmatrix} \Delta z_{i+1} \\ \Delta z_{i+3} \\ \Delta z_{i+5} \end{pmatrix} = \Delta x_i \oplus \Delta x_{i+6} \ .$$

Since $[M_{i+1}|M_{i+3}|M_{i+5}]$ has maximum branch number, and from Remark 2, $w_n(\Delta z_{i+1})$, $w_n(\Delta z_{i+3})$, and $w_n(\Delta z_{i+5})$ cannot be simultaneously 0, we get that

$$w_n(\Delta z_{i+1}) + w_n(\Delta z_{i+3}) + w_n(\Delta z_{i+5}) + w_n(\Delta x_i \oplus \Delta x_{i+6}) \geq m+1 \ .$$

By assuming $\Delta x_i = 0$ and $\Delta x_{i+6} = 0$, we derive the condition $(d7)$. □

From the additional condition $(d7)$, we derive the following lemma.

Lemma 5. *Let E be a (m,n,r)-SPMFC block cipher where $r \geq 9$. If every $[M_i|M_{i+2}|M_{i+4}]$ is an optimal diffusion mapping, then any 9 consecutive rounds in E guarantee at least $3(m+1)$ differentially active S-boxes.*

Proof. Consider the total number of active S-boxes in 9 consecutive rounds,

$$\sum_{k=i}^{i+8} D_k = D_i + D_{i+1} + D_{i+2} + D_{i+3} + D_{i+4} + D_{i+5} + D_{i+6} + D_{i+7} + D_{i+8} \ .$$

If $D_{i+1} \neq 0$ then$D_i + D_{i+1} + D_{i+2} \geq m+1$ from (d2), and Lemma 3 guarantees that the sum of the remaining 6 consecutive rounds is equal to $\sum_{k=i+3}^{i+8} D_k \geq 2(m+1)$. Consequently $\sum_{k=i}^{i+8} D_k \geq 3(m+1)$. Similarly, if $D_{i+7} \neq 0$, at least $3(m+1)$ active S-boxes are guaranteed.

If $D_{i+1} = D_{i+7} = 0$, we obtain

$$\sum_{k=i}^{i+8} D_k = D_i + D_{i+2} + D_{i+3} + D_{i+4} + D_{i+5} + D_{i+6} + D_{i+8} \ .$$

From (d1),

$$\begin{aligned}\sum_{k=i}^{i+8} D_k &= 2 \cdot D_{i+2} + D_{i+3} + D_{i+4} + D_{i+5} + 2 \cdot D_{i+6} \\ &= (D_{i+2} + D_{i+3}) + (D_{i+2} + D_{i+4} + D_{i+6}) + (D_{i+5} + D_{i+6}) \ .\end{aligned}$$

From (d3), (d7) and (d4),

$$\sum_{k=i}^{i+8} D_k \geq (m+1) + (m+1) + (m+1) = 3(m+1) \ .$$

As a consequence, we have shown that any 9 consecutive rounds of E guarantee at least $3(m+1)$ differentially active S-boxes. □

We conclude this section with

Theorem 1. *Let E be an (m, n, r)-SPMFC block cipher where $r \geq 6$. If every $[M_i | M_{i+2} | M_{i+4}]$ is an optimal diffusion mapping, any $3R$ consecutive rounds in E guarantees at least $R(m+1)$ differentially active S-boxes.*

Proof. Any integer $3R$ ($R \geq 2$) can be written as $3R = 6k + 9j$ ($k + j \geq 1, 2k + 3j = R$). From lemmata 3 and 5, 6 and 9 consecutive rounds of E guarantee $2(m+1)$ and $3(m+1)$ differentially active S-boxes, respectively. Therefore, E guarantees $k*2(m+1)+j*3(m+1) = (2k+3j)(m+1) = R(m+1)$ differentially active S-boxes. □

4.2 Proofs for the Lower Bound of Linearly Active S-Boxes

In this subsection, we will show the proof of the guaranteed number of linearly active S-boxes of (m, n, r)-SPMFC with *ODM-MR design*.

Theorem 2. *Let E be an (m, n, r)-SPMFC block cipher. If every $[{}^t M_i^{-1} | {}^t M_{i+2}^{-1}]$ is an optimal diffusion mapping for any i, any $3R$ consecutive rounds in E has at least $R(m+1)$ linearly active S-boxes.*

Proof. From the 3-round linear mask relation,

$$\Gamma x_{i+1} = {}^tM_i^{-1}\Gamma z_i \oplus {}^tM_{i+2}^{-1}\Gamma z_{i+2} \ .$$

Then,

$$\Gamma x_{i+1} = [{}^tM_i^{-1}{}^tM_{i+2}^{-1}] \begin{pmatrix} \Gamma z_i \\ \Gamma z_{i+2} \end{pmatrix} \ .$$

Since $[{}^tM_i^{-1}|{}^tM_{i+2}^{-1}]$ has maximum branch number $m+1$, and from Remark 2, $w_n(\Gamma z_i)$ and $w_n(\Gamma z_{i+2})$ cannot be simultaneously 0, we obtain

$$w_n(\Gamma z_i) + w_n(\Gamma x_{i+1}) + w_n(\Gamma z_{i+2}) \geq m+1 \ .$$

By using the notion of L_i, this implies,

$$(l1) \quad L_i + L_{i+1} + L_{i+2} \geq m+1 \ .$$

This shows that every 3 consecutive rounds guarantees at least $m+1$ linearly active S-boxes. Consequently, any $3R$ consecutive rounds in E guarantees $\sum_{k=i}^{i+3R-1} L_k \geq R(m+1)$. □

Finally, by simply combining Theorems 1 and 2, the claimed Corollary 1 follows directly. Appendix A contains example matrices that satisfy the *ODM-MR design*.

5 Discussion

5.1 Comparison of *ODM-MR* and MDS-Feistel

To discuss the implications of this new design approach, we show empirical search results for the cases $r = 12, m = 4, .., 8$. To obtain these results we employed a weight based search method. This approach has been used by Shirai and Shibutani before [18]. Our results are shown in Table 1. In the table, the values for more than 13-rounds are interpolated by the corollary and Shimizu's conjecture. Note that the simulation results completely match the lower bound values predicted by the corollary, which are denoted by the underlined values. These results show the superiority of *ODM-MR design* over MDS-Feistel explicitly.

Fig. 2 shows graphs of the results in Table 1, and five auxiliary lines $y = (m+1)x/3$ are added where $m = 4, .., 8$. These lines connect the lower bounds values of every $3R$-round.

The left side of Fig. 3 shows an estimated lower bound of MDS-Feistel and approximate lines $y = (m+2)x/4-1$. To see the effect of the *ODM-MR* approach graphically, the right side of Fig. 3 includes the approximated lines for *ODM-MR* and MDS-Feistel for $m = 4$ and $m = 8$. The differences of the gradients show explicitly the advantage of the *ODM-MR* approach.

Table 1. Lower Bounds of MDS-Feistel and *ODM-MR design*

	$m=4$			$m=5$			$m=6$			$m=7$			$m=8$		
Round	M1	D	L	M1	D	L	M1	D	L	M1	D	L	M1	D	L
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	2	2	5	2	2	6	2	2	7	2	2	8	2	2	9
4	5	5	5	6	6	6	7	7	7	8	8	8	9	9	9
5	6	6	6	7	7	7	8	8	8	9	9	9	10	10	10
6	7	10	10	8	12	12	9	14	14	10	16	16	11	18	18
7	8	10	10	9	12	12	10	14	14	11	16	16	12	18	18
8	11	12	11	13	14	13	15	16	15	17	18	17	19	20	19
9	12	15	15	14	18	18	16	21	21	18	24	24	20	27	27
10	13	16	15	15	18	18	17	22	21	19	24	24	21	28	27
11	14	17	16	16	20	19	18	23	22	20	26	25	22	29	28
12	17	20	20	20	24	24	23	28	28	26	32	32	29	36	36
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
15	20	**25**	**25**	23	**30**	**30**	26	**35**	**35**	29	**40**	**40**	32	**45**	**45**
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
18	25	**30**	**30**	29	**36**	**36**	33	**42**	**42**	37	**48**	**48**	41	**54**	**54**

M1: numbers of active S-boxes of MDS-Feistel
D: numbers of differentially active S-boxes of *ODM-MR*
L: numbers of linearly active S-boxes of *ODM-MR*

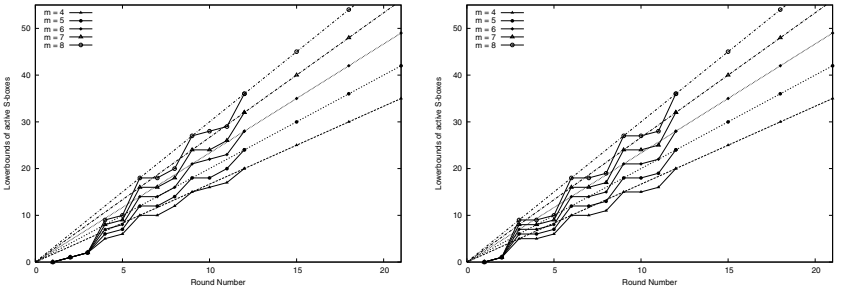


Fig. 2. Lower bounds of the *ODM-MR design* ($m = 8$)

5.2 Active S-Box Ratio

In this subsection, we compare the *ODM-MR* approach to other design approaches using the new type of approach. Since we obtained a formal bound for the lower bound of the *ODM-MR design* approach, we can compare it to other well known design approaches based on the concept of active S-box ratio introduced by Shirai and Shibutani [18].

Let $active(r, m)$ be the number of guaranteed active S-boxes for an r-round cipher which employs $m \times m$ diffusion matrices over $GF(2^n)$ in its diffusion layer.For example, $active(r, m)$ of the MDS-Feistel design can be written as

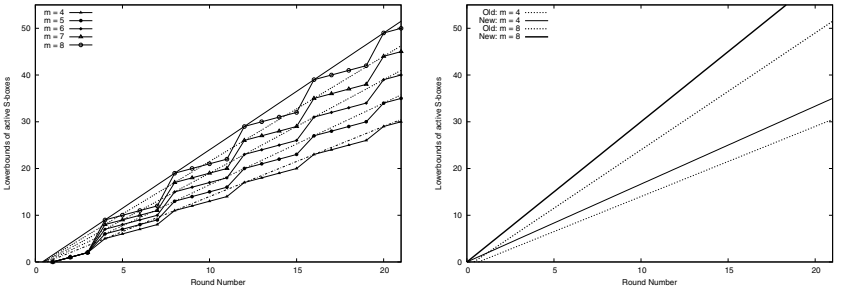


Fig. 3. Comparison of MDS-Feistel and *ODM-MR design*

$$active(r, m) = (m + 2)\lfloor r/4 \rfloor + \alpha_{r,m} \ ,$$

where $\alpha_{r,m} = (r \mod 4) - 1$. Generally, $\alpha_{r,m}$ is a function which maximum absolute value is proportional to m and $lim_{r\to\infty}\alpha_{r,m}/r = 0$.

Next, let $total(r, m)$ be the total number of S-boxes in an r-round cipher. The ratio of the number of active S-boxes to the total number of S-boxes becomes

$$ratio(r, m) = \frac{active(r, m)}{total(r, m)} = \frac{(m + 2)\lfloor r/4 \rfloor + \alpha_{r,m}}{rm} \ .$$

By using the definition of active S-box ratio, we can study the characteristic of the MDS-Feistel design. For example, consider a 128-bit block cipher employing 8-bit S-boxes. For $m = 8$, $ratio(r, 8)$ will converge to a specific value when r goes to infinity,

$$\lim_{r\to\infty} ratio(r, 8) = \lim_{r\to\infty} \frac{10\lfloor r/4 \rfloor + \alpha_{r,8}}{8r} = \frac{10}{32} = 0.3125 \ .$$

This implies that about 31% of all S-boxes will be active for a very large number of rounds. This limit can be considered as a potentially guaranteed ratio of active S-boxes corresponding to the chosen m.

Also, we can take the limit of $ratio(r, m)$ when both r and m tend to infinity,

$$\lim_{r,m\to\infty} ratio(r, m) = \frac{(m + 2)\lfloor r/4 \rfloor + \alpha_{r,m}}{rm} = \frac{1}{4} = 0.25 \ .$$

Even though huge r and m are not practical in the real world, the value can be understood as an indicator of the potential efficiency of a particular design strategy.

We propose these limits as a reference to evaluate the efficiency of the linear diffusion layer of the cipher and use them to compare ciphers employing different design strategies. The following table contains the convergence values of the "MDS-Feistel" and the"*ODM-MR*" design. Additionally, the following "Rijndael type" and "SHARK type" design approaches are also evaluated for reference.

Rijndael Type: A nm^2-bit SPN block cipher design whose round function consists of key-addition, $m \times m$ parallel n-bit S-boxes, a MixColumn employing m $m \times m$ matrices over $GF(2^n)$ and a ShiftRow operation [4].

Table 2. Comparison of the Active S-box Ratio

Type	$active(r,m)$	$total(r,m)$	128bit blk.	$\lim_{r\to\infty}$	$\lim_{m,r\to\infty}$
MDS-Feistel	$(m+2)\lfloor r/4\rfloor+\alpha_{r,m}$	rm	$m=8$	0.313	0.25
ODM-MR	$(m+1)\lfloor r/3\rfloor+\beta_{r,m}$	rm	$m=8$	0.371	0.33
Rijndael type	$(m+1)^2\lfloor r/4\rfloor+\gamma_{r,m}$	rm^2	$m=4$	0.391	0.25
SHARK type	$(m+1)\lfloor r/2\rfloor+\theta_{r,m}$	rm	$m=16$	0.531	0.5

SHARK Type: A nm-bit SPN block cipher design where m parallel n-bit S-boxes, an $m \times m$ matrix over $GF(2^n)$ are employed [12].

Note that all four designs employ optimal diffusion mappings in their diffusion layers; they have block length of 128 bits with 8-bit S-boxes. The result shows that the *ODM-MR* approach has the better limit than MDS-Feistel in the 128-bit block setting which is also confirmed by the empirical results in the previous section. We also know that *ODM-MR*'s limit is closer to that of the Rijndael design approach than MDS-Feistel.

Moreover, the limit value of the *ODM-MR* approach, when both r and m tend to infinity, exceeds that of the Rijndael type construction. This is due to the fact that the *ODM-MR* approach guarantees a certain number of active S-boxes for 3 consecutive rounds, while the Rijndael-type approach has such a property for 4 consecutive rounds.

The values of SHARK are still the highest, because the design strategy has a 2-round property. However, there seems to be a tradeoff for the implementation cost, as SHARK-type design requires matrices which are twice as large as the matrices in the MDS-Feistel and *ODM-MR* and four times as large as in the Rijndael approach.

6 Conclusion

We provide a theoretical motivation for the *ODM-MR design*. We first give a theoretical reason of *ODM-MR*, and found additional conditions and proofs to improve the immunity against differential *and* linear cryptanalysis. As a result, we showed that the *ODM-MR design* approach guarantees at least $R(m+1)$ active S-boxes in $3R$ consecutive rounds ($R \geq 2$) where m is the number of S-boxes in a round. This guaranteed number of active S-boxes was compared with the design approach of other well-known designs namely SHARK, Rijndael, and MDS-Feistel ciphers. We were able to show that our design approach outperforms some of the other designs.

Acknowledgments. We thank An Braeken and Christopher Wolf for carefully reading the earlier version of the paper. We thank the anonymous referees for helpful comments.

References

1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platforms." in *Proceedings of Selected Areas in Cryptography – SAC 2000* (D. R. Stinson and S. E. Tavares, eds.), no. 2012 in LNCS, pp. 41–54, Springer-Verlag, 2001.
2. P. S. L. M. Barreto and V. Rijmen, "The Whirlpool hashing function." Primitive submitted to NESSIE, Sept. 2000. Available at `http://www.cryptonessie.org/`.
3. E. Biham and A. Shamir, "Differential cryptanalysis of des-like cryptosystems." *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.
4. J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard (Information Security and Cryptography)*. Springer, 2002.
5. H. Feistel, "Cryptography and computer privacy." *Scientific American*, vol. 228, pp. 15–23, May 1973.
6. Data Encryption Standard, "Federal Information Processing Standard (FIPS)." National Bureau of Standards, U.S. Department of Commerce, Washington D.C., Jan. 1977.
7. M. Kanda, "Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function." in *Proceedings of Selected Areas in Cryptography – SAC'00* (D. R. Stinson and S. E. Tavares, eds.), no. 2012 in LNCS, pp. 324–338, Springer-Verlag, 2001.
8. M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions." *SIAM Journal on Computing*, vol. 17, pp. 373–386, 1988.
9. M. Matsui, "Linear cryptanalysis of the data encryption standard." in *Proceedings of Eurocrypt'93* (T. Helleseth, ed.), no. 765 in LNCS, pp. 386–397, Springer-Verlag, 1994.
10. M. Matsui, "New structure of block ciphers with provable security against differential and linear cryptanalysis." in *Proceedings of Fast Software Encryption – FSE'96* (D. Gollmann, ed.), no. 1039 in LNCS, pp. 205–218, Springer-Verlag, 1996.
11. K. Nyberg and L. R. Knudsen, "Provable security against a differential cryptanalysis." in *Proceedings of Crypto'92* (E. F. Brickell, ed.), no. 740 in LNCS, pp. 566–574, Springer-Verlag, 1993.
12. V. Rijmen, J. Daemen, B. Preneel, A. Bossalaers, and E. D. Win, "The cipher SHARK." in *Proceedings of Fast Software Encryption – FSE'96* (D. Gollmann, ed.), no. 1039 in LNCS, pp. 99–111, Springer-Verlag, 1996.
13. R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 block cipher." Primitive submitted to AES, 1998. Available at `http://www.rsasecurity.com/`.
14. B. Schneier and J. Kelsey, "Unbalanced Feistel networks and block cipher design." in *Proceedings of Fast Software Encryption – FSE'96* (D. Gollmann, ed.), no. 1039 in LNCS, pp. 121–144, Springer-Verlag, 1996.
15. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-bit block cipher." Primitive submitted to AES, 1998. Available at `http://www.schneier.com/`.
16. H. Shimizu, "On the security of Feistel cipher with SP-type F function." in *Proceedings of SCIS – SCIS 2001*, 2001.
17. T. Shirai, S. Kanamaru, and G. Abe, "Improved upper bounds of differential and linear characteristic probability for Camellia." in *Proceedings of Fast Software Encryption – FSE'02* (J. Daemen and V. Rijmen, eds.), no. 2365 in LNCS, pp. 128–142, Springer-Verlag, 2002.

18. T. Shirai and K. Shibutani, "Improving immunity of Feistel ciphers against differential cryptanalysis by using multiple MDS matrices." in *Proceedings of Fast Software Encryption – FSE'04* (B. Roy and W. Meier, eds.), no. 3017 in LNCS, pp. 260–278, Springer-Verlag, 2004.

Appendix A

We show one of methods to construct a Feistel cipher satisfying the *ODM-MR design*. To construct a concrete cipher, at least three $m \times m$ matrices over $GF(2^n)$ are required to satisfy all the *ODM-MR* conditions. The construction steps are:

1. Choose $m \times m$ matrices A_0, A_1, A_2 over $GF(2^n)$ such that,
   (a) Every square submatrix of $[A_0|A_1|A_2]$ is nonsingular,
   (b) Every square submatrix of $\left[\frac{A_0^{-1}}{A_1^{-1}}\right]$, $\left[\frac{A_1^{-1}}{A_2^{-1}}\right]$ and $\left[\frac{A_2^{-1}}{A_0^{-1}}\right]$ is nonsingular.
2. Set these three matrices as $M_{2i+1} = M_{2r-2i} = A_{\ i \bmod 3}$, for $(0 \le i < r)$ in a Feistel cipher with $2r$ rounds.

Note that all operations in Step 1 are over $GF(2^n)$ although the optimal diffusion conditions for $[M_i M_{i+2} M_{i+4}]$ and $[{}^t M_j^{-1\,t} M_{j+2}^{-1}]$ are given over $GF(2)$.

Here we show an example of three matrices A_0, A_1, A_2 for the case $m = 4$.

Example 1. The following matrices A_0, A_1, A_2 satisfy the *ODM-MR* conditions.

$$A_0 = \begin{pmatrix} \texttt{9d b4 d3 5d} \\ \texttt{29 34 39 60} \\ \texttt{67 6a d2 e3} \\ \texttt{8e d7 e6 1b} \end{pmatrix}, \quad A_1 = \begin{pmatrix} \texttt{ae ec b9 3e} \\ \texttt{81 25 13 d4} \\ \texttt{db 9d 4 1b} \\ \texttt{9e 3a 91 39} \end{pmatrix}, \quad A_2 = \begin{pmatrix} \texttt{b8 f1 65 ef} \\ \texttt{3a f6 2d 6a} \\ \texttt{4a 97 a3 b9} \\ \texttt{82 5f a2 c1} \end{pmatrix}.$$

Each element is expressed as hexadecimal value corresponding to a binary representation of elements in $GF(2^8)$ with a primitive polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$. From the corollary, a $(4, 8, 12)$-SPNFC employing the above matrices A_0, A_1, A_2 as outlined in Fig. 4 guarantees 10, 15 and 20 differentially and linearly active S-boxes in 6, 9 and 12 consecutive rounds, respectively.

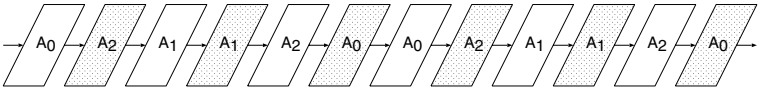


Fig. 4. Example Allocation of Matrices A_0, A_1, A_2

Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC

Phillip Rogaway

Dept. of Computer Science, University of California,
Davis CA 95616 USA
Dept. of Computer Science, Chiang Mai University,
Chiang Mai 50200 Thailand
rogaway@cs.ucdavis.edu
www.cs.ucdavis.edu/∼rogaway

Abstract. We describe highly efficient constructions, XE and XEX, that turn a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ into a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ having tweak space $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I}$ is a set of tuples of integers such as $\mathbb{I} = [1\,..\,2^{n/2}] \times [0\,..\,10]$. When tweak T is obtained from tweak S by incrementing one if its numerical components, the cost to compute $\widetilde{E}_K^T(M)$ having already computed some $\widetilde{E}_K^S(M')$ is one blockcipher call plus a small and constant number of elementary machine operations. Our constructions work by associating to the i^{th} coordinate of $\mathbb{I}$ an element $\alpha_i \in \mathbb{F}_{2^n}^*$ and multiplying by α_i when one increments that component of the tweak. We illustrate the use of this approach by refining the authenticated-encryption scheme OCB and the message authentication code PMAC, yielding variants of these algorithms that are simpler and faster than the original schemes, and yet have simpler proofs. Our results bolster the thesis of Liskov, Rivest, and Wagner [10] that a desirable approach for designing modes of operation is to start from a tweakable blockcipher. We elaborate on their idea, suggesting the kind of tweak space, usage-discipline, and blockcipher-based instantiations that give rise to simple and efficient modes.

1 Introduction

Liskov, Rivest and Wagner [10] defined the notion of a tweakable blockcipher and put forward the thesis that these objects make a good starting point for doing blockcipher-based cryptographic design. In this paper we describe a good way to build a tweakable blockcipher $\widetilde{E}$ out of an ordinary blockcipher E. Used as intended, our constructions, XE and XEX, add just a few machine instructions to the cost of computing E. We illustrate the use of these constructions by improving on the authenticated-encryption scheme OCB [15] and the message authentication code PMAC [4].

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 16–31, 2004.

TWEAKABLE BLOCKCIPHERS. Schroeppel [16] designed a blockcipher, Hasty Pudding, wherein the user supplies a non-secret *spice* and changing this spice produces a completely different permutation. Liskov, Rivest, and Wagner [10] formally defined the syntax and security measures for such a *tweakable* blockcipher, and they suggested that this abstraction makes a desirable starting point to design modes of operation and prove them secure. They suggested ways to build a tweakable blockcipher $\widetilde{E}$ out of a standard blockcipher E, as well as ways to modify existing blockcipher designs to incorporate a tweak. They illustrated the use of these objects. Formally, a tweakable blockcipher is a map $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where each $\widetilde{E}_K^T(\cdot) = \widetilde{E}(K,T,\cdot)$ is a permutation and $\mathcal{T}$ is the set of *tweaks*.

OUR CONTRIBUTIONS. We propose efficient ways to turn a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ into a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$. (See Appendix A for the best constructions formerly known.) Our *powering-up* constructions, XE and XEX, preserve the key space and blocksize of E but endow $\widetilde{E}$ with a tweak space $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I}$ is a set of tuples of integers, like $\mathbb{I} = [1\,..\,2^{n/2}] \times [0\,..\,10]$. The XE construction turns a CPA-secure blockcipher into a CPA-secure tweakable blockcipher, while XEX turns a CCA-secure blockcipher into a CCA-secure tweakable blockcipher. (CPA stands for chosen-plaintext attack and CCA for chosen-ciphertext attack.) The methods are highly efficient when tweaks arise in sequence, with most tweaks $(N, \mathbf{i})$ being identical to the prior tweak $(N, \mathbf{i}')$ except for incrementing a component of $\mathbf{i}$.

As an illustrative and useful example, consider turning a conventional blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ into a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ by defining $\widetilde{E}_K^{N\,i\,j}(M) = E_K(M \oplus \Delta) \oplus \Delta$ where offset $\Delta = 2^i 3^j \,\mathsf{N}$ and $\mathsf{N} = E_K(N)$. Arithmetic is done in the finite field $\mathbb{F}_{2^n}$. For concreteness, assume $n = 128$ and a tweak space of $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{64}] \times [0\,..\,10]$. We show that $\widetilde{E}$ is secure (as a strong, tweakable PRP) as long as E is secure (as a strong, untweakable PRP). Computing $\widetilde{E}_K^{N\,i\,j}(X)$ will usually cost about 1 shift, 1 conditional, and 3–4 xors more than computing $E_K(X)$.

We illustrate how the use of tweakable blockciphers during mode design, followed by the instantiation of the tweakable blockcipher with an ordinary blockcipher using one of our constructions, can give rise to modes that are simpler, faster, and easier to prove correct than what designing directly from a blockcipher has delivered. We do this by refining two already-optimized modes, OCB [15] and PMAC [4], yielding new modes, OCB1 and PMAC1, that are are easier to understand, easier to implement, and faster. Computing offsets in the new modes does not involve Gray-code sequence or counting the number of trailing zero bits in successive integers. OCB1 eliminates the utility of preprocessing, saving a blockcipher call.

INTUITION. The idea behind the powering-up constructions can be explained like this. Apart from Gray-code reordering, PMAC authenticates an m-block message using a sequence of offsets $L, 2L, 3L, \ldots, (m-1)L$, where multiplication is in the finite field $\mathbb{F}_{2^n}$ and $L = E_K(0^n)$ is a variant of the underlying

key K. When a special kind of offset is needed, a value $\mathit{huge} \cdot L$ is added (xored) into the current offset, where *huge* is so large that it could never be among $\{1, 2, \ldots, m-1\}$. What we now do instead is to use the easier-to-compute sequence of offsets $2^1L, 2^2L, \ldots, 2^{m-1}L$. We insist that our field be represented using a primitive polynomial instead of merely an irreducible one, which ensures that $2^1, 2^2, 2^3, \ldots, 2^{2^n-1}$ will all be distinct. When a special offset is needed we can no longer add to the current offset some huge constant times L and expect this never to land on a point in $2^1L, 2^2L, \ldots, 2^{m-1}L$. Instead, we multiply the current offset by 3 instead of 2. If the index of 3 (in $\mathbb{F}^*_{2^n}$) is enormous relative to the base 2 then multiplying by 3 is equivalent to multiplying by 2^{huge} and $2^i 3L$ won't be among of $2^1L, 2^2L, \ldots, 2^{m-1}L$ for any reasonable value of m. The current paper will make all of the ideas of this paragraph precise.

Further related work. Halevi and Rogaway [7] used the sequence of offsets $2L, 2^2L, 2^3L, \ldots$, in their EME mode. They give no general results about this construction, and EME did not use tweakable blockciphers, yet this offset ordering was our starting point.

2 Preliminaries

The field with 2^n points. Let $\mathbb{F}_{2^n}$ denote the field with 2^n points and let $\mathbb{F}^*_{2^n}$ be its multiplicative subgroup. We interchangeably think of a point $a \in \mathbb{F}_{2^n}$ as an n-bit string, a formal polynomial of degree $n-1$, or as an integer in $[0\,..\,2^n{-}1]$. To represent points select a primitive polynomial, say the lexicographically first one among the degree n polynomials having a minimum number of nonzero coefficients. For $n = 128$ the indicated polynomial is $p_{128}(\mathtt{x}) = \mathtt{x}^{128} + \mathtt{x}^7 + \mathtt{x}^2 + \mathtt{x} + 1$. Saying that $p_n(\mathtt{x})$ is primitive means that it is irreducible over $\mathbb{F}_2$ and 2 (i.e., $\mathtt{x}$) generates all of $\mathbb{F}^*_{2^n}$. It is computationally simple to multiply $a \in \{0,1\}^n$ by 2. To illustrate for $n = 128$, $2a = a{\ll}1$ if $\mathsf{firstbit}(a) = 0$ and $2a = (a{\ll}1) \oplus 0^{120}10^41^3$ if $\mathsf{firstbit}(a) = 1$. One can easily multiply by other small constants, as $3a = 2a \oplus a$ and $5a = 2(2a) \oplus a$ and so forth.

Blockciphers and tweakable blockciphers. We review the standard definitions for blockciphers and their security [2] and the extension of these notions to tweakable blockciphers [10]. A *blockcipher* is a function $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $n \geq 1$ is a number and $\mathcal{K}$ is a finite nonempty set and $E(K,\cdot) = E_K(\cdot)$ is a permutation for all $K \in \mathcal{K}$. A *tweakable blockcipher* is a function $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ where n and $\mathcal{K}$ are as above and $\mathcal{T}$ is a nonempty set and $\widetilde{E}(K,T,\cdot) = \widetilde{E}^T_K(\cdot)$ is a permutation for all $K \in \mathcal{K}$ and $T \in \mathcal{T}$. For blockciphers and tweakable blockciphers we call n the *blocksize* and $\mathcal{K}$ the *key space*. For tweakable blockciphers we call $\mathcal{T}$ the *tweak space*.

Let $\mathrm{Perm}(n)$ be the set of all permutations on n bits. Let $\mathrm{Perm}(\mathcal{T}, n)$ be the set of all mappings from $\mathcal{T}$ to permutations on n bits. In writing $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ we are choosing a random permutation $\pi(\cdot)$ on $\{0,1\}^n$. In writing $\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T}, n)$ we are choosing a random permutation $\pi(T,\cdot) = \pi_T(\cdot)$ on $\{0,1\}^n$ for each $T \in \mathcal{T}$.

If $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is a blockcipher then its inverse is the blockcipher $D = E^{-1}$ where $D\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is defined by $D(K,Y) = D_K(Y)$ being the unique point X such that $E_K(X) = Y$. If $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is a tweakable blockcipher then its inverse is the tweakable blockcipher $\widetilde{D} = \widetilde{E}^{-1}$ where $\widetilde{D}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is defined by $\widetilde{D}(K,T,Y) = \widetilde{D}_K^T(Y)$ being the unique point X such that $\widetilde{E}_K^T(X) = Y$.

An adversary is a probabilistic algorithm with access to zero or more oracles. Without loss of generality, adversaries never ask a query for which the answer is trivially known: an adversary does not repeat a query, does not ask $D_K(Y)$ after receiving Y in response to a query $E_K(X)$, and so forth. Oracles will have an implicit domain of valid queries and, for convenience, we assume that all adversarial queries lie within that domain. This is not a significant restriction because membership can be easily tested for all domains of interest to us.

Definition 1 (Blockcipher/Tweakable-Blockcipher Security). *Let* $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *be a blockcipher and let* $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *be a tweakable blockcipher. Let* A *be an adversary. Then* $\mathbf{Adv}^{\mathrm{prp}}_{E}(A)$, $\mathbf{Adv}^{\pm\mathrm{prp}}_{E}(A)$, $\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(A)$, *and* $\mathbf{Adv}^{\pm\widetilde{\mathrm{prp}}}_{\widetilde{E}}(A)$ *are defined by:*

$$\Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{E_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\pi(\cdot)} \Rightarrow 1]$$
$$\Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{E_K(\cdot)\, D_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n) :\ A^{\pi(\cdot)\, \pi^{-1}(\cdot)} \Rightarrow 1]$$
$$\Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{\widetilde{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n) :\ A^{\pi(\cdot,\cdot)} \Rightarrow 1]$$
$$\Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{\widetilde{E}_K(\cdot,\cdot)\, \widetilde{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n) :\ A^{\pi(\cdot,\cdot)\, \pi^{-1}(\cdot,\cdot)} \Rightarrow 1] \quad \square$$

Of course D and $\widetilde{D}$ denote the inverses of blockciphers E and $\widetilde{E}$. In writing $A \Rightarrow 1$ we are referring to the event that the adversary A outputs the bit 1.

In the usual way we lift advantage measures that depend on an adversary to advantage measures that depend on named resources: $\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(\mathcal{R}) = \max_A\{\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(A)\}$ over all adversaries A that use resources at most $\mathcal{R}$. The resources of interest to us are the total number of oracle queries q and the total length of those queries σ and the running time t. For convenience, the total length of queries will be measured in n-bit blocks, for some understood value of n, so a query X contributes $|X|_n$ to the total, where $|X|_n$ means $\max\{|X|/n, 1\}$. Running time, by convention, includes the description size of the algorithm relative to some standard encoding. When we speak of authenticity, the block length of the adversary's output is included in σ.

3 The XE and XEX Constructions

Goals. We want to support tweak sets that look like $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I}$ is a set of tuples of integers. In particular, we want to be able to make $\mathbb{I}$ the cross product of a large subrange of integers, like $[1\,..\,2^{n/2}]$, by the cross product of small ranges of integers, like $[0\,..\,10] \times [0\,..\,10]$. Thus an example tweak

space is $\mathcal{T} = \{0,1\}^n \times [1\,..\,2^{n/2}] \times [0\,..\,10] \times [0\,..\,10]$. Tweaks arise in some sequence $T_1, T_2, \ldots$ and we will obtain impressive efficiency only to the extent that most tweaks are an increment of the immediately prior one. When we say that tweak $T = (N, i_1, \ldots, i_k)$ is an increment of another tweak we mean that one of $i_1, \ldots, i_k$ got incremented and everything else stayed the same. The second component of tweak $(N, i_1, \ldots, i_k)$, meaning i_1, is the component that we expect to get incremented most often. We want there to be a simple, constant-time procedure to increment a tweak at any given component of $\mathbb{I}$. To increment a tweak it shouldn't be necessary to go to memory, consult a table, or examine which number tweak this in in sequence. Incrementing tweaks should be endian-independent and avoid extended-precision arithmetic. Efficiently incrementing tweaks shouldn't require precomputation. Tweaks that are not the increment of a prior tweak will also arise, and they will typically look like $(N, 1, 0 \ldots, 0)$. Constructions should be reasonably efficient in dealing with such tweaks.

We emphasize that the efficiency measure we are focusing on is not the cost of computing $\widetilde{E}_K^T(X)$ from scratch—by that measure our constructions will not be particularly good. Instead, we are interested in the cost of computing $\widetilde{E}_K^T(X)$ given that one has just computed $\widetilde{E}_K^S(X')$ and T is obtained by incrementing S at some component. Most often that component will have been the second component of S. It is a thesis underlying our work, supported by the design of OCB1 and PMAC1, that one will often be able to arrange that most tweaks are an increment to the prior one.

Tweaking with $\Delta = 2^i\,\mathsf{N}$. Recall that we have chosen to represent points in $\mathbb{F}_{2^n}$ using a primitive polynomial, not just an irreducible one. This means that the point 2 is a generator of $\mathbb{F}_{2^n}$: the points $1, 2, 2^2, 2^3, \ldots, 2^{2^n-2}$ are all distinct. This property turns out to be the crucial one that lets us construct from a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times [1\,..\,2^n - 2]) \times \{0,1\}^n \to \{0,1\}^n$ by way of

$$\widetilde{E}_K^{N\,i}(M) = E_K(M \oplus \Delta) \oplus \Delta \quad \text{where} \quad \Delta = 2^i\,\mathsf{N} \quad \text{and} \quad \mathsf{N} = E_K(N).$$

The tweak set is $\mathcal{T} = \{0,1\}^n \times \mathbb{I}$ where $\mathbb{I} = [1\,..\,2^n - 2]$ and the tweakable blockcipher just described is denoted $\widetilde{E} = \mathrm{XEX}[E, 2^{\mathbb{I}}]$. When computing the sequence of values $\widetilde{E}_K^{N\,1}(M_1), \ldots, \widetilde{E}_K^{N\,m-1}(M_{m-1})$ each $\widetilde{E}_K^{N\,i}(M_i)$ computation but the first uses one blockcipher call and one doubling operation. Doubling takes a shift followed by a conditional xor. We call the construction above, and all the subsequent constructions of this section, *powering-up* constructions.

Tweaking by $\Delta = 2^i 3^j\,\mathsf{N}$. To facilitate mode design we may want tweaks that look like (N, i, j) where $N \in \{0,1\}^n$ and i is an integer from a large set $\mathbb{I}$, like $\mathbb{I} = [1\,..\,2^{n/2}]$, and j is an integer from some small set $\mathbb{J}$, like $\mathbb{J} = \{0, 1\}$. To get the "diversity" associated to the various j-values we just multiply by 3 instead of 2. That is, we construct from a blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times \mathbb{I} \times \mathbb{J}) \times \{0,1\}^n \to \{0,1\}^n$ by way of

$$\widetilde{E}_K^{N\,i\,j}(M) = E_K(M \oplus \Delta) \oplus \Delta \quad \text{where} \quad \Delta = 2^i 3^j\,\mathsf{N} \quad \text{and} \quad \mathsf{N} = E_K(N).$$

The tweakable blockcipher just described is denoted $\widetilde{E} = \mathrm{XEX}[E, 2^{\mathbb{I}}3^{\mathbb{J}}]$. Incrementing the tweak at component i is done by doubling, while incrementing the tweak at component j is done by tripling.

The XEX construction. Generalizing the two examples above, we have the following definition.

Definition 2 (XEX). *Let* $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *be a blockcipher, let* $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*_{2^n}$*, and let* $\mathbb{I}_1, \ldots, \mathbb{I}_k \subseteq \mathbb{Z}$*. Then* $\widetilde{E} = \mathsf{XEX}[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$ *is the tweakable blockcipher* $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times \mathbb{I}_1 \times \cdots \times \mathbb{I}_k) \times \{0,1\}^n \to \{0,1\}^n$ *defined by* $\widetilde{E}_K^{Ni_1 \ldots i_k}(M) = E_K(M \oplus \Delta) \oplus \Delta$ *where* $\Delta = \alpha_1^{i_1}\alpha_2^{i_2} \cdots \alpha_k^{i_k}\,\mathsf{N}$ *and* $\mathsf{N} = E_K(N)$.

The XE construction. As made clear in the work of Liskov, Rivest, and Wagner [10], constructions of the form $\widetilde{E}_K^T(M) = E_K(M \oplus \Delta) \oplus \Delta$ aim for chosen-ciphertext attack (CCA) security, while for chosen-plaintext attack (CPA) security one can omit the outer xor. Thus we consider the construction $E_K(M \oplus \Delta)$. This is slightly more efficient than XEX, saving one xor.

Definition 3 (XE). *Let* $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *be a blockcipher,* $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*_{2^n}$*, and* $\mathbb{I}_1, \ldots, \mathbb{I}_k \subseteq \mathbb{Z}$*. Then* $\widetilde{E} = \mathsf{XE}[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$ *is the tweakable blockcipher* $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times \mathbb{I}_1 \times \cdots \times \mathbb{I}_k) \times \{0,1\}^n \to \{0,1\}^n$ *defined by* $\widetilde{E}_K^{Ni_1 \ldots i_k}(M) = E_K(M \oplus \Delta)$ *where* $\Delta = \alpha_1^{i_1}\alpha_2^{i_2} \cdots \alpha_k^{i_k}\,\mathsf{N}$ *and* $\mathsf{N} = E_K(N)$. □

4 Parameter Sets Yielding Unique Representations

It is easy to see that the XE and XEX constructions can only "work" if $\alpha_1^{i_1} \cdots \alpha_k^{i_k}$ are distinct throughout $(i_1, \ldots, i_k) \in \mathbb{I}_1 \times \cdots \times \mathbb{I}_k$. This motivates the following definition.

Definition 4 (Unique Representations). *Fix a group* G*. A* ***choice of parameters*** *is a list* $\alpha_1, \ldots, \alpha_k \in G$ *of* ***bases*** *and a set* $\mathbb{I}_1 \times \cdots \times \mathbb{I}_k \subseteq \mathbb{Z}^k$ *of* ***allowed indices****. We say that the choice of parameters provides* ***unique representations*** *if for every* $(i_1, \ldots, i_k)$, $(j_1, \ldots, j_k) \in \mathbb{I}_1 \times \cdots \times \mathbb{I}_k$ *we have that* $\alpha_1^{i_1} \cdots \alpha_k^{i_k} = \alpha_1^{j_1} \cdots \alpha_k^{j_k}$ *implies* $(i_1, \ldots, i_k) = (j_1, \ldots, j_k)$. □

In other words, representable points are uniquely representable: any group element $\alpha_1^{i_1} \cdots \alpha_k^{i_k}$ that can be represented using allowed indices can be represented in only one way (using allowed indices).

For tweak spaces of practical interest, discrete-log calculations within $\mathbb{F}^*_{2^n}$ can be used to help choose and verify that a given choice of parameters provides unique representations. The following result gives examples for $\mathbb{F}^*_{2^{128}}$.

Proposition 1. [Can Use 2, 3, 7 When $n = 128$] *In the group* $\mathbb{F}^*_{2^{128}}$ *the following choices for parameters provide unique representations:*

(1) $\alpha_1 = 2$ *and* $\mathbb{I}_1 = [-2^{126} .. 2^{126}]$.

(2) $\alpha_1, \alpha_2 = 2, 3$ *and* $\mathbb{I}_1 \times \mathbb{I}_2 = [-2^{115} .. 2^{115}] \times [-2^{10} .. 2^{10}]$.

(3) $\alpha_1, \alpha_2, \alpha_3 = 2, 3, 7$ *and* $\mathbb{I}_1 \times \mathbb{I}_t \times \mathbb{I}_3 = [-2^{108} .. 2^{108}] \times [-2^7 .. 2^7] \times [-2^7 .. 2^7]$.

Proof. For statement (1) recall that 2 is a generator of the group (by our choice of irreducible polynomial) and the order of the group $\mathbb{F}^*_{2^{128}}$ is $2^{128}-1$ and so $2^i = 2^j$ iff $i = j \pmod{2^{128}-1}$ and so any contiguous range of $2^{128}-1$ or fewer integers will provide unique representations with respect to base 2.

To prove statement (2) we need to compute $\log_2 3$ in the group $\mathbb{F}^*_{2^{128}}$:

$$\log_2 3 = 338793687469689340204974836150077311399 \quad \text{(decimal)}$$

This and subsequent discrete logs were computed using a Maple-implementation combining the Pohlig-Hellman [11] and Pollard-rho [12] algorithms. (A naive implementation computes discrete logs in $\mathbb{F}^*_{2^{128}}$ in a few hours.) Now note that $2^a 3^b = 2^{a'} 3^{b'}$ iff $2^a 2^{b \log_2 3} = 2^{a'} 2^{b' \log_2 3}$ iff $2^{a+b\log_2 3} = 2^{a'+b'\log_2 3}$ iff $a + b\log_2 3 = a' + b' \log_2 3 \pmod{2^{128}-1}$ because 2 is a generator of the group $\mathbb{F}^*_{2^{128}}$. Thus $2^a 3^b = 2^{a'} 3^{b'}$ iff $a - a' = (b' - b)\log_2 3 \pmod{2^{128}-1}$. If $b, b' \in [-2^{10} .. 2^{10}]$ then $\Delta_b = b' - b \in [-2^{11} .. 2^{11}]$ and computer-assisted calculation then shows that the smallest value of $\Delta_b \log_2 3 \pmod{2^{128}-1}$ for $\Delta_b \in [-2^{11}..2^{11}]$ and $\Delta_b \neq 0$ is $1600 \log_2 3 = 00113a0ce508326c006763c0b80c59f9$ (in hexadecimal) which is about $2^{116.1}$. (By "smallest" we refer to the distance from 0, modulo $2^{128}-1$, so 2^{100} and $(2^{128}-1) - 2^{100}$ are equally small, for example.) Thus if a, a' are restricted to $[-2^{115} .. 2^{115}]$ and b, b' are restricted to $[-2^{10} .. 2^{10}]$ then $\Delta_a = a - a' \leq 2^{116}$ can never equal $\Delta_b \log_2 3 \pmod{2^{128}-1} > 2^{116}$ unless $\Delta_b = 0$. This means that the only solution to $2^a 3^b = 2^{a'} 3^{b'}$ within the specified range is $a = a'$ and $b = b'$.

To prove statement (3) is similar. First we need the value

$$\log_2 7 = 30504680247268818232978065568589919539 6 \quad \text{(decimal)}$$

Now $2^a 3^b 7^c = 2^{a'} 3^{b'} 7^{c'}$ iff $a - a' = (b' - b)\log_2 3 + (c' - c)\log_2 7 \pmod{2^{128}-1}$. The smallest value for $\Delta_b \log_2 3 + \Delta_c \log_2 7 \pmod{2^{128}-1}$ when $\Delta_b, \Delta_c \in [-2^8 .. 2^8]$ and at least one of these is non-zero is $-48\log_2 3 + 31 \log_2 7 \pmod{2^{128}-1} = 00003bfabac91e02b278b7e69a379d18$ (hexadecimal) which is about $2^{109.9}$. So restricting the index for base-2 to $[-2^{108} .. 2^{108}]$ ensures that $a - a' \leq 2^{109}$ while $(b' - b)\log_2 3 + (c' - c)\log_2 7 > 2^{109}$ unless $b = b'$ and $c = c'$ and $a = a'$. □

We emphasize that not just any list of bases will work. Notice, for example, that $3^2 = 5$ in $\mathbb{F}^*_{2^n}$ so the list of bases 2, 3, 5 does *not* give unique representations, even for a tiny list of allowed indices like $\mathbb{I}_1 \times \mathbb{I}_2 \times \mathbb{I}_3 = \{0, 1, 2\}^3$.

Similar calculations can be done in other groups; here we state the analogous result for $\mathbb{F}^*_{2^{64}}$.

Proposition 2. [Can Use 2, 3, 11 When $n = 64$] *In the group $\mathbb{F}^*_{2^{64}}$ the following choices for parameters provide unique representations:*

(1) $\alpha_1 = 2$ *and* $[-2^{62} .. 2^{62}]$.
(2) $\alpha_1, \alpha_2 = 2, 3$ *and* $[-2^{51} .. 2^{51}] \times [-2^{10} .. 2^{10}]$.
(3) $\alpha_1, \alpha_2, \alpha_3 = 2, 3, 11$ *and* $[-2^{44} .. 2^{44}] \times [-2^7 .. 2^7] \times [-2^7 .. 2^7]$. □

This time 2, 3, 7 does *not* work as a list of bases, even with a small set of allowed indices like $[1 .. 64] \times \{0, 1, 2\} \times \{0, 1, 2\}$, due to the fact that $2^{64} = 3^2 \cdot 7$

in this group. Machine-assisted verification seems essential here; a relation like that just given is found immediately when computing the possible values for $\Delta_b \log_2 3 + \Delta_c \log_2 7 \pmod{2^{64}-1}$ but it might not otherwise be anticipated.

5 Security of XE

The following result quantifies the security of the XE construction.

Theorem 1. [Security of XE] *Fix $n \geq 1$ and let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*_{2^n}$ be base elements and let $\mathbb{I}_1 \times \cdots \times \mathbb{I}_k$ be allowed indices such that these parameters provide unique representations. Fix a blockcipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and let $\widetilde{E} = \mathsf{XE}[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$. Then $\mathbf{Adv}^{\widetilde{\text{prp}}}_{\widetilde{E}}(t,q) \leq \mathbf{Adv}^{\text{prp}}_{E}(t', 2q) + \frac{4.5\, q^2}{2^n}$ where $t' = t + ckn(q+1)$ for some absolute constant c.* □

In English, the XE construction promotes a CPA-secure blockcipher to a CPA-secure tweakable blockcipher, assuming that the chosen base elements and range of allowed indices provide unique representations. The proof is in [14].

6 Security of XEX

Some added care is needed to address the security of XEX. Suppose, to be concrete, that we are looking at $\mathsf{XEX}[E, 2^{\mathbb{I}}]$ and $\mathbb{I} = [0\,..\,2^{n-2}]$. Let the adversary ask a deciphering query with ciphertext $C = 0^n$ and tweak $(0^n, 0)$. If the adversary has a construction-based deciphering oracle then it will get a response of $M = \widetilde{D}_K^{0^n\,0}(0^n) = D_K(\Delta) \oplus \Delta = D_K(\mathsf{N}) \oplus \mathsf{N} = 0^n \oplus \mathsf{N} = \mathsf{N}$, where $\mathsf{N} = E_K(0^n) = \Delta$. This allows the adversary to defeat the CCA-security. For example, enciphering $2M = 2\mathsf{N}$ with a tweak of $(0^n, 1)$ and enciphering $4M = 4\mathsf{N}$ with a tweak of $(0^n, 2)$ will give identical results (if the adversary has the construction-based enciphering oracle). Corresponding to this attack we exclude any tweak $(N, i_1, \ldots, i_k)$ for which $(i_1, \ldots, i_k)$ is a representative of 1—that is, any tweak $(N, i_1, \ldots, i_k)$ for which $\alpha_1^{i_1} \ldots \alpha_k^{i_k} = 1$. In particular, this condition excludes any tweak $(N, 0, \ldots, 0)$. The proof of the following is omitted, as Theorem 3 will be more general.

Theorem 2 (Security of XEX). *Fix $n \geq 1$ and let $\alpha_1, \ldots, \alpha_k \in \mathbb{F}^*_{2^n}$ be base elements and let $\mathbb{I}_1 \times \cdots \times \mathbb{I}_k$ be allowed indices such that these parameters provide unique representations. Assume $\alpha_1^{i_1} \cdots \alpha_k^{i_k} \neq 1$ for all $(i_1, \ldots, i_k) \in \mathbb{I}_1 \times \cdots \times \mathbb{I}_k$. Fix a blockcipher E: $\mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and let $\widetilde{E} = \mathsf{XEX}[E, \alpha_1^{\mathbb{I}_1} \cdots \alpha_k^{\mathbb{I}_k}]$. Then $\mathbf{Adv}^{\pm\widetilde{\text{prp}}}_{\widetilde{E}}(t,q) \leq \mathbf{Adv}^{\pm\text{prp}}_{E}(t', 2q) + \frac{9.5\, q^2}{2^n}$ where $t' = t + ckn(q+1)$ for some absolute constant c.* □

7 An Almost-Free Alternative to Key Separation

When combining two blockcipher-based cryptographic mechanisms into a composite mechanism, it is, in general, essential to use two different keys. Either

these two keys together comprise the key for the joint mechanism, or else each key is obtained from an underlying one by a key-derivation technique. The first possibility increases the key length in the composite mechanism while the second involves extra computation at key setup. Both possibilities incur the inefficiency of blockcipher re-keying when the combined mode runs. For all of these reasons, some new "composite" modes of operation have gone to considerable trouble in order to make do (for their particular context) with a *single* blockcipher key. Examples include EAX, CCM, and OĊB [3, 13, 17]. Using a single key complicates proofs—when the mechanism works at all—because one can no longer reason about generically combining lower-level mechanisms.

Tweakable blockciphers open up a different possibility: the same underlying key is used across the different mechanisms that are being combined, but one arranges that the tweaks are disjoint across different mechanisms. In this way one retains the modularity of design and analysis associated to using separate keys—one reasons in terms of generic composition—yet one can instantiate in a way that avoids having extra key material or doing extra key setups. Because the tweak space for XE and XEX is a Cartesian product of ranges of integers, it is easy, for these constructions, to separate the different tweaks.

8 Combining XE and XEX

Some blockcipher-based constructions need CCA-security in some places and CPA-security in other places. One could assume CCA-security throughout, later instantiating all blockcipher calls with a CCA-secure construction, but it might be better to use a CPA-secure construction where sufficient and a CCA-secure one where necessary. Regardless of subsequent instantiation, it is good to be able to talk, formally, about *where* in a construction one needs *what* assumption.

To formalize where in a construction one is demanding what, we *tag* each blockcipher call with an extra bit. We say that a tweakable blockcipher $\tilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ is *tagged* if $\mathcal{T} = \{0,1\} \times \mathcal{T}^*$ for some nonempty set $\mathcal{T}^*$. Think of $\mathcal{T}^*$, the *effective tweak space*, as the tweak space actually used by the mode. The extra bit indicates what is demanded for each tweak. A first bit of 0 indicates a demand of CPA security, and 1 indicates a demand for CCA security. For a given $T \in \mathcal{T}$ one should be asking for one or the other.

An adversary A launching an attack on a tagged blockcipher is given two oracles, $e(\cdot,\cdot)$ and $d(\cdot,\cdot)$, where the second oracle computes the inverse of the first (meaning $d(T,Y)$ is the unique X such that $e(T,X) = Y$). The adversary must respect the semantics of the tags, meaning that the adversary may not make any query $d(T,Y)$ where the first component of T is 0, and if the adversary makes an oracle query with a tweak $(b,\ T^*)$ then it may make no subsequent query with a tweak $(1-b,\ T^*)$. As always, we insist that there be no pointless queries: an adversary may not repeat an $e(T,X)$ query or a $d(T,Y)$ query, and it may not ask $d(T,Y)$ after having learned $Y = e(T,X)$, nor ask $e(T,X)$ after having learned $X = d(T,Y)$. The definition for security is now as follows.

Definition 5 (Security of a Tagged, Tweakable Blockcipher). *Let* $\widetilde{E}\colon \mathcal{K}\times \mathcal{T}\times\{0,1\}^n \to \{0,1\}^n$ *be a tagged, tweakable blockcipher and let* A *be an adversary. Then* $\mathbf{Adv}^{[\pm]\widetilde{\mathrm{prp}}}_{\widetilde{E}}(A)$ *is defined as* $\Pr[K \xleftarrow{\$} \mathcal{K}\colon\ A^{\widetilde{E}_K(\cdot,\cdot)\,\widetilde{D}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{T},n)\colon\ A^{\pi(\cdot,\cdot)\,\pi^{-1}(\cdot,\cdot)} \Rightarrow 1]$ □

Naturally $\widetilde{D}$, above, is the inverse of $\widetilde{E}$. Security in the $\widetilde{\mathrm{prp}}$-sense and security in the $\pm\widetilde{\mathrm{prp}}$-sense are special cases of security in the $[\pm]\widetilde{\mathrm{prp}}$ sense (but for the enlarged tweak space).

If we combine XE and XEX using our tagging convention we get the tagged, tweakable blockcipher XEX*.

Definition 6 (XEX*). *Let* $E\colon \mathcal{K}\times\{0,1\}^n \to \{0,1\}^n$ *be a blockcipher, let* $\alpha_1,\ldots,\alpha_k \in \mathbb{F}^*_{2^n}$, *and let* $\mathbb{I}_1,\ldots,\mathbb{I}_k \subseteq \mathbb{Z}$. *Then* $\widetilde{E} = XEX^*[E,\ \alpha_1^{\mathbb{I}_1}\cdots\alpha_k^{\mathbb{I}_k}]$ *is the tweakable blockcipher* $\widetilde{E}\colon \mathcal{K}\times(\{0,1\}\times\{0,1\}^n\times\mathbb{I}_1\cdots\times\mathbb{I}_k)\times\{0,1\}^n\to\{0,1\}^n$ *defined by* $\widetilde{E}_K^{0\,N\,i_1\ldots i_k}(M) = E_K(M\oplus\Delta)$ *and* $\widetilde{E}_K^{1\,N\,i_1\ldots i_k}(M) = E_K(M\oplus\Delta)\oplus\Delta$ *where* $\Delta = \alpha_1^{i_1}\alpha_2^{i_2}\cdots\alpha_k^{i_k}\,\mathsf{N}$ *and* $\mathsf{N} = E_K(N)$. □

9 Security of the Combined Construction

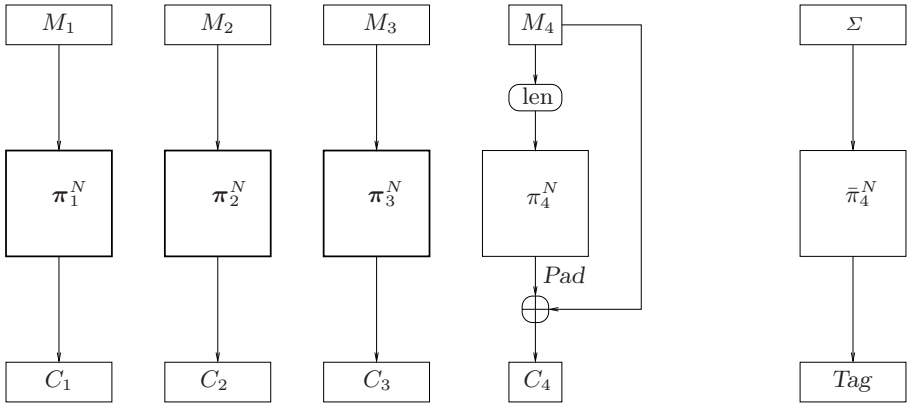
We now specify the security of the XEX* construction. The result encompasses that XE is $\widetilde{\mathrm{prp}}$-secure and XEX is $\pm\widetilde{\mathrm{prp}}$-secure. The proof is in [14].

Theorem 3 (Security of XEX*). *Fix* $n\ge 1$ *and let* $\alpha_1,\ldots,\alpha_k \in \mathbb{F}^*_{2^n}$ *be base elements and let* $\mathbb{I}_1\times\cdots\times\mathbb{I}_k$ *be allowed indices such that these parameters provide unique representations and such that* $\alpha_1^{i_1}\cdots\alpha_k^{i_k} \mathrel{/=} 1$ *for all* $(i_1,\ldots,i_k)\in\mathbb{I}_1\times\cdots\times\mathbb{I}_k$. *Fix a blockcipher* $E\colon \mathcal{K}\times\{0,1\}^n\to\{0,1\}^n$ *and let* $\widetilde{E} = XEX^*[E,\alpha_1^{\mathbb{I}_1}\cdots\alpha_k^{\mathbb{I}_k}]$. *Then* $\mathbf{Adv}^{[\pm]\widetilde{\mathrm{prp}}}_{\widetilde{E}}(t,q) \le \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t',2q) + \frac{9.5\,q^2}{2^n}$ *where* $t' = t + ckn(q+1)$ *for some absolute constant* c. □

10 The OCB1 Authenticated-Encryption Scheme

We recast OCB [15] to use a tweakable blockcipher instead of a conventional blockcipher. Liskov, Rivest, and Wagner first did this [10], but our formulation is different from theirs. First, guided by what we have done so far, we choose a tweak space of $\mathcal{T} = \{0,1\}\times\{0,1\}^n\times[1\,..\,2^{n/2}]\times\{0,1\}$. The first bit of the tweak is the tag; the effective tweak space is $\mathcal{T}^* = \{0,1\}^n\times[1\,..\,2^{n/2}]\times\{0,1\}$. Second, we want tweaks to increase monotonically, and so we switch the "special" processing done in OCB from the penultimate block to the final block. The resulting algorithm is shown in Fig. 1. Algorithm OCB1 is parameterized by a tweakable blockcipher $\widetilde{E}\colon K\times\mathcal{T}\times\{0,1\}^n\to\{0,1\}^n$ and a number $\tau\in[0\,..\,n]$. For clarity, we write $\boldsymbol{\pi}_i^N$ for $\widetilde{E}_K^{1\,N\,i\,0}$ and π_i^N for $\widetilde{E}_K^{0\,N\,i\,0}$ and $\bar{\pi}_i^N$ for $\widetilde{E}_K^{0\,N\,i\,1}$.

The security of OCB1[Perm($\mathcal{T},n$)] is much simpler to prove than the security of OCB[Perm(n)]. (Liskov, Rivest, and Wagner [10] had made the same point for their tweakable-blockcipher variant of OCB.) To state the result we



Algorithm OCB1.Encrypt$_K^N$ (M)

Partition M into $M[1] \cdots M[m]$
for $i \in [1 \,..\, m-1]$ **do** $C[i] \leftarrow \boldsymbol{\pi}_i^N(M[i])$
$Pad \leftarrow \pi_m^N(\mathrm{len}(M[m]))$
$C[m] \leftarrow M[m] \oplus Pad$
$C \leftarrow C[1] \cdots C[m]$
$\Sigma \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus$
$\quad C[m]0^* \oplus Pad$
$Tag \leftarrow \bar{\pi}_m^N(\Sigma)$
$T \leftarrow Tag$ [first τ bits]
return $\mathcal{C} \leftarrow C \,\|\, T$

Algorithm OCB1.Decrypt$_K^N$ $(\mathcal{C})$

Partition $\mathcal{C}$ into $C[1] \cdots C[m]\, T$
for $i \in [1 \,..\, m-1]$ **do** $M[i] \leftarrow \left(\boldsymbol{\pi}_i^N\right)^{-1}(C[i])$
$Pad \leftarrow \pi_m^N(\mathrm{len}(C[m]))$
$M[m] \leftarrow C[m] \oplus Pad$
$M \leftarrow M[1] \cdots M[m]$
$\Sigma \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^* \oplus Pad$
$Tag \leftarrow \pi_m^N(\Sigma)$
$T' \leftarrow Tag$ [first τ bits]
if $T = T'$ **then return** M
else return INVALID

Fig. 1. OCB1$[\widetilde{E}, \tau]$ with a tweakable blockcipher $\widetilde{E}$: $\mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ and tweak space $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1 \,..\, 2^{n/2}] \times \{0,1\}$ and tag length $\tau \in [0 \,..\, n]$. We write $\boldsymbol{\pi}_i^N$ and π_i^N and $\bar{\pi}_i^N$ for $\widetilde{E}_K^{1\,N\,i\,0}$ and $\widetilde{E}_K^{0\,N\,i\,0}$ and $\widetilde{E}_K^{0\,N\,i\,1}$

give a couple of definitions from [15]. For privacy of a nonce-based encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ we use the notion of indistinguishability-from-random-strings, which defines $\mathbf{Adv}_{\Pi}^{\mathrm{priv}}(A)$ as $\Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[A^{\$(\cdot,\cdot)} \Rightarrow 1]$. Here $\$(\cdot,\cdot)$ is an oracle that, on input (N, M), returns $|M|$ random bits. The adversary is not allowed to repeat a nonce N. For authenticity we use the nonce-based notion of integrity of ciphertexts: the adversary is given an encryption oracle $\mathcal{E}_K(\cdot,\cdot)$ and is said to *forge* if it outputs an $(N, \mathcal{C})$ that is valid and $\mathcal{C}$ was not the result of any prior (N, M) query. The adversary is not allowed to repeat a nonce N while it queries its encryption oracle. We write $\mathbf{Adv}_{\Pi}^{\mathrm{auth}}(A)$ for $\Pr[K \xleftarrow{\$} \mathcal{K} :\ A^{\mathcal{E}_K(\cdot,\cdot)}$ forges $]$. We have the following theorem for the information-theoretic security of OCB1. The proof is in [14].

Theorem 4 (OCB1 with an Ideal Tweakable Blockcipher). *Fix $n \geq 1$, $\tau \in [0 \,..\, n]$, and $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1 \,..\, 2^{n/2}] \times \{0,1\}$. Let A be an adver-*

Algorithm OCB1.Encrypt$_K^N$ (M)
Partition M into $M[1] \cdots M[m]$
$\Delta \leftarrow 2\,E_K(N)$
$\Sigma \leftarrow 0^n$
for $i \in [1\,..\,m-1]$ **do**
  $C[i] \leftarrow E_K(M[i] \oplus \Delta) \oplus \Delta$
  $\Delta \leftarrow 2\Delta$
  $\Sigma \leftarrow \Sigma \oplus M[i]$
$Pad \leftarrow E_K(\mathrm{len}(M[m]) \oplus \Delta)$
$C[m] \leftarrow M[m] \oplus Pad$
$C \leftarrow C[1] \cdots C[m]$
$\Sigma \leftarrow \Sigma \oplus C[m]0^* \oplus Pad$
$\Delta \leftarrow 3\Delta$
$Tag \leftarrow E_K(\Sigma \oplus \Delta)$
$T \leftarrow Tag$ [first τ bits]
return $\mathcal{C} \leftarrow C \,\|\, T$

Algorithm OCB1.Decrypt$_K^N$ $(\mathcal{C})$
Partition $\mathcal{C}$ into $C[1] \cdots C[m]\,T$
$\Delta \leftarrow 2\,E_K(N)$
$\Sigma \leftarrow 0^n$
for $i \in [1\,..\,m-1]$ **do**
  $M[i] \leftarrow E_K^{-1}(C[i] \oplus \Delta) \oplus \Delta$
  $\Delta \leftarrow 2\Delta$
  $\Sigma \leftarrow \Sigma \oplus M[i]$
$Pad \leftarrow E_K(\mathrm{len}(C[m]) \oplus \Delta)$
$M[m] \leftarrow C[m] \oplus Pad$
$M \leftarrow M[1] \cdots M[m]$
$\Sigma \leftarrow \Sigma \oplus C[m]0^* \oplus Pad$
$\Delta \leftarrow 3\Delta$
$Tag \leftarrow E_K(\Sigma \oplus \Delta)$
$T' \leftarrow Tag$ [first τ bits]
if $T = T'$ **then return** M
  else return INVALID

Fig. 2. OCB1$[E, \tau]$ with a conventional blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ and a tag length $\tau \in [0\,..\,n]$. This coincides with OCB1$[\widetilde{E}, \tau]$ where $\widetilde{E} = \mathrm{XEX}[E, 2^{[1\,..\,2^{n/2}]}3^{\{0,1\}}]$

sary. Then $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB1}[\mathrm{Perm}(\mathcal{T},n),\tau]}(A) = 0$ *and* $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB1}[\mathrm{Perm}(\mathcal{T},n),\tau]}(A) \le 2^{n-\tau}/(2^n - 1)$. □

Note that the authenticity bound is close to $2^{-\tau}$; in particular, $2^{n-\tau}/(2^n - 1) \le 1/(2^\tau - 1)$ for all $\tau \ge 2$. The bounds do not degrade with the number of queries asked by the adversary, the length of these queries, or the time the adversary runs. For the complexity-theoretic analog we have the following.

Corollary 1 (OCB1 with a Tweakable Blockcipher). *Fix* $n \ge 1$, $\tau \in [0\,..\,n]$, $\mathcal{T} = \{0,1\} \times \{0,1\}^n \times [1\,..\,2^{n/2}] \times \{0,1\}$, *and* $\widetilde{E}\colon \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ *a tagged, tweakable blockcipher. Then* $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB1}[\widetilde{E},\tau]}(t, \sigma) \le \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\widetilde{E}}(t', \sigma)$ *and* $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB1}[\widetilde{E},\tau]}(t, \sigma) \le \mathbf{Adv}^{\widetilde{[\pm]\mathrm{prp}}}_{\widetilde{E}}(t', \sigma) + 2^{n-\tau}/(2^n - 1)$, *where* $t' = t + cn\sigma$ *for some absolute constant* c. □

The proof requires CPA-security for privacy but authenticity uses the notion that combines CPA- and CCA-security (Definition 5). It is here that one has formalized the intuition that the first $m-1$ tweakable-blockcipher calls to OCB1 need to be CCA-secure but the last two calls need only be CPA-secure.

To realize OCB1 with a conventional blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, use XEX*, instantiating OCB1$[\widetilde{E}, \tau]$ by way of $\widetilde{E} = \mathrm{XEX}^*[E, 2^{\mathbb{I}}3^{\mathbb{J}}]$ where $\mathbb{I} = [1\,..\,2^{n/2}]$ and $\mathbb{J} = \{0,1\}$. Overloading the notation, we write this scheme as OCB1$[E, \tau]$. The method is rewritten in Fig. 2.

Corollary 2 (OCB1 with a Blockcipher). *Fix $n \geq 1$ and $\tau \in [0\,..\,n]$. Assume that 2, 3 provide unique representations on $[1\,..\,2^{n/2}] \times \{0,1\}$ and $2^i 3^j \neq 1$ for all $(i,j) \in [1\,..\,2^{n/2}] \times \{0,1\}$. Let $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Then*

- $\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB1}[E,\tau]}(t,\sigma) \;\leq\; \mathbf{Adv}^{\mathrm{prp}}_{E}(t',2\sigma) + \dfrac{4.5\sigma^2}{2^n}$ *and*
- $\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB1}[E,\tau]}(t,\sigma) \;\leq\; \mathbf{Adv}^{\pm\mathrm{prp}}_{E}(t',2\sigma) + \dfrac{9.5\sigma^2}{2^n} + \dfrac{2^{n-\tau}}{2^n-1}$

where $t' = t + cn\sigma$ for some absolute constant c. □

Propositions 1 and 2 establish that $n = 128$ and $n = 64$ satisfy the requirement for unique representations. They also guarantee that there is no representative of 1 within $[1\,..\,2^{n/2}] \times \{0,1\}$. To see this, note that the propositions imply that $(0,0)$ is the only representative for 1 within a space $\mathbb{I}_1 \times \mathbb{I}_2$ that includes $[1\,..\,2^{n/2}] \times \{0,1\}$, and so there can be *no* representative of 1 within a subspace of $\mathbb{I}_1 \times \mathbb{I}_2$ that excludes $(0,0)$.

Blockcipher-based OCB1 is more efficient than OCB. With OCB one expects to use preprocessing to compute a value $L = E_K(0^n)$ and a collection of $2^i L$-values. This is gone in OCB1; preprocessing is not useful there beyond setting up the underlying blockcipher key. Beyond this, with OCB processing the j^{th} block involved xoring into the current offset a value $L(i) = 2^i L$ where $i = \mathsf{ntz}(j)$ was the number of trailing zero-bits in the index j. In the absence of preprocessing, offset-calculations were not constant time. This too is gone.

The previous paragraph notwithstanding, the time difference or chip-area difference between optimized implementations of OCB and OCB1 will be small, since the overhead of OCB over a mode like CBC was already small. The larger gain is that the mode is simpler to understand, implement, and prove correct.

11 The PMAC1 Message Authentication Code

As with OCB, one can recast PMAC [4] to use a tweakable blockcipher and, having done so, one can instantiate the tweakable blockcipher, this time with the XE construction. The resulting algorithm, PMAC1, is simpler and more efficient than PMAC. In the latter construction one had to xor into the current offset a value $L(i) = 2^i L$ where i was the number of trailing zero-bits in the current block index j. This is gone in PMAC1, and an implementation no longer needs to concern itself with Gray codes, precomputing $L(i)$-values, or finding the most efficient way to bring in the right $L(i)$ value. Details are in [14].

To make an authenticated encryption scheme that handles associated-data, combine OCB1 and PMAC1 [13]. Encrypt message M under key K, nonce N, and header H by $\mathrm{OCB1}.\mathsf{Encrypt}^N_K(M) \oplus \mathrm{PMAC1}_K(H)$ where the $\oplus$ xors into the end. Omit the $\oplus\ \mathrm{PMAC1}_K(H)$ if $H = \varepsilon$. We call this scheme AEM.

12 Comments

Under the approach suggested by this paper, to get good efficiency for a design that uses a tweakable-blockcipher, the designer must accept certain design rules. In particular, the tweak space needs to look like $\{0,1\}^n \times \mathbb{BIG} \times \mathbb{SMALL}$ for appropriate sets $\mathbb{BIG}$ and $\mathbb{SMALL}$, and one needs to arrange that most tweaks be obtained by incrementing the prior one. It is a thesis implicit in this work that these restrictions are not overly severe.

Besides simplifying the design and proof for OCB and PMAC, we have improved their efficiency. The improvement are not large (the modes were already highly efficient), but performance improvements, of any size, was not a benefit formerly envisaged as flowing from the tweakable-blockcipher abstraction.

Somewhat strangely, our constructions depend on the relative *easiness* of computing discrete logarithms. I know of no other example where one needs to compute discrete logs in order to design or verify a mode of operation.

I end this paper by acknowledging that everyone writes *block cipher*, not *blockcipher*. Still, the time has come to spell this word solid. I invite you to join me.

Acknowledgments

Thanks to David Wagner for pointing out an oversight in an early draft. Useful comments were also received from John Black and the anonymous referees.

This research was supported by NSF 0208842 and by a gift from Cisco System. Thanks to the NSF (particularly Carl Landwehr) and to Cisco (particularly David McGrew) for their kind support of my research.

References

1. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. *Symposium on Foundations of Computer Science, FOCS '97*, IEEE Computer Society, pp. 394–403, 1997.
2. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, vol. 61, no. 3, Dec 2000. Earlier version in *CRYPTO '94*.
3. M. Bellare, P. Rogaway, and D. Wagner. The EAX Mode of operation. *Fast Software Encryption, FSE 2004*. Lecture Notes in Computer Science, vol. 3017, Springer-Verlag, pp. 389–407, 2004.
4. J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology — Eurocrypt '02*. Lecture Notes in Computer Science, vol. 2332, Springer-Verlag, pp. 384–397, 2002.
5. V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption, FSE 2001*. Lecture Notes in Computer Science, vol. 2355, Springer-Verlag, pp. 92–108, 2001.

6. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, vol. 28, April 1984, pp. 270–299.
7. S. Halevi and P. Rogaway. A parallelizable enciphering mode. *Topics in Cryptology — CT-RSA 2004*. Lecture Notes in Computer Science, vol. 2964, Springer-Verlag, pp. 292–304, 2004.
8. J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *J. of Cryptology*, vol. 14, no. 1, pp. 17–35, 2001.
9. C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology — EUROCRYPT 2001*. Lecture Notes in Computer Science, vol. 2045, Springer-Verlag, pp. 529–544, 2001.
10. M. Liskov, R. Rivest, and D. Wagner. Tweakable block ciphers. *Advances in Cryptology — CRYPTO '02*. Lecture Notes in Computer Science, vol. 2442, Springer-Verlag, pp. 31–46, 2002.
11. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, vol 24, pp. 106–110, 1978.
12. J. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, vol. 32, pp. 918–924, 1978.
13. P. Rogaway. Authenticated-encryption with associated-data. *ACM Conference on Computer and Communications Security 2002, CCS 2002*. ACM Press, pp. 98–107, 2002.
14. P. Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. Manuscript, 2004. Full version of this paper, available from the author's web page.
15. P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security*, vol. 6, no. 3, pp. 365–403, 2003. Earlier version, with T. Krovetz, in *CCS 2001*.
16. R. Schroeppel. The hasty pudding cipher. AES candidate submitted to NIST, 1998.
17. D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). Network Working Group RFC 3610. The Internet Society, September 2003.

A Tweakable Blockciphers Implicit in Prior Work

When tweaks increase in sequence, the most efficient constructions formerly known for a tweakable blockcipher are those implicit in earlier modes [4, 5, 9, 15], recast in view of Liskov, Rivest, and Wagner [10]. In particular:

- Jutla [9] might be seen as suggesting a construction (among others) of $\widetilde{E}$: $(\mathcal{K} \times \mathcal{K}') \times (\{0,1\}^n \times \mathbb{Z}_p^+) \times \{0,1\}^n \to \{0,1\}^n$ by way of $\widetilde{E}^{N,i}_{KK'}(X) = E_K(X \oplus \Delta) \oplus \Delta$ where $\Delta = i\ell \bmod p$ and $\ell = E_{K'}(N)$ and p is the largest prime less than 2^n.
- Gligor and Donescu [5] might be seen as suggesting constructions like $\widetilde{E}$: $(\mathcal{K} \times \{0,1\}^n) \times [1\,..\,2^n - 1] \to \{0,1\}^n$ by $\widetilde{E}^i_{K,r}(X) = E_K(X + \delta)$ where $\delta = ir$ and addition is done modulo 2^n.

- Rogaway, Bellare, and Black [15] might be seen as implicitly suggesting making a tweakable blockcipher $\widetilde{E}\colon \mathcal{K} \times (\{0,1\}^n \times [0\,..\,2^{n-2}]) \times \{0,1\}^n \to \{0,1\}^n$ from an ordinary blockcipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ by way of $\widetilde{E}_K^{N,i}(X) = E_K(X \oplus \Delta) \oplus \Delta$ where $\Delta = \gamma_i L \oplus R$ and $L = E_K(0^n)$ and $R = E_K(N \oplus L)$ and γ_i is the i-th Gray-code coefficient.
- Black and Rogaway [4] might be seen as making $\widetilde{E}\colon \mathcal{K} \times [0\,..\,2^{n-2}] \times \{0,1\}^n \to \{0,1\}^n$ out of $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ by $\widetilde{E}_K^i(X) = E_K(X \oplus \Delta)$ where $\Delta = \gamma_i L$ and $L = E_K(0^n)$ and γ_i is as before.
- The last two definitions ignore the "special" treatment afforded to blocks modified by xoring in $2^{-1}L$. The implicit intent [4, 15] was to use this mechanism to enlarge the tweak space by one bit, effectively taking the cross product with $\{0,1\}$.

Eliminating Random Permutation Oracles in the Even-Mansour Cipher

Craig Gentry and Zulfikar Ramzan

DoCoMo Communications Laboratories USA, Inc.
{cgentry, ramzan}@docomolabs-usa.com

Abstract. Even and Mansour [EM97] proposed a block cipher construction that takes a publicly computable random permutation oracle P and XORs different keys prior to and after applying P: $C = k_2 \oplus P(M \oplus k_1)$. They did not, however, describe how one could instantiate such a permutation securely. It is a fundamental open problem whether their construction could be proved secure outside the random permutation oracle model. We resolve this question in the affirmative by showing that the construction can be proved secure in the random *function* oracle model. In particular, we show that the random permutation oracle in their scheme can be replaced by a construction that utilizes a four-round Feistel network (where each round function is a random function oracle publicly computable by all parties including the adversary). Further, we prove that the resulting cipher is super pseudorandom – the adversary's distinguishing advantage is at most $2q^2/2^n$ if he makes q total queries to the cipher, its inverse, as well as any random oracles. Even and Mansour, on the other hand, only showed security against inversion and forgery. *One noteworthy aspect of this result is that the cipher remains secure even though the adversary is permitted separate oracle access to all of the round functions.* One can achieve a two-fold and four-fold reduction respectively in the amount of key material by a closer inspection of the proof and by instantiating the scheme using group operations other than exclusive-OR. On the negative side, a straightforward adaption of an advanced slide attack recovers the $4n$-bit key with approximately $\sqrt{2} \cdot 2^n$ work using roughly $\sqrt{2} \cdot 2^n$ known plaintexts. Finally, if only three Feistel rounds are used, the resulting cipher is pseudorandom, but not super pseudorandom.

1 Introduction

The Even-Mansour Construction. Even and Mansour [EM97] proposed a block cipher construction based on XORing secret key material just prior to and just after applying a random permutation oracle P: $C = k_2 \oplus P(M \oplus k_1)$, where M is the plaintext, C is the ciphertext, and k_1, k_2 is the key material. The permutation P (as well as its inverse P^{-1}) is computable by all parties, including the adversary (see fig. 1). Even-Mansour proved that a polynomial-time adversary with black-box query access to the cipher and its inverse, as well

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 32–47, 2004.

as black-box query access to the internal permutation and its inverse cannot invert an un-queried ciphertext of his choice, except with negligible probability. They also proved an analogous result about computing the cipher's forward direction.

While there are practical limitations to their construction [Dae91, BW00], the Even-Mansour work is well known and theoretically interesting. In particular, it is an example of a cipher for which an adversary has black-box access to the only real "cryptographic" component; i.e., the random permutation oracle. The only secrets are simply XORed at the beginning and the end, and everything else is publicly accessible.

Fundamental Open Problems. The Even-Mansour work may be described within the framework of the random-oracle model [BR93] in which their cipher makes use of a random *permutation* oracle. Naturally, the need for such a permutation oracle is unpleasant, especially since Even and Mansour did not indicate how one might instantiate such a random permutation oracle while maintaining security. This motivates the following problem:

Open Problem 1: How can one go about instantiating the random permutation oracle in the Even-Mansour scheme?

Furthermore, Even and Mansour only proved security against inversions and forgeries. However, for block ciphers, the current bar is to prove super pseudorandomess [LR88]. That is, the cipher should be indistinguishable from a randomly chosen permutation on the same message space even if the adversary is granted black-box access to the forward and inverse directions of the cipher[1]. This motivates a second problem:

Open Problem 2: Can one prove that an Even-Mansour type construction yields a super pseudorandom permutation?

Our Contributions. We address the first question by demonstrating that the random permutation oracle can be replaced by a construction involving random *function* oracles; i.e., the underlying oracle (which must be accessible to all parties) does not have to be bijective, but we construct a permutation using it that is bijective. By supplanting the use of random permutation oracles by random function oracles, we have a result based on a less restrictive model. Our construction uses a Feistel ladder in which the random function oracle is used as a round function (see fig. 1). *However, what is different in this setting is that the adversary not only has access to the forward and reverse directions of the cipher, but also to each of the individual round functions.*

We address the second problem by proving that the construction is *super pseudorandom*. We remark the one can construe the Kilian-Rogaway analysis

[1] Their is also a notion of *pseudorandomness* for block ciphers wherein the adversary must distinguish it from a random permutation given black-box access to only the forward direction of the cipher.

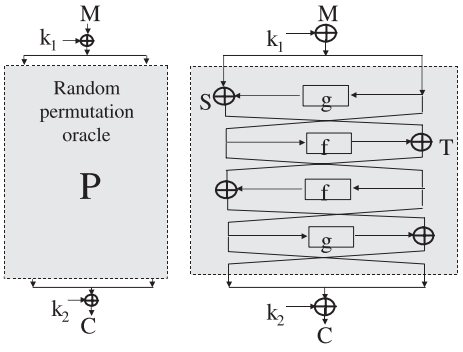


Fig. 1. The diagram on the left depicts the Even-Mansour scheme where P is a random *permutation* oracle; i.e., the adversary has black-box access to P and P^{-1}. The diagram on the right depicts our scheme in which the permutation oracle is instantiated by a Feistel network consisting of *publicly-accessible* random *function* oracles f, g

of DESX [KR96] as a proof that Even-Mansour is pseudorandom. Recall that in DESX, the Even-Mansour random permutation is supplanted with a *keyed* block cipher, such as DES. The Kilian-Rogaway proof allowed the adversary oracle access to the internal permutation P (modeled as an ideal block cipher) as well as P^{-1}, to simulate that an adversary had correctly guessed the key – this maneuver isolates the benefits of the pre- and post-whitening keys. However, in their published proof the adversary was not given access to the inverse of the block cipher – so *super* pseudorandomness *was not* proved[2].

In addition, Ramzan-Reyzin [RR00] noted that one could use their round security framework to prove that Even-Mansour is super pseudorandom, but their focus was different, so no proof was supplied. Also their comment was limited to the original Even-Mansour construction (which used the random permutation oracle). Therefore, we consider addressing the first fundamental open problem as our main technical contribution; a side benefit of our work is a proof of super-pseudorandomness for Even-Mansour style block ciphers.

Our results help us better understand block cipher design. First, they point to the benefit of pre- and post- whitening. In particular, our construction shows that, in the random function oracle model, one can construct a super pseudorandom block cipher in which the all key material is only incorporated during the pre- and post-whitening phases and in a very simple way. This is despite the fact that the adversary has access to the internals of the cipher. Second, our constructions show that it may be possible to obtain a middle ground between

[2] Kilian and Rogaway mentioned that one could extend their proof to address chosen ciphertext queries, however, they did not provide the proof, nor did they state a formal security theorem where such access is given.

pure black-box analysis and one in which an adversary has some meaningful knowledge about the internal design of the black box. This can be thought of as a "gray-box" analysis. We also remark that the random permutation oracle model seems less appealing than the random function oracle model. Instantiating a random function oracle while maintaining security seems more plausible since such functions could be made sufficiently complex that their behavior is ill understood. On the other hand, when instantiating a random permutation oracle with an actual permutation, one is limited in the complexity of the design since the function must remain bijective and efficient to invert. Our results give hope that one may be able to base future cryptosystems on random permutation oracles and replace them with constructions based on random function oracles in a provably secure way. Finally, our work helps bridge the gap between the theory and practice of Feistel ciphers. In particular, the theoretical work on Feistel ciphers (e.g., [LR88]) considers round functions that are strong (e.g., pseudorandom) and potentially complex keying mechanisms (e.g., the functions themselves are keyed). This departs from practice in two ways. First, round functions in practice are weak. Second, block cipher round keys are introduced in some simple way, for example by XORing them prior to applying an un-keyed function (c.f., DES [FIPS46]). Our work sits somewhere in between since it considers complex round functions (random oracles), but simple keying procedures (XORing). Therefore, we can view our work as providing better mathematical insight into the security of DES-like ciphers.

Other Results. Our proof of security holds even if the amount of key material is reduced twofold. Also, if we permit group operations other than XOR, we can recycle keying material, yielding a fourfold reduction; interestingly, if XOR is used with recycled keying material, the cipher behaves like an involution and is trivially distinguishable from a random permutation. This idea of considering different group operations has previously been applied to Luby-Rackoff ciphers [PRS02]. On the negative side, a "sliding with a twist" attack [BW00] allows an adversary to recover the key using $\sqrt{2} \cdot 2^n$ *known* plaintexts and $\sqrt{2} \cdot 2^n$ work. Finally, if we instantiate the permutation with three Feistel rounds, the construction is pseudorandom, but is not super pseudorandom. The attack adapts the standard distinguisher for three-round Luby-Rackoff ciphers [LR88]. Due to space constraints, as well as the fact that these results follow easily from existing techniques, we omit a further discussion. For details, see the full version of the paper [GR04].

Caveat(s) Emptor. While the random-oracle model is an extremely useful cryptographic tool, there are instances of schemes that are secure in the random oracle model, but are insecure for any instantiation of the random oracle by a polynomial-time computable function [CGH98, GK03, BBP04]. We further note that the lower bounds we present indicate that n should be chosen so that $2^{n/2}$ is sufficiently large to thwart distinguishing attacks. We also remark that Even

and Mansour gave a $O(2^{-n})$ upper bound on the adversary's success probability, whereas our bound resembles $O(2^{-n/2})$. However, Even and Mansour only proved security against inversions and forgeries whereas we show super pseudorandomness. Moreover, we eliminate the random permutation oracle requirement and also give the adversary access to the innards of the cipher. Therefore, we expect there to be a sizeable gap in the respective security guarantees. In light of these caveats, *we stress that our main contribution is in resolving fundamental issues from the Even-Mansour work and gaining more theoretical insight into block cipher design; we do not recommend this as a practical approach to building a block cipher.* In fact, efficient random oracle model based block ciphers are desired, then Ramzan and Reyzin have a four-round Feistel block cipher construction in which the middle two rounds use a random oracle, and the outer two rounds involve universal hash functions [RR00].

Organization. Section 2 reviews prior definitions and constructions. Section 3 discusses our main construction and security proof. Finally, we make concluding remarks in Section 4.

2 Prior Definitions and Constructions

We describe definitions and prior constructions that are germane to our work. We avoid asymptotic analysis in favor of the "concrete" (or "exact") security model as laid out by Bellare, Kilian, and Rogaway [BKR94], and Bellare, Canetti, Krawczyk [BCK96]. However, our results can be adapted to either model.

Notation. For a natural number n, we let I_n denote the set of bit strings of length n: $\{0,1\}^n$. For a bit string x, we let $|x|$ denote its length. If $|x|$ is even, then x^L and x^R denote the left and right halves of the bits respectively; we sometimes write $x = (x^L, x^R)$. If x and y are two bit strings with $|x| = |y|$, we denote by $x \oplus y$ their bitwise exclusive OR. If S is a probability space, then $x \xleftarrow{R} S$ denotes the process of picking an element from S according to the underlying probability distribution. Unless otherwise specified, the underlying distribution is assumed to be uniform. By a finite function (or permutation) family $\mathcal{F}$, we denote a set of functions with common domain and common range. Let $\mathsf{Rand}^{k \to \ell}$ be the set of all functions going from I_k to I_ℓ, and let Perm^m be the set of all permutations on I_m. We call a finite function (or permutation) family *keyed* if every function in it can be specified (not necessarily uniquely) by a key a. We denote the function given by a as f_a. We assume that given a, it is possible to efficiently evaluate f_a at any point (as well as f_a^{-1} in case of a keyed permutation family). For a given keyed function family, a key can be any string from I_s, where s is known as "key length." (Sometimes it is convenient to have keys from a set other than I_s; we do not consider such function families simply for clarity of exposition, but our results continue to apply in such cases.) For functions f and g, $g \circ f$ denotes the function $x \mapsto g(f(x))$.

Model of Computation. We model the adversary $\mathcal{A}$ as a program for a Random Access Machine (RAM) with black-box access to some number k of oracles, each computing some specified function. If $(f_1, \ldots, f_k)$ is a k-tuple of functions, then $\mathcal{A}^{f_1,\ldots,f_k}$ denotes a k-oracle adversary who is given black-box oracle access to each of the functions $f_1, \ldots, f_k$. We define $\mathcal{A}$'s "running time" to be the number of time steps it takes plus the length of its description (to prevent one from embedding arbitrarily large lookup tables in $\mathcal{A}$'s description).

Pseudorandom Functions and Block Ciphers. The pseudorandomness of a keyed function family $\mathcal{F}$ with domain I_k and range I_ℓ captures its computational indistinguishability from $\mathsf{Rand}^{k\to\ell}$. The following definition is adapted from [GGM84]:

Definition 1. *A pseudorandom function family $\mathcal{F}$ is a keyed function family with domain I_k, range I_ℓ, and key length s. Let $\mathcal{A}$ be a 1-oracle adversary. Then we define $\mathcal{A}$'s advantage as*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A}) \triangleq \left|\Pr[a \xleftarrow{R} I_s : \mathcal{A}^{f_a} = 1] - \Pr[f \xleftarrow{R} \mathsf{Rand}^{k\to\ell} : \mathcal{A}^{f} = 1]\right|.$$

For any integers $q, t \geq 0$, we define $\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(q,t) \triangleq \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(\mathcal{A})\}$, as an insecurity function, where the maximum is taken over choices of adversary $\mathcal{A}$ such that:

- *$\mathcal{A}$ makes at most q oracle queries, and*
- *the running time of $\mathcal{A}$, plus the time necessary to select a $\xleftarrow{R} I_s$ and answer $\mathcal{A}$'s queries, is at most t.*

Recall that the Even-Mansour cipher [EM97] operates on a $2n$-bit string x as follows $E(x) = k_2 \oplus P(x \oplus k_1)$ where $k_1, k_2 \in I_{2n}$ constitutes the keying material and P is a random *permutation* oracle. Here P and P^{-1} are publicly computable (in a black-box fashion) by all parties. Even and Mansour proved that E is hard to invert on a point C_0 of the adversary's choice even if the adversary has oracle access to E, E^{-1}, P, P^{-1} subject to the restriction that the adversary cannot query the E^{-1} oracle on the point C_0; i.e., it is hard to find M_0 such that $M_0 = E^{-1}_{k_1,k_2}(C_0)$. Similarly, they showed that the adversary cannot compute the ciphertext corresponding to a message point M_0 of its choice with access to these same oracles, but this time subject to the restriction that the adversary cannot query the E oracle on point M_0; i.e., it is hard to find C_0 such that $C_0 = E_{k_1,k_2}(M_0)$. While these results capture some of the security requirements needed for a block cipher, there are stronger notions of security for a block cipher. One such notion, proposed by Luby and Rackoff [LR88], is called *super pseudorandomness*. The notion captures the pseudorandomness of a permutation family on I_ℓ in terms of its indistinguishability from Perm^ℓ, where the adversary is given access to both directions of the permutation. In other words, it measures security of a block cipher against chosen plaintext and ciphertext attacks. We now describe such notions and how to achieve them.

Definition 2. *A block cipher $\mathcal{P}$ is a keyed permutation family with domain and range I_ℓ and key length s. Let $\mathcal{A}$ be a 2-oracle adversary. Then we define $\mathcal{A}$'s advantage as*

$$\mathsf{Adv}^{\mathsf{sprp}}_{\mathcal{P}}(\mathcal{A}) \triangleq \left|\Pr[a \xleftarrow{R} I_s : \mathcal{A}^{p_a, p_a^{-1}} = 1] - \Pr[p \xleftarrow{R} \mathsf{Perm}^\ell : \mathcal{A}^{p, p^{-1}} = 1]\right|.$$

For any integers $q, t \geq 0$, $\mathsf{Adv}^{\mathsf{sprp}}_F(q,t)$ specifies the insecurity function (analogous to Definition 1).

Luby and Rackoff showed how to construct a secure block cipher using Feistel permutations.

Definition 3 (Basic Feistel Permutation). *Let $\mathcal{F}$ be a function family with domain and range I_n. Let $f \in \mathcal{F}$. Let $x = (x^L, x^R)$ with $x^L, x^R \in I_n$. We denote by $\overline{f}$ the permutation on I_{2n} defined as $\overline{f}(x) = (x^R, x^L \oplus f(x^R))$. Note that it is a permutation because $\overline{f^{-1}}(y) = (y^R \oplus f(y^L), y^L)$. Similarly, let $\overline{\mathcal{F}} = \{\overline{f} \mid f \in \mathcal{F}\}$.*

Definition 4 (Feistel Network). *If $f_1, \ldots, f_s$ are mappings with domain and range I_n, then we denote by $\Phi(f_1, \ldots, f_s)$ the permutation on I_{2n} defined as $\Phi(f_1, \ldots, f_s) = \overline{f_s} \circ \ldots \circ \overline{f_1}$.*

Theorem 1 (Luby-Rackoff). *Let h_1, f_1, f_2, h_2 be independently-keyed functions from a keyed function family $\mathcal{F}$ with domain and range I_n and key space I_s. Let $\mathcal{P}$ be the family of permutations on I_{2n} with key space I_{4s} defined by $\mathcal{P} = \Phi(h_1, f_1, f_2, h_2)$ (the key for an element of $\mathcal{P}$ is simply the concatenation of keys for h_1, f_1, f_2, h_2). Then, $\mathsf{Adv}^{\mathsf{sprp}}_{\mathcal{P}}(q,t) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(q,t) + \binom{q}{2}\left(2^{-n+1} + 2^{-2n+1}\right)$.*

The Luby-Rackoff result proved security when the adversary has access to the permutation and its inverse. In our case, we will show security of the Even-Mansour cipher when the adversary has black-box access to the cipher, its inverse, and to each of the internal round functions.

Having presented the classical definitions of block ciphers and Feistel networks, we now describe notions of the Ramzan-Reyzin round security framework [RR00] which we make use of in the present work.

Definition 5 (Round Decomposition [RR00]). *Let $\mathcal{P}, \mathcal{F}^1, \mathcal{F}^2, \ldots, \mathcal{F}^r$ be keyed permutation families, each with domain and range I_ℓ and key length s, such that for any key $a \in I_s$, $p_a = f^r_a \circ \cdots \circ f^1_a$. Then $\mathcal{F}^1, \ldots, \mathcal{F}^r$ is called an* r-round decomposition *for $\mathcal{P}$. For $i \leq j$, denote by $(i \to j)_a$ the permutation $f^j_a \circ \ldots \circ f^i_a$, and by $(i \leftarrow j)_a$ the permutation $\left(f^j_a \circ \ldots \circ f^i_a\right)^{-1}$. Denote by $i \to j$ and $i \leftarrow j$ the corresponding keyed function families.*

Note that having oracle access to a member of $i \to j$ means being able to give inputs to round i of the forward direction of a block cipher and view outputs after round j. Likewise, having oracle access to $i \leftarrow j$ corresponds to being able to give inputs to round j of the *reverse* direction of the block cipher and view outputs after round i. Thus, the oracle for $1 \to r = \mathcal{P}$ corresponds to the oracle

for a chosen plaintext attack, and the oracle for $1 \leftarrow r = \mathcal{P}^{-1}$ corresponds to the oracle for a chosen ciphertext attack.

We now give a formal security definition of a block cipher when an adversary has access to internal rounds. Note that the adversary is allowed oracle access to some subset K of the set $\{i \rightarrow j, i \leftarrow j : 1 \leq i \leq j \leq r\}$, and the insecurity function additionally depends on K.

Definition 6 (Round Security [RR00]). *Let $\mathcal{P}$ be a block cipher with domain and range I_ℓ, key length s, and some r-round decomposition $\mathcal{F}^1, \ldots, \mathcal{F}^r$. Fix some subset $K = \{\pi^1, \ldots, \pi^k\}$ of the set $\{i \rightarrow j, i \leftarrow j : 1 \leq i \leq j \leq r\}$, and let $\mathcal{A}$ be a $k+2$-oracle adversary. Then we define $\mathcal{A}$'s advantage as*

$$\mathsf{Adv}^{\mathsf{sprp}}_{\mathcal{P},\mathcal{F}^1,\ldots,\mathcal{F}^r,K}(\mathcal{A}) = \left|\Pr[a \xleftarrow{R} I_s : \mathcal{A}^{p_a, p_a^{-1}, \pi_a^1, \ldots, \pi_a^k} = 1] - \Pr[p \xleftarrow{R} \mathsf{Perm}^\ell, a \xleftarrow{R} I_s : \mathcal{A}^{p, p^{-1}, \pi_a^1, \ldots, \pi_a^k} = 1]\right|$$

For any integers $q, t \geq 0$ and set K, $\mathsf{Adv}^{\mathsf{sprp}}_{\mathcal{P},\mathcal{F}^1,\ldots,\mathcal{F}^r,K}(q,t)$ specifies our insecurity function (analogous to Definition 2).

Ramzan and Reyzin [RR00] were the first to consider what happens when internal round functions of a Feistel network are available to an external adversary.

Theorem 2 (Ramzan-Reyzin). *Let f_1, f_2, f_3, f_4 be independently-keyed functions from a keyed function family $\mathcal{F}$ with domain and range I_n and key space I_s. Let $\mathcal{P}$ be the family of permutations on I_{2n} with key space I_{4s} defined by $\mathcal{P} = \Phi(f_1, f_2, f_3, f_4)$ with the natural 4-round decomposition $\overline{\mathcal{F}}, \overline{\mathcal{F}}, \overline{\mathcal{F}}, \overline{\mathcal{F}}$. Let $K = \{i \rightarrow j, i \leftarrow j : 2 \leq i \leq j \leq 3\}$. Then*

$$\mathsf{Adv}^{\mathsf{sprp}}_{\mathcal{P},\overline{\mathcal{F}},\overline{\mathcal{F}},\overline{\mathcal{F}},\overline{\mathcal{F}},K}(q,t) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(q,t) + \binom{q}{2}\left(2^{-n+1} + 2^{-2n+1}\right) + q^2\left(2^{-n-1}\right).$$

Ramzan-Reyzin consider the case where all parties have black-box access to the internal permutations $\overline{f_2}, \overline{f_3}$. They noted that if the underlying round functions f_1, and f_2 are chosen from $\mathsf{Rand}^{n \rightarrow n}$, then one could translate their results to the random oracle model wherein f_2, f_3 are modeled as random function oracles that are accessible to all parties, including the adversary.

3 Our Main Result

We now prove our main result. We use the Ramzan-Reyzin round-security framework [RR00] to analyze our construction and leverage their techniques to obtain the desired result. However, for technical reasons, the proof must also incorporate an additional hybrid distribution into the argument. Now, let $\Psi^{f,g}_{k_1,k_2}$ denote the Even-Mansour construction when the internal permutation is replaced by a four-round Feistel network with outer round function g and inner round function f: $\Psi^{f,g}_{k_1,k_2}(x) = k_2 \oplus \Phi(g, f, f, g)(x \oplus k_1)$. Here $k_1, k_2 \in I_{2n}$ are the keys and

f, g are modeled as random function oracles; i.e., they are publicly accessible to all parties (including the adversary) and behave like random functions. Observe then that the adversary can compute not only the Even-Mansour permutation, but also knows its internal structure and has black-box access to the functions f and g around which it is designed. We view this construction as consisting of the composition of six round permutations:

- $\pi_1^{k_1}(x) = x \oplus k_1$
- $\pi_3, \pi_4 = \overline{f}$. Recall that $\overline{f}$ denotes a permutation on I_{2n} defined as $\overline{f}(x) = (x^R, x^L \oplus f(x^R))$.
- $\pi_2, \pi_5 = \overline{g}$.
- $\pi_6^{k_2}(x) = x \oplus k_2$.

Observe that $\Psi_{k_1,k_2}^{f,g}(M) = \pi_6^{k_2} \circ \pi_5 \circ \cdots \circ \pi_2 \circ \pi_1^{k_1}$. We now state our main result in the following theorem:

Theorem 3 (Main Result). *Suppose $K \subseteq \{i \to j, i \leftarrow j \mid 2 \leq i \leq j \leq 5\}$. Let f be modeled as a random oracle, and let k_1 and k_2 be picked randomly and independently from I_{2n}. Let $\Psi_{k_1,k_2}^{f,g}(x) = k_2 \oplus \Phi(g, f, f, g)(x \oplus k_1)$, and R be a random permutation on I_{2n}. Then*

$$\mathsf{Adv}^{\mathsf{sprp}}_{\mathcal{P},\pi_1,\pi_2,\ldots,\pi_6}(q, t, K) \leq \left(2q^2 - q\right) \cdot 2^{-n} + \binom{q}{2} \cdot 2^{-2n+1}.$$

Observe that we do not consider any terms of the form $\mathsf{Adv}^{\mathsf{prf}}_{\mathcal{F}}(q, t)$ since we assume that the underlying round functions are modeled as random oracles in which case such terms will evaluate to 0.

Recasting the problem in the round-security framework allows us to apply the techniques of Ramzan and Reyzin [RR00] (who generalized the techniques of Naor and Reingold [NR99] to deal with the extra queries from an oracle with internal access). We note that access to the oracles of K is equivalent to access to the oracles for f and g[3]. Now, consider the following theorem.

Theorem 4. *Let f and g be modeled as random oracles, and let k_1 and k_2 be picked randomly and independently from I_{2n}. Let $\Psi_{k_1,k_2}^{f,g}(x) = k_2 \oplus \Phi(g, f, f, g)(x \oplus k_1)$, and let R be a random element of Perm^{2n}. Then, for any 4-oracle adversary $\mathcal{A}$ (we do not restrict the running time of $\mathcal{A}$) that makes at most q_c queries to its first two oracles (either Ψ, Ψ^{-1} or R, R^{-1}) and at most q_{of} and q_{og} queries to its second two oracles (f and g) respectively, it follows that:*

$$\left|\Pr[\mathcal{A}^{\Psi,\Psi^{-1},f,g} = 1] - \Pr[A^{R,R^{-1},f,g} = 1]\right| \leq (q_c^2 + 2q_{of}q_c + 2q_{og}q_c + q_c^2 - q_c)2^{-n} + \binom{q_c}{2}\left(2 \cdot 2^{-n} + 2^{-2n+1}\right).$$

[3] We remark, however, that one query to an oracle in K may need to be simulated by multiple queries to f, g. Therefore, the total number of queries made to f and g is an upper bound on the number of queries that would need to be made to an oracle in K.

Observing that the total number of queries $q = q_c + q_{of} + q_{og}$, it is straightforward to see that

$$(q_c^2 + 2q_{of}q_c + 2q_{og}q_c + q_c^2 - q_c) \leq 2q^2 - q.$$

Therefore, we see that theorem 4 implies theorem 3. In the sequel, we describe the proof of theorem 4. The first part of the proof focuses on the adversary's transcript (i.e., his "view") and shows that each possible transcript is about as likely to occur when $\mathcal{A}$ is given Ψ, f, g as when $\mathcal{A}$ is given R, f, g. This part of the proof also relies on a hybrid distribution $\tilde{\Psi}$ to facilitate the proof. The second part uses a standard probability argument to show that if the distributions on transcripts are similar, then $\mathcal{A}$ will have a small advantage in distinguishing Ψ from R.

Proof of Theorem 4. To start with, let P denote the permutation oracle (either Ψ or R) that $\mathcal{A}$ accesses. From now on, for notational convenience we ignore the superscripts f, g and the subscripts k_1, k_2 associated with Ψ. Let $\mathcal{O}^f$ and $\mathcal{O}^g$ denote the oracles that compute the functions f and g (note that when $\mathcal{A}$ gets Ψ as its permutation oracle, f and g are actually used as the round function in the computation of the oracle $P = \Psi$; when $\mathcal{A}$ gets R as its permutation oracle, f and g are independent of $P = R$). The machine $\mathcal{A}$ makes two types of queries to the oracle P: $(+, x)$ which asks to obtain the value of $P(x)$, or $(-, y)$ which asks to obtain the value of $P^{-1}(y)$ – where both x and y are in I_{2n}. We call these *cipher queries.* We define the query-answer pair for the i^{th} cipher query as $\langle x_i, y_i \rangle \in I_{2n} \times I_{2n}$ if $\mathcal{A}$'s query was $(+, x_i)$ and y_i is the answer it received from P or its query was $(-, y_i)$ and x_i is the answer it received. We assume that $\mathcal{A}$ makes exactly q_c cipher queries and we call the sequence $\{\langle x_1, y_1 \rangle, \ldots, \langle x_{q_c}, y_{q_c} \rangle\}_P$ the cipher-transcript of $\mathcal{A}$. In addition, $\mathcal{A}$ can make queries to $\mathcal{O}^f$ and $\mathcal{O}^g$. We call these *oracle queries.* We denote these queries as: $(\mathcal{O}^f, x')$ (*resp.* $(\mathcal{O}^g, x')$) which asks to obtain $f(x')$ (*resp.* $g(x')$). We define the query-answer pair for the i^{th} oracle query as $\langle x'_i, y'_i \rangle \in I_n \times I_n$ if $\mathcal{A}$'s query was $(\mathcal{O}^f, x')$ and the answer it received was y' and as $\langle x''_i, y''_i \rangle \in I_n \times I_n$ if $\mathcal{A}$'s query was $(\mathcal{O}^g, x'')$ and the answer it received was y''. We assume that $\mathcal{A}$ makes q_{of} and q_{og} queries to $\mathcal{O}^f$ and $\mathcal{O}^g$ respectively. We call the sequence $\{\langle x'_1, y'_1 \rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}} \rangle\}_{\mathcal{O}^f}$ the f-oracle-transcript of $\mathcal{A}$ and $\{\langle x''_1, y''_1 \rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}} \rangle\}_{\mathcal{O}^g}$ the g-oracle-transcript of $\mathcal{A}$. Note that since $\mathcal{A}$ is computationally unbounded, we can make the standard assumption that $\mathcal{A}$ is a deterministic machine. Under this assumption, the exact next query made by $\mathcal{A}$ can be determined by the previous queries and the answers received. We formalize this as follows:

Definition 7. *We denote the $i + j + k + 1^{st}$ query $\mathcal{A}$ makes as a function of the first $i + j + k$ query-answer pairs in $\mathcal{A}$'s cipher and oracle transcripts (where either $i < q_c$ or $j < q_{of}$ or $k < q_{og}$) by:*

$$C_{\mathcal{A}}[\{\langle x_1, y_1 \rangle, \ldots, \langle x_i, y_i \rangle\}_P, \{\langle x'_1, y'_1 \rangle, \ldots, \langle x'_j, y'_j \rangle\}_{\mathcal{O}^f}, \{\langle x''_1, y''_1 \rangle, \ldots, \langle x''_k, y''_k \rangle\}_{\mathcal{O}^g}].$$

For the case that all queries have been made (i.e., $i = q_c, j = q_{of}, k = q_{og}$), we define the above expression to denote $\mathcal{A}$'s output as a function of its cipher and oracle transcripts.

Definition 8. *Let $\sigma = (T_P, T_f, T_g)$ be a three tuple comprising the sequences $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$, $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$, and $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$, and where for $1 \leq i \leq q_c$ we have that $\langle x_i, y_i\rangle \in I_{2n} \times I_{2n}$, for $1 \leq j \leq q_{of}$, we have that $\langle x'_j, y'_j\rangle \in I_n \times I_n$, and for $1 \leq k \leq q_{og}$, we have that $\langle x''_k, y''_k\rangle \in I_n \times I_n$. Then, σ is a possible $\mathcal{A}$-transcript if for every $1 \leq i \leq q_c$, for every $1 \leq j \leq q_{of}$ and for every $1 \leq k \leq q_{og}$,*

$$C_{\mathcal{A}}[\{\langle x_1, y_1\rangle, \ldots, \langle x_i, y_i\rangle\}_P, \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_j, y'_j\rangle\}_{\mathcal{O}^f} \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_k, y''_k\rangle\}_{\mathcal{O}^g}] \in \{(+, x_{i+1}), (-, y_{i+1}), (\mathcal{O}^f, x'_{j+1}), (\mathcal{O}^g, x''_{k+1})\}.$$

We now consider two useful processes for answering $\mathcal{A}$'s *cipher* queries.

Definition 9. *Let $\tilde{\Psi}$ denote the process in which the cipher queries and f-oracle queries are answered as they would be for Ψ, however the g-oracle queries are answered by another independent random function oracle h.*

Definition 10. *Let $\tilde{R}$ denote the process that answers all oracle queries as Ψ would, but answers the i^{th} cipher query of $\mathcal{A}$ as follows:*

1. *If $\mathcal{A}$'s query is $(+, x_i)$ and for some $1 \leq j < i$ the j^{th} query-answer pair is $\langle x_i, y_i\rangle$, then $\tilde{R}$ answers y_i.*
2. *If $\mathcal{A}$'s query is $(-, y_i)$ and for some $1 \leq j < i$ the j^{th} query-answer pair is $\langle x_i, y_i\rangle$, then $\tilde{R}$ answers x_i.*
3. *If neither of the above happens, then $\tilde{R}$ answers with a uniformly chosen element in I_{2n}.*

We formalize the fact that $\tilde{R}$'s answers may not be consistent with any function, let alone any permutation.

Definition 11. *Let $\sigma' = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$ be any possible $\mathcal{A}$-cipher transcript. We say that σ' is inconsistent if for some $1 \leq j < i \leq q_c$ the corresponding query-answer pairs satisfy $x_i = x_j$ but $y_i \neq y_j$, or $x_i \neq x_j$ but $y_i = y_j$. Likewise, we call any $\mathcal{A}$-transcript σ that contains σ' inconsistent.*

Note 1. If $\sigma = (T_P, T_f, T_g)$, with sub-transcripts $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$, $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$, and $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$, is a possible $\mathcal{A}$-transcript, we assume from now on that if σ is consistent and if $i \neq j$ then $x_i \neq x_j$, $y_i \neq y_j$, $x'_i \neq x'_j$, and $x''_i \neq x''_j$. This formalizes the concept that $\mathcal{A}$ never repeats a query if it can determine the answer from a previous query-answer pair.

Fortunately, the process $\tilde{R}$ often behaves like a permutation. It turns out that if $\mathcal{A}$ is given oracle access to either $\tilde{R}$ or R to answer its cipher queries, it will have a negligible advantage in distinguishing between the two. Proposition 1 states this more formally. Before doing so, we first consider the distributions on

the various transcripts seen by $\mathcal{A}$ as a function of the different distributions on answers it can get.

Definition 12. *The discrete random variables $T_\Psi, T_{\tilde{\Psi}}, T_R, T_{\tilde{R}}$ denote the cipher and oracle transcripts seen by $\mathcal{A}$ when its cipher queries are answered by Ψ, $\tilde{\Psi}$, R, $\tilde{R}$ respectively, and its oracle queries are answered by $\mathcal{O}^f$ or $\mathcal{O}^g$.*

Remark 1. Observe that according to our definitions and assumptions, $\mathcal{A}^{\Psi,\Psi^{-1},f,g}$ and $C_{\mathcal{A}}(T_\Psi)$ denote the same random variable. The same is true for the other discrete random variables.

Proposition 1. $|\Pr_{\tilde{R}}[C_{\mathcal{A}}(T_{\tilde{R}}) = 1] - \Pr_R[C_{\mathcal{A}}(T_R) = 1]| \leq \binom{q_c}{2} \cdot 2^{-2n}$.

The proof of this proposition has appeared in numerous places [NR99, RR00]. The idea is to observer that T_R, $T_{\tilde{R}}$ have the same distribution conditioned on $T_{\tilde{R}}$ being consistent. One can then bound the probability that $T_{\tilde{R}}$ is inconsistent by $\binom{q_c}{2} \cdot 2^{-2n}$. The proof can be completed by a standard probability argument. We omit the details, though they are available in the full version [GR04]. We now proceed to obtain a bound on the advantage that $\mathcal{A}$ will have in distinguishing between T_Ψ and $T_{\tilde{R}}$. We first show that T_Ψ and $T_{\tilde{\Psi}}$ are identically distributed, unless the input to g in a cipher query related to Ψ matches the input to g in an oracle query related to Ψ. We can compute the likelihood of such an event as a function of only k_1 and k_2 – we term this event BadG and define it next; we then compute the probability that it occurs.

Definition 13. *For every specific pair of keys $k_1, k_2 \in I_{2n}$, we define $\mathsf{BadG}(k_1, k_2)$ to be the set of all possible and consistent transcripts $\sigma = (T_P, T_f, T_g)$, with sub-transcripts $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$, $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$, and $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$ satisfying at least one of the following events:*

- **BG1:** *there exists $1 \leq i \leq q_c$, $1 \leq j \leq q_{og}$ such that $x_i^R \oplus k_1^R = x''_j$, or*
- **BG2:** *there exists $1 \leq i \leq q_c$, $1 \leq j \leq q_{og}$ such that $y_i^L \oplus k_2^L = x''_j$.*

Proposition 2. *Let k_1, k_2 be randomly and independently chosen from I_{2n}. For any possible and consistent $\mathcal{A}-$transcript $\sigma = (T_P, T_f, T_g)$, with sub-transcripts $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$, $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$, and $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$, we have that*

$$\Pr_{k_1,k_2}[\sigma \in \mathsf{BadG}(k_1, k_2)] \leq 2q_{og}q_c \cdot 2^{-n}.$$

Proof. (Sketch) A transcript σ is in $\mathsf{BadG}(k_1, k_2)$ if one of **BG1** or **BG2** occur. We obtain an upper bound on the probabilities of each of these events separately by using the fact that k_1, k_2 are chosen uniformly at random from I_{2n}. Applying the union bound to sum the individual probabilities yields the desired result.

We now show that T_Ψ and $T_{\tilde{\Psi}}$ are identically distributed if neither **BG1** nor **BG2** occur.

Lemma 1. *Let* $\sigma = (T_P, T_f, T_g)$*, where* $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$*,* $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$*, and* $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$*, be any possible and consistent* $\mathcal{A}-$*transcript, then*

$$\Pr_{\Psi}[T_\Psi = \sigma | \sigma \notin \mathsf{BadG}(k_1, k_2)] = \Pr_{\tilde{\Psi}}[T_{\tilde{\Psi}} = \sigma].$$

Proof. (Sketch) Observe that $x_i^R \oplus k_1^R \not= x''_j$ and $y_i^L \oplus k_2^L \not= x''_j$ for all i, j whenever $\sigma \notin \mathsf{BadG}(k_1, k_2)$. In such a case, the inputs to g during the cipher queries are distinct from the inputs to g during the g-oracle queries. Since there is no overlap in the two sets of queries, since g is modeled as a random oracle, and since the events depend only on the choice of k_1 and k_2 (which are chosen independently of g), the distribution is identical to one in which g is replaced by another independently chosen random oracle h.

We now focus on $T_{\tilde{\Psi}}$. It turns out that $T_{\tilde{\Psi}}$ and $T_{\tilde{R}}$ are identically distributed unless the same value is input to the inner random oracle f on different occasions (we show this in Lemma 2). We can compute the likelihood of this event as a function of *only* k_1, k_2, and g. We call this event "Bad" (in the next definition) and obtain a bound on the probability that it actually occurs (in Proposition 3).

Definition 14. *For every specific pair of keys* $k_1, k_2 \in I_{2n}$ *and oracle* $g \in \mathsf{Rand}^{n\to n}$*, define* $\mathsf{Bad}(k_1, k_2, g)$ *to be the set of all possible and consistent transcripts* $\sigma = (T_P, T_f, T_g)$*, with sub-transcripts* $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$*,* $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$*, and* $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$*, satisfying at least one of the following events:*

- **B1:** $\exists\ 1 \le i < j \le q_c$ *such that* $g(x_i^R \oplus k_1^R) \oplus x_i^L = g(x_j^R \oplus k_1^R) \oplus x_j^L$
- **B2:** $\exists\ 1 \le i < j \le q_c$ *such that* $y_i^R \oplus g(y_i^L \oplus k_2^L) = y_j^R \oplus g(y_j^L \oplus k_2^L)$
- **B3:** $\exists\ 1 \le i, j \le q_c$ *such that* $g(x_i^R \oplus k_1^R) \oplus x_i^L \oplus k_1^L = k_2^R \oplus y_j^R \oplus g(y_j^L \oplus k_2^L)$
- **B4:** $\exists\ 1 \le i \le q_c,\ 1 \le j \le q_{of}$ *such that* $g(x_i^R \oplus k_1^R) \oplus x_i^L \oplus k_1^L = x'_j$
- **B5:** $\exists\ 1 \le i \le q_c,\ 1 \le j \le q_{of}$ *such that* $k_2^R \oplus y_i^R \oplus g(y_i^L \oplus k_2^L) = x'_j$.

Proposition 3. *Let* k_1, k_2 *be randomly and independently chosen from* I_{2n}*. Then, for any possible and consistent* $\mathcal{A}-$*transcript* $\sigma = (T_P, T_f, T_g)$*, with sub-transcripts* $T_P = \{\langle x_1, y_1\rangle, \ldots, \langle x_{q_c}, y_{q_c}\rangle\}_P$*,* $T_f = \{\langle x'_1, y'_1\rangle, \ldots, \langle x'_{q_{of}}, y'_{q_{of}}\rangle\}_{\mathcal{O}^f}$*, and* $T_g = \{\langle x''_1, y''_1\rangle, \ldots, \langle x''_{q_{og}}, y''_{q_{og}}\rangle\}_{\mathcal{O}^g}$*, we have that*

$$\Pr_{k_1,k_2,g}[\sigma \in \mathsf{Bad}(k_1, k_2, g)] \le \left(q_c^2 + 2q_{of}q_c + 2 \cdot \binom{q_c}{2}\right) \cdot 2^{-n}.$$

Proof. (Sketch) Recall that a transcript $\sigma \in \mathsf{Bad}(k_1, k_2, g)$ if at least one of the above events occurs. We obtain an upper bound on the probabilities of each of these events separately using the fact that k_1, k_2 are chosen uniformly at

random from I_{2n} and that g is chosen uniformly at random from $\mathsf{Rand}^{n\to n}$. Applying the union bound to sum the probabilities for each event yields the desired result.

Lemma 2. *Let σ be defined as in Lemma 1. Then,*

$$\Pr_{\tilde{\Psi}}[T_{\tilde{\Psi}} = \sigma | \sigma \notin \mathsf{Bad}(k_1, k_2, g)] = \Pr_{\tilde{R}}[T_{\tilde{R}} = \sigma].$$

Proof. It is easy to see that $\Pr_{\tilde{R}}[T_{\tilde{R}} = \sigma] = 2^{-(2q_c+q_{of}+q_{og})n}$ (following an argument in [NR99], [RR00]). Now, fix k_1, k_2, g to be such that $\sigma \notin \mathsf{Bad}(k_1, k_2, g)$. We will now compute $\Pr_{f,h}[T_{\tilde{\Psi}} = \sigma]$ (recall that in the definition of $\tilde{\Psi}$, h is a random oracle independent of f and g, and note that the probability is now only over the choice of f and h). Since σ is a possible $\mathcal{A}$-transcript, it follows that $T_{\tilde{\Psi}} = \sigma$ if and only if $y_i = k_1 \oplus \tilde{\Psi}(g, f, f, g)(x_i \oplus k_2)$ for all $1 \le i \le q_c$, $y'_j = f(x'_j)$, for all $1 \le j \le q_{of}$, and $y''_j = g(x''_j)$ for all $1 \le j \le q_{og}$. If we define $S_i = k_1^L \oplus x_i^L \oplus g(x_i^R \oplus k_1^R)$ and $T_i = k_2^R \oplus y_i^R \oplus g(y_i^L \oplus k_2^L)$, then

$$(y_i^L, y_i^R) = \tilde{\Psi}(x_i^L, x_i^R) \Leftrightarrow f(S_i) \oplus k_1^R = T_i \oplus x_i^R \text{ and } f(T_i) \oplus k_2^L = y_i^L \oplus S_i.$$

Now observe that for all $1 \le i < j \le q_c$, $S_i \not= S_j$ and $T_i \not= T_j$ (otherwise $\sigma \in \mathsf{Bad}(k_1, k_2, g)$). Similarly, for all $1 < i, j < q_c$, $S_i \not= T_j$. In addition, it follows again from the fact that $\sigma \notin \mathsf{Bad}(k_1, k_2, g)$ that for all $1 \le i \le q_c$ and $1 \le j \le q_{og}$, $x'_i \not= S_j$ and $x'_i \not= T_j$. So, if $\sigma \notin \mathsf{Bad}(k_1, k_2, g)$ all the inputs to f are distinct. Since f is modeled as a random oracle, $\Pr_{f,h}[T_{\tilde{\Psi}} = \sigma] = 2^{-(2q_c+q_{of}+q_{og})n}$ (the cipher transcript contributes 2^{-2nq_c} and the oracle transcripts contribute $2^{-q_{of}n-q_{og}n}$ to the probability). Thus, for every choice of k_1, k_2, g such that $\sigma \notin \mathsf{Bad}(k_1, k_2, g)$, the probability that $T_{\tilde{\Psi}} = \sigma$ is exactly the same: $2^{-(2q_c+q_{of}+q_{og})n}$. Therefore: $\Pr_{\tilde{\Psi}}[T_{\tilde{\Psi}} = \sigma | \sigma \notin \mathsf{Bad}(k_1, k_2, g)] = 2^{-(2q_c+q_{of}+q_{og})n}$.

The rest of the proof consists of using the above lemma and Propositions 1, 2 and 3, as well as Lemmas 1 and 2, in a probability argument. The idea is to first express the adversary's advantage as a function of how its distinguishing machine behaves on specific transcripts. Then, these probabilities are re-expressed to incorporate the conditions Bad and BadG. By basic manipulation of probabilities, we can show that the adversary's advantage is bounded above by the probability of the conditions Bad or BadG occurring, plus the probability that the transcript is inconsistent. An additional term of the form $\binom{q_c}{2} \cdot 2^{-2n}$ also appears because of an application of the triangle inequality. The complete details are omitted due to space constraints, though are available in the full version [GR04].

4 Conclusions

We resolved a fundamental open problem of the Even-Mansour work by demonstrating that the underlying random permutation oracle could be instantiated

with a construction based on random function oracles. There are many avenues for future work. For example, we may be able to apply our techniques to other situations where random permutation oracles are useful. Also, there is a sizeable gap between the best known key-recovery attack and the bound achieved in our security proof. Perhaps that gap can be decreased by developing a variant on the slide-with-twist that exploits the structure of our construction.

References

[BBP04] M. Bellare, A. Boldyreva, and A. Palacio. An un-instantiable random-oracle-model scheme for a hybrid-encryption problem. In C. Cachin and J. Camenisch, editors, *Proc. EUROCRYPT 2004*, Lecture Notes in Computer Science. Springer-Verlag, 2004.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th Annual Symposium on Foundations of Computer Science*, pages 514–523. IEEE, 1996.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer-Verlag, 21–25 August 1994.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993.

[BW00] A. Biryukov and D. Wagner. Advanced slide attacks. In Advances in Cryptology – Proc. of Eurocrypt 2000. Springer-Verlag.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. 30th ACM Symp. on Theory of Computing*, 1998.

[Dae91] J. Daemen. Limitations of the Even-Mansour construction. In *Advances in Cryptology – ASIACRYPT '91, vol. 739, 495–498, 1992.* Springer-Verlag. Initially Presented at the Rump Session.

[EM97] S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, Summer 1997. Earlier version in *Proc. ASIACRYPT 1991.* Lecture Notes in Computer Science, vol. 739, 210–224, Springer Verlag (1992).

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1984.

[GK03] S. Goldwasser and Y. Tauman Kalai. On the (in)security of the Fiat-Shamir Paradigm. In *Proceedings of FOCS 2003*, 2003.

[GR04] C. Gentry and Z. Ramzan. Eliminating random permutation oracles in the Even-Mansour cipher. *Cryptology ePrint archive*, 2004.

[KR96] J. Kilian and P. Rogaway. How to protect against exhaustive search. In *Proc. CRYPTO 96*, Lecture Notes in Computer Science. Springer-Verlag, 1996.

[LR88] M. Luby and C. Rackoff. How to construct pseudorandom permutations and pseudorandom functions. *SIAM J. Computing*, 17(2):373–386, April 1988.

[FIPS46] National Bureau of Standards. FIPS publication 46: Data encryption standard, 1977. Federal Information Processing Standards Publication 46.

[NR99] M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-Rackoff revisited. *J. of Cryptology*, 12:29–66, 1999. Preliminary version in: *Proc. STOC 97*.

[PRS02] S. Patel, Z. Ramzan, and G. Sundaram. Luby-Rackoff ciphers: Why XOR is not exclusive. In Proc. of Selected Areas of Cryptography, Lecture Notes in Computer Science, Vol. 2595, pages 271–290.

[RR00] Z. Ramzan and L. Reyzin. On the Round Security of Symmetric-Key Cryptographic Primitives. In *Proc. CRYPTO 00*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

Towards Plaintext-Aware Public-Key Encryption Without Random Oracles

Mihir Bellare and Adriana Palacio

Dept. of Computer Science & Engineering, University of California,
San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA
{mihir, apalacio}@cs.ucsd.edu
http://www-cse.ucsd.edu/users/{mihir, apalacio}

Abstract. We consider the problem of defining and achieving plaintext-aware encryption without random oracles in the classical public-key model. We provide definitions for a hierarchy of notions of increasing strength: PA0, PA1 and PA2, chosen so that PA1+IND-CPA → IND-CCA1 and PA2+IND-CPA → IND-CCA2. Towards achieving the new notions of plaintext awareness, we show that a scheme due to Damgård [12], denoted DEG, and the "lite" version of the Cramer-Shoup scheme [11], denoted CS-lite, are both PA0 under the DHK0 assumption of [12], and PA1 under an extension of this assumption called DHK1. As a result, DEG is the most efficient proven IND-CCA1 scheme known.

1 Introduction

The theory of encryption is concerned with defining and implementing notions of security for encryption schemes [22, 23, 17, 25, 27, 15]. One of the themes in its history is the emergence of notions of security of increasing strength that over time find applications and acceptance.

Our work pursues, from the same perspective, a notion that is stronger than any previous ones, namely plaintext awareness. Our goal is to strengthen the foundations of this notion by lifting it out of the random-oracle model where it currently resides. Towards this end, we provide definitions of a hierarchy of notions of plaintext awareness, relate them to existing notions, and implement some of them. We consider this a first step in the area, however, since important questions are left unresolved. We begin below by reviewing existing work and providing some motivation for our work.

1.1 Background

Intuitively, an encryption scheme is plaintext aware (PA) if the "only" way that an adversary can produce a valid ciphertext is to apply the encryption algorithm to the public key and a message. In other words, any adversary against a PA scheme that produces a ciphertext "knows" the corresponding plaintext.

Random-Oracle model work. The notion of PA encryption was first suggested by Bellare and Rogaway [6], with the motivation that PA+IND-CPA

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 48–62, 2004.

should imply IND-CCA2. That is, security against chosen-plaintext attack coupled with plaintext awareness should imply security against adaptive chosen-ciphertext attack. The intuition, namely, that if an adversary knows the plaintext corresponding to a ciphertext it produces, then a decryption oracle must be useless to it, goes back to [8, 9]. Bellare and Rogaway [6] provided a formalization of PA in the random oracle (RO) model. They asked that for every adversary A taking the public key and outputting a ciphertext, there exist an extractor that, given the same public key and a transcript of the interaction of A with its RO, is able to decrypt the ciphertext output by A. We will refer to this notion as PA-BR.

Subsequently, it was found that PA-BR was too weak for PA-BR+IND-CPA to imply IND-CCA2. Bellare, Desai, Pointcheval and Rogaway [4] traced the cause of this to the fact that PA-BR did not capture the ability of the adversary to obtain ciphertexts via eavesdropping on communications made to the receiver. (Such eavesdropping can put into the adversary's hands ciphertexts whose decryptions it does not know, lending it the ability to create other ciphertexts whose decryptions it does not know.) They provided an appropriately enhanced definition (still in the RO model) that we denote by PA-BDPR, and showed that PA-BDPR+IND-CPA $\rightarrow$ IND-CCA2.

Plaintext awareness is exploited, even though typically implicitly rather than explicitly, in the proofs of the IND-CCA2 security of numerous RO-model encryption schemes, e.g., [16, 28, 7].

PA AND THE RO MODEL. By restricting the above-mentioned RO-model definitions to schemes and adversaries that do not query the RO, one obtains natural counterpart standard (i.e., non-RO) model definitions of PA. These standard-model definitions turn out, however, not to be achievable without sacrificing privacy, because the extractor can simply be used for decryption. This indicates that the use of the RO model in the definitions of [6, 4] is central.

Indeed, PA as per [6, 4] is "designed" for the RO model in the sense that the definition aims to capture certain properties of certain RO-model schemes, namely, the fact that possession of the transcript of the interaction of an adversary with its RO permits decryption of ciphertexts formed by this adversary. It is not clear what counterpart this intuition has in the standard model.

The lack of a standard-model definition of PA results in several gaps. One such arises when we consider that RO-model PA schemes are eventually instantiated to get standard-model schemes. In that case, what property are these instantiated schemes even supposed to possess? There is no definition that we might even discuss as a target.

PA VIA KEY REGISTRATION. PA without ROs was first considered by Herzog, Liskov and Micali [21], who define and implement it in an extension of the usual public-key setting. In their setting, the sender (not just the receiver) has a public key, and, in a key-registration phase that precedes encryption, proves knowledge of the corresponding secret key to a key-registration authority via an interactive proof of knowledge. Encryption is a function of the public keys of both the sender

and the receiver, and the PA extractor works by extracting the sender secret key using the knowledge extractor of the interactive proof of knowledge.

Their work also points to an application of plaintext-aware encryption where the use of the latter is crucial in the sense that IND-CCA2-secure encryption does not suffice, namely to securely instantiate the ideal encryption functions of the Dolev-Yao model [14].

1.2 Our Goals and Motivation

The goal of this work is to provide definitions and constructions for plaintext-aware public-key encryption in the standard and classical setting of public-key encryption, namely the one where the receiver (but not the sender) has a public key, and anyone (not just a registered sender) can encrypt a message for the receiver as a function of the receiver's public key. In this setting there is no key-registration authority or key-registration protocol akin to [21].

Motivations include the following. As in the RO model, we would like a tool enabling the construction of public-key encryption schemes secure against chosen-ciphertext attack. We would also like to have some well-defined notion that can be viewed as a target for instantiated RO-model PA schemes. (One could then evaluate these schemes with regard to meeting the target.)

Additionally, we would like to enable the possibility of instantiating the ideal encryption functions of the Dolev-Yao model [14] without recourse to either random oracles or the key-registration model. Note that the last is an application where, as per [21], PA is required and IND-CCA2 does not suffice, meaning plaintext-awareness is crucial. (However, see also [1].)

As we will see later, consideration of PA in the standard model brings other benefits, such as some insight, or at least an alternative perspective, on the design of existing encryption schemes secure against chosen-ciphertext attack. Let us now discuss our contributions.

1.3 Definitions

The first contribution of this paper is to provide definitions for plaintext-aware encryption in the standard model and standard public-key setting.

Overview. We provide a hierarchy consisting of three notions of increasing strength that we denote by PA0, PA1 and PA2. There are several motivations for this. One is that these will be seen (in conjunction with IND-CPA) to imply security against chosen-ciphertext attacks of different strengths. Another is that, as will become apparent, PA is difficult to achieve, and progress can be made by first achieving it in weaker forms. Finally, it is useful, pedagogically, to bring in new definitional elements incrementally.

A closer look. Our basic definitional framework considers a polynomial-time adversary $\boldsymbol{C}$, called a ciphertext creator, that takes input the public key and can query ciphertexts to an oracle. A polynomial-time algorithm $\boldsymbol{C}^*$ is said to be a successful extractor for $\boldsymbol{C}$ if it can provide replies to the oracle queries of $\boldsymbol{C}$

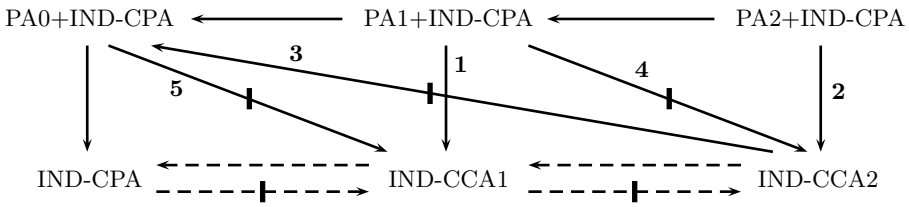


Fig. 1. An arrow is an implication, and, in the directed graph given by the arrows, there is a path from A to B if and only if A implies B. The hatched arrows represent separations. Solid lines represent results from this paper, while dashed lines represent results from prior work [4, 15]. The number on an arrow or hatched arrow refers to the theorem in this paper that establishes this relationship. Absence of a number on a solid arrow means the result is trivial

that are computationally indistinguishable from those provided by a decryption oracle.

An important element of the above framework is that the extractor gets as input *the same public key as the ciphertext creator, as well as the coin tosses of the ciphertext creator*. This reflects the intuition that the extractor is the "subconscious" of the adversary, and begins with exactly the same information as the adversary itself.

We say that an encryption scheme is PA0 (respectively, PA1) if there exists a successful extractor for any ciphertext creator that makes only a single oracle query (respectively, a polynomial number of oracle queries).

Eavesdropping capability in PA2 is captured by providing the ciphertext creator $\boldsymbol{C}$ with an additional oracle that returns ciphertexts, but care has to be taken in defining this oracle. It does *not* suffice to let it be an encryption oracle because we want to model the ability of the adversary to obtain ciphertexts whose decryptions it may not know. Our formalization of PA2 allows the additional oracle to compute a plaintext, as a function of the query made to it and coins unknown to $\boldsymbol{C}$, and return the encryption of this plaintext to $\boldsymbol{C}$.

Formal definitions of PA0, PA1 and PA2 are in Section 3.

1.4 Relations

PA by itself is not a notion of privacy, and so we are typically interested in PA coupled with the minimal notion of privacy, namely IND-CPA [22, 23]. We consider six notions, namely, PA0+IND-CPA, PA1+IND-CPA and PA2+IND-CPA, on the one hand, and the standard notions of privacy IND-CPA, IND-CCA1 [25] and IND-CCA2 [27], on the other. We provide implications and separations among these six notions in the style of [4, 15]. The results are depicted in Figure 1. For notions A, B, an implication, represented by A $\rightarrow$ B, means that every encryption scheme satisfying notion A also satisfies notion B, and a separation, represented by A $\not\rightarrow$B, means that there exists an encryption scheme satisfying notion A but not satisfying notion B. (The latter assumes there exists some encryption scheme satisfying notion A, since otherwise the question is vacuous.)

Figure 1 shows a minimal set of arrows and hatched arrows, but the relation between any two notions is resolved by the given relations. For example, IND-CCA1 $\not\rightarrow$PA1+IND-CPA, because, otherwise, there would be a path from IND-CCA2 to PA0+IND-CPA, contradicting the hatched arrow labeled 3. Similarly, we get PA0 $\not\rightarrow$PA1 $\not\rightarrow$PA2, meaning the three notions of plaintext awareness are of increasing strength.

The main implications are that PA1+IND-CPA implies IND-CCA1 and PA2+IND-CPA implies IND-CCA2. The PA1+IND-CPA $\rightarrow$ IND-CCA1 result shows that even a notion of PA not taking eavesdropping adversaries into account is strong enough to imply security against a significant class of chosen-ciphertext attacks. Since the PA+IND-CPA $\rightarrow$ IND-CCA2 implication has been a motivating target for definitions of PA, the PA2+IND-CPA $\rightarrow$ IND-CCA2 result provides some validation for the definition of PA2.

Among the separations, we note that IND-CCA2 does not imply PA0, meaning even the strongest form of security against chosen-ciphertext attack is not enough to guarantee the weakest form of plaintext awareness.

1.5 Constructions

The next problem we address is to find provably-secure plaintext-aware encryption schemes.

Approaches. A natural approach to consider is to include a non-interactive zero-knowledge proof of knowledge [13] of the message in the ciphertext. However, as we explain in [2], this fails to achieve PA.

As such approaches are considered and discarded, it becomes apparent that achieving even the weaker forms of PA in the standard (as opposed to RO) model may be difficult. We have been able to make progress, however, under some strong assumptions that we now describe.

DHK assumptions. Let G be the order q subgroup of $\mathbb{Z}^*_{2q+1}$, where $q, 2q+1$ are primes, and let g be a generator of G. Damgård [12] introduced and used an assumption that states, roughly, that an adversary given g^a and outputting a pair of the form (g^b, g^{ab}) must "know" b. The latter is captured by requiring an extractor that given the adversary coins and inputs can output b. We call our formalization of this assumption (cf. Assumption 2) DHK0.[1] We also introduce an extension of this assumption called DHK1 (cf. Assumption 1), in which the adversary does not just output one pair (g^b, g^{ab}), but instead interacts with the

[1] Another formalization, called DA-1, is used by Hada and Tanaka [19]. (We refer to the full version of their paper [19], which points out that the formalization of the preliminary version [20] is wrong.) This differs from DHK0 in being for a non-uniform setting. DA-1 is called KEA1 by [5], based on Naor's terminology [24]: KEA stands for "knowledge of exponent." Hada and Tanaka [19] also introduced and used another assumption, that they call DA-2 and is called KEA2 in [5], but the latter show that this assumption is false. The DHK0/DA-1/KEA1 assumptions, to the best of our knowledge, are not known to be false.

extractor, feeding it such pairs adaptively and each time expecting back the discrete logarithm of the first component of the pair.

The DEG scheme. Damgård presented a simple ElGamal variant that we call DEG. It is efficient, requiring only three exponentiations to encrypt and two to decrypt.

We prove that DEG is PA0 under the DHK0 assumption and PA1 under the DHK1 assumption. Since DEG is easily seen to be IND-CPA under the DDH assumption, and we saw above that PA1+IND-CPA → IND-CCA1, a consequence is that DEG is IND-CCA1 assuming DHK1 and DDH. DEG is in fact the most efficient IND-CCA1 scheme known to date to be provably secure in the standard model.

Damgård [12] claims that DEG meets a notion of security under ciphertext attack that we call RPR-CCA1, assuming DHK0 and assuming the ElGamal scheme meets a notion called RPR-CPA. (Both notions are recalled in the full version of this paper [2], and are weaker than IND-CCA1 and IND-CPA, respectively). As we explain in [2], his proof has a flaw, but his overall approach and intuition are valid, and the proof can be fixed by simply assuming DHK1 in place of DHK0. In summary, our contribution is (1) to show that DEG meets a stronger and more standard notion of security than RPR-CCA1, namely IND-CCA1, and (2) to show it is PA0 and PA1, indicating that it has even stronger properties, and providing some formal support for the intuition given in [12] about the security underlying the scheme.

CS-Lite. CS-lite is a simpler and more efficient version of the Cramer-Shoup encryption scheme [11] that is IND-CCA1 under the DDH assumption. We show that CS-lite is PA0 under the DHK0 assumption and PA1 under the DHK1 assumption. (IND-CPA under DDH being easy to see, this again implies CS-lite is IND-CCA1 under DHK1 and DDH, but in this case the conclusion is not novel.) What we believe is interesting about our results is that they show that some form of plaintext awareness underlies the CS-lite scheme, and this provides perhaps an alternative viewpoint on the source of its security. We remark, however, that DEG is more efficient than CS-lite.

Warning and discussion. DHK0 and DHK1 are strong and non-standard assumptions. As pointed out by Naor [24], they are not efficiently falsifiable. (However, such assumptions can be shown to be false as exemplified in [5]). However standard-model schemes, even under strong assumptions, might provide better guarantees than RO model schemes, for we know that the latter may not provide real-world security guarantees at all [10, 26, 18, 3]. Also, PA without random oracles is challenging to achieve, and we consider it important to "break ground" by showing it is possible, even if under strong assumptions.

Open questions. The central open question is to find an IND-CPA+PA2 scheme provably secure under some plausible assumption. We suggest, in particular, that an interesting question is whether the Cramer-Shoup scheme, already known to be IND-CCA2, is PA2 under some appropriate assumption. (Intu-

itively, it seems to be PA2.) It would also be nice to achieve PA0 or PA1 under weaker and more standard assumptions than those used here.

2 Notation and Standard Definitions

We let $\mathbb{N} = \{1, 2, 3, \ldots\}$. We denote by ε the empty string, by $|x|$ the length of a string x, by $\bar{x}$ the bitwise complement of x, by "$\|$" the string-concatenation operator, and by 1^k the string of $k \in \mathbb{N}$ ones. We denote by $[\,]$ the empty list. Given a list L and an element x, $L \,@\, x$ denotes the list consisting of the elements in L followed by x. If S is a randomized algorithm, then $S(x, y, \ldots; R)$ denotes its output on inputs $x, y, \ldots$ and coins R; $s \xleftarrow{\$} S(x, y, \ldots)$ denotes the result of picking R at random and setting $s = S(x, y, \ldots; R)$; and $[S(x, y, \ldots)]$ denotes the set of all points having positive probability of being output by S on inputs $x, y, \ldots$. Unless otherwise indicated, an algorithm is randomized.

Encryption schemes. We recall the standard syntax. An asymmetric (also called public-key) encryption scheme is a tuple $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ whose components are as follows. The polynomial-time key-generation algorithm $\mathcal{K}$ takes input 1^k, where $k \in \mathbb{N}$ is the security parameter, and returns a pair (pk, sk) consisting of a public key and matching secret key. The polynomial-time encryption algorithm $\mathcal{E}$ takes a public key pk and a message M to return a ciphertext C. The deterministic, polynomial-time decryption algorithm $\mathcal{D}$ takes a secret key sk and a ciphertext C to return either a message M or the special symbol $\perp$ indicating that the ciphertext is invalid. The polynomial-time computable message-space function MsgSp associates to each public key pk a set $\mathrm{MsgSp}(pk)$ called the message space of pk. It is required that for every $k \in \mathbb{N}$

$$\Pr\left[\, (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\,;\; M \xleftarrow{\$} \mathrm{MsgSp}(pk)\,;\; C \xleftarrow{\$} \mathcal{E}(pk, M) \;:\; \mathcal{D}(sk, C) = M \,\right] \;=\; 1\,.$$

Standard security notions. We recall the definitions of IND-CPA, IND-CCA1, and IND-CCA2 security that originate in [22], [25], and [27], respectively. We use the formalizations of [4]. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ be an asymmetric encryption scheme, let $k \in \mathbb{N}$ and $b \in \{0, 1\}$. Let $\boldsymbol{X}$ be an algorithm with access to an oracle. For $\mathrm{aaa} \in \{\mathrm{cpa}, \mathrm{cca1}, \mathrm{cca2}\}$, consider the following experiment

$$\begin{array}{l}\text{Experiment } \mathbf{Exp}^{\text{ind-aaa-}b}_{\mathcal{AE},\boldsymbol{X}}(k)\\ \quad (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\,;\; (M_0, M_1, \mathrm{St}) \xleftarrow{\$} \boldsymbol{X}^{\mathcal{O}_1(\cdot)}(\mathsf{find}, pk)\,;\; C \xleftarrow{\$} \mathcal{E}(pk, M_b)\\ \quad d \leftarrow \boldsymbol{X}^{\mathcal{O}_2(\cdot)}(\mathsf{guess}, C, \mathrm{St})\,;\; \text{Return } d\end{array}$$

where

$$\begin{array}{ll}\text{If aaa} = \mathrm{cpa} \text{ then } \mathcal{O}_1(\cdot) = \varepsilon & \text{and } \mathcal{O}_2(\cdot) = \varepsilon\\ \text{If aaa} = \mathrm{cca1} \text{ then } \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) & \text{and } \mathcal{O}_2(\cdot) = \varepsilon\\ \text{If aaa} = \mathrm{cca2} \text{ then } \mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot) & \text{and } \mathcal{O}_2(\cdot) = \mathcal{D}_{sk}(\cdot)\end{array}$$

In each case it is required that $M_0, M_1 \in \mathrm{MsgSp}(pk)$ and $|M_0| = |M_1|$. In the case of IND-CCA2, it is also required that $\boldsymbol{X}$ not query its decryption oracle with ciphertext C. We call $\boldsymbol{X}$ an *ind-aaa-adversary*. The *ind-aaa-advantage* of $\boldsymbol{X}$ is

Experiment $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $x \xleftarrow{\$} \boldsymbol{C}^{\mathcal{D}(sk,\cdot)}(pk)$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

---

Experiment $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$
Choose coins $R[\boldsymbol{C}], R[\boldsymbol{C}^*]$ for $\boldsymbol{C}, \boldsymbol{C}^*$, respectively ; $\text{St}[\boldsymbol{C}^*] \leftarrow (pk, R[\boldsymbol{C}])$
Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{C}$ makes query Q then
$(M, \text{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \text{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return M to $\boldsymbol{C}$ as the reply EndIf
Let x denote the output of $\boldsymbol{C}$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

Fig. 2. Experiments used to define PA1 and PA0

$$\mathbf{Adv}^{\text{ind-aaa}}_{\mathcal{AE},\boldsymbol{X}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{ind-aaa-1}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-aaa-0}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1\right].$$

For AAA $\in \{\text{CPA}, \text{CCA1}, \text{CCA2}\}$, $\mathcal{AE}$ is said to be IND-AAA secure if $\mathbf{Adv}^{\text{ind-aaa}}_{\mathcal{AE},\boldsymbol{X}}(\cdot)$ is negligible for every polynomial-time ind-aaa-adversary $\boldsymbol{X}$.

3 New Notions of Plaintext Awareness

In this section we provide our formalizations of plaintext-aware encryption. We provide the formal definitions first and explanations later. We begin with PA1, then define PA0 via this, and finally define PA2.

Definition 1. [**PA1**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \text{MsgSp})$ be an asymmetric encryption scheme. Let $\boldsymbol{C}$ be an algorithm that has access to an oracle, takes as input a public key pk, and returns a string. Let $\boldsymbol{D}$ be an algorithm that takes a string and returns a bit. Let $\boldsymbol{C}^*$ be an algorithm that takes a string and some state information, and returns a message or the symbol $\perp$, and a new state. We call $\boldsymbol{C}$ a *ciphertext-creator* adversary, $\boldsymbol{D}$ a *distinguisher*, and $\boldsymbol{C}^*$ a *pa1-extractor*. For $k \in \mathbb{N}$, we define the experiments shown in Figure 2. The *pa1-advantage* of $\boldsymbol{C}$ relative to $\boldsymbol{D}$ and $\boldsymbol{C}^*$ is

$$\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) = 1\right].$$

We say that $\boldsymbol{C}^*$ is a *successful pa1-extractor for* $\boldsymbol{C}$ if for every polynomial-time distinguisher $\boldsymbol{D}$ the function $\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. We say $\mathcal{AE}$ is *PA1 secure* if for any polynomial-time ciphertext creator there exists a successful polynomial-time pa1-extractor. ▌

Definition 2. [**PA0**] Let $\mathcal{AE}$ be an asymmetric encryption scheme. We call a ciphertext-creator adversary that makes *exactly one* oracle query a *pa0 ciphertext creator*. We call a pa1-extractor for a pa0 ciphertext creator a *pa0-extractor*. We say that $\mathcal{AE}$ is *PA0 secure* if for any polynomial-time pa0 ciphertext creator there exists a successful polynomial-time pa0-extractor. ▌

Experiment $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D}}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $\textsc{List} \leftarrow [\,]$
Choose coins $R[\boldsymbol{C}], R[\boldsymbol{P}]$ for $\boldsymbol{C}, \boldsymbol{P}$, respectively ; $\text{St}[\boldsymbol{P}] \leftarrow \varepsilon$
Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:

- If $\boldsymbol{C}$ makes query (dec, Q) then
  $M \leftarrow \mathcal{D}(sk, Q)$; Return M to $\boldsymbol{C}$ as the reply EndIf
- If $\boldsymbol{C}$ makes query (enc, Q) then
  $(M, \text{St}[\boldsymbol{P}]) \leftarrow \boldsymbol{P}(Q, \text{St}[\boldsymbol{P}]; R[\boldsymbol{P}])$; $C \xleftarrow{\$} \mathcal{E}(pk, M)$; $\textsc{List} \leftarrow \textsc{List} \mathbin{@} C$
  Return C to $\boldsymbol{C}$ as the reply EndIf

Let x denote the output of $\boldsymbol{C}$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

---

Experiment $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $\textsc{List} \leftarrow [\,]$
Choose coins $R[\boldsymbol{C}], R[\boldsymbol{P}], R[\boldsymbol{C}^*]$ for $\boldsymbol{C}, \boldsymbol{P}, \boldsymbol{C}^*$, respectively
$\text{St}[\boldsymbol{P}] \leftarrow \varepsilon$; $\text{St}[\boldsymbol{C}^*] \leftarrow (pk, R[\boldsymbol{C}])$
Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:

- If $\boldsymbol{C}$ makes query (dec, Q) then
  $(M, \text{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \textsc{List}, \text{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$
  Return M to $\boldsymbol{C}$ as the reply EndIf
- If $\boldsymbol{C}$ makes query (enc, Q) then
  $(M, \text{St}[\boldsymbol{P}]) \leftarrow \boldsymbol{P}(Q, \text{St}[\boldsymbol{P}]; R[\boldsymbol{P}])$; $C \xleftarrow{\$} \mathcal{E}(pk, M)$; $\textsc{List} \leftarrow \textsc{List} \mathbin{@} C$
  Return C to $\boldsymbol{C}$ as the reply EndIf

Let x denote the output of $\boldsymbol{C}$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

Fig. 3. Experiments used to define PA2

We now explain the ideas behind the above formalisms. The core of the formalization of plaintext awareness of asymmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \text{MsgSp})$ considers a polynomial-time ciphertext-creator adversary $\boldsymbol{C}$ that takes input a public key pk, has access to an oracle and returns a string. The adversary tries to distinguish between the cases that its oracle is $\mathcal{D}(sk, \cdot)$, or it is an *extractor* algorithm $\boldsymbol{C}^*$ that takes as input the same public key pk. PA1 security requires that there exist a polynomial-time $\boldsymbol{C}^*$ such that $\boldsymbol{C}$'s outputs in the two cases are indistinguishable. We allow $\boldsymbol{C}^*$ to be stateful, maintaining state $\text{St}[\boldsymbol{C}^*]$ across invocations. Importantly, $\boldsymbol{C}^*$ *is provided with the coin tosses of* $\boldsymbol{C}$; otherwise, $\boldsymbol{C}^*$ would be functionally equivalent to the decryption algorithm and thus could not exist unless $\mathcal{AE}$ were insecure with regard to providing privacy. We remark that this formulation is stronger than one not involving a distinguisher $\boldsymbol{D}$, in which $\boldsymbol{C}$ simply outputs a bit representing its guess, since $\boldsymbol{C}^*$ gets the coins of $\boldsymbol{C}$, but not the coins of $\boldsymbol{D}$.

PA0 security considers only adversaries that make a *single* query in their attempt to determine if the oracle is a decryption oracle or an extractor.

Definition 3. [**PA2**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ be an asymmetric encryption scheme. Let $\boldsymbol{C}$ be an algorithm that has access to an oracle, takes as input a public key pk, and returns a string. Let $\boldsymbol{P}$ be an algorithm that takes a string and some state information, and returns a message and a new state. Let $\boldsymbol{D}$ be an algorithm that takes a string and returns a bit. Let $\boldsymbol{C}^*$ be an algorithm that takes a string, a list of strings and some state information, and returns a message or the symbol $\perp$, and a new state. We call $\boldsymbol{C}$ a *ciphertext-creator* adversary, $\boldsymbol{P}$ a *plaintext-creator* adversary, $\boldsymbol{D}$ a *distinguisher*, and $\boldsymbol{C}^*$ a *pa2-extractor*. For $k \in \mathbb{N}$, we define the experiments shown in Figure 3. It is required that, in these experiments, $\boldsymbol{C}$ not make a query (dec, C) for which $C \in \textsc{Clist}$. The *pa2-advantage* of $\boldsymbol{C}$ relative to $\boldsymbol{P}$, $\boldsymbol{D}$ and $\boldsymbol{C}^*$ is

$$\mathbf{Adv}^{\mathrm{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\mathrm{pa2\text{-}d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D}}(k) = 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\mathrm{pa2\text{-}x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(k) = 1\,\right].$$

We say that $\boldsymbol{C}^*$ is a *successful pa2-extractor for* $\boldsymbol{C}$ if for every polynomial-time plaintext creator $\boldsymbol{P}$ and distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\mathrm{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. We say $\mathcal{AE}$ is *PA2 secure* if for any polynomial-time ciphertext creator there exists a successful polynomial-time pa2-extractor. ▌

In the definition of PA2, the core setting of PA1 is enhanced to model the real-life capability of a ciphertext creator to obtain ciphertexts via eavesdropping on communications made by a third party to the receiver (cf. [4]). Providing $\boldsymbol{C}$ with an encryption oracle does not capture this because eavesdropping puts into $\boldsymbol{C}$'s hands ciphertexts of which it does *not* know the corresponding plaintext, and, although we disallow $\boldsymbol{C}$ to query these to its oracle, it might be able to use them to create other ciphertexts whose corresponding plaintext it does not know and on which the extractor fails.

Modeling eavesdropping requires balancing two elements: providing $\boldsymbol{C}$ with a capability to obtain ciphertexts of plaintexts it does not know, yet capturing the fact that $\boldsymbol{C}$ might have partial information about the plaintexts, or control of the distribution from which these plaintexts are drawn. We introduce a companion *plaintext-creator* adversary $\boldsymbol{P}$ who, upon receiving a communication from $\boldsymbol{C}$, creates a plaintext and forwards it to an encryption oracle. The ciphertext emanating from the encryption oracle is sent to both $\boldsymbol{C}$ and $\boldsymbol{C}^*$. $\boldsymbol{C}$ has some control over $\boldsymbol{P}$ via its communication to $\boldsymbol{P}$, but we ensure this is not total by *denying* $\boldsymbol{C}$ *and* $\boldsymbol{C}^*$ *the coin tosses of* $\boldsymbol{P}$, and also by asking that $\boldsymbol{C}^*$ depend on $\boldsymbol{C}$ but not on $\boldsymbol{P}$.

The extractor $\boldsymbol{C}^*$ is, as before, provided with the coin tosses of $\boldsymbol{C}$. Two types of oracle queries are allowed to $\boldsymbol{C}$. Via a query (dec, Q), it can ask its oracle to decrypt ciphertext Q. Alternatively, it can make a query (enc, Q) to call $\boldsymbol{P}$ with argument Q, upon which the latter computes a message M and forwards it to the encryption oracle, which returns the resulting ciphertext to $\boldsymbol{C}$, and $\boldsymbol{C}^*$ in the case that $\boldsymbol{C}$'s oracle is $\boldsymbol{C}^*$. We observe that if an asymmetric encryption scheme is PA2 secure then it is PA1 secure, and if it is PA1 secure then it is PA0 secure.

See [2] for extensive comparisons of these definitions with previous ones, and also for stronger, statistical versions of these notions.

4 Relations Among Notions

We now state the formal results corresponding to Figure 1, beginning with the two motivating applications of our notions of plaintext awareness. Proofs of these results are provided in the full version of this paper [2].

Theorem 1. [**PA1+IND-CPA** $\Rightarrow$ **IND-CCA1**] *Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is PA1 secure and IND-CPA secure, then it is IND-CCA1 secure.* ▌

Theorem 2. [**PA2+IND-CPA** $\Rightarrow$ **IND-CCA2**] *Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is PA2 secure and IND-CPA secure, then it is IND-CCA2 secure.* ▌

Theorem 3. [**IND-CCA2** $\not\Rightarrow$**PA0+IND-CPA**] *Assume there exists an IND-CCA2-secure asymmetric encryption scheme. Then there exists an IND-CCA2-secure asymmetric encryption scheme that is not PA0 secure.* ▌

Theorem 4. [**PA1+IND-CPA** $\not\Rightarrow$**IND-CCA2**] *Assume there exists a PA1 secure and IND-CPA-secure asymmetric encryption scheme. Then there exists a PA1 secure and IND-CPA-secure asymmetric encryption scheme that is not IND-CCA2 secure.* ▌

Theorem 5. [**PA0+IND-CPA** $\not\Rightarrow$**IND-CCA1**] *Assume there exists a PA0 secure and IND-CPA-secure asymmetric encryption scheme. Then there exists a PA0 secure and IND-CPA-secure asymmetric encryption scheme that is not IND-CCA1 secure.* ▌

5 Constructions

Prime-order groups. If p, q are primes such that $p = 2q + 1$, then we let G_q denote the subgroup of quadratic residues of $\mathbb{Z}_p^*$. Recall this is a cyclic subgroup of order q. If g is a generator of G_q then $\mathrm{dlog}_{q,g}(X)$ denotes the discrete logarithm of $X \in G_q$ to base g. A *prime-order-group generator* is a polynomial-time algorithm G that on input 1^k returns a triple (p, q, g) such that p, q are primes with $p = 2q + 1$, g is a generator of G_q, and $2^{k-1} < p < 2^k$ (p is k bits long).

The DHK assumptions. Let G be a prime-order-group generator, and suppose $(p, q, g) \in [G(1^k)]$. We say that (A, B, W) is a *DH-triple* if there exist $a, b \in Z_q$ such that $A = g^a \bmod p$, $B = g^b \bmod p$ and $W = g^{ab} \bmod p$. We say that (B, W) is a *DH-pair relative to* A if (A, B, W) is a DH-triple. One way for an adversary $\boldsymbol{H}$ taking input p, q, g, A to output a DH-pair (B, W) relative to A is to pick —and thus "know"— some $b \in \mathbb{Z}_q$, set $B = g^b \bmod p$ and $W = A^b \bmod p$, and output (B, W). Damgård [12] makes an assumption which, informally, says that this is the "only" way that a polynomial-time adversary $\boldsymbol{H}$ can output a DH-pair relative to A. His framework to capture this requires that there exist a suitable

Experiment $\mathbf{Exp}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k)$
$(p,q,g) \xleftarrow{\$} G(1^k)$; $a \xleftarrow{\$} \mathbb{Z}_q$; $A \leftarrow g^a \bmod p$
Choose coins $R[\boldsymbol{H}], R[\boldsymbol{H}^*]$ for $\boldsymbol{H}, \boldsymbol{H}^*$, respectively ; $\mathrm{St}[\boldsymbol{H}^*] \leftarrow ((p,q,g,A), R[\boldsymbol{H}])$
Run $\boldsymbol{H}$ on input p,q,g,A and coins $R[\boldsymbol{H}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{H}$ makes query (B,W) then
$(b, \mathrm{St}[\boldsymbol{H}^*]) \leftarrow \boldsymbol{H}^*((B,W), \mathrm{St}[\boldsymbol{H}^*]; R[\boldsymbol{H}^*])$
If $W \equiv B^a \pmod p$ and $B \not\equiv g^b \pmod p$ then return 1
Else return b to $\boldsymbol{H}$ as the reply EndIf EndIf
Return 0

Fig. 4. Experiment used to define the DHK1 and DHK0 assumptions

extractor $\boldsymbol{H}^*$ that can compute $\mathrm{dlog}_{q,g}(B)$ whenever $\boldsymbol{H}$ outputs some DH-pair (B,W) relative to A.

We provide a formalization of this assumption that we refer to as the DHK0 (DHK stands for Diffie-Hellman Knowledge) assumption. We also present a natural extension of this assumption that we refer to as DHK1. Here the adversary $\boldsymbol{H}$, given p,q,g,A, interacts with the extractor, querying it adaptively. The extractor is required to be able to return $\mathrm{dlog}_{q,g}(B)$ for each DH-pair (B,W) relative to A that is queried to it. Below we first present the DHK1 assumption, and then define the DHK0 assumption via this.

Assumption 1. [**DHK1**] Let G be a prime-order-group generator. Let $\boldsymbol{H}$ be an algorithm that has access to an oracle, takes two primes and two group elements, and returns nothing. Let $\boldsymbol{H}^*$ be an algorithm that takes a pair of group elements and some state information, and returns an exponent and a new state. We call $\boldsymbol{H}$ a dhk1-*adversary* and $\boldsymbol{H}^*$ a dhk1-*extractor*. For $k \in \mathbb{N}$ we define the experiment shown in Figure 4. The dhk1-*advantage* of $\boldsymbol{H}$ relative to $\boldsymbol{H}^*$ is

$$\mathbf{Adv}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k) = 1\,\right].$$

We say that G satisfies the DHK1 assumption if for every polynomial-time dhk1-adversary $\boldsymbol{H}$ there exists a polynomial-time dhk1-extractor $\boldsymbol{H}^*$ such that $\mathbf{Adv}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(\cdot)$ is negligible. ▌

Assumption 2. [**DHK0**] Let G be a prime-order-group generator. We call a dhk1-adversary that makes *exactly one* oracle query a dhk0-*adversary*. We call a dhk1-extractor for a dhk0-adversary a dhk0-*extractor*. We say that G satisfies the Diffie-Hellman Knowledge (DHK0) assumption if for every polynomial-time dhk0-adversary $\boldsymbol{H}$ there exists a polynomial-time dhk0-extractor $\boldsymbol{H}^*$ such that $\mathbf{Adv}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(\cdot)$ is negligible. ▌

We observe that DHK1 implies DHK0 in the sense that if a prime-order-group generator satisfies the former assumption then it also satisfies the latter assumption.

Algorithm $\mathcal{K}(1^k)$ $(p,q,g) \xleftarrow{\$} G(1^k)$ $x_1 \xleftarrow{\$} \mathbb{Z}_q$; $X_1 \leftarrow g^{x_1} \bmod p$ $x_2 \xleftarrow{\$} \mathbb{Z}_q$; $X_2 \leftarrow g^{x_2} \bmod p$ Return $((p,q,g,X_1,X_2),(p,q,g,x_1,x_2))$	Algorithm $\mathcal{E}((p,q,g,X_1,X_2),M)$ $y \xleftarrow{\$} \mathbb{Z}_q$; $Y \leftarrow g^y \bmod p$ $W \leftarrow X_1^y \bmod p$; $V \leftarrow X_2^y \bmod p$ $U \leftarrow V \cdot M \bmod p$ Return (Y,W,U)
Algorithm $\mathcal{D}((p,q,g,x_1,x_2),(Y,W,U))$ If $W \not\equiv Y^{x_1} \pmod p$ then return $\perp$ Else $M \leftarrow U \cdot Y^{-x_2} \bmod p$; Return M EndIf	$\mathrm{MsgSp}((p,q,g,X_1,X_2)) = G_q$

Fig. 5. Algorithms of the encryption scheme $\mathrm{DEG} = (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ based on prime-order-group generator G

Constructions. We would like to build an asymmetric encryption scheme that is PA0 secure (and IND-CPA secure) under the DHK0 assumption. An obvious idea is to use ElGamal encryption. Here the public key is $X = g^x$, where x is the secret key, and an encryption of message $M \in G_q$ has the form (Y,U), where $Y = g^y \bmod p$ and $U = X^y \cdot M \bmod p = g^{xy} \cdot M \bmod p$. However, we do not know whether this scheme is PA0 secure.

We consider a modification of the ElGamal scheme that was proposed by Damgård [12]. We call this scheme *Damgård ElGamal* or DEG. It is parameterized by a prime-order group generator G, and its components are depicted in Figure 5. The proof of the following is in the full version of this paper [2]:

Theorem 6. *Let G be a prime-order-group generator and let* $\mathrm{DEG} = (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ *be the associated Damgård ElGamal asymmetric encryption scheme defined in Figure 5. If G satisfies the DHK0 and DDH assumptions then* DEG *is PA0+IND-CPA secure. If G satisfies the DHK1 and DDH assumptions then* DEG *is PA1+IND-CPA secure.* ▌

As a consequence of the above and Theorem 1, DEG is IND-CCA1 secure under the DHK1 and DDH assumptions. DEG is in fact the most efficient known IND-CCA1 scheme with some proof of security in the standard model.

Next we consider the "lite" version of the Cramer-Shoup asymmetric encryption scheme [11]. The scheme, denoted CS-lite, is parameterized by a prime-order group generator G, and its components are depicted in Figure 6. This scheme is known to be IND-CCA1 secure under the DDH assumption [11]. We are able to show the following. The proof can be found in [2].

Theorem 7. *Let G be a prime-order-group generator, and let* $\text{CS-lite} = (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ *be the associated Cramer-Shoup lite asymmetric encryption scheme defined in Figure 6. If G satisfies the DHK0 and DDH assumptions then* CS-lite *is PA0+IND-CPA secure. If G satisfies the DHK1 and DDH assumptions then* CS-lite *is PA1+IND-CPA secure.* ▌

Algorithm $\mathcal{K}(1^k)$ $(p, q, g_1) \xleftarrow{\$} G(1^k)$; $g_2 \xleftarrow{\$} G_q \setminus \{1\}$ $x_1 \xleftarrow{\$} \mathbb{Z}_q$; $x_2 \xleftarrow{\$} \mathbb{Z}_q$; $z \xleftarrow{\$} \mathbb{Z}_q$ $X \leftarrow g_1^{x_1} \cdot g_2^{x_2} \bmod p$; $Z \leftarrow g_1^z \bmod p$ Return $((p, q, g_1, g_2, X, Z), (p, q, g_1, g_2, x_1, x_2, z))$	Algorithm $\mathcal{E}((p, q, g_1, g_2, X, Z), M)$ $r \xleftarrow{\$} \mathbb{Z}_q$ $R_1 \leftarrow g_1^r \bmod p$ $R_2 \leftarrow g_2^r \bmod p$ $E \leftarrow Z^r \cdot M \bmod p$ $V \leftarrow X^r \bmod p$ Return (R_1, R_2, E, V)
Algorithm $\mathcal{D}((p, q, g_1, g_2, x_1, x_2, z), (R_1, R_2, E, V))$ If $V \not\equiv R_1^{x_1} \cdot R_2^{x_2} \pmod p$ then return $\perp$ Else $M \leftarrow E \cdot R_1^{-z} \bmod p$; Return M EndIf	$\mathrm{MsgSp}((p, q, g_1, g_2, X, Z)) = G_q$

Fig. 6. Algorithms of the encryption scheme CS-lite $= (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ based on prime-order-group generator G

Again, the above and Theorem 1 imply that CS-lite is IND-CCA1 secure under the DHK1 and DDH assumptions. This however is not news, since we already know that DDH alone suffices to prove it IND-CCA1 [11]. However, it does perhaps provide a new perspective on why the scheme is IND-CCA1, namely that this is due to its possessing some form of plaintext awareness.

In summary, we have been able to show that plaintext awareness without ROs is efficiently achievable, even though under very strong and non-standard assumptions.

References

1. M. Backes, B. Pfitzmann and M. Waidner. A composable cryptographic library with nested operations. CCS 03.
2. M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. Full version of this extended abstract. Available at `http://www-cse.ucsd.edu/users/mihir`.
3. M. Bellare, A. Boldyreva and A. Palacio. An un-instantiable random oracle model scheme for a hybrid encryption problem. EUROCRYPT '04.
4. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations among notions of security for public-key encryption schemes. CRYPTO '98.
5. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. CRYPTO '04.
6. M. Bellare and P. Rogaway. Optimal asymmetric encryption. EUROCRYPT '94.
7. D. Boneh. Simplified OAEP for the RSA and Rabin functions. CRYPTO '01.
8. M. Blum, P. Feldman and S. Micali. Non-interactive zero-knowledge and its applications. STOC 88.
9. M. Blum, P. Feldman and S. Micali. Proving security against chosen ciphertext attacks. CRYPTO '88.
10. R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. STOC 98.

11. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, Vol. 33, No. 1, 2003, pp. 167–226.
12. I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. CRYPTO '91.
13. A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. FOCS 92.
14. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, Vol. 29, 1983, pp. 198–208.
15. D. Dolev, C. Dwork, and M. Naor. Non-Malleable cryptography. *SIAM Journal on Computing*, Vol. 30, No. 2, 2000, pp. 391–437.
16. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP is secure under the RSA assumption. CRYPTO '01.
17. O. Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, Vol. 6, No. 1, 1993, pp. 21–53.
18. S. Goldwasser and Y. Taumann. On the (in)security of the Fiat-Shamir paradigm. FOCS 03.
19. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. IACR Cryptology ePrint Archive, Report 1999/009, March 1999. Available at `http://eprint.iacr.org/1999/009/`. [Revised version of [20].]
20. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. CRYPTO '98. [Preliminary version of [19].]
21. J. Herzog, M. Liskov and S. Micali. Plaintext awareness via key registration. CRYPTO '03.
22. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, 1984, pp. 270–299.
23. S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, Vol. 17, No. 2, 1988, pp. 412–426.
24. M. Naor. Cryptographic assumptions and challenges. CRYPTO '03.
25. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. STOC 90.
26. J. B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. CRYPTO '02
27. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. CRYPTO '91.
28. V. Shoup. OAEP reconsidered. *Journal of Cryptology* Vol. 15, No. 4, 2002, pp. 223–249.

OAEP 3-Round: A Generic and Secure Asymmetric Encryption Padding

Duong Hieu Phan and David Pointcheval

École normale supérieure – Dépt d'informatique,
45 rue d'Ulm, 75230 Paris Cedex 05,
France
{duong.hieu.phan, david.pointcheval}@ens.fr

Abstract. The OAEP construction is already 10 years old and well-established in many practical applications. But after some doubts about its actual security level, four years ago, the first efficient and provably IND-CCA1 secure encryption padding was formally and fully proven to achieve the expected IND-CCA2 security level, when used with any trapdoor permutation. Even if it requires the partial-domain one-wayness of the permutation, for the main application (with the RSA permutation family) this intractability assumption is equivalent to the classical (full-domain) one-wayness, but at the cost of an extra quadratic-time reduction. The security proof which was already not very tight to the RSA problem is thus much worse.

However, the practical optimality of the OAEP construction is two-fold, hence its attractivity: from the efficiency point of view because of two extra hashings only, and from the length point of view since the ciphertext has a minimal bit-length (the encoding of an image by the permutation.) But the bandwidth (or the ratio ciphertext/plaintext) is not optimal because of the randomness (required by the semantic security) and the redundancy (required by the plaintext-awareness, the sole way known to provide efficient CCA2 schemes.)

At last Asiacrypt '03, the latter intuition had been broken by exhibiting the first IND-CCA2 secure encryption schemes without redundancy, and namely without achieving plaintext-awareness, while in the random-oracle model: the OAEP 3-round construction. But this result achieved only similar practical properties as the original OAEP construction: the security relies on the partial-domain one-wayness, and needs a trapdoor permutation, which limits the application to RSA, with still a quite bad reduction.

This paper improves this result: first we show the OAEP 3-round actually relies on the (full-domain) one-wayness of the permutation (which improves the reduction), then we extend the application to a larger class of encryption primitives (including ElGamal, Paillier, etc.) The extended security result is still in the random-oracle model, and in a relaxed CCA2 model (which lies between the original one and the replayable CCA scenario.)

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 63–77, 2004.

1 Introduction

The OAEP construction [4, 12, 13] is now well-known and widely used, since it is an efficient and secure padding. However, the latter property had been recently called into question: indeed, contrarily to the widely admitted result, the security cannot be based on the sole one-wayness of the permutation [28], but the partial-domain one-wayness [12, 13]. For an application to RSA, the main trapdoor one-way permutation, the two problems are equivalent, but the security reduction is much worse than believed, because of a quadratic reduction between the two above problems.

There is also a second drawback of the OAEP construction, since its use is limited to permutations. It can definitely not apply to any function, as tried and failed on the NTRU primitive [15].

Finally, the optimality, as claimed in the name of the construction, is ambiguous and not clear: from the efficiency point of view, the extra cost for encryption and decryption is just two more hashings which is indeed quite good. But the most important optimality was certainly from the length point of view: the ciphertext is just an image by the permutation, and thus the shortest as possible. However, another important parameter is the bandwidth, or the ratio ciphertext/plaintext, which is not optimal: the construction requires a randomness over $2k$ bits for a semantic security in 2^{-k}, and redundancy over k bits for preventing chosen-ciphertext attacks (plaintext-awareness): the ciphertext is thus at least $3k$ bits as large as the plaintext.

1.1 Related Work

Right after the Shoup's remark about the security of OAEP [28], several alternatives to OAEP have been proposed: OAEP+ (by Shoup himself) and SAEP, SAEP+ (by Boneh [6]) but either the bandwidth, or the reduction cost remain pretty bad. Furthermore, their use was still limited to permutations.

About generic paddings applicable to more general encryption primitives, one had to wait five years after the OAEP proposal to see the first efficient suggestions: Fujisaki–Okamoto [10, 11] proposed the first constructions, then Pointcheval [23] suggested one, and eventually Okamoto–Pointcheval [18] introduced the most efficient construction, called REACT. However, all these proposals are far to be optimal for the ciphertext size. They indeed apply, in the random-oracle model, the general approach of symmetric and asymmetric components integration [27]: an ephemeral key is first encrypted using key-encapsulation, then this key is used on the plaintext with a symmetric encryption scheme (which is either already secure against chosen-ciphertext attacks, or made so by appending a MAC – or a tag with a random oracle, for achieving plaintext-awareness.)

Plaintext-awareness [4, 3] was indeed the essential ingredient to achieve IND-CCA2 security in the random-oracle model: it makes the simulation of the decryption oracle quite easy, by rejecting almost all the decryption queries, unless the plaintext is clearly known. But this property reduces the bandwidth since

"unnecessary" redundancy is introduced. Randomness is required for the semantic security, but this is the sole mandatory extra data for constructing a secure ciphertext. At last Asiacrypt [21], the first encryption schemes with just such a randomness, but no redundancy, has been proposed: plaintext-awareness is no longer achieved, since any ciphertext is valid and corresponds to a plaintext. But this does not exclude the IND-CCA2 security level. In that paper [21], we indeed proved that an extension of OAEP, with 3 rounds but without redundancy, provides an IND-CCA2 secure encryption scheme, with any trapdoor permutation, but again under the partial-domain one-wayness. Hence a bad security reduction.

Note 1. The classical OAEP [4] construction can be seen as a 2-round Feistel network, while our proposal [21] was a 3-round network, hence the name *OAEP 3-round*. By the way, one should notice that SAEP [6] can be seen as a 1-round Feistel network.

1.2 Achievements

In this paper, we address the two above problems: the bad security reduction of the OAEP constructions, because of the need of the intractability of the partial-domain one-wayness; and the restriction to permutations.

First, we show that, contrarily to the OAEP (2-round) construction which cannot rely on the (full-domain) one-wayness, the OAEP 3-round simply requires the (full-domain) one-wayness: because of the third round, the adversary looses any control on the r value. It is not able to make ciphertexts with the same r, without querying it.

Then, we extend the application of OAEP 3-round to a larger class of encryption primitives: it applies to any efficiently computable probabilistic injection $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$, which maps any $x \in \mathrm{E}$ into F in a probabilistic way according to the random string $\rho \in \mathrm{R}$. We need this function to be one-way: given $y \in \mathrm{F}$, it must be hard to recover $x \in \mathrm{E}$ (we do not mind about the random string ρ); this probabilistic function also needs to satisfy uniformity properties which are implied by a simple requirement: f is a bijection from $\mathrm{E} \times \mathrm{R}$ onto F. Some additional restrictions will appear in the security proof:

- we cannot really consider the CCA2 scenario, but a relaxed one denoted RCCA, which is between the usual one and the replayable CCA2 introduced last year [7] and considered enough in many applications.
- the simulation will need a decisional oracle which checks whether two elements in F have the same pre-images in E. The security result will thus be related to the well-known gap-problems [19, 18].

This extension allows *almost* optimal bandwidths for many very efficient asymmetric encryption schemes, with an IND-RCCA security level related to gap-problems (e.g. an ElGamal variant related to the Gap Diffie-Hellman problem.) Note that the application to trapdoor one-way permutations like RSA results in a much more efficient security result, and provides an IND-CCA2 encryption scheme under the sole one-wayness intractability assumption.

This paper is then organized as follows: in the next section, we review the classical security model for asymmetric encryption, and present our new CCA-variant. In section 3, we present the OAEP 3-round construction for any probabilistic injection, with some concrete applications. The security result is presented and proven in section 4.

2 Security Model

In this section, we review the security model widely admitted for asymmetric encryption. Then, we consider some relaxed CCA-variants. First, let us briefly remind that a public-key encryption scheme $\mathcal{S}$ is defined by three algorithms: the key generation algorithm $\mathcal{K}(1^k)$, which produces a pair of matching public and private keys $(\mathsf{pk}, \mathsf{sk})$; the encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$ which outputs a ciphertext c corresponding to the plaintext $m \in \mathcal{M}$, using random coins $r \in \mathcal{R}$; and the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ which outputs the plaintext m associated to the ciphertext c.

2.1 Classical Security Notions

Beyond *one-wayness*, which is the basic security level for an encryption scheme, it is now well-admitted to require *semantic security* (*a.k.a. polynomial security* or *indistinguishability of encryptions* [14], denoted IND): if the attacker has some *a priori* information about the plaintext, it should not learn more with the view of the ciphertext. More formally, this security notion requires the computational indistinguishability between two messages, chosen by the adversary, one of which has been encrypted, which one has been actually encrypted with a probability significantly better than one half: the advantage $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{S}}(\mathcal{A})$, where the adversary $\mathcal{A}$ is seen as a 2-stage Turing machine (A_1, A_2), should be negligible, where $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{S}}(\mathcal{A})$ is formally defined as

$$2 \times \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow A_1(\mathsf{pk}),\\ b \stackrel{R}{\leftarrow} \{0,1\}, c = \mathcal{E}_{\mathsf{pk}}(m_b) : A_2(m_0, m_1, s, c) = b\end{array}\right] - 1.$$

Stronger security notions have also been defined thereafter (namely the *non-malleability* [8]), but we won't deal with it since it is similar to the semantic security in several scenarios [3, 5].

On the other hand, an attacker can use many kinds of attacks, according to the available information: since we are considering asymmetric encryption, the adversary can encrypt any plaintext of its choice with the public key, hence the basic *chosen-plaintext attack*. But the strongest attack is definitely when the adversary has an unlimited access to the decryption oracle itself, *adaptive chosen-ciphertext attacks* [25], denoted CCA or CCA2 (by opposition to the earlier *lunchtime attacks* [17], denoted CCA1, where this oracle access is limited until the challenge is known.) From now, we simply use CCA instead of CCA2 since we focus on adaptive adversaries.

The strongest security notion that we now widely consider is the *semantic security against adaptive chosen-ciphertext attacks* denoted IND-CCA —where the adversary just wants to distinguish which plaintext, between two messages of its choice, had been encrypted; it can ask any query to a decryption oracle (except the challenge ciphertext).

2.2 Relaxed CCA-Security

First, at Eurocrypt '02, An *et al* [1] proposed a "generalized CCA" security notion, where the adversary is restricted not to ask, to the decryption oracle, ciphertexts which are in *relation* with the challenge ciphertext. This relation must be an equivalence relation, publicly and efficiently computable, and decryption-respecting: if two ciphertexts are in relation, they necessarily encrypt identical plaintexts. This relaxation was needed in that paper, so that extra bits in the ciphertext, which can be easily added or suppressed, should not make the scheme theoretical insecure, while its security is clearly the same from a practical point of view.

More recently, another relaxation (an extra one beyond the above one) has been proposed by Canetti *et al* [7]: informally, it extends the above relation to the (possibly non-computable) equality of plaintexts. More precisely, if the adversary asks for a ciphertext c to the decryption oracle, c is first decrypted into m. Then, if m is one of the two plaintexts output in the first stage by the adversary, the decryption oracle returns test, otherwise the actual plaintext m is returned. They called this variant the "replayable CCA" security. They explain that this security level, while clearly weaker than the usual CCA one, is enough in most of the practical applications. The classical CCA security level is indeed very strong, *too strong* for the same reasons as explained above for the first relaxation.

In this paper, we could work with the latter relaxation, the "replayable CCA" scenario. But for a simpler security proof, as well as a more precise security result (with nice corollaries for particular cases, such as the RSA one) we restrict it a little bit into the "relaxed CCA" scenario, denoted RCCA. A scheme which is secure in this scenario is trivially secure in the "replayable CCA" one, but not necessarily in the "generalized CCA" or the usual CCA scenario. The actual relations between these scenarios depend on the way the random string is split. In the formal notation of the encryption algorithm, we indeed split the randomness in two parts r and ρ: $c = \mathcal{E}_{\mathsf{pk}}(m; r, \rho)$. The encryption algorithm is thus a function from $\mathcal{M} \times \mathcal{R} \times \mathrm{R}$ into the ciphertext set. We know that for being an encryption scheme, this function must be an injection with respect to $\mathcal{M}$ (several elements in $\mathcal{M} \times \mathcal{R} \times \mathrm{R}$ can map to the same ciphertext, but all these elements must project uniquely on $\mathcal{M}$: the plaintext.) In our new relaxation, we split the randomness in $\mathcal{R} \times \mathrm{R}$ so that this function is also an injection with respect to $\mathcal{M} \times \mathcal{R}$.

Let us assume that the challenge ciphertext is $c^\star = \mathcal{E}_{\mathsf{pk}}(m^\star; r^\star, \rho^\star)$. Let us consider the ciphertext $c = \mathcal{E}_{\mathsf{pk}}(m; r, \rho)$. According to the above comment, $(m^\star, r^\star)$ and (m, r) are uniquely defined from $c^\star$ and c respectively, while $\rho^\star$ and ρ may not be unique. Upon receiving c, the *relaxed decryption oracle* first checks whether $(m^\star, r^\star) = (m, r)$ in which case it outputs test. Otherwise, it outputs m.

Definition 2 (Relaxed CCA). *In the "relaxed CCA" scenario, an adversary has an unlimited access to the relaxed decryption oracle.*

Property 3. Security in the "relaxed CCA" scenario implies security in the "replayable CCA" one.

Proof. As already noticed, this is a trivial relation, since the decryption oracle in the latter scenario can be easily simulated by the relaxed decryption oracle: if its output is test, this value is forwarded, else the returned plaintext m is compared to the output of the adversary at the end of the first stage. According to the result of the comparison, either a test-answer is also given (if $m \in \{m_0, m_1\}$), or m.

This property was just to make clear that we do not relax more the CCA security, but still keep it beyond what is clearly acceptable for practical use. Namely, note that if R is the empty set, then the RCCA scenario is exactly the usual CCA one: if f is a permutation from E onto F (the RSA case.)

3 OAEP 3-Round: A General and Efficient Padding

3.1 The Basic Primitive

Our goal is to prove that OAEP 3-round can be used with a large class of one-way functions. More precisely, we need an injective *probabilistic* trapdoor one-way function family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$ from a set E_{pk} to a set F_{pk}, respectively to the index pk: almost any encryption primitive, where the plaintext set is denoted E_{pk} and the ciphertext set is denoted F_{pk}, is fine: for any parameter pk (the public key), there exists the inverse function ψ_{sk} (where sk is the private key) which returns the pre-image in E_{pk}. An injective *probabilistic* trapdoor one-way function f from E to F is actually a function $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$, which takes as input a pair (x, ρ) and outputs $y \in \mathrm{F}$. The element x lies in E and is the important input, ρ is the random string in R which makes the function to be probabilistic. Injectivity means that for any y there is at most one x (but maybe several ρ) such that $y = f(x, \rho)$. The function g which on input y outputs x is the inverse of the probabilistic function f. Clearly, we need the function f to be efficiently computable, but the one-wayness means that computing the unique x (if it exists) such that $y = f(x, \rho)$ is intractable (unless one knows the trapdoor g.) These are the basic requirement for an asymmetric encryption primitive. But for our construction to work, we need two additional properties:

- the function $f : \mathrm{E} \times \mathrm{R} \to \mathrm{F}$ is a bijection;
- without knowing the trapdoor, it is intractable to invert f in E, even for an adversary which has access to the decisional oracle $\mathsf{Same}_f(y, y')$ which answers whether $g(y) = g(y')$.

The latter property is exactly the "gap problem" notion, which is defined by the following success probability $\mathsf{Succ}_f^{\mathsf{gap}}(t, q)$, for any adversary $\mathcal{A}$ whose

running time is limited by t, and the number of queries to the decisional oracle Same_f is upper-bounded by q:

$$\mathsf{Succ}_f^{\mathsf{gap}}(t,q) = \max_{\mathcal{A}}\{x \stackrel{R}{\leftarrow} \mathrm{E}, \rho \stackrel{R}{\leftarrow} \mathrm{R}, y = f(x,\rho) : \mathcal{A}^{\mathsf{Same}_f}(y) = x\}.$$

For a family of functions, this success probability includes the random choice of the keys in the probability space, and assumes the inputs randomly drawn from the appropriate sets, hence the notation $\mathsf{Succ}_{\varphi}^{\mathsf{gap}}(t,q)$ for a family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$.

3.2 Examples

Let us see whether the two above additional properties are restrictive or not in practice:

- The first example is clearly the RSA permutation [26], where for a given public key $\mathsf{pk} = (n,e)$, the sets are $\mathrm{E} = \mathrm{F} = \mathbb{Z}_n^\star$ and R is the empty set. Then, this is clearly an injective (but deterministic) function, which is furthermore a bijection. Because of the determinism, the decisional oracle $\mathsf{Same}(y,y')$ simply checks whether $y = y'$: the gap problem is thus the classical RSA problem.
- The goal of our extension of OAEP is to apply it to the famous ElGamal encryption [9] in a cyclic group $\mathbb{G}$ of order q, generated by g. Given a public key $\mathsf{pk} = y \in \mathbb{G}$, the sets are $\mathrm{E} = \mathbb{G}$, $\mathrm{R} = \mathbb{Z}_q$ and $\mathrm{F} = \mathbb{G} \times \mathbb{G}$: $\varphi_y(x,\rho) = (g^\rho, x \times y^\rho)$, which is a probabilistic injection from E onto F, and a bijection from $\mathrm{E} \times \mathrm{R}$ onto F. About the decisional oracle, it should check, on inputs $(a = g^\rho, b = x \times y^\rho)$ and $(a' = g^{\rho'}, b' = x' \times y^{\rho'})$, whether $x = x'$, which is equivalent to decide whether $(g, y, a'/a = g^{\rho'-\rho}, b'/b = (x'/x) \times y^{\rho'-\rho})$ is a Diffie-Hellman quadruple: the gap problem is thus the well-known Gap Diffie-Hellman problem [18, 19].
- One can easily see that the Paillier's encryption [20] also fits this formalism.

3.3 Description of OAEP 3-Round

Notations and Common Parameters. For a simpler presentation, and an easy to read analysis, we focus on the case where $\mathrm{E} = \{0,1\}^n$ (is a binary set). A similar analysis as in [21] could be performed to deal with more general sets. On the other hand, any function can be mapped into this formalism at some low cost [2].

The encryption and decryption algorithms use three hash functions: $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$ (assumed to behave like random oracles in the security analysis) where the security parameters satisfy $n = k + \ell$:

$$\mathcal{F} : \{0,1\}^k \to \{0,1\}^\ell \qquad \mathcal{G} : \{0,1\}^\ell \to \{0,1\}^k \qquad \mathcal{H} : \{0,1\}^k \to \{0,1\}^\ell.$$

The encryption scheme uses any probabilistic injection family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$, whose inverses are respectively denoted ψ_{sk}, where sk is the private key associated to the public key pk. The symbol "$\|$" denotes the bit-string concatenation and identifies $\{0,1\}^k \times \{0,1\}^\ell$ to $\{0,1\}^n$.

Encryption Algorithm. The space of the plaintexts is $\mathcal{M} = \{0,1\}^\ell$, the encryption algorithm uses random coins, from two distinct sets $r \in \mathcal{R} = \{0,1\}^k$ and $\rho \in \mathrm{R}$, and outputs a ciphertext c into F: on a plaintext $m \in \mathcal{M}$, one computes

$$s = m \oplus \mathcal{F}(r) \qquad t = r \oplus \mathcal{G}(s) \qquad u = s \oplus \mathcal{H}(t) \qquad c = \varphi_{\mathsf{pk}}(t\|u, \rho).$$

Decryption Algorithm. On a ciphertext c, one first computes $t\|u = \psi_{\mathsf{sk}}(c)$, where $t \in \{0,1\}^k$ and $u \in \{0,1\}^\ell$, and then

$$s = u \oplus \mathcal{H}(t) \qquad r = t \oplus \mathcal{G}(s) \qquad m = s \oplus \mathcal{F}(r).$$

4 Security Result

In this section, we state and prove the security of this construction. A sketch is provided in the body of the paper, the full proof can be found in the full version [22].

Theorem 4. *Let $\mathcal{A}$ be an* IND-RCCA *adversary against the OAEP 3-round construction with any trapdoor one-way probabilistic function family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$, within time τ. Let us assume that after q_f, q_g, q_h and q_d queries to the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$, and the decryption oracle respectively, its advantage $\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathsf{oaep\text{-}3}}(\tau)$ is greater than ε. Then, $\mathsf{Succ}^{\mathsf{gap}}_{\varphi}(\tau', q_d(q_g q_h + q_d))$ is upper-bounded by*

$$\frac{\varepsilon}{2} - q_d^2 \times \left(\frac{1}{2^\ell} + \frac{6}{2^k}\right) - (4q_d + 1) \times \left(\frac{q_g}{2^\ell} + \frac{q_f}{2^k}\right) - q_d \times \frac{q_f + 1}{2^k},$$

with $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_d^2 T_{\mathsf{Same}} + (q_d + 1)q_g q_h (T_\varphi + T_{\mathsf{Same}})$, where T_φ is the time complexity for evaluating any function φ_{pk}, T_{Same} is the time for the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ to give its answer, and T_{lu} is the time complexity for a look up in a list.

4.1 Trapdoor Permutations

Before proving this general result, let us consider the particular case where φ_{pk} is a permutation from E onto F (*i.e.*, a deterministic function.) The general result has indeed several drawbacks:

- the reduction cost introduces a cubic factor $q_d q_g q_h$ which implies larger keys for achieving a similar security level as for some other constructions;
- the security relies on a gap problem, which is a strong assumption in many cases;
- and we cannot achieve the usual IND-CCA security level.

These drawbacks are acceptable as the price of generality: this becomes one of the best padding for ElGamal or Paillier strongly secure variants. However,

for trapdoor permutations, such as RSA, several OAEP variants achieve much better efficiency.

But one should interpret the above result in this particular case: first, the gap-problem becomes the classical one-wayness, since the decisional oracle is simply the equality test; Furthermore, the RCCA scenario becomes the classical CCA one; Finally, because of the determinism of the permutation, with proper bookkeeping, one can avoid the cubic factor, and fall back to the usual quadratic factor $q_g q_h$, as for any OAEP-like constructions (OAEP+, SAEP and SAEP+). Then, one can claim a much better security result:

Theorem 5. *Let $\mathcal{A}$ be an* IND-CCA *adversary against the OAEP 3-round construction with a trapdoor one-way permutation family $(\varphi_{\mathsf{pk}})_{\mathsf{pk}}$, within time τ. Let us assume that after q_f, q_g, q_h and q_d queries to the random oracles $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$, and the decryption oracle respectively, its advantage $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{oaep\text{-}3}}(\tau)$ is greater than ε. Then, $\mathsf{Succ}^{\mathsf{ow}}_{\varphi}(\tau')$ is upper-bounded by*

$$\frac{\varepsilon}{2} - q_d^2 \times \left(\frac{1}{2^\ell} + \frac{6}{2^k}\right) - (4q_d + 1) \times \left(\frac{q_g}{2^\ell} + \frac{q_f}{2^k}\right) - q_d \times \frac{q_f + 1}{2^k},$$

with $\tau' \leq \tau + (q_f + q_g + q_h + q_d)T_{lu} + q_g q_h T_\varphi$, where T_φ is the time complexity for evaluating any function φ_{pk} and T_{lu} is the time complexity for a look up in a list.

4.2 Sketch of the Proof

The proof is very similar to the one in [21], but the larger class (injective probabilistic functions), and the improved security result (relative to the one-wayness) make some points more intricate: for a permutation f, each value x maps to a unique image $y = f(x)$; whereas for a function f, each value x maps to several images $y = f(x, \rho)$, according to the random string ρ. Consequently, when used as an asymmetric encryption primitive, the adversary may have the ability to build another y' whose pre-image is identical to the one of y: $x = g(y) = g(y')$. Such a query to the decryption oracle is not excluded in the CCA scenario, while we may not be able to either detect or answer. Hence the relaxed version of chosen-ciphertext security, and the decisional oracle Same_f: the latter helps to detect ciphertexts with identical pre-images, the relaxed scenario gives the ability to answer test in this case. Granted the decisional oracle Same_f, we can also detect whether a decryption query c has the same pre-image as a previous decryption query c' in which case we output the same plaintext. If it is a really new ciphertext, by using again the decisional oracle Same_f, we can check whether s and t have both been asked to $\mathcal{G}$ and $\mathcal{H}$, respectively, which immediately leads to the plaintext m. In the negative case, a random plaintext can be safely returned.

4.3 More Details

The full proof can be found in the full version [22], but here are the main steps, since the proof goes by successive games in order to show that the above decryp-

tion simulation is almost indistinguishable for the adversary. Then, a successful IND-RCCA adversary can be easily used for inverting the one-way function.

$\mathbf{G}_0$: We first start from the real IND-RCCA attack game.

$\mathbf{G}_1$–$\mathbf{G}_2$: We then simulate the view of the adversary, first, as usual with lists for the random oracles and the decryption oracle (see figures 1 and 2.) We then modify the generation of the challenge ciphertext, using a random mask $f^\star$, totally independent of the view of the adversary: the advantage of any adversary is then clearly zero. The plaintext is indeed unconditionally hidden.

The only way for any adversary to detect this simulation is to ask $\mathcal{F}(r^\star)$ and then detect that the answer differs from any possible $f^\star$. We are thus interested in this event, termed AskF, which denotes the event that $r^\star$ is asked to $\mathcal{F}$.

The main difference with the OAEP 2-round construction, as shown by Shoup with his counter-example [28], is that here an adversary cannot make another ciphertext with the same r as $r^\star$, in the challenge ciphertext, but either by chance, or if it had asked for *both* $\mathcal{G}(s^\star)$ and $\mathcal{H}(t^\star)$. We now try to show this fact.

$\mathbf{G}_3$–$\mathbf{G}_8$: We thus modify the decryption process so that it makes no new query to $\mathcal{G}$ and $\mathcal{H}$. The sequence of games leads to the following new rules:

▶**Rule** Decrypt-noT$^{(8)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.

▶**Rule** Decrypt-TnoS$^{(8)}$

> Choose $m \xleftarrow{R} \{0,1\}^\ell$.

▶**Rule** Decrypt-TSnoR$^{(8)}$

> If $s = s^\star$ but $s^\star$ has not been directly asked by the adversary yet: $m \xleftarrow{R} \{0,1\}^\ell$.
> Else, one chooses $m \xleftarrow{R} \{0,1\}^\ell$, computes $f = m \oplus s$ and adds (r, f) in $\mathcal{F}$-List.

▶**Rule** EvalGAdd$^{(8)}$

> For each $(t, h) \in \mathcal{H}$-List and each $(m, c) \in \mathcal{D}$-List, choose an arbitrary random $\rho \in \mathrm{R}$ and ask for $(c, c' = \varphi_{\mathsf{pk}}(t \| h \oplus s, \rho))$ to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. If the record is found (the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$ answers "yes"), we compute $r = t \oplus g$ and $f = m \oplus s$, and finally add (r, f) in $\mathcal{F}$-List.

Some bad cases may appear, which make our simulation to fail. But they are very unlikely, we thus can safely cancel executions, applying the following rule

$\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ Oracles

Query $\mathcal{F}(r)$: if a record (r, f) appears in $\mathcal{F}$-List, the answer is f.
Otherwise the answer f is chosen randomly in $\{0,1\}^k$ and the record (r, f) is added in $\mathcal{F}$-List.

Query $\mathcal{G}(s)$: if a record (s, g) appears in $\mathcal{G}$-List, the answer is g.
Otherwise the answer g is chosen randomly in $\{0,1\}^\ell$ and the record (s, g) is added in $\mathcal{G}$-List.

▶**Rule** EvalGAdd$^{(1)}$

| Do nothing `% To be defined later`

Query $\mathcal{H}(t)$: if a record (t, h) appears in $\mathcal{H}$-List, the answer is h.
Otherwise the answer h is chosen randomly in $\{0,1\}^k$ and the record (t, h) is added in $\mathcal{H}$-List.

$\mathcal{D}$ Oracle

Query $\mathcal{D}_{\mathsf{sk}}(c)$: first, if we are in the second stage (the challenge $c^\star$ as been defined), ask for $(c, c^\star)$ to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. In case of positive decision, the answer is test.
Else, for each (m', c') in $\mathcal{D}$-List, ask for (c, c') to the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$. In case of a positive decision, the answer is the corresponding'm'.
Otherwise, the answer m is defined according to the following rules:

▶**Rule** Decrypt-Init$^{(1)}$

| Compute $t \| u = \psi_{\mathsf{sk}}(c)$;

Look up for $(t, h) \in \mathcal{H}$-List:

- if the record is found, compute $s = u \oplus h$.
  Look up for $(s, g) \in \mathcal{G}$-List:
  - if the record is found, compute $r = t \oplus g$.
    Look up for $(r, f) \in \mathcal{F}$-List:
    - if the record is found
      ▶**Rule** Decrypt-TSR$^{(1)}$
      | $h = \mathcal{H}(t)$,
      | $s = u \oplus h, \quad g = \mathcal{G}(s)$,
      | $r = t \oplus g, \quad f = \mathcal{F}(r)$,
      | $m = s \oplus f$.
    - else
      ▶**Rule** Decrypt-TSnoR$^{(1)}$
      | same as rule Decrypt-TSR$^{(1)}$.
  - else
    ▶**Rule** Decrypt-TnoS$^{(1)}$
    | same as rule Decrypt-TSR$^{(1)}$.
- else
  ▶**Rule** Decrypt-noT$^{(1)}$
  | same as rule Decrypt-TSR$^{(1)}$.

Answer m and add (m, c) to $\mathcal{D}$-List.

Fig. 1. Formal Simulation of the IND-RCCA Game: Oracles

Challenger

For two messages (m_0, m_1), flip a coin b and set $m^\star = m_b$, choose randomly $r^\star$ then answer $c^\star$ where

►**Rule** $\mathsf{Chal}^{(1)}$

$f^\star = \mathcal{F}(r^\star), \quad s^\star = m^\star \oplus f^\star,$
$g^\star = \mathcal{G}(s^\star), \quad t^\star = r^\star \oplus g^\star,$
$h^\star = \mathcal{H}(t^\star), \quad u^\star = s^\star \oplus h^\star.$

►**Rule** $\mathsf{ChalC}^{(1)}$

and $c^\star = \varphi_{\mathsf{pk}}(t^\star \| u^\star, \rho^\star)$, for random string $\rho^\star$.

Fig. 2. Formal Simulation of the IND-RCCA Game: Challenger

►**Rule** $\mathsf{Abort}^{(1)}$

Abort and output a random bit:

- If $s^\star$ has been asked to $\mathcal{G}$ by the adversary, while the latter did not ask for $\mathcal{H}(t^\star)$.
- If a Decrypt-TSR/Decrypt-TSnoR rule has been applied with $t = t^\star$, while $\mathcal{H}(t^\star)$ had not been asked by the adversary yet.
- If a Decrypt-TSR rule has been applied with $s = s^\star$, while $\mathcal{G}(s^\star)$ had not been asked by the adversary yet.

The remaining bad case (termed AskGHA) is if both $s^\star$ and $t^\star$ have been asked to $\mathcal{G}$ and $\mathcal{H}$ by the adversary. Such a case helps the adversary to distinguish our simulation. On the other hand, this case helps to invert φ_{pk}.

G$_9$: With these new rules for decryption, the simulation of the decryption oracle does not use at all the queries previously asked to $\mathcal{G}$ and $\mathcal{H}$ by the generation of the challenge, but just the queries directly asked by the adversary, which are available to the simulator (we remind that we are in the random-oracle model.) One can thus make $g^\star$ and $h^\star$ to be values independent to the view of the adversary:

►**Rule** $\mathsf{Chal}^{(9)}$

The two values $r^+ \xleftarrow{R} \{0,1\}^k$ and $f^+ \xleftarrow{R} \{0,1\}^\ell$ are given, as well as $g^+ \xleftarrow{R} \{0,1\}^k$ and $h^+ \xleftarrow{R} \{0,1\}^\ell$ then $r^\star = r^+$, $f^\star = f^+$, $s^\star = m^\star \oplus f^+$, $g^\star = g^+$, $t^\star = r^+ \oplus g^\star$, $h^\star = h^+$ and $u^\star = s^\star \oplus h^\star$.

And then the decryption oracle can be simply replaced by the classical plaintext-extractor which looks up in the lists $\mathcal{G}$-List and $\mathcal{H}$-List (which only contain the queries directly asked by the adversary) to obtain the values (s, g) and (t, h) which match with $c = \varphi_{\mathsf{pk}}(t \| s \oplus h, \rho)$, using the decisional oracle $\mathsf{Same}_{\varphi_{\mathsf{pk}}}$, but without using anymore ψ_{sk}. In case of failure, one answers a random plaintext m.

> We simply conclude, since our reduction does not use any oracle, but can answer any query of the adversary, in an indistinguishable way, unless the bad case AskGHA happens: in which case we have inverted φ_{sk}.

The time complexity of one simulation is thus upper-bounded by $q_g q_h \times (T_\varphi + T_{\mathsf{Same}})$, where T_φ is the time to evaluate one function in the φ family, and T_{Same} the time for the decisional oracle, plus the initial look up in the $\mathcal{D}$-List: $T_{lu} + q_d T_{\mathsf{Same}}$. Thus the global running time is bounded by (including all the list look up):

$$\tau' \leq \tau + q_d q_g q_h \times (T_\varphi + T_{\mathsf{Same}}) + q_d^2 \times T_{\mathsf{Same}} + (q_f + q_g + q_h + q_d) \times T_{lu}.$$

In the particular case where φ_{pk} is a permutation from E onto F (a deterministic one), one can improve it, using an extra list of size $q_g q_h$, which stores all the tuples $(s, g = \mathcal{G}(s), t, h = \mathcal{H}(t), c' = \varphi_{\mathsf{pk}}(t \| s \oplus h))$. The time complexity then falls down to $\tau + q_g q_h \times T_\varphi + (q_f + q_g + q_h + q_d) \times T_{lu}$.

5 Conclusion

All the OAEP variants [28, 6] applied to RSA, with general exponents (*i.e.*, not Rabin nor $e = 3$) admit, in the best cases, reductions to the RSA problem with a quadratic loss in time complexity [24] – the original OAEP is even worst because of the reduction to the partial-domain case, which requires a more time consuming reduction to the full-domain RSA problem. Furthermore, for a security level in 2^{-k}, a randomness of $2k$ bits is required, plus a redundancy of k bits.

In this paper, we show that the variant of OAEP with 3 rounds admits a reduction as efficient as the best OAEP variants (to the full-domain RSA, when applied to the RSA family) without having to add redundancy: one can thus earn k bits. But this is not the main advantage.

Considering any criteria, OAEP with 3 rounds is at least as good as all the other OAEP variants, but from a more practical point of view

- since no redundancy is required, implementation becomes easier, namely for the decryption process [16];
- it applies to more general families than just (partial-domain) one-way trapdoor permutations, but to any probabilistic trapdoor one-way function. It is thus safer to use it with a new primitive [15].

As a conclusion, OAEP with 3 round is definitely the most generic and the simplest padding to use with almost all the encryption primitives.

References

1. J. H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signatures and Encryption. In *Eurocrypt '02*, LNCS 2332, pages 83–107. Springer-Verlag, Berlin, 2002.

2. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Asiacrypt '01*, LNCS 2248, pages 566–582. Springer-Verlag, Berlin, 2001.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.
4. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
5. M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Crypto '99*, LNCS 1666, pages 519–536. Springer-Verlag, Berlin, 1999.
6. D. Boneh. Simplified OAEP for the RSA and Rabin Functions. In *Crypto '01*, LNCS 2139, pages 275–291. Springer-Verlag, Berlin, 2001.
7. R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing Chosen-Ciphertext Security. In *Crypto '03*, LNCS 2729, pages 565–582. Springer-Verlag, Berlin, 2003.
8. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
9. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
10. E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.
11. E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E83-A(1):24–32, January 2000.
12. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '01*, LNCS 2139, pages 260–274. Springer-Verlag, Berlin, 2001.
13. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. *Journal of Cryptology*, 17(2):81–104, 2004.
14. S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
15. E. Jaulmes and A. Joux. A Chosen Ciphertext Attack on NTRU. In *Crypto '00*, LNCS 1880, pages 20–35. Springer-Verlag, Berlin, 2000.
16. J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1. In *Crypto '01*, LNCS 2139, pages 230–238. Springer-Verlag, Berlin, 2001.
17. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437. ACM Press, New York, 1990.
18. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *CT – RSA '01*, LNCS 2020, pages 159–175. Springer-Verlag, Berlin, 2001.
19. T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *PKC '01*, LNCS 1992. Springer-Verlag, Berlin, 2001.
20. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, Berlin, 1999.

21. D. H. Phan and D. Pointcheval. Chosen-Ciphertext Security without Redundancy. In *Asiacrypt '03*, LNCS 2894, pages 1–18. Springer-Verlag, Berlin, 2003. Full version available from `http://www.di.ens.fr/users/pointche/`.
22. D. H. Phan and D. Pointcheval. OAEP 3-Round: A Generic and Secure Asymmetric Encryption Padding. In *Asiacrypt '04*, LNCS. Springer-Verlag, Berlin, 2004. Full version available from `http://www.di.ens.fr/users/pointche/`.
23. D. Pointcheval. Chosen-Ciphertext Security for any One-Way Cryptosystem. In *PKC '00*, LNCS 1751, pages 129–146. Springer-Verlag, Berlin, 2000.
24. D. Pointcheval. How to Encrypt Properly with RSA. *CryptoBytes*, 5(1):10–19, winter/spring 2002.
25. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992.
26. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
27. V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption, december 2001. ISO/IEC JTC 1/SC27.
28. V. Shoup. OAEP Reconsidered. In *Crypto '01*, LNCS 2139, pages 239–259. Springer-Verlag, Berlin, 2001.

Stream Ciphers: Dead or Alive?

Adi Shamir

Weizmann Institute of Science, Israel
adi.shamir@weizmann.ac.il

Secret key cryptography was traditionally divided into block ciphers and stream ciphers, but over the last 30 years the balance had steadily shifted, and today stream ciphers have become an endangered species. In this talk I'll survey the current state of the art in stream ciphers: who needs them, who uses them, how they are attacked, and how they can be protected by new types of constructions.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, p. 78, 2004.

On the Generalized Linear Equivalence of Functions Over Finite Fields

Luca Breveglieri, Alessandra Cherubini, and Marco Macchetti

Politecnico di Milano, Milan, Italy
{brevegli, macchett}@elet.polimi.it
aleche@mate.polimi.it

Abstract. In this paper we introduce the concept of generalized linear equivalence between functions defined over finite fields; this can be seen as an extension of the classical criterion of linear equivalence, and it is obtained by means of a particular geometric representation of the functions. After giving the basic definitions, we prove that the known equivalence relations can be seen as particular cases of the proposed generalized relationship and that there exist functions that are generally linearly equivalent but are not such in the classical theory. We also prove that the distributions of values in the Difference Distribution Table (DDT) and in the Linear Approximation Table (LAT) are invariants of the new transformation; this gives us the possibility to find some Almost Perfect Nonlinear (APN) functions that are not linearly equivalent (in the classical sense) to power functions, and to treat them accordingly to the new formulation of the equivalence criterion. This answers a question posed in [8].

Keywords: Boolean functions, linear equivalence, differential cryptanalysis, linear cryptanalysis, APN functions, S-boxes.

1 Introduction

The design criteria for symmetric key algorithms can be traced back to the work of Shannon [18], where the concepts of *confusion* and *diffusion* are formalized. Today, a significant number of block ciphers are built by alternating nonlinear substitution layers with linear diffusion layers, in the so called Substitution-Permutation Networks (SPNs). It has been proved that the usage of sufficiently strong substitution functions, or S-boxes, leads to construction of strong block ciphers, see for instance the Wide-Trail design technique [6]. The strength of each S-box is often measured by means of the resistance to differential [4] and linear [14],[3] cryptanalysis.

For a given function $f : F_{p^m} \rightarrow F_{p^n}$ with p prime and $m, n \geq 1$ we can build the DDT by computing the number $\delta_f(a, b)$ of solutions x of the equation

$$f(x + a) - f(x) = b \qquad a \in F_{p^m}, b \in F_{p^n} \tag{1}$$

The lower the value of the maximum entry in the table, $\Delta_f = \max_{a \neq 0, b}(\delta_f(a, b))$, the more robust function f is versus differential cryptanalysis.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 79–91, 2004.

In a similar way, we can construct the LAT of f by counting the number $\lambda_f(a,b)$ of solutions x of the equation

$$a \bullet x = b \bullet f(x) \qquad a \in {F_p}^m, b \in {F_p}^n \tag{2}$$

where the inner product is indicated with $\bullet$ and gives a value in F_p. The robustness to linear cryptanalysis is measured with the maximum value $\Lambda_f = \max_{a,b\neq 0}(||\lambda_f(a,b) - p^{m-1}||)$. Good S-boxes have both small Λ_f and Δ_f values, and usually have a complex algebraic expression; most of the results focus on the case $p = 2$ which is of interest for practical applications.

Two functions are said to be equivalent if they differ by a group operation on the input or output variables; Lorens [12] and Harrison [10],[11] have considered the special case of invertible n-bit vectorial Boolean functions and have derived the exact number of equivalence classes (along with asymptotic estimates) for $n \leq 5$ when different transformations such as complementation, permutation, linear and affine transformations are applied on the input and output bits. Similar results can be found in [1],[13] regarding the case of Boolean functions with 5 and 6 input bits and an asymptotic estimate for the number of equivalence classes of Boolean functions under the transformation $g(x) = f(Ax + b) + L(x)$ (where L is a linear transformation) can be found in [7]. We can say that, in the most general case of classical linear equivalence, two functions $f, g : {F_p}^m \rightarrow {F_p}^n$ are linearly equivalent if there are two non-singular matrices A, B and a matrix C over F_p such that

$$g(x) = Bf(Ax) + Cx \tag{3}$$

The fact that two functions belong to the same equivalence class is rather important from a cryptanalytic point of view; it is well known that the distributions of values in the DDT and LAT as defined by (1) and (2) are invariant under the transformation (3). It is also true that if f is invertible, then $g(x) = f^{-1}(x)$ has the same cryptographic robustness of f [15],[2]. This has motivated the fact that the inverse of a function is also quoted as being *equivalent* to it [8]; while this is understandable from the point of view of cryptography[1], there is not formal consistency in the theory, because clearly the operation of inversion is very different from the transformation in (3).

To fill this gap, in Sect. 2 we propose a re-definition of the criterion of linear equivalence that permits us to treat the classical case of linear equivalence and the inversion operation with a unified approach. The criterion of generalized linear equivalence can be applied to functions over finite fields, provided that they are represented geometrically by set of vectors in an appropriate linear space S. The set of vectors representing function f is denoted with $\mathcal{F}$ and called the implicit embedding of f (in the space S); the implicit embedding contains the information of the truth-table of the function.

[1] A significant example is that of power functions over F_{p^n}, as it happens that the inverse of a power monomial is again a power monomial, generally belonging to a different cyclotomic coset.

Two functions f and g are said to be *generally linearly equivalent* if $\mathcal{G}$ can be obtained from $\mathcal{F}$ with an invertible linear transformation T that acts on the space S, i.e. $\mathcal{G} = T(\mathcal{F})$. We show that there exist couples of functions that are generally linearly equivalent but are not correlated in the classical theory of equivalence; thus the proposed criterion is in fact an *extension* of the classical concept of equivalence.

In Sect. 3 we prove that the cryptographic robustness of a function versus differential and linear cryptanalysis is invariant under the transformations considered in the framework of generalized linear equivalence, completing the proof for the classical case.

In Sect. 4 we apply the criterion to power functions; we give an example of an APN function that is not classically linearly equivalent to any power monomial, but is easily obtainable using the generalized equivalence criterion. This answers a question posed in [8].

Sect. 5 concludes the paper.

2 Extension of the Linear Equivalence Relation

2.1 A Geometric Representation

Let us consider a completely specified function $f : {F_p}^m \to {F_p}^n$, with no restrictions on the values of m, n. There are different possible representations for the *object* f; we are particularly interested in the truth table of f, that lists the output values of f associated with the corresponding (actually, all the possible) input values. If we view the truth table as a matrix, there are p^m rows, $m + n$ columns, and each entry belongs to the field F_p. The ordering of the rows is not important, in the sense that two truth tables that contain the same rows in different order specify the same function and thus can be considered as the same truth table.

We can build a geometric representation of the function in the following way. Let S be a linear space of dimension $k = m + n$, the elements (or vectors) of which are defined over the finite field $F_{p^{m+n}}$. Such vectors can thus be conceived both as elements of the extension field $F_{p^{m+n}}$ and as vectors of the space S, each vector consisting of $m + n$ components over the basic field F_p. Denote by $+$ and $\cdot$ (or nothing) the addition and multiplication of elements in $F_{p^{m+n}}$; by extension $+$ denotes vector addition in S, and $\cdot$ (or nothing) denotes scalar-vector multiplication in S. Consider the set $\mathcal{F}$ of p^m vectors in this space formed by the rows of the truth-table of f, i.e. the concatenation of the input vectors with the corresponding output vectors of f. Formally,

$$\mathcal{F} = \{(x|f(x)), x \in {F_p}^m, f(x) \in {F_p}^n\} \tag{4}$$

where with $|$ we indicate the simple concatenation of two vectors with components over F_p. Each vector of the set represents one complete row of the truth table and thus the same information is contained in both representations; since the vectors are not ordered, we can see that different orderings of the rows of

the truth table, as we would write it down on a piece of paper, actually identify the same set of vectors, i.e. the same geometric entity. Two different functions have different information in the truth table and therefore they are represented with different set of vectors. We conclude that each function f can be unambiguously represented with a particular set of vectors $\mathcal{F}$, which we call its *implicit embedding* (in the linear space S).

A natural question is when a given set of vectors actually represents a function. The following three conditions must be satisfied:

1. The set must have cardinality p^m for some positive m. In fact, we consider completely specified functions, and the number of rows in the truth table must be p^m if the function has m input variables (belonging to F_p).
2. The dimension of the vectors must be $m + n$ for some positive n, i.e. the function must have at least one output variable.
3. If we consider the first m components of all the vectors, we must find all possible configurations once and only once. This is because there cannot be a missing configuration (there would be a missing row in the truth table, but the function must be completely specified) and there cannot be multiple instances of the same configuration (there would be some missing configurations because the cardinality of the set is p^m).

We can see that there are sets of vectors which do not represent functions; thus the representation defines a relation from the set of all functions $f : {F_p}^m \rightarrow {F_p}^n$ to the set of all the sets of vectors in the space ${F_p}^{m+n}$ that is one-to-one but not onto.

2.2 Linear Transformations Over S

We have seen that all the information contained in the function specification (truth table) is contained also in its geometric counterpart; the *shape* of the set of vectors is thus a unique property of the represented function. If we apply a linear transformation of coordinates to the space that is invertible, the information contained in the set of vectors is not changed; instead, we change the way we are looking at every geometric object (curves, hyperplanes, etc...) that is contained in the linear space S, including the function represented as a set of vectors.

Every invertible linear transformation over the whole space is governed by a non-singular $(m + n) \times (m + n)$ matrix T over F_p. The non-singularity of the matrix assures that we do not loose information while transforming the coordinates, and also that the transformation has always an inverse.

Each vector of the implicit embedding of f is transformed into a new one, but the essential shape of the configuration is invariant (we shall study the cryptographic invariants of f in Sect. 3). Thus if one vector set is obtained from another one by a change of basis governed by matrix T, then the two corresponding functions are said to be *generally linearly equivalent*.

Definition 1. *Two functions* $f, g : {F_p}^m \to {F_p}^n$ *are called generally linearly equivalent*[2] *if and only if the implicit embedding* $\mathcal{G}$ *of* g *can be obtained from the implicit embedding* $\mathcal{F}$ *of* f *with*

$$\mathcal{G} = T(\mathcal{F})$$

where T *is an invertible linear transformation over the space* ${F_p}^{m+n}$ *corresponding to the non-singular matrix* T*.*

We can treat the classical notion of linear equivalence as a particular case of the generalized linear equivalence. We first consider the case $m > n$. Then:

1. If matrix T of the change of basis is defined as

$$T = \left(\begin{array}{c|c} A & 0 \\ \hline 0 & B \end{array}\right)$$

where A is a non-singular $m \times m$ matrix and B is a non-singular $n \times n$ matrix over F_p, then:
   - Matrix T is non-singular
   - If we examine the transformed set of vectors, we see that it still describes a function g which has the following relation with function f:

$$g(x) = Bf(A^{-1}x)$$

The relation is easy to prove, once we remember that the first m components of the vectors in the implicit embeddings of f and g represent the input values, and the last n components represent the corresponding output values. Thus carrying out the matrix-vector multiplication at block level, we obtain $y = Ax$ and $g(y) = Bf(x)$ and substituting we have the above relation between f and g. Obviously, if $A = I_m$ (the $m \times m$ identity matrix over F_p) and $B = I_n$ (the $n \times n$ identity matrix over F_p) we obtain again f because the global transformation is the identity.
2. If matrix T of the change of basis is defined as

$$T = \left(\begin{array}{c|c} A & 0 \\ \hline C & B \end{array}\right)$$

where A is a non-singular $m \times m$ matrix, B is an $n \times n$ non-singular matrix and $C \; / = 0$ is an $n \times m$ matrix over F_p, then
   - Matrix T is non-singular.
   - If we examine the transformed set of vectors, we see that it still describes a function g which has the following relation with function f:

$$g(x) = Bf(A^{-1}x) + CA^{-1}x$$

[2] We observe that the concept of generalized affine equivalence could be defined along the same line, to remove the artificial restriction that if two S-boxes are equivalent and one maps 0 to 0, the other must also.

Thus we obtain all the functions that are linearly equivalent (in the classical sense) to f, according to (3).

3. If matrix T of the change of basis is defined as

$$T = \left(\begin{array}{c|c} A & D \\ \hline C & B \end{array}\right)$$

where $A \neq 0$ is a $m \times m$ matrix, B is an $n \times n$ matrix, C is an $n \times m$ matrix and $D \neq 0$ is a $m \times n$ matrix over F_p, then if matrix T is non-singular, we can examine the transformed set of vectors. Two possibilities arise:

(a) It may happen that the transformed set does not describe a function anymore because the non-singularity of T does not always imply that condition 3 in Sect. 2.1 is satisfied.

(b) The transformed set satisfies condition 3 in Sect. 2.1, and function g is generally equivalent to function f, although it is not obtainable within the classical theory. The link between g and f is non-trivial: the output vectors of g (the last n components of the transformed vectors) are obtained by mixing information contained in the input vectors of f by means of matrix C and information contained in the output vectors of f by means of matrix B. The difference from the previous case is that the same thing happens also to the input vectors of g by means of matrices A and D. As a result it is not possible to express the relation between f and g with a simple equation as before; nonetheless the two functions are generally linearly equivalent. The truth tables of the two functions can be expressed as:

$$f : x \rightarrow f(x)$$

$$g : Ax + Df(x) \rightarrow Cx + Bf(x)$$

Note that the reason why the transformed vector set is still representing a function is simply that the function $h : x \rightarrow Ax + Df(x)$ is a permutation over ${F_p}^m$.

If $m = n$ holds, the above cases are still valid; however, if it happens that f is invertible, more cases can be considered. In particular:

4. If matrix T of the change of basis is defined as

$$T = \left(\begin{array}{c|c} 0 & D \\ \hline C & 0 \end{array}\right)$$

where C, D are non-singular $m \times m$ matrices over F_p, then:

- Matrix T is non-singular
- If we examine the transformed set of vectors, we see that it still describes a function g, and it holds that:

$$g(x) = Cf^{-1}(D^{-1}x)$$

This happens because the blocks C, D swap the input and the output parts of all the vectors belonging to the implicit embedding of f in the implicit embedding of g. Obviously, if $C = D = I_m$ we obtain the inverse of f. We have thus reduced the operation of inversion of a function to a linear transformation over the space where the implicit embedding of the function is defined. This is surely a convenient feature of the proposed formulation.

5. If matrix T of the change of basis is defined as

$$T = \left(\begin{array}{c|c} 0 & D \\ \hline C & B \end{array}\right)$$

where $B \neq 0$ is an $m \times m$ matrix and C, D are non-singular $m \times m$ matrices over F_p, then
   - Matrix T is non-singular
   - The relation between f and g is the following:

$$g(x) = Cf^{-1}(D^{-1}x) + BD^{-1}x$$

i.e. we obtain all the functions that are linearly equivalent (in the classical sense) to the inverse of f.

Last, we consider the case $m < n$. The following considerations can be made:

- Cases 1,2,3 are still valid; however in the conditions for case 3 we should substitute $A \neq 0$ with $B \neq 0$.
- Case 4 is not applicable.
- Under some assumptions for matrix D and function f, case 5 can still be valid. However, we loose the relationship with the inverse transformation (which is not defined when the numbers of input and output variables are different); moreover this case in fact becomes a special instance of 3, thus it does not deserve a separate mention.

In all the remaining cases, either it can be proved that matrix T is singular, or the transformed set of vectors cannot represent a function, so we have no interest in examining them.

In the following, an example of a family of functions belonging to case 3 for $m > n$ is given.

Example 1. The family of functions $f : {F_p}^{2m} \rightarrow {F_p}^m$ with p prime and $m \geq 1$ is given, where the input vector x and $f(x)$ are defined as:

$$x = (x_1)|(x_2) \quad x \in {F_p}^{2m} \quad x_1, x_2 \in F_{p^m}$$

$$f(x) = f((x_1)|(x_2)) = x_1^{-1} + x_2^{-1}$$

where we indicate with | the simple concatenation of two vectors (actually, x_1 and x_2 represented as vectors over F_p are concatenated). When function f is transformed into function g using a suitable matrix T, we can simply write

$g = T(f)$ as the same equation holds for the implicit embeddings of the two functions. The implicit embeddings of f and g can be *visually* represented, along with a block decomposition of T; we write explicitly:

$$g = T(f) = \left(\begin{array}{cc|c} I_m & 0 & 0 \\ 0 & 0 & I_m \\ \hline I_m & I_m & I_m \end{array}\right) \bullet \begin{pmatrix} x_1 \\ x_2 \\ x_1^{-1} + x_2^{-1} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_1^{-1} + x_2^{-1} \\ x_1^{-1} + x_2^{-1} + x_1 + x_2 \end{pmatrix}$$

It can be observed that matrix T is non-singular and that the transformed set of vectors still represents a function, because the input part $(x_1)|(x_1^{-1} + x_2^{-1})$ is still a permutation over ${F_p}^{2m}$ when x_1, x_2 vary over F_{p^m} (i.e. all the possible input values for g are specified in the implicit embedding). We make here the underlying assumption that, with an abuse of notation, $0^{-1} = 0$.

By Def. 1 the two functions f, g are generally linearly equivalent, although there is no way to express the link using the classical theory of equivalence, since every function that is classically linearly equivalent to f is obtained with a matrix T characterized by a null upper-right block. The truth-table of g is written in compact form as

$$((x_1)|(x_1^{-1} + x_2^{-1})) \rightarrow (x_1^{-1} + x_2^{-1} + x_1 + x_2)$$

Any property that is invariant under the considered transformation is common between f and g. In the next Section we present a result on the invariance of cryptographic robustness.

3 Cryptographic Robustness of Generally Equivalent Functions

We start by recalling a fundamental result of the classical theory [15], [2]:

Theorem 1. *Given two functions f and g, if they are linearly equivalent i.e. if there exist two non-singular matrices A, B such that*

$$g(x) = Bf(Ax) \tag{5}$$

then the distributions of the values in DDTs and LATs of f and g are equal.

Corollary 1. *As a consequence of Theorem 1, we have that $\Delta_f = \Delta_g$ and $\Lambda_f = \Lambda_g$.*

It is also known that the same parameters are conserved when we consider the inverse of a function (the DDTs and LATs are merely transposed), or when we add a linear combination of the input variables of the function directly to its output variables [9].

Since we proved that these relations are particular occurrences of the generalized linear equivalence, it is therefore natural to ask whether the same parameters are also invariant in the general case. We answer with the following theorem.

Theorem 2. *Given two functions $f, g : {F_p}^m \to {F_p}^n$ and a non-singular $(m+n) \times (m+n)$ matrix T over F_p, if $g = T(f)$ then the distributions of values in the linear and differential tables of f and g are equal.*

Proof. We first prove the relation regarding the DDTs of f and g.

A cell of the DDT of f located in the i-th row and j-th column contains the number of the input vector couples (x, y) such that $y = x+i$ and $f(y) = f(x)+j$, according to (1).

Thus, if we consider the geometric representation for function f we have that the cell contains the number of vector couples (w, z) belonging to the implicit embedding of f such that $w = z + k$ where $k = (i)|(j)$ (the concatenation of i and j); note that $i \in {F_p}^m$, $j \in {F_p}^n$ and $k \in {F_p}^{m+n}$.

These couples will be transformed by the change of basis into other couples (w', z') belonging to the implicit embedding of function g such that $w' = Tw$, $z' = Tz$ and $w' = z' + k'$ with $k' = Tk$.

Since matrix T is non-singular, there is a bijection between the values of k and those of k', i.e. the cells of the DDT of g are just a (linear) rearrangement of the cells of the DDT of f.

A similar reasoning can be applied to prove the relation between the LATs. A cell of the LAT table of f located in the i-th row and j-th column contains the number of the input vectors x such that $i^+ \bullet x + j^+ \bullet f(x) = 0$, where we denote the inner-product with $\bullet$ and, for sake of clearness, the transposed of a vector with $^+$.

Thus, if we consider the geometric representation for function f we have that the cell contains the number of vectors w belonging to the implicit embedding of f such that $k^+ \bullet w = 0$ where $k = (i)|(j)$; note that $i \in {F_p}^m$, $j \in {F_p}^n$ and $k \in {F_p}^{m+n}$.

These vectors will be transformed by the change of basis into other vectors w' belonging to the implicit embedding of function g such that $w' = Tw$. We can rewrite the equation as:

$$k^+ \bullet Tw = 0 \quad \Leftrightarrow \quad (T^+k)^+ \bullet w = 0 \quad \Leftrightarrow \quad (k')^+ \bullet w = 0$$

Since matrix T is non-singular, there is a bijection between the values of k and those of $k' = T^+k$, i.e. the cells of the LAT of g are just a (linear) rearrangement of the cells of the LAT of f. □

Corollary 2. *As a consequence of Theorem 2 we have that if f and g are generally linearly equivalent, then $\Delta_g = \Delta_f$ and $\Lambda_g = \Lambda_f$.*

We thus conclude that two generally linearly equivalent functions are characterized by the same cryptographic robustness; since the general case extends the classical relation, we can justify the common robustness of previously unrelated functions, such as f and g in Example 1.

It is a rather computationally difficult problem to decide whether two given functions are linearly equivalent: besides exhaustive search on the space of all possible matrices, it is possible to classify the functions basing on the distribution of values in the Walsh-Hadamard transform. Recently, Fuller and Millan

[9] have developed a classification method which exploits the concept of connectivity between two functions $f, g : {F_2}^m \to F_2$. They applied the method to the case $m = 8$ and to the Rijndael S-box, being able to prove that all the output variables of the only nonlinear step of the algorithm are linearly equivalent. Also, a description of optimized algorithms being able to find out whether two given invertible S-boxes are equivalent under a linear (or affine) transformation can be found in [5].

The result of Theorem 2 states that the whole distributions of values in the cryptographic tables are equal, not only the maximum values; such information could be used as a necessary condition for the generalized equivalence of two functions: if the two distributions differ, it can be immediately concluded that the two functions are not generally equivalent[3]. The check of this condition is not considered in [5]; we think that the check could speed up considerably the algorithms in most cases of negative answer. Obviously the condition is not sufficient and further techniques are needed to conclude that the two functions are generally (or classically) linearly equivalent.

It may be useful, at the end of this Section, to give also the geometric meaning of the parameters that measure cryptographic robustness.

In particular, the entries in the DDT of function f represent the number of vector couples belonging to the implicit embedding of f, that sum up to the same fixed vector, i.e. the (composed) difference vector. We can mentally view the process if we figure that the usual parallelogram rule is used to sum the vectors, as it would be done in standard Euclidean spaces; in practice, we are searching the vector couples that lead to the same path in the space S. This is evidently a measure of the redundancy of the information that characterizes the particular set of function vectors, i.e. the function itself.

The entries in the LAT, instead, can be seen as the number of vectors belonging to the implicit embedding of f that are orthogonal to a given fixed vector, since the inner product is the scalar product in S; the fixed vector is obtained by concatenating the masks that are classically applied to the function input and output values to compute the LAT. This can also be thought as a measure of the redundancy of the directions of the function vectors, and eventually of the function itself.

Finally, note that when the classical notion of linear equivalence is considered, we have linear rearrangements of the rows and the columns of the cryptographic tables; when generalized equivalence is applied, we have a linear rearrangement of the *cells* within the tables. There may exist couples of functions where the distributions of the values in the cryptographic tables are equal, but the actual arrangements of the cells cannot be linearly correlated. In these cases we can prove that the functions are not generally equivalent if we show that there are no possible linear rearrangements of the cells of one table that lead exactly to the other table.

[3] Since the classical equivalence is a special case of the generalized equivalence, the two functions are not equivalent also in the classical theory.

4 Application of the Criterion to Power Functions

The set of monomial power functions over F_{p^m} is interesting, since significant examples of functions with minimum possible Δ_f can be found in this class.

If $p = 2$ the minimum possible value for Δ_f when $f : \{0,1\}^m \to \{0,1\}^n$ is 2^{m-n}; functions reaching this limit are called Perfect Nonlinear (PN) [16] and exist only for m even and $m \geq 2n$. If we consider the important class of S-boxes, i.e. $f : \{0,1\}^m \to \{0,1\}^m$, then the minimum possible value for Δ_f is 2; functions reaching this limit are called Almost Perfect Nonlinear (APN) [17]. The only known examples of APN functions (up to classical linear equivalence) are power monomials; the list of known values for the exponent d such that $f(x) = x^d$ is APN can be found in [8]. Such functions find applications in symmetric key cryptography.

When $p > 2$ the minimum possible value for Δ_f is 1; functions reaching this limit are again called Perfect Nonlinear (PN). There are examples of PN and APN power functions over F_{p^m} and there is also one known example of a function that is not a power monomial but is PN over F_{3^m} for certain values of m [8].

Normally power monomials in even characteristic are classified into cyclotomic cosets, where a coset contains all the power monomials $\{x^d, x^{2d}, \ldots, x^{2^{m-1}d}\}$; the value d is called the coset leader and the power functions belonging to the same coset are classically linearly equivalent. Also, the inverse function $x^{d^{-1}}$ has the same cryptographic robustness of x^d, although it (in general) belongs to a different coset and is *not* linearly equivalent to x^d. Cosets, expanded with the usual classical equivalence criterion of Eq. 3, constitute the equivalence classes of power functions.

Using the criterion of generalized linear equivalence, different classical equivalence classes are merged into one: this is the case for instance of the classical equivalence classes of x^d and $x^{d^{-1}}$, since we have shown that in the new formalism the operation of inversion is nothing but a special case of linear transformation.

Moreover, we can show the existence of some functions that are not classically linearly equivalent to any power monomial, but still are APN.

Example 2. Consider the finite field F_{2^3}; the classification of all the possible exponents into cyclotomic cosets is given by:

$$C_0 = \{0\}$$
$$C_1 = \{1, 2, 4\}$$
$$C_3 = \{3, 6, 5\}$$

where the cosets C_i are numbered accordingly to the coset leader i. Coset C_0 contains only the constant function; coset C_1 contains the power monomials that are linear; coset C_3 contains non-linear APN power monomials. Since the inverse of x^6 is again x^6 and the inverse of x^3 is x^5 this coset is its own inverse[4].

[4] Note that this always happens to the coset that contains the inverse power function x^{-1} which in this case is actually x^6.

Coset C_3 can be expanded into a (classical) linear equivalence class of $f(x) = x^3$ by considering all the functions $g(x)$ such that

$$g(x) = T(f(x)) = \left(\begin{array}{c|c} A & 0 \\ \hline C & B \end{array}\right) \bullet \begin{pmatrix} x \\ x^3 \end{pmatrix}$$

Obviously, all these functions are APN and x^5, x^6 are some members of this class.

Now, consider the function $h(x)$ such that

$$h(x) = T'(f(x)) = \left(\begin{array}{c|c} I+S & I \\ \hline I & 0 \end{array}\right) \bullet \begin{pmatrix} x \\ x^3 \end{pmatrix}$$

where S is the matrix that gives the square of x (x^2 is a linear transformation in even characteristic, thus it can be represented by a matrix multiplication). The implicit embedding of h, and thus its truth-table, is described by:

$$x^3 + x^2 + x \quad \rightarrow \quad x$$

This implicit embedding still defines a function because $x^3 + x^2 + x$ is a permutation polynomial over F_{2^m} with m odd, see Corollary 2.10 of [19]. Since matrix T is non-singular, h is generally linearly equivalent to f and thus is APN. However, h does not belong to the classical equivalence class that extends C_3 because all the functions in this class are obtainable from $f(x)$ only using matrices T with a null upper-right block. We conclude that h belongs to a (classical) equivalence class that contains APN functions but is different from that of $f(x)$, which is the only one obtainable from power functions over F_{2^3}. Both these equivalence classes will be merged into one, when the general equivalence classes are considered; thus, this is another example of class merging.

Note that function h can actually be obtained from function f using classical *means*, i.e. by first transforming f into a classically linear equivalent function g and then inverting, since:

$$\left(\begin{array}{c|c} I+S & I \\ \hline I & 0 \end{array}\right) = \left(\begin{array}{c|c} 0 & I \\ \hline I & 0 \end{array}\right) \bullet \left(\begin{array}{c|c} I & 0 \\ \hline I+S & I \end{array}\right)$$

However, this does not lead to a function that is classically equivalent to f; while this may be difficult to prove classically, it becomes evident when general linear equivalence is introduced and one considers that matrix T' cannot belong to the family of matrices T indicated in the example.

5 Conclusions

In this paper we have presented the criterion of generalized linear equivalence. We have shown that the criterion extends the classical notion of linear equivalence; all the known cases of transformations that lead to invariance of the

cryptographic robustness can be treated as special instances of the proposed relation. Also, it can be shown that there are functions that cannot be correlated using the classical theory but become equivalent under the proposed criterion. We have used general equivalence to show that there are APN functions that are not classically linearly equivalent to power monomials, and that these equivalence classes are merged under the extended criterion.

References

1. Berlekamp, E.R., Welch, L.R.: Weight Distributions of the Cosets of the (32,6) Reed-Muller Code. IEEE Transactions on Information Theory, 18(1):203–207, 1972.
2. Beth, T., Ding, C.: On Almost Perfect Nonlinear Permutations. Proceedings of EUROCRYPT '93, 65–76, 1994.
3. Biham, E.: On Matsui's Linear Cryptanalysis. Proceedings of EUROCRYPT '94, 341–355, 1994.
4. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, 4(1):3–72, 1991.
5. Biryukov, A., De Canniere, C., Braeken, A., Preneel, B.: A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. Proceedings of EUROCRYPT 2003, 33–50, 2003.
6. Daemen, J., Rijmen, V.: The Design of Rijndael: AES-The Advanced Encryption Standard. Springer-Verlag, 2002.
7. Denev, J.D., Tonchev, V.D.: On the Number of Equivalence Classes of Boolean Functions under a Transformation Group. IEEE Transactions on Information Theory, 26(5):625–626, 1980.
8. Dobbertin, H., Mills, D., Muller, E.N., Pott, A., Willems, W.: APN functions in odd characteristic. Discrete Mathematics, 267(1-3):95–112, 2003.
9. Fuller, J., Millan, W.,: On linear Redundancy in the AES S-Box. Available online on http://eprint.iacr.org, 2002.
10. Harrison, M.A.: The Number of Classes of Invertible Boolean Functions. Journal of ACM, 10:25–28, 1963.
11. Harrison, M.A.: On Asymptotic Estimates in Switching and Automata Theory. Journal of ACM, 13(1):151–157, 1966.
12. Lorens, C.S.: Invertible Boolean Functions. IEEE Transactions on Electronic Computers, EC-13:529–541, 1964.
13. Maiorana, J.A.: A Classification of the Cosets of the Reed-Muller code r(1,6). Mathematics of Computation, 57(195):403–414, 1991.
14. Matsui, M.: Linear Cryptanalysis method for DES cipher. Proceedings of EUROCRYPT '93, 386–397, 1994.
15. Nyberg, K.: Differentially Uniform Mappings for Cryptography. Proceedings of EUROCRYPT '93, 55–64, 1994.
16. Nyberg, K.: Perfect Nonlinear S-Boxes. Proceedings of EUROCRYPT '91, 378–386, 1991.
17. Nyberg, K., Knudsen, L. R.: Provable security against differential cryptanalysis. Proceedings of CRYPTO '92, 566–574, 1992.
18. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell System Technical Journal, 28:656–715, 1949.
19. Small, C.: Arithmetics of Finite Fields. Dekker, New York, 1991.

Sieving Using Bucket Sort*

Kazumaro Aoki and Hiroki Ueda

NTT, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239-0847, Japan
{maro, ueda}@isl.ntt.co.jp

Abstract. This paper proposes a new sieving algorithm that employs a bucket sort as a part of a factoring algorithm such as the number field sieve. The sieving step requires an enormous number of memory updates; however, these updates usually cause cache hit misses. The proposed algorithm dramatically reduces the number of cache hit misses when the size of the sieving region is roughly less than the square of the cache size, and the memory updates are several times faster than the straightforward implementation.

1 Introduction

The integer factoring problem is one of the most important topics for public-key cryptography, because the RSA cryptosystem is the most widely used public-key cryptosystem, and its security is based on the difficulty of the integer factoring problem. Over a few hundred bits, the number field sieve [1] is currently the most fastest algorithm to factor an RSA modulus.

The number field sieve consists of many steps. It is known that the sieving step is theoretically and experimentally the most time-consuming step. It is noted that a straightforward implementation of the sieving step on a PC causes a long delay in memory reading and writing, and the sieving program is several dozen times faster if all memory accesses utilize the cache memory.

This paper focuses on memory access in the software implementation of the sieving step on a PC, and introduces an algorithm that reduces the number of cache hit misses. The experimental results confirm that the proposed sieving algorithm is several times faster than that in the straightforward implementation.

2 Preliminaries

2.1 Number Field Sieve

This section briefly describes the number field sieve algorithm that is relevant to the scope of the paper. Details regarding this algorithm can be found in (e.g. [1]).

* This work is supported in part by a consignment research from the Telecommunications Advancement Organization of Japan (now the National Institute of Information and Communications Technology) and by the CRYPTREC project.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 92–102, 2004.

Let N be a composite number and it will be factored. Find an irreducible polynomial $f(X) \in \mathbb{Z}[X]$ and its root M such that $f(M) \equiv 0 \pmod{N}$. The purpose of the sieving step is to collect many coprime pairs $(a, b) \in \mathbb{Z}^2$ such that $N_R(a, b) = |a + bM|$ is B_R-smooth and $N_A(a, b) = |(-b)^{\deg f} f(-a/b)|$ is B_A-smooth[1]. Such a coprime, (a, b), is called a *relation*.

We describe the line-by-line sieve (hereafter we simply referred to as *line sieve*) as Algorithm 1, and it is the most basic algorithm used to find relations. Hereafter, we omit the details on the algebraic side, because very similar algorithms are used for the algebraic side. Algorithm 1 assumes that $2H_a$ elements are allocated for array $\mathtt{S}$. The sieving region may be divided if $2H_a$ is greater than the suitable size for the implementation platform. The size of each element, $\mathtt{S}[a]$, is typically 1 byte, and the base for $\log p$ is selected such that it does not to exceed the maximum representable value of $\mathtt{S}[a]$. In Step 8 in the inset,

Algorithm 1 (line sieve for rational side (basic version)).

1: **for** $b \leftarrow 1$ **to** H_b
2: for all a ($-H_a \le a < H_a$), initialize $\mathtt{S}[a]$ to $\log N_R(a, b)$
3: **for prime** $p \leftarrow 2$ **to** B_R
4: Compute $a \ge -H_a$ as the first sieving point depending on b and p
5: **while** $a < H_a$
6: $\mathtt{S}[a] \leftarrow \mathtt{S}[a] - \log p$
7: $a \leftarrow a + p$
8: Completely factor $N_R(a, b)$ for all a if $\mathtt{S}[a] <$ some threshold

the threshold is determined by considering the error generated by the logarithm rounded to the nearest integer in Steps 2 and 6, and the omission of prime powers[2].

2.2 Large Prime Variation

If B_R is close to or greater than H_a, the while-loop in Step 5 is hardly activated, and the first sieving point computation in Step 4 may dominate the sieving time. For this case, we can use the *large prime variation*. The changes compared to the basic version are as follows:

1. Set the bound for p at Step 3 to B_R^L ($< B_R$).
2. Relax the threshold at Step 8 in Algorithm 1.

The faster the primality testing and factoring for small integers greater than B_R^L become available, the more relaxed the threshold can become.

Based on the experience, the most time-consuming part in large prime variation is reading and writing to memory to update $\mathtt{S}[a]$ in Step 6. This paper optimizes the memory read/write process.

[1] "x is y-smooth" means that all prime factors of x are less than or equal to y.

[2] By regarding prime power p^e as prime and $\log p^e$ as $\log p$, prime powers can be easily incorporated into Algorithm 1.

2.3 Memory Latency of a PC

Recent PCs have incorporated cache memory, and cache memory can usually be classified into several levels. A low level cache represents fast access but low capacity. For better understanding, we provide an example. Let us consider the Pentium 4 memory characteristics for logical operations performed by general purpose registers as shown in Table 1.

Table 1. Pentium 4 Northwood [2, p.1-17,1-19,1-20]

	Line size	Size	Latency
Register	(4 B)	32 B	$\frac{1}{2}$ processor cycle
Level 1 cache	64 B	8 KB	2 processor cycles
Level 2 cache	64 B+64 B	512 KB	7 processor cycles
Main memory	(4 KB)	≈1 GB	12 processor cycles + 6-12 bus cycles

The memory system in a PC is constructed to provide good performance for continuous address access, that is, random address access is very poor. A line sieve algorithm updates $\mathtt{S}[a]$ by step p in Step 6 in Algorithm 1. When p is greater than the size of cache memory, the updates seem to be random access. A read from the main memory requires at least $12 + 6 \times (2.53/0.533) = 40.5$ processor cycles, where the Pentium 4 frequency is 2.53 GHz and FSB is 533 MHz, according to Table 1. However, the user probably feels that the time required for main memory access requires more processor cycles. An experiment shows that the time for a random read from the main memory requires several hundred processor cycles.

2.4 Previous Work

Sieving can be considered as waiting for memory because other steps in the innermost loop are small and very simple, according to Steps 5 to 7 in Algorithm 1. To overcome cache hit misses, [3] proposed the *block sieving* algorithm. There are two differences between the basic version of the line sieve in Algorithm 1 and the block sieving algorithm: the addition of Algorithm 2 between Steps 2 and 3, and the initial p in Step 3 is modified to the smallest prime greater than B_R^S. The block sieving algorithm classifies factor base primes into *smallish primes* ($\in (0, B^S]$) and *largish primes* ($\in (B^S, B^L]$), and updates each small region whose size is H_a^S by smallish primes. To achieve better performance, H_a^S and B_R^S are set to approximately the size of the cache memory. Note that the computation of the first sieving point in Step 3 in Algorithm 2 can be omitted if the last sieving point computed in Step 4 is available. Focusing on the memory hierarchy, the performance of the sieving step may be better optimized in order to consider more parameters in classifying smallish primes in some environments.

Algorithm 2 (Additional steps from line sieve to block sieving algorithm).

1: **for** $a^S \leftarrow -H_a$ **to** H_a **step** $+H_a^S$
2: **for prime** $p \leftarrow 2$ **to** B_R^S
3: Compute $a \geq a^S$ as the first sieving point
4: **while** $a < a^S + H_a^S$
5: $\mathtt{S}[a] \leftarrow \mathtt{S}[a] - \log p$
6: $a \leftarrow a + p$

3 Sieving Using Bucket Sort

The number of cache hit misses for smallish primes greatly decreases using the block sieving algorithm described in Sect. 2.4; however, the sieving for largish primes still generate many cache hit misses. This section describes the reduction in the number of cache hit misses for largish primes using the bucket sorting algorithm [4, Sect. 5.2.5].

As mentioned in Sect. 2.3, memory updates between close addresses are not penalized, and the $\log p$ minuses which are memory update operations are commutative. Sorting $(a, \log p)$ using key a can reduce the number of cache hit misses; however, the sorting should be done very quickly, because the number of $\mathtt{S}[a]$ updates is roughly $2H_a(\log\log B^L - \log\log B^S)$, that is, it is almost linear to H_a. While complete sorting is not required and recent PC models have very large memory capacity, we use the bucket sorting algorithm to address this issue.

3.1 Proposed Algorithm

The proposed algorithm replaces the largish prime sieving in Algorithm 1, that is, the algorithm has the same function as Algorithm 1 for sieving largish primes.

The algorithm is based on bucket sorting. Let n be the number of buckets, and r be $\left\lceil \frac{n_\mathtt{S}}{n} \right\rceil$, where $n_\mathtt{S}$ denotes the number of elements in $\mathtt{S}$. Note that $n_\mathtt{S} = 2H_a$ for Algorithm 1. The algorithm comprises the continuous runs of Algorithms 3 and 4.

Algorithm 3.

1: Let all buckets empty
2: **for prime** $p \leftarrow B_R^S + 1$ **to** B_R^L
3: Compute $a \geq -H_a$ as the first sieving point
4: **while** $a < H_a$
5: Store $(a, \log p)$ to the $\left\lfloor \frac{a + H_a}{r} \right\rfloor$th bucket
6: $a \leftarrow a + p$

Algorithm 4.

1: **for** all buckets that are numbered i $(0 \leq i < n)$
2: **for** all $(a, \log p)$ in the bucket i
3: $\mathtt{S}[a] \leftarrow \mathtt{S}[a] - \log p$

Algorithm 3 throws $(a, \log p)$ in the buckets, and Algorithm 4 updates $\mathtt{S}[a]$ using the elements in the buckets.

3.2 Why Can Proposed Algorithm Hit Cache Memory?

Figure 1 forms the basis for the following discussion. We first consider Algorithm 4. All elements in a bucket will only updates the memory in range r.

Thus,

$$r \times (\text{Size of each } \mathtt{S}[a]) \leq (\text{Size of cache memory}) \tag{1}$$

should hold. Next, we consider Algorithm 3. For each bucket, the addresses for memory writes are continuous. It is sufficient if

$$n \times (\text{Size of cache line}) \leq (\text{Size of cache memory}) \tag{2}$$

holds. Note that the cache memory can only be updated in units called cache lines. We assume that the size of $(a, \log p)$ is less than the size of a cache line. When combining (1) and (2), n exists if

$$n_{\mathtt{S}} \times (\text{Size of each } \mathtt{S}[a]) = (\text{Size of } \mathtt{S}[\bullet]) \leq \frac{(\text{Size of cache memory})^2}{(\text{Size of cache line})} \tag{3}$$

holds.

Let us consider a typical parameter using Table 1. The size of the cache memory is 512 KB, and the size of the cache line is 128 B. Therefore, the right hand side of (3) is 2^{31} B. If we allocate each $\mathtt{S}[a]$ as 1 B, then $\mathtt{S}$ can occupy up to 2 GB. This means that the proposed algorithm is effective for most PCs. The proposed algorithm increases the number of memory accesses, but dramatically reduces the number of cache hit misses with appropriate prefetching.

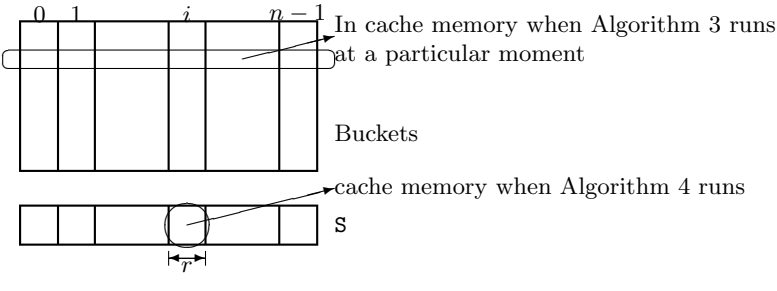


Fig. 1. Memory usage for buckets and $\mathtt{S}$

3.3 Related Work

[5], which follows the inspiring work [6], independently proposed sieving hardware, which sorts $(a, \log p)$. The paper does not consider the cache memory; however, their algorithm is similar in that *sieving* is converted to *sorting*.

4 Optimizations and Improvements

This section considers optimization techniques and improvements to the proposed algorithm.

4.1 $(a, \log p)$ Size Reduction

The size of a stored in a bucket can be reduced. $a' = a + H_a \bmod r$ is sufficient to recover a, because $a = ir + a' - H_a$ for the ith bucket.

Moreover, the number of bits for $\log p$ can be reduced to 1 bit, because $(a, \log p)$ can be generated in ascending order on p and $\log p$ in a bucket increases very slow.

4.2 Number of Buckets

For efficient computation of Step 5 in Algorithm 3 and the technique described in Sect. 4.1, r should be a power of 2 on most PCs.

4.3 Hierarchical Buckets

Considering the idea of radix sort and cache hierarchy, Algorithm 4 can be modified to Algorithms 3 and 4 using smaller buckets.

4.4 Reduction in Memory for Buckets

Consider the case that a PC does not have enough memory to allocate buckets to store all $(a, \log p)$s. Whenever a bucket is full at Step 5 in Algorithm 3, call Algorithm 4 and empty the buckets.

4.5 Reduction in Sieving Memory S

First, perform sieving for largish primes using Algorithms 3 and 4. When executing Algorithm 4, smallish prime can be sieved between Steps 1 and 2. In the ith bucket, a is in $[ir - H_a, (i+1)r - H_a)$. Thus, r elements for $\mathtt{S}[a]$ are sufficient for the ith bucket.

Note that this idea cannot be used with the idea described in Sect. 4.4.

4.6 Bucket Reuse for Trial Division

The trial sieving algorithm [7] was proposed to reduce the time in Step 8 in Algorithm 1. The algorithm acts almost the same as the sieving algorithm discussed above, but it only considers a small set of (a, b). When filling buckets in Algorithm 3, store p in addition to $(a, \log p)$, and the buckets can be used for

trial sieving. This can reduce the computational cost of the first sieving points for largish primes. However, storing p probably doubles the memory allocation for the buckets. It may be a good idea to avoid storing small primes that are classified as largish primes.

4.7 Application to Lattice Sieve

The idea behind the proposed algorithm can be applied to any algorithm if the memory update operation is commutative. There are no problems in using the proposed algorithm for the lattice sieve.

4.8 Tiny Primes

[8, p.334] suggests that $\mathtt{S}[a]$ is initialized by the logarithm of tiny primes. It can be efficiently achieved by the following idea. First, compute the sieving pattern for tiny primes, which are less than B^T, and their small powers. Once the pattern is computed, the initialization of $\mathtt{S}[a]$ can be done by duplicating the pattern by adjusting the correct starting position.

5 Implementation on Pentium 4

We implemented Algorithms 3 and 4 in the lattice sieve using all the techniques in Sect. 4 except for Sect. 4.4 and last haf of Sect. 4.1 on a Pentium 4 (Northwood) with 1 GB main memory and 533 MHz FSB. The specifications are the same as those described in Table 1. The prime bounds are described in Table 2. These names are from [9]. We tried to obtain the best Bs using the factor base parameter for c158 as described in [10].

Table 2. Prime Bounds and Algorithms

Range	Name	Algorithm
$p \le B^T$	p: Tiny prime	Sieving pattern
$B^T < p \le B^S$	p: Smallish prime	Block sieving
$B^S < p \le B^L$	p: Largish prime	Bucket sorting
$B^L < p \le B$	p: Large prime	Primality testing and factoring

5.1 Parameter Selection

We assign 1 B for $\log p$ and 4 B for each $(a, \log p)$, because the smallest memory read and write unit is 1 B and the basic memory data unit is 4 B for the Pentium 4.

On the factorization of c158, the sieving rectangle was $2H_c \times H_d = 2^{14} \times 2^{13}$. To translate the rectangle to a line sieve case, we can interpret $2H_a = 2^{14} \times 2^{13} = 2^{27}$. The large primes in each relation and the values of B_R^L and B_A^L are unclear. Therefore, we select two large primes for both sides in each relation, and set $B_R^L = 30 \times 10^6$, $B_A^L = 0.9 \times Q$, and $B_R^S = B_A^S = 512 \times 2^{10}$, where Q denotes the

special-Q according to our factoring code, the primality testing for large prime products, factor base bound for the line sieve, and the size of level 2 cache. We tried the depths of 1, 2, and 3 for the hierarchical buckets with all powers of 2 for r, and found that the best hierarchy depth is 2. Surprisingly, the best rs are not the combination of the size of the level 2 cache and level 1 cache, but 2 MB and 256 KB.

Next, we tried to find the best B_R^S and B_A^S. Based on dozens of experments, we find that $B_R^S = 2H_c$ and $B_A^S = 5H_c$ achieve almost the best performance.

Remark 1. We sieve prime powers less than $\sqrt{B^L}$, and select $B_R^T = B_A^T = 5$.

Remark 2. We classify smallish primes into small sets taking into account the size of the caches and sieving range.

Remark 3. After executing Algorithm 3, the numbers of elements in each bucket are roughly the same. We found that a 2% difference is the largest in our experiments.

Remark 4. We used base-2 Solovay-Strassen primality testing [11, pp.90–91], and ρ [11, pp.177–183] and SQUFOF [11, pp.186–193] as the factoring algorithm for large primes.

5.2 Factoring Example

We factor 164-digit cofactor c164 in $2^{1826}+1$ using GNFS, and 248-digit cofactor c248 in $2^{1642}+1$ using SNFS employing the above implementation. Refer to the Appendix for detailed information. The parameters used in the factoring of c164 and c248 are summarized in Table 3. For comparison purposes, Table 3 also includes the parameters used in the factoring of RSA-512 [12].

Table 3. Factoring Parameters for Lattice Sieve

	H_c	H_d	B_R^L	B_A^L	B	max sp-Q	#sp-Q	#LP	rel/MY
c164	16 K	8 K	40 m	0.95Q	4 g	194 m	8.2 m	2+2	29 k
c248	16 K	8 K	0.95Q	100 m	4 g	200 m	10.2 m	2+2	22 k
RSA-512	4 K	5 k	16 M	16 M	1 g	15.7 m	308 m	2+2	14 k

k: 10^3, K: 2^{10}, m: 10^6, M: 2^{20} g: 10^9, G: 2^{30}
rel/MY: Generated relations per MIPS year

The proposed siever yields more relations per MIPS year despite that c164 is larger than RSA-512. However, a straightforward comparison should be avoided because the characteristics of computers used for the above factoring are quite different, and MIPS is not optimal for comparing the sieving complexity.

Remark 1. The lattice siever used for RSA-512 is intended to factor RSA-130 [12, Sect. 3.2].

Remark 2. We timed MIPS using the output of a "BYTE benchmark." We obtained 3969679.6 lps for Dhrystone 2 without register variables. Thus, MIPS is computed by $3969679.6/1767 \approx 2246.6$. This number is used for c164 and c248 in column rel/MY.

Remark 3. We noticed that numbers larger than RSA-512 such as RSA-576 are already factored using GNFS [13] and that their siever seems faster than one that was used for RSA-512. However, not enough information is provided to estimate the timings. We used the records that were published and the largest values [12].

6 Conclusion

We proposed a sieving algorithm that cleverly uses the cache memory. The algorithm accelerates the memory update processes in the sieving step to several times faster than that of the simple $\log p$ subtraction. Moreover, we implemented the proposed algorithm in the lattice sieve on a Pentium 4, and successfully factored a 164-digit number using GNFS, and a 248-digit number using SNFS.

Acknowledgments

The authors gratefully thank Prof. Yuji Kida for fruitful discussions, his ideas regarding Sect. 4.5 and the last half of Sect. 4.6. Moreover, the authors thank him for his contribution in approximating B^S in Sect. 5.1. We also thank Dr. Takeshi Shimoyama of Fujitsu Labs and Dr. Soichi Furuya of Hitachi for suggesting a hint for reducing the number of bits for $\log p$ in Sect. 4.1.

References

1. Lenstra, A.K., Lenstra, Jr., H.W., eds.: The development of the number field sieve. Volume 1554 of Lecture Notes in Mathematics. Springer-Verlag, Berlin, Heidelberg (1993)
2. Intel Corporation: IA-32 Intel Architecture Optimization Reference Manual. (2004) Order Number: 248966-010 (`http://support.intel.com/design/pentium4/manuals/248966.htm`).
3. Wambach, G., Wettig, H.: Block sieving algorithms. Technical Report 190, Informatik, Universität zu Köln (1995) (`http://www.zaik.uni-koeln.de/~paper/index.html?show=zpr95-190`).
4. Knuth, D.E.: Sorting and Searching. Second edn. Volume 3 of The Art of Computer Programming. Addison Wesley (1998)
5. Geiselmann, W., Steinwandt, R.: A dedicated sieving hardware. In Desmedt, Y.G., ed.: Public Key Cryptography — PKC2003. Volume 2567 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2003) 254–266
6. Bernstein, D.J.: Circuits for integer factorizaion: a proposal. (available at `http://cr.yp.to/factorization.html\#nfscircuit`) (2002)
7. Golliver, R.A., Lenstra, A.K., McCurley, K.S.: Lattice sieving and trial division. In Adleman, L.M., Huang, M.D., eds.: Algorithmic Number Theory — First International Symposium, ANTS-I. Volume 877 of Lecture Notes in Computer Science., Berlin, Heidelberg, New York, Springer-Verlag (1994) 18–27

8. Silverman, R.D.: The multiple polynomial quadratic sieve. Mathematics of Computation **48** (1987) 329–339
9. Shamir, A., Tromer, E.: Factoring large numbers with the TWIRL device. In Boneh, D., ed.: Advances in Cryptology — CRYPTO 2003. Volume 2729 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2003) 1–26
10. Bahr, F., Franke, J., Kleinjung, T.: Factorization of 158-digit cofactor of $2^{953}+1$. (available at `http://www.crypto-world.com/announcements/c158.txt`) (2002)
11. Riesel, H.: Prime Numbers and Computer Methods for Factorization. Second edn. Volume 126 of Progress in Mathematics. Birkhäuser, Boston, Basel, Berlin (1993)
12. Cavallar, S., Dodson, B., Lenstra, A.K., Lioen, W., Montgomery, P.L., Murphy, B., te Riele, H., Aardal, K., Gilchrist, J., Guillerm, G., Leyland, P., Marchand, J., Morain, F., Muffett, A., Putnam, C., Zimmermann, P.: Factorization of a 512-bit RSA modulus. In Preneel, B., ed.: Advances in Cryptology — EUROCRYPT 2000. Volume 1807 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2000) 1–18
13. Contini, S.: Factor world! (`http://www.crypto-world.com/FactorWorld.html`) (2002)
14. Wagstaff, S.S.: The Cunningham Project. (2003) (`http://www.cerias.purdue.edu/homes/ssw/cun/`).
15. Aoki, K., Kida, Y., Shimoyama, T., Sonoda, Y., Ueda, H.: GNFS164. (`http://www.rkmath.rikkyo.ac.jp/~kida/gnfs164e.htm`) (2003)
16. Aoki, K., Kida, Y., Shimoyama, T., Sonoda, Y., Ueda, H.: SNFS248. (`http://www.rkmath.rikkyo.ac.jp/~kida/snfs248e.htm`) (2004)

Appendix: Factoring Parameters and Statistics for c164 and c248

c164 and c248 are selected from Cunningham table [14].

c164 is the 164-digit cofactor of $2^{1826}+1$. $2^{1826}+1$ can be trivially factored to $2,1826\mathrm{L} \times 2,1826\mathrm{M}$, where $2,1826\mathrm{L} = 2^{913}-2^{457}+1$, and its several factors are already known:

$$\begin{aligned} 2,1826\mathrm{L} = {} & 997 \times 2113 \times 10957 \times 46202197673 \times 209957719973 \\ & \times 457905185813813 \times 911842581496373502008405006 9 \\ & \times 75898404523976541436629048015451408 9 \times \mathrm{c164} \end{aligned}$$

c164 is factored into two primes, p68 × p97, where

$$\begin{aligned} \mathrm{p68} = {} & 343346448861824465 \\ & 462730089242420846343270897895597712158640922548 49. \end{aligned}$$

c248 is the 248-digit cofactor of $2^{1642}+1$. $2^{1642}+1$ can be trivially factored to $2,1642\mathrm{L} \times 2,1642\mathrm{M}$, where $2,1642\mathrm{M} = \mathrm{c248} = 2^{821}+2^{411}+1$. c248 is factored into two primes, p88 × p160, where

$$\begin{aligned} \mathrm{p88} = {} & 750529374601164176649246785489326160 36 \\ & 640381023147128390479077762437121481797484501905 33. \end{aligned}$$

$$
\begin{aligned}
\text{c164poly} = {} & 8293702863045600\, x^5 \\
& + 5627796025215486707\, x^4 \\
& + 55752455642730993190 2111\, x^3 \\
& + 17691721660250881843016 1036\, x^2 \\
& - 13601173202899548432935219131949\, x \\
& - 12171622290476241497444980012311021 \\
M = {} & 41126877593272575259693918 4846
\end{aligned}
\qquad
\begin{aligned}
\text{c248poly} = {} & x^6 \\
& + 2\, x^3 \\
& + 2 \\
M = {} & 2^{137}
\end{aligned}
$$

Fig. 2. Polynomials used to factor c164 and c248

We used the polynomials described in Fig. 2 to factor c164 and c248.

Statistics are summarized in Table 4. CPU years for sieving are converted for the Pentium 4 2.53 GHz. Line sieve is used for c164 factoring, and it yields 49 m relations. Free relations are not used for both factorings. Linear algebra is computed by a 16 PC cluster with GbE using block Lanczos with 128-bit block. The Pentium 4 is used for both factoring, but its frequency is about 2.53 GHz for c164 and 3.2 GHz for c248. The programs used for the factoring are basically the same except that minor improvements are included for c248. More detailed information can be found at [15, 16].

Table 4. Statistics

	Sieve		Linear algebra		
	CPU years	Yields	Matrix size	Row weight	Calendar days
c164	7	458 m	7.5 m	167	12
c248	8.2	558 m	7.4 m	208	9.5

Right-Invariance: A Property for Probabilistic Analysis of Cryptography Based on Infinite Groups

Eonkyung Lee

Dept. of Applied Mathematics, Sejong University, Seoul, Korea
eonkyung@sejong.ac.kr

Abstract. Infinite groups have been used for cryptography since about twenty years ago. However, it has not been so fruitful as using finite groups. An important reason seems the lack of research on building a solid mathematical foundation for the use of infinite groups in cryptography. As a first step for this line of research, this paper pays attention to a property, the so-called right-invariance, which makes finite groups so convenient in cryptography, and gives a mathematical framework for correct, appropriate use of it in infinite groups.

1 Introduction

In modern cryptography, many schemes are designed based on groups. The most popular problems used for cryptography may be the integer factorization and discrete logarithm problems in finite groups. From these problems, many schemes have been developed. However, on quantum computer they turned out to be efficiently solved by Shor's algorithms [19].

Not to put all eggs in one basket as well as to enrich cryptography, people have attempted to use infinite groups for cryptography. Compared to finite groups, in infinite groups there are only a few types of schemes (e.g. key agreement protocol or public key encryption) [24, 9, 21–23, 13, 2] and a few ways of analyses of attacks (e.g. deterministic or empirical) [3, 10, 17, 12, 11, 16, 7]. A natural question is how we can proceed one more step. An impediment to this seems to be connected with "probability". Indeed, many cryptographic schemes have checkpoints concerning probability for their basic security, and many cases of cryptanalysis rely on probabilistic analysis. Furthermore, we do not see that we can build a provably secure cryptosystem without probability. However, there is nothing discussed seriously for it in the literature on infinite-group-based cryptography.

Our Results. When cryptosystems are designed or analyzed using infinite groups, we sometimes feel attracted to use nice properties or tools which are commonly used in finite groups. However, we do not since either it looks wrong or we are not sure if it is right or wrong. A possible approach to resolve this problem is to extract a nice property of finite groups, to generalize it in arbitrary

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 103–118, 2004.

groups, and then to construct a rigorous theory by which we can decide when we can or cannot use this property in infinite groups.

This paper follows this way focusing on a particular property, the so-called *right-invariance*: we define a probability measure (cf. probability distribution in probability theory) P on a group G as right-invariant if $P(E) = P(Ex)$ for all $E \subset G$ on which P is defined and for all $x \in G$. We show that right-invariance property depends on a particular *subgroup* and the index of the subgroup determines when right-invariance can or cannot be used in infinite groups.

For the situations where this property is allowable, one may be curious about how it can be handled in practice. It is easy to find a probability measure which is right-invariant *only* in a particular situation. However, what is more meaningful is to find a probability measure which is right-invariant in *all* situations where such property is allowable. Namely, a right-invariant probability measure that can be used universally on a given group. As to this, we prove that most infinite groups dealt with in cryptography do not have such a probability measure. So we discuss weaker, yet practical alternatives with concrete examples. Using these, we illustrate how our theory is applied to infinite-group-based cryptography via two opposite types of situations.

Organization. Sec. 2 gives basic notations and brief definitions for reading this paper. Sec. 3 discusses why right-invariance is attractive, and formalizes the notion. Sec. 4 explores right-invariance property through building a mathematical framework. Sec. 5 discusses the notion of universally right-invariant probability measure and its alternatives. Sec. 6 shows how the results developed in the previous sections can be applied to practice. This paper concludes with Sec. 7.

2 Preliminaries

$\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ denote the sets of all positive integers, all integers, and all real numbers, respectively. For $a < b$, $(a, b) = \{x \in \mathbb{R} \mid a < x < b\}$ and $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$. For $n \in \mathbb{N}$, $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ and $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$. For sets S and T, $S \backslash T = \{x \in S \mid x \notin T\}$. $|S|$ and 2^S denote the cardinal number of S and the collection of all subsets of S, respectively. $S^{-1} = \{x^{-1} \mid x \in S\}$. A *partition* of S means a family $\{S_i\}_{i \in I}$ of non-empty, mutually disjoint subsets of S such that $S = \cup_{i \in I} S_i$. $\emptyset$ denotes the empty set.

Definition 1. (a) Let $\mathcal{M} \subset 2^X$ for a non-empty set X. $\mathcal{M}$ is called a *σ-algebra in X* if (i) $\emptyset \in \mathcal{M}$, (ii) $E \in \mathcal{M}$ implies $X \backslash E \in \mathcal{M}$, and (iii) $E_1, E_2, \ldots \in \mathcal{M}$ implies $\cup_{i=1}^{\infty} E_i \in \mathcal{M}$.

(b) If $\mathcal{M}$ is a σ-algebra in a non-empty set X, then $(X, \mathcal{M})$ is called a *measurable space* and the members of $\mathcal{M}$ are called the *measurable sets* in X.

If S is any collection of subsets of X, there exists a smallest σ-algebra $\mathcal{M}$ in X such that $S \subset \mathcal{M}$. This $\mathcal{M}$ is called the *σ-algebra generated by S*.

Definition 2. (a) For a measurable space $(X, \mathcal{M})$, a set function $\mu : \mathcal{M} \to [0, 1]$ is called a *probability measure on* $\mathcal{M}$ if it satisfies that (i) $\mu(X) = 1$ and (ii) if $E_1, E_2, \ldots \in \mathcal{M}$ are mutually disjoint, $\mu(\cup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \mu(E_i)$.
(b) For a measurable space $(X, \mathcal{M})$, if μ is a probability measure on $\mathcal{M}$, then $(X, \mathcal{M}, \mu)$ is called a *probability space*. In particular, it is called *atomic* if $\mathcal{M} = 2^X$. Measurable sets of a probability space are called *events*.

Let G be a group and H a subgroup of G. For $x \in G$, let $Z_H(x) = \{y \in H \mid yx = xy\}$, which is a subgroup of H. $Hx = \{hx \mid h \in H\}$ is called a *right coset* of H in G and $xH = \{xh \mid h \in H\}$ a *left coset* of H in G. The *index of* H *in* G, denoted by $[G : H]$, is the cardinal number of the set of distinct right (or left) cosets of H in G. For a normal subgroup H of G, G/H denotes $\{Hx \mid x \in G\}$ and is called the *factor group of* G *over* H. 1_G denotes the identity of G.

Definition 3. (a) For a set X, $w = w_1 \cdots w_\ell$ is called a *reduced word on* X if w is the empty word or w satisfies that (i) $\ell \in \mathbb{N}$; (ii) $w_i \in X \cup X^{-1}$ for all $1 \leq i \leq \ell$; (iii) $w_{i+1} \neq w_i^{-1}$ for all $1 \leq i < \ell$. $|w| = 0$ (if w is the empty word) or ℓ (otherwise) denotes the word length of w.
(b) $F(X)$ is called the *free group generated by* X. It is the set of all reduced words on X with the binary operation: for any $w_1, w_2 \in F(X)$, $w_1 \cdot w_2$ is the reduced form of the word obtained by the juxtaposition $w_1 w_2$ of the two words. The symbol '$\cdot$' is omitted if there is no confusion.

3 Role of Right-Invariance in Cryptography

This section shows why this paper selects right-invariance as a useful property.

Role in Random Self-Reducibility. Informally, a problem is called *random self-reducible* if solving it on *any* instance is efficiently reduced to solving it on a *random* instance. For a random self-reducible problem, if breaking a cryptographic scheme implies solving the problem on average, it means solving it in the worst case. Thus, since Blum and Micali [4] introduced this notion, it has played an invaluable role in showing provable security of many schemes. We refer to [1, 8] for detailed references on it and the cryptographic significance of this feature. We state it roughly in terms of the discrete logarithm problem with proper parameters; a prime p and a generator g of $\mathbb{Z}_p^*$. n is the length of p when it is represented in a bit-string.

Let $a, b \in \mathbb{N}$ and let $\mathcal{A}$ be a probabilistic polynomial time algorithm such that

$$\Pr_x[\mathcal{A}(p, g, g^x \bmod p) = x] > \frac{1}{n^a},$$

where x is taken uniformly at random from $\mathbb{Z}_{p-1}$. Then, there exists a probabilistic polynomial time algorithm $\mathcal{D}$ such that for all $y \in \mathbb{Z}_{p-1}$,

$$\Pr[\mathcal{D}(p, g, g^y \bmod p) = y] > 1 - \frac{1}{n^b}.$$

$\mathcal{D}$ is built based on the following idea: for any fixed $y \in \mathbb{Z}_{p-1}$, $\mathcal{D}$ chooses $x \in \mathbb{Z}_{p-1}$ uniformly at random, gets w by running $\mathcal{A}$ on an input $(p, g, g^y g^x \bmod p)$, outputs $w - x \bmod p - 1$ if $g^w = g^y g^x \bmod p$, otherwise repeats this process some polynomial times. A basic property used in computing the success probability of $\mathcal{D}$ is that for any $y \in \mathbb{Z}_{p-1}$

$$\Pr_x[\mathcal{A}(p, g, g^{y+x} \bmod p) = y + x \bmod p - 1] = \Pr_x[\mathcal{A}(p, g, g^x \bmod p) = x], \quad (1)$$

where x is taken uniformly at random from $\mathbb{Z}_{p-1}$.

Equation (1) can be generalized as follows: given a group G, for all $r \in G$

$$\Pr(f(X) = 0) = \Pr(f(Xr) = 0) \quad \text{or} \quad (2)$$
$$\Pr(f(X) = 0) = \Pr(f(rX) = 0), \quad (3)$$

where X is a random variable over G and $f : G \to \{0, 1\}$ is a predicate. Without loss of generality (WLOG), in this paper we focus on (2).

If G is a finite group and X has the uniform distribution, (2) is true. In this case, it is being used as an underlying assumption in probabilistically analyzing many kinds of cryptographic schemes. However, it is not true in general if G is an infinite group or if one cannot uniformly generate elements from even a finite group. We know that no probability distribution can ever be uniform on any infinite group, however the concept of uniformity makes infinite groups more flexibly handled in cryptography. A natural question is what distribution on an infinite group is an analogue of the uniform distribution on a finite group.

For an arbitrary group G, let's recall the meaning of a random variable. The fact that X is a random variable over G with a probability distribution P means that P is the probability measure on the atomic measurable space $(G, 2^G)$ and $\Pr[X \in E] = P(E)$ for any $E \subset G$. In order for (2) to hold when G is an infinite group, we see it from a measure-theoretic point of view. Namely, we consider not only 2^G but also a smaller σ-algebra $\mathcal{G}$ for P. By restricting P originally defined on 2^G to $\mathcal{G}$, $(G, 2^G, P)$ induces another probability space $(G, \mathcal{G}, P)$.

Definition 4. Let $(G, \mathcal{G}, P)$ be a probability space. $E \in \mathcal{G}$ is called a *right-invariant event* (resp. *left-invariant event*) if, for all $x \in G$, $Ex \in \mathcal{G}$ (resp. $xE \in \mathcal{G}$) and $P(E) = P(Ex)$ (resp. $P(E) = P(xE)$). $(G, \mathcal{G}, P)$ (or shortly P) is called *right-invariant* (resp. *left-invariant*) if all events are right-invariant (resp. left-invariant).

For a situation in which one is interested (e.g. points where one wants to compute probabilities or to compare them), if a σ-algebra covering all the events in question (i.e. containing all the events in question as its measurable sets) can be constructed and there exists a right-invariant probability measure thereon, then we say that *right-invariance is allowable (or can be used, etc.) in the situation.*

4 Right-Invariant Probability Space

In order to discuss right-invariance from a measure-theoretic point of view, we first analyze the structure of an arbitrary σ-algebra in infinite groups, and then

a special type of σ-algebra. From this we formulate a way of deciding whether or not right-invariance property is allowable in a given situation.

Throughout this paper, we deal with only finitely generated groups since groups with infinitely many generators are not practical. Note that any finitely generated infinite group is a countable set.

σ-Algebra in Finitely Generated Infinite Groups. Let G be a finitely generated infinite group and $\mathcal{G}$ be a σ-algebra in G. For $x \in G$, define

$$\mathcal{M}_{\mathcal{G}}(x) = \{E \in \mathcal{G} \mid x \in E\} \quad \text{and} \quad M_{\mathcal{G}}(x) = \cap_{E \in \mathcal{M}_{\mathcal{G}}(x)} E.$$

In particular, denote $M_{\mathcal{G}}(1_G)$ by $M_{\mathcal{G}}$. The following proposition shows that $M_{\mathcal{G}}(x)$ is the smallest measurable set containing x.

Proposition 1. *For a finitely generated infinite group G, let $\mathcal{G}$ be any σ-algebra in it. Then, $M_{\mathcal{G}}(x) \in \mathcal{G}$ for all $x \in G$. Furthermore, any measurable set is partitioned into $M_{\mathcal{G}}(x)$'s.*

Proof. Let $x \in G$. Since $G \in \mathcal{M}_{\mathcal{G}}(x)$ and $x \in M_{\mathcal{G}}(x)$, $M_{\mathcal{G}}(x) \neq \emptyset$. We show that $M_{\mathcal{G}}(x)$ can be expressed as an intersection of a countable number of measurable sets. For $y \in G$, define a set A_y as follows.

$$A_y = \begin{cases} G & \text{if } y \in M_{\mathcal{G}}(x), \\ E \text{ such that } y \notin E \in \mathcal{M}_{\mathcal{G}}(x) & \text{if } y \notin M_{\mathcal{G}}(x). \end{cases}$$

Since G is a countable set, it suffices to show that $M_{\mathcal{G}}(x) = \cap_{y \in G} A_y$. (i) $M_{\mathcal{G}}(x) \subset \cap_{y \in G} A_y$: If $w \notin \cap_{y \in G} A_y$, there exists $y \in G$ such that $w \notin A_y$. Since $A_y \in \mathcal{M}_{\mathcal{G}}(x)$, $w \notin M_{\mathcal{G}}(x)$. (ii) $M_{\mathcal{G}}(x) \supset \cap_{y \in G} A_y$: If $w \notin M_{\mathcal{G}}(x)$, $w \notin A_w$. Thus, $w \notin \cap_{y \in G} A_y$. Therefore, $M_{\mathcal{G}}(x) \in \mathcal{G}$.

Let $E \in \mathcal{G}$. Since, for any $x \in E$, $M_{\mathcal{G}}(x) \subset E$, $E = \cup_{x \in E} M_{\mathcal{G}}(x)$. Thus it suffices to show that any distinct $M_{\mathcal{G}}(x)$ and $M_{\mathcal{G}}(y)$ are disjoint. Assume $M_{\mathcal{G}}(x) \cap M_{\mathcal{G}}(y) \neq \emptyset$. If $x \notin M_{\mathcal{G}}(y)$, then $M_{\mathcal{G}}(x) \backslash M_{\mathcal{G}}(y) \in \mathcal{M}_{\mathcal{G}}(x)$ since $M_{\mathcal{G}}(x) \backslash M_{\mathcal{G}}(y) \in \mathcal{G}$ and $x \in M_{\mathcal{G}}(x) \backslash M_{\mathcal{G}}(y)$. Since $M_{\mathcal{G}}(x)$ is the intersection of all members of $\mathcal{M}_{\mathcal{G}}(x)$, $M_{\mathcal{G}}(x) \subset M_{\mathcal{G}}(x) \backslash M_{\mathcal{G}}(y)$. In particular, $M_{\mathcal{G}}(x) \cap M_{\mathcal{G}}(y) = \emptyset$ which contradicts to the assumption. Thus $x \in M_{\mathcal{G}}(y)$, so $M_{\mathcal{G}}(x) \subset M_{\mathcal{G}}(y)$. By the same argument, $M_{\mathcal{G}}(y) \subset M_{\mathcal{G}}(x)$. Therefore, $M_{\mathcal{G}}(x) = M_{\mathcal{G}}(y)$. □

Right-Closed σ-Algebra in Finitely Generated Infinite Groups

Definition 5. A measurable space $(G, \mathcal{G})$ (or a σ-algebra $\mathcal{G}$ in G) is called *right-closed* (resp. *left-closed*) if, for any $E \in \mathcal{G}$ and any $x \in G$, $Ex \in \mathcal{G}$ (resp. $xE \in \mathcal{G}$).

A σ-algebra generated by a subgroup and all its right cosets is right-closed. The following shows that right-closed σ-algebras have only this form.

Theorem 1. *For a finitely generated infinite group G, the following conditions on a measurable space $(G, \mathcal{G})$ are equivalent.*
(i) $\mathcal{G}$ is right-closed.
(ii) $M_{\mathcal{G}}(x) = M_{\mathcal{G}} x$ for all $x \in G$.
(iii) $M_{\mathcal{G}}$ is a subgroup of G, and $\mathcal{G}$ is generated by $M_{\mathcal{G}}$ and all its right cosets.

Proof. (i)⇒(ii): Suppose that (i) holds. Let $x \in G$. Since $M_{\mathcal{G}}(x) = \cap_{A \in \mathcal{M}_{\mathcal{G}}(x)} A$ and $M_{\mathcal{G}}x = (\cap_{A \in \mathcal{M}_{\mathcal{G}}(1_G)} A)x = \cap_{A \in \mathcal{M}_{\mathcal{G}}(1_G)} (Ax) = \cap_{B \in \mathcal{M}_{\mathcal{G}}(1_G)x} B$, it suffices to show that $\mathcal{M}_{\mathcal{G}}(x) = \mathcal{M}_{\mathcal{G}}(1_G)x$.

Let Ax, where $A \in \mathcal{M}_{\mathcal{G}}(1_G)$, be an arbitrary element of $\mathcal{M}_{\mathcal{G}}(1_G)x$. Since $1_G \in A$, $x = 1_G x \in Ax$ and so $Ax \in \mathcal{M}_{\mathcal{G}}(x)$ by (i). Thus $\mathcal{M}_{\mathcal{G}}(1_G)x \subset \mathcal{M}_{\mathcal{G}}(x)$. Conversely, if $A \in \mathcal{M}_{\mathcal{G}}(x)$, then $1_G = xx^{-1} \in Ax^{-1} \in \mathcal{M}_{\mathcal{G}}(1_G)$ by (i). Thus, $\mathcal{M}_{\mathcal{G}}(x) \subset \mathcal{M}_{\mathcal{G}}(1_G)x$.

(ii)⇒(iii): Suppose that (ii) holds. Let $a, b \in M_{\mathcal{G}}$. Since $b \in M_{\mathcal{G}}$, $M_{\mathcal{G}} = M_{\mathcal{G}}(b)$ by Proposition 1. Then, $a \in M_{\mathcal{G}}(b) = M_{\mathcal{G}}b$ by (ii), and so $ab^{-1} \in M_{\mathcal{G}}$. Therefore, $M_{\mathcal{G}}$ is a subgroup of G.

For any $E \in \mathcal{G}$, $E = \cup_{x \in E} M_{\mathcal{G}}(x)$ by Proposition 1. $M_{\mathcal{G}}(x) = M_{\mathcal{G}}x \in \mathcal{G}$ by (ii), and so $E = \cup_{x \in E} M_{\mathcal{G}}x$. Thus, $\mathcal{G}$ is generated by all right cosets of $M_{\mathcal{G}}$.

(iii)⇒(i): It is trivial. □

Analogous result holds for left-closed σ-algebras. By combining these, we get the following.

Corollary 1. *For a finitely generated infinite group G, the following conditions on a measurable space $(G, \mathcal{G})$ are equivalent.*

(i) $\mathcal{G}$ is both left- and right-closed.
(ii) $xM_{\mathcal{G}} = M_{\mathcal{G}}(x) = M_{\mathcal{G}}x$ for all $x \in G$.
(iii) $M_{\mathcal{G}}$ is a normal subgroup of G and $\mathcal{G}$ is generated by $M_{\mathcal{G}}$ and all its cosets.

Right-Invariance Property of Finitely Generated Infinite Groups. Right-invariance property is what belongs to a probability measure defined on a right-closed σ-algebra. When a probability space is right-invariant, any measurable set is, of course, right-invariant. Conversely, Proposition 1 and Theorem 1 imply that right-invariance of $M_{\mathcal{G}}$ is extended to the whole space.

Theorem 2. *For a finitely generated infinite group G, let $\mathcal{G}$ be a right-closed σ-algebra in G. $P(M_{\mathcal{G}}) = P(M_{\mathcal{G}}x)$ for all $x \in G$ if and only if $P(E) = P(Ex)$ for all $E \in \mathcal{G}$ and all $x \in G$.*

From Theorems 1 and 2, we have the following.

Corollary 2. *Let G be a finitely generated infinite group. If $(G, \mathcal{G}, P)$ is a right-invariant probability space, then $[G : M_{\mathcal{G}}]$ is finite and $P(M_{\mathcal{G}}x) = [G : M_{\mathcal{G}}]^{-1}$ for all $x \in G$. Therefore, if $[G : M_{\mathcal{G}}]$ is infinite, $(G, \mathcal{G}, P)$ cannot be right-invariant for any probability measure P.*

5 Universally Right-Invariant Probability Measure and Alternatives

Now we can decide whether or not right-invariance is allowable in a given situation. Suppose that it is allowable. Then, what are the concrete examples of the probability measure which is both *useful* and *practical* for such property?

5.1 Universally Right-Invariant Probability Measure

Given a right-closed measurable space $(G, \mathcal{G})$, if $M_{\mathcal{G}}$ is of finite-index, it is easy to get a probability measure that is right-invariant *only* on $(G, \mathcal{G})$. However, what is more meaningful is the one that is right-invariant on *any* right-closed σ-algebra $\mathcal{G}$ with finite-index $M_{\mathcal{G}}$. By Corollary 2, it can be defined as follows.

Definition 6. A probability measure P defined on an atomic measurable space $(G, 2^G)$ is called a *universally right-invariant probability measure on* G if $P(H) = P(Hx)$ for any finite-index subgroup H of G and any $x \in G$.

Most infinite groups that have emerged in cryptography are finitely generated residually-finite groups (e.g. free groups, groups of automorphisms of free groups, braid groups, etc.). A group is *residually-finite* if the intersection of all finite-index normal subgroups consists of only the identity. Here, we consider a larger class of groups, finitely generated groups with infinitely many finite-index subgroups. Finitely-generated residually-finite infinite groups belong to this class.

Theorem 3. *Let G be a finitely generated group with infinitely many finite-index subgroups. Then the intersection of all finite-index subgroups of G is a subgroup of G with infinite-index. Furthermore, G has no universally right-invariant probability measure.*

Proof. For the proof, we use the following fact.

Fact 1. Let G be a finitely generated infinite group. Then, for any $m \in \mathbb{N}$, G has only finitely many subgroups of index m.

Let $\mathcal{H}$ be the collection of all finite-index subgroups of G and $H_0 = \cap_{H \in \mathcal{H}} H$. Clearly H_0 is a subgroup of G. Assume that $[G : H_0] = k$ is finite. Then any $H \in \mathcal{H}$ has index k or less. By Fact 1, $\mathcal{H}$ is a finite set which contradicts to the hypothesis. Therefore, $[G : H_0]$ is infinite.

Assume that P is a universally right-invariant probability measure on G. Then for any $x \in G$ and any $H \in \mathcal{H}$,

$$P(H_0 x) \leq P(Hx) = P(H) = [G : H]^{-1}$$

by Corollary 2. Note that for any integer m there exists a finite-index subgroup H such that $[G : H] \geq m$ by Fact 1 and by the hypothesis. Thus $P(H_0 x) = 0$. Since H_0 is an infinite-index subgroup of G, there exist $x_1, x_2, \ldots \in G$ such that G is partitioned into $H_0 x_1, H_0 x_2, \ldots$. So $P(G) = \sum_{i=1}^{\infty} P(H_0 x_i) = 0$ which contradicts to $P(G) = 1$. Therefore, P cannot be universally right-invariant. □

Corollary 3. *Any finitely-generated residually-finite infinite group has no universally right-invariant probability measure.*

5.2 Alternatives

From Theorem 3, a question arises: what are weaker, yet practical alternatives to the universally right-invariant probability measure? We approach this question

via random walk on a free group $F = F(X)$, where $X = \{x_1, \ldots, x_m\}$. It is because any finitely generated infinite group is a homomorphic image of a finitely generated free group, and random walk yields a natural probability measure on F in the following sense: it generates all words of F with positive probability, and the longer the word is, the lower its occurrence probability is.

On the other hand, Theorems 1 and 2 reduce finding such an alternative measure to finding an atomic probability measure in an infinite group which is close to the uniform distribution over the family of all right-cosets of any finite-index subgroup. The latter has been studied independently in group theory for a long time. So we attempt to search for alternatives in the results from this area.

For $s \in (0,1)$, let W_s be a no-return random walk on the Cayley graph $C(F,X)$ of F with respect to the generating set X. See Appendix for Cayley graph. W_s starts at 1_F and either does nothing with probability s, or moves to one of the $2m$ adjacent vertices with equal probabilities $\frac{1-s}{2m}$. If W_s is at a vertex $v \neq 1_F$, it either stops at v with probability s, or moves with probability $\frac{1-s}{2m-1}$ to one of the $2m-1$ adjacent vertices lying away from 1_F producing a new freely reduced word $vx_i^{\pm 1}$. So $\Pr(|w| = k) = s(1-s)^k$ and the resulting atomic probability measure on F is

$$\mu_s(w) = \begin{cases} s & \text{if } w = 1_F, \\ \frac{s(1-s)^{|w|}}{2m(2m-1)^{|w|-1}} & \text{otherwise.} \end{cases}$$

Thus, $\mu_s(w)$ is the probability that the random walk W_s stops at w. From the results of Woess [25] and Borovik, Myasnikov, and Remeslennikov [5], for any finite-index subgroup H of F and any $x \in F$

$$\lim_{s \to 0} \mu_s(Hx) = [F : H]^{-1}.$$

On the other hand, for the case that we are working with only sufficiently long words, let's consider a variant of μ_s. For $k \in \mathbb{N}$, define

$$\bar{\mu}_k(w) = \begin{cases} 0 & \text{if } w \in B_k, \\ \frac{\mu_s(w)}{\mu_s(F \setminus B_k)} & \text{otherwise,} \end{cases}$$

where $B_k = \{w \in F \mid |w| \leq k\}$ is a ball of radius k. Then $\bar{\mu}_k$ is a probability measure on $(F, 2^F)$. From the results of Pak [18] and Borovik, Myasnikov, and Shpilrain [6], for any finite-index normal subgroup H of F

$$\frac{1}{2} \sum_{\bar{x} \in F/H} \left| \bar{\mu}_k(\bar{x}) - [F : H]^{-1} \right| = o(e^{-k}). \tag{4}$$

Discussion of Property of μ_s and $\bar{\mu}_k$. Let $(F, \mathcal{F})$ be a right-closed measurable space with $[F : M_{\mathcal{F}}] < \infty$. Suppose that $P_{\mathcal{F}}$ is the right-invariant probability measure on $(F, \mathcal{F})$. Then, by Proposition 1 and Theorem 1, μ_s has the following property. For any $E \in \mathcal{F}$

$$|\mu_s(E) - P_{\mathcal{F}}(E)| = \left| \sum_{i=1}^{t} \mu_s(M_{\mathcal{F}} x_i) - t P_{\mathcal{F}}(M_{\mathcal{F}}) \right|$$
$$\leq \sum_{i=1}^{t} \left| \mu_s(M_{\mathcal{F}} x_i) - [F : M_{\mathcal{F}}]^{-1} \right| \ \rightarrow \ 0 \quad \text{as } s \rightarrow 0,$$

where $M_{\mathcal{F}} x_i$'s are distinct right-cosets of $M_{\mathcal{F}}$ in F such that $E = \cup_{i=1}^{t} M_{\mathcal{F}} x_i$.

On the other hand, by the normality of H in (4), $\bar{\mu}_k$ has a slightly different property, so that it can be used in two cases. In the first case, let $(F, \mathcal{F})$ be a both left- and right-closed measurable space with $[F : M_{\mathcal{F}}] < \infty$. Then, by Corollary 1, $M_{\mathcal{F}}$ is a normal subgroup of F. Suppose that $P_{\mathcal{F}}$ is the right-invariant probability measure on $(F, \mathcal{F})$. Then, for any $E \in \mathcal{F}$

$$|\bar{\mu}_k(E) - P_{\mathcal{F}}(E)| \ \leq \ \frac{1}{2} \sum_{\bar{x} \in F/M_{\mathcal{F}}} \left| \bar{\mu}_k(\bar{x}) - [F : M_{\mathcal{F}}]^{-1} \right| \ = \ o(e^{-k}) \qquad (5)$$

for $k \rightarrow \infty$. The above inequality comes from the following fact.

Fact 2. Let Ω be a finite set, and let P_1 and P_2 be probability measures on $(\Omega, 2^{\Omega})$. Then,

$$\max_{E \subset \Omega} |P_1(E) - P_2(E)| = \frac{1}{2} \sum_{\omega \in \Omega} |P_1(\omega) - P_2(\omega)|.$$

In the second case, let $(F, \mathcal{F})$ be a right-closed measurable space such that $M_{\mathcal{F}}$ contains a finite-index normal subgroup N of F. Then, there exist distinct cosets, $Nx_1, \ldots, Nx_t$, of N in F such that $M_{\mathcal{F}} = \cup_{i=1}^{t} N x_i$. Let $P_{\mathcal{F}}$ be the right-invariant probability measure on $(F, \mathcal{F})$. Then, from Fact 2, for any $E \in \mathcal{F}$

$$|\bar{\mu}_k(E) - P_{\mathcal{F}}(E)| \leq \frac{1}{2} \sum_{M_{\mathcal{F}} x \in \mathcal{R}} \left| \bar{\mu}_k(M_{\mathcal{F}} x) - [F : M_{\mathcal{F}}]^{-1} \right|$$
$$\leq \frac{1}{2} \sum_{M_{\mathcal{F}} x \in \mathcal{R}} \sum_{i=1}^{t} \left| \bar{\mu}_k(N x_i x) - [F : N]^{-1} \right| \ = \ o(e^{-k})$$

for $k \rightarrow \infty$, where $\mathcal{R}$ is the set of all right-cosets of $M_{\mathcal{F}}$ in F.

Discussion of Alternatives. Given a group G, a good alternative to the universally right-invariant probability measure may be a probability measure P on $(G, 2^G)$ such that for any right-invariant probability space $(G, \mathcal{G}, P_{\mathcal{G}})$ and for any $E \in \mathcal{G}$, $|P(E) - P_{\mathcal{G}}(E)|$ is very small. Here, we should be careful with the word, "small". Small in what? The factors which determine the value of $|P(E) - P_{\mathcal{G}}(E)|$ come from the characteristics of G, $\mathcal{G}$, and P. Note that the group G is given, the σ-algebra $\mathcal{G}$ is arbitrarily selected to some extent, and we are discussing the measure P. So focusing on P, it seems more reasonable to view P not as a single probability measure but as a family of probability measures indexed by factors representing its characteristics. For example, $\mu = \{\mu_s\}_{s \in (0,1)}$ and $\bar{\mu} = \{\bar{\mu}_k\}_{k \in \mathbb{N}}$. From this point of view, let's define our alternative in general terms.

Let $P = \{P_\alpha\}_{\alpha \in \mathcal{A}}$ be a family of probability measures on $(G, 2^G)$ for an index set $\mathcal{A}$. And let some α_0 be given. For any right-invariant probability space $(G, \mathcal{G}, P_\mathcal{G})$ and for any $E \in \mathcal{G}$, P has the following property.

$$\lim_{\alpha \to \alpha_0} |P_\alpha(E) - P_\mathcal{G}(E)| = 0$$

μ serves as a good example of this alternative. On the other hand, $\bar{\mu}$ can serve as another example if $(G, \mathcal{G}, P_\mathcal{G})$ is a both left- and right-invariant probability space, or if $(G, \mathcal{G}, P_\mathcal{G})$ is a right-invariant probability space and $M_\mathcal{G}$ contains a finite-index normal subgroup of G. In these cases, $|P_\alpha(E) - P_\mathcal{G}(E)|$ decreases exponentially.

6 Applications

This section shows two basic examples of how to apply our theory to real situations via recent works. These works are based on braid groups. For a survey of braid-group-based cryptography, see [14].

For $n \geq 2$, the *n-braid group* B_n can be presented by $(n-1)$-generators $\sigma_1, \ldots, \sigma_{n-1}$ and two kinds of relations: $\sigma_i \sigma_j = \sigma_j \sigma_i$ for $|i-j| > 1$ and $\sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j$ for $|i-j| = 1$. For the symmetric group S_n on n-letters, there is a natural projection $\pi : B_n \to S_n$ sending σ_i to the transposition $(i, i+1)$. $\pi(x)$ is written interchangeably with π_x. Define $P_n = \ker(\pi)$ and call its elements *pure braids*.

6.1 The Case That Right-Invariance Is Not Allowable

Sibert, Dehornoy, and Girault [20] proposed entity authentication schemes using braid groups: Schemes I, II, II', III. As a two-pass scheme, Scheme I is perfectly honest-verifier zero-knowledge. As three-pass protocols, the other schemes were shown to be zero-knowledge under the assumption that the probability space is right-invariant (to polynomial-time distinguishers). Their assumption was made from some experiment over a certain finite subset of B_n.

This section discusses the security of Scheme II on the whole group B_n by disproving the assumption for zero-knowledge. Analogous arguments apply to Schemes II', III. Let's see Scheme II. Prover's secrete key is $z \in B_n$, and public key is $(b, b') \in B_n^2$, where $b' = zbz^{-1}$. Its three-pass process is given in Fig. 1.

Assumption for Perfect Zero-Knowledge. For perfect zero-knowledge of Scheme II, it is assumed that the distributions of r and rz^{-1} are identical, where $r \in_R B_n$. We show that they cannot be identical by defining a distinguisher $\mathcal{A}$ as follows.

$$\mathcal{A}: \text{ “On an input } x \in B_n, \text{ output 1 if } x = 1_{B_n}, \text{ and 0 otherwise.”} \qquad (6)$$

Since verifying that any two braids are identical can be done very efficiently, $\mathcal{A}$ is also efficient. Then the situation comparing the distributions of r and rz^{-1} by using the algorithm $\mathcal{A}$ yields the atomic σ-algebra 2^{B_n} as the right-closed σ-algebra in B_n. So, right-invariance is not allowable in this situation.

Prover		Verifier
$r \in_R B_n$		
$x = rbr^{-1}$	$\xrightarrow{x}$	
	$\xleftarrow{\epsilon}$	$\epsilon \in_R \{0,1\}$
$y = \begin{cases} r & \text{if } \epsilon = 0, \\ rz^{-1} & \text{otherwise.} \end{cases}$	$\xrightarrow{y}$	$x = \begin{cases} yby^{-1} & \text{if } \epsilon = 0, \\ yb'y^{-1} & \text{otherwise.} \end{cases}$

Fig. 1. Scheme II

Assumption for Computational Zero-Knowledge. For computational zero-knowledge of Scheme II, it is assumed that the distributions of r and rz^{-1} are computationally indistinguishable, where $r \in_R B_n$. This means that, for any polynomial-time distinguisher $\mathcal{A}$, $\left|\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Xz^{-1}) = 1]\right|$ is negligible. Here X is a random variable over B_n.

By using the algorithm (6), we show that it is not negligible in the word length of the secrete key z with respect to the probability measure μ_s which is defined on a free group F generated by $\{x_1, \ldots, x_{n-1}\}$. Considering a natural projection $\phi : F \to B_n$ defined by $x_i \mapsto \sigma_i$, let $K = \phi^{-1}(1_{B_n})$ and let the random variable X have the probability distribution induced by μ_s. Then

$$\Pr[\mathcal{A}(X) = 1] = \mu_s(K) \geq \mu_s(1_F) = s.$$

Let $\ell = \min_{w \in \phi^{-1}(z)} |w|$, and let $w_0 \in \phi^{-1}(z)$ satisfy $|w_0| = \ell$. Then

$$\Pr[\mathcal{A}(Xz^{-1}) = 1] = \mu_s(Kw_0) = \sum_{k=0}^{\infty} \mu_s(Kw_0 \cap C_k) = \sum_{k=0}^{\infty} s(1-s)^k \frac{|Kw_0 \cap C_k|}{|C_k|},$$

where $C_k = \{w \in F \mid |w| = k\}$. Note that $Kw_0 \cap C_k = \emptyset$ for $0 \leq k < \ell$. Thus,

$$\Pr[\mathcal{A}(Xz^{-1}) = 1] = s(1-s)^\ell \sum_{k=0}^{\infty} (1-s)^k \frac{|Kw_0 \cap C_{\ell+k}|}{|C_{\ell+k}|} \leq s(1-s)^\ell \sum_{k=0}^{\infty} (1-s)^k = (1-s)^\ell.$$

Therefore, $\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Xz^{-1}) = 1] \geq s - (1-s)^\ell$.

6.2 The Case That Right-Invariance Is Allowable

For notational convenience, this section assumes that n is even. Define B_ℓ (resp. B_u) be a subgroup of B_n generated by $\sigma_1, \ldots, \sigma_{n/2-1}$ (resp. $\sigma_{n/2+1}, \ldots, \sigma_{n-1}$). Likewise, define S_ℓ (resp. S_u) be a subgroup of the symmetric group S_n generated by $(1,2), \ldots, (\frac{n}{2}-1, \frac{n}{2})$ (resp. $(\frac{n}{2}+1, \frac{n}{2}+2), \ldots, (n-1, n)$). Then, any two elements chosen from B_ℓ and B_u (resp. S_ℓ and S_u) commute with each other. The *decisional Diffie-Hellman-type conjugacy problem in* B_n is defined as follows.

> Given $(a, w_\ell^{-1} a w_\ell, w_u^{-1} a w_u, x_u^{-1} x_\ell^{-1} a x_\ell x_u)$, distinguish $x_u^{-1} x_\ell^{-1} a x_\ell x_u$ and $w_u^{-1} w_\ell^{-1} a w_\ell w_u$, where $a \in B_n$, $w_\ell, x_\ell \in B_\ell$, and $w_u, x_u \in B_u$.

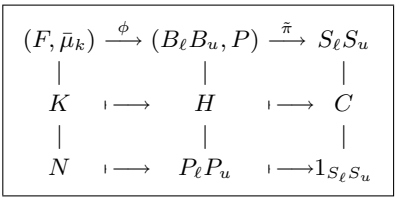


Fig. 2. Correspondences among groups

This problem is used as an underlying problem of a public-key encryption [13], pseudorandom number generator, and pseudorandom synthesizer [15]. Gennaro and Micciancio [10] proposed how to solve it for some parameters. We supplement their attack with quantifying the success probability of their adversary. The adversary is described as follows.

$\mathcal{A}$: "On an input $(a,\ w_\ell^{-1}aw_\ell,\ w_u^{-1}aw_u,\ x_u^{-1}x_\ell^{-1}ax_\ell x_u)$
where $a \in B_n \backslash P_n$, $w_\ell, x_\ell \in B_\ell$, and $w_u, x_u \in B_u$,
1. find any $\theta \in S_\ell$ such that $\theta^{-1}\pi_a\theta = \pi(w_\ell^{-1}aw_\ell)$;
2. output 1 if $\pi(x_u^{-1}x_\ell^{-1}ax_\ell x_u) = \theta^{-1}\pi(w_u^{-1}aw_u)\theta$, and 0 otherwise."

Define $B_\ell B_u = \{xy \mid x \in B_\ell, y \in B_u\}$ and $S_\ell S_u = \{\tau\omega \mid \tau \in S_\ell, \omega \in S_u\}$. Then they are subgroups of B_n and S_n, respectively. Let $C = Z_{S_\ell S_u}(\pi_a)$. Since θ (at Step 1) can be easily and perfectly computed and such θ satisfies $\theta^{-1}\pi(w_u^{-1}aw_u)\theta = \pi(w_u^{-1}w_\ell^{-1}aw_\ell w_u)$, the success probability equals

$$\Pr[\mathcal{A}(a, w_\ell^{-1}aw_\ell, w_u^{-1}aw_u, X^{-1}aX) = 0] = \Pr[\pi(X) \in C\pi(w_\ell w_u)], \qquad (7)$$

where X is a random variable over $B_\ell B_u$.

Deciding Whether Right-Invariance Is Allowable or Not. Restricting π defined on B_n to $B_\ell B_u$ induces another natural projection $\tilde{\pi} : B_\ell B_u \to S_\ell S_u$. Define $H = \tilde{\pi}^{-1}(C)$ and $P_\ell P_u = \ker(\tilde{\pi})$. See Fig. 2. Then H is a subgroup of $B_\ell B_u$, $\Pr[\pi(X) \in C\pi(w_\ell w_u)] = \Pr[X \in Hw_\ell w_u]$, and $P_\ell P_u$ is a normal subgroup of $B_\ell B_u$ contained in H. Define $\mathcal{B}$ as the σ-algebra in $B_\ell B_u$ generated by all cosets of $P_\ell P_u$. Then $H \in \mathcal{B}$ and $\mathcal{B}$ is both left- and right-closed. Since $[B_\ell B_u : M_\mathcal{B}] = [B_\ell B_u : P_\ell P_u] = ((\frac{n}{2})!)^2$ is finite, we can use right-invariance property in order to compute the success probability $\Pr[X \in Hw_\ell w_u]$.

Computing the Success Probability. Let $F = F(\{x_1, \ldots, x_{n/2-1}, x_{n/2+1}, \ldots, x_{n-1}\})$ be a free group. Then, there is a natural projection $\phi : F \to B_\ell B_u$ defined by $x_i \mapsto \sigma_i$. Let $K = \phi^{-1}(H)$ and $N = \phi^{-1}(P_\ell P_u)$. See Fig. 2. Let $\mathcal{F}$ be the σ-algebra in F generated by all cosets of N. Since N is a finite-index normal subgroup of F and $M_\mathcal{F} = N$, $\bar{\mu}_k$ can be used on $(F, \mathcal{F})$ for right-invariance.

Define a set function $P : \mathcal{B} \to [0, 1]$ by $P(E) = \bar{\mu}_k(\phi^{-1}(E))$ for all $E \in \mathcal{B}$. Since $\mathcal{F} = \{\phi^{-1}(E) \mid E \in \mathcal{B}\}$, P is a probability measure on $(B_\ell B_u, \mathcal{B})$. Let the

random variable X in (7) induce P. Then, $\Pr[X \not\in Hw_\ell w_u] = 1 - P(Hw_\ell w_u)$. On the other hand, from the definition of P and (5)

$$\left|P(Hw_\ell w_u) - [B_\ell B_u : H]^{-1}\right| = \left|\bar{\mu}_k(K\phi^{-1}(w_\ell w_u)) - [K : N]/[F : N]\right| = o(e^{-k}).$$

Therefore, the success probability of the adversary is

$$1 - [B_\ell B_u : H]^{-1} - o(e^{-k}) \le 1 - P(Hw_\ell w_u) \le 1 - [B_\ell B_u : H]^{-1} + o(e^{-k}).$$

Note that $[B_\ell B_u : H] = [S_\ell S_u : C]$ and $C = Z_{S_\ell S_u}(\pi_a)$. So $[B_\ell B_u : H]$ can be evaluated if π_a is specified. For all $a \in B_n \backslash P_n$, its upper bound is $(n/2)!^2$, and lower bound is $n(n-2)/8$ for $n \ge 10$ from the following theorem.

Theorem 4. *If $\alpha \in S_n \backslash \{1_{S_n}\}$, $Z_{S_\ell S_u}(\alpha)$ is a proper subgroup of $S_\ell S_u$ for $n \ge 6$. Precisely,*

$$[S_\ell S_u : Z_{S_\ell S_u}(\alpha)] \;\ge\; \begin{cases} n(n-2)/8 & \text{for } n \ge 10, \\ 3 & \text{for } n = 8, \\ 2 & \text{for } n = 6. \end{cases}$$

Proof. Let $\alpha \in S_n$, and let $\alpha_1, \ldots, \alpha_s$ be disjoint cycles in S_n such that

$$\alpha = \alpha_1 \cdots \alpha_s \quad \text{and} \quad \alpha_i \in \begin{cases} S_\ell & \text{for } 1 \le i \le t_\ell, \\ S_n \backslash S_\ell S_u & \text{for } t_\ell < i \le t_u, \\ S_u & \text{for } t_u < i \le s, \end{cases}$$

for some $0 \le t_\ell \le t_u \le s$. Let

$$\alpha_\ell = \alpha_1 \cdots \alpha_{t_\ell} \in S_\ell,\ \tilde{\alpha} = \alpha_{t_\ell+1} \cdots \alpha_{t_u} \in (S_n \backslash S_\ell S_u) \cup \{1_{S_n}\},\ \alpha_u = \alpha_{t_u+1} \cdots \alpha_s \in S_u.$$

For every $1 \le i \le s$, let $\alpha_i = (a_{k_{i-1}+1}, \ldots, a_{k_i})$ with $k_0 = 0$. Then

$$\alpha = (a_1, \ldots, a_{k_1})(a_{k_1+1}, \ldots, a_{k_2}) \cdots (a_{k_{s-1}+1}, \ldots, a_{k_s}).$$

Note that for any $\tau \in S_n$, the cycle decomposition of $\tau\alpha\tau^{-1}$ is as follows.

$$\tau\alpha\tau^{-1} = (\tau(a_1), \ldots, \tau(a_{k_1}))(\tau(a_{k_1+1}), \ldots, \tau(a_{k_2})) \cdots (\tau(a_{k_{s-1}+1}), \ldots, \tau(a_{k_s}))$$

Let $\tau \in Z_{S_\ell S_u}(\alpha)$. Then $\tau\alpha_1\tau^{-1}, \ldots, \tau\alpha_s\tau^{-1}$ are disjoint cycles of α. If $\alpha_i \in S_\ell$, $\alpha_j \in S_n \backslash S_\ell S_u$, and $\alpha_k \in S_u$, then $\tau\alpha_i\tau^{-1} \in S_\ell$, $\tau\alpha_j\tau^{-1} \in S_n \backslash S_\ell S_u$, and $\tau\alpha_k\tau^{-1} \in S_u$ for all i, j, k. So $\tau\alpha_\ell\tau^{-1} = \alpha_\ell$, $\tau\tilde{\alpha}\tau^{-1} = \tilde{\alpha}$, and $\tau\alpha_u\tau^{-1} = \alpha_u$. Namely, $\tau \in Z_{S_\ell S_u}(\alpha_\ell) \cap Z_{S_\ell S_u}(\tilde{\alpha}) \cap Z_{S_\ell S_u}(\alpha_u)$. On the other hand, it is clear that $Z_{S_\ell S_u}(\alpha_\ell) \cap Z_{S_\ell S_u}(\tilde{\alpha}) \cap Z_{S_\ell S_u}(\alpha_u) \subset Z_{S_\ell S_u}(\alpha)$. So

$$Z_{S_\ell S_u}(\alpha) = Z_{S_\ell S_u}(\alpha_\ell) \cap Z_{S_\ell S_u}(\tilde{\alpha}) \cap Z_{S_\ell S_u}(\alpha_u).$$

Let $\alpha \not=_{S_n} 1$, and let $\tau = \tau_\ell\tau_u \in S_\ell S_u$ mean that $\tau_\ell \in S_\ell$ and $\tau_u \in S_u$.

Case 1. $\alpha_\ell\alpha_u \not=_{S_n} 1$: WLOG, let $\alpha_\ell \not=_{S_n} 1$. Define $\ell_1 = |\{1 \le i \le n/2 \mid \alpha_\ell(i) = i\}|$ and ℓ_i as the number of i-cycles of α_ℓ for $2 \le i \le n/2$. Then

$$|Z_{S_\ell}(\alpha_\ell)| \;=\; \prod_{i=1}^{n/2} i^{\ell_i}(\ell_i)!.$$

Table 1. Maximum of $|Z_{S_\ell}(\alpha_\ell)|$ and minimum of $[S_\ell : Z_{S_\ell}(\alpha_\ell)]$

$\frac{n}{2}$	max. of $\lvert Z_{S_\ell}(\alpha_\ell)\rvert$	min. of $[S_\ell : Z_{S_\ell}(\alpha_\ell)]$	number of cycles
3	3	2	$\ell_3 = 1$, $\ell_k = 0$ for $k \neq 3$
4	8	3	$\ell_2 = 2$, $\ell_k = 0$ for $k \neq 2$
≥ 5	$2 \times (\frac{n}{2} - 2)!$	$\frac{n}{4}(\frac{n}{2} - 1)$	$\ell_1 = \frac{n}{2} - 2$, $\ell_2 = 1$, $\ell_k = 0$ for $k \geq 3$

Table 1 shows the maximum values of $|Z_{S_\ell}(\alpha_\ell)|$ and the corresponding values of $[S_\ell : Z_{S_\ell}(\alpha_\ell)]$ over $\alpha_\ell \in S_\ell \backslash \{1_{S_n}\}$. Since $[S_\ell S_u : Z_{S_\ell S_u}(\alpha)] \geq [S_\ell S_u : Z_{S_\ell S_u}(\alpha_\ell)] = [S_\ell : Z_{S_\ell}(\alpha_\ell)]$, for all $\alpha \in S_n$ such that $\alpha_\ell \alpha_u \neq 1_{S_n}$

$$[S_\ell S_u : Z_{S_\ell S_u}(\alpha)] \;\geq\; \begin{cases} n(n-2)/8 & \text{for } n \geq 10, \\ 3 & \text{for } n = 8, \\ 2 & \text{for } n = 6. \end{cases}$$

Case 2. $\alpha_\ell \alpha_u = 1_{S_n}$: In this case, $Z_{S_\ell S_u}(\alpha) = Z_{S_\ell S_u}(\tilde{\alpha})$. Define

$$A_\ell = \left\{ 1 \leq i \leq \frac{n}{2} \mid \tilde{\alpha}(i) \neq i \right\}, \; A_u = \left\{ \frac{n}{2} < i \leq n \mid \tilde{\alpha}(i) \neq i \right\}, \; N_\ell = |A_\ell|, \; N_u = |A_u|.$$

WLOG, we assume $1 \leq N_u \leq N_\ell \leq n/2$. Note that for any $\tau_\ell \tau_u \in Z_{S_\ell S_u}(\tilde{\alpha})$, $\{(i, \tau_\ell(i)) \mid i \in A_\ell\}$ is uniquely determined by $\{(i, \tau_u(i)) \mid i \in A_u\}$. So

$$|Z_{S_\ell S_u}(\tilde{\alpha})| \leq \left(\frac{n}{2} - N_\ell\right)! \left(\frac{n}{2} - N_u\right)! N_u! \leq \begin{cases} (n/2-1)!^2 & \text{if } N_u < n/2, \\ (n/2)! & \text{if } N_u = n/2, \end{cases}$$

$$\leq \begin{cases} (n/2-1)!^2 & \text{if } n \geq 8, \\ 6 & \text{if } n = 6. \end{cases}$$

Therefore, for all $\alpha \in S_n \backslash \{1_{S_n}\}$ such that $\alpha_\ell \alpha_u = 1_{S_n}$

$$[S_\ell S_u : Z_{S_\ell S_u}(\alpha)] \;\geq\; \begin{cases} (n/2)^2 & \text{if } n \geq 8, \\ 6 & \text{if } n = 6. \end{cases}$$

From Cases 1 and 2, the conclusion follows. □

7 Conclusions

We know that it is impossible to overestimate the role of the uniform distribution in cryptography. However, no infinite group has such a nice distribution. Noticing that this fact is an impediment to the use of infinite groups for cryptography, this paper has formalized the notion of right-invariance on an infinite group which in a sense corresponds to the uniform distribution on a finite set, and then shown *when* and *how* this notion can be used for infinite-group-based cryptography.

Our work is a first attempt to formalize and resolve probability-theoretic problems arising in the process of using infinite groups for cryptography. Although our work cannot resolve all the problems, we hope that it contributes to widening the scope of what provably secure cryptosystems can be built on. We close this paper with the following research topics.

- Find different types of alternatives to the universally right-invariant probability measure from ours.
- Find more various examples of practical problems which right-invariance can resolve in cryptography.
- For complex problems (e.g. proving security of a cryptosystem), discover, formalize, and solve its constituent problems other than right-invariance.

Acknowledgements

The author would like to thank Prof. Kouichi Sakurai and the anonymous referees for helpful remarks and suggestions.

References

1. D. Angluin and D. Lichtenstein, *Provable Security of Cryptosystmes: A Survey*, Computer Science Department, Yale University, TR-288, 1983
2. I. Anshel, M. Anshel, B. Fisher, and D. Goldfeld, *New Key Agreement Protocols in Braid Group Cryptography*, CT-RSA 2001, LNCS **2020**, 13–27, 2001
3. S.R. Blackburn, *Cryptanalysis of two cryptosystems based on group actions*, ASIACRYPT '99, LNCS **1716**, 52–61, 1999
4. M. Blum and S. Micali, *How to Generate Cryptographically Strong Sequences of Pseudorandom Bits*, SIAM J. Comput. **13**, 850–864, 1984
5. A.V. Borovik, A.G. Myasnikov, and V.N. Remeslennikov, *Multiplicative Measures on Free Groups*, To appear in Internat. J. Algebra Comp.
6. A.V. Borovik, A.G. Myasnikov, and V. Shpilrain, *Measuring Sets in Infinite Groups*, Contemporary Mathematics **298**, 21–42, 2002
7. J.H. Cheon and B. Jun, *A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem*, CRYPTO 2003, LNCS **2729**, 212–225, 2003
8. J. Feigenbaum, *Locally Random Reductions in Interactive Complexity Theory*, Advances in Computational Complexity Theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **13**, AMS, 73–98, 1993
9. M. Garzon and Y. Zalcstein, *The Complexity of Grigorchuk Groups with Application to Cryptography*, Theoretical Computer Sciences **88**, 83–88, 1991
10. R. Gennaro and D. Micciancio, *Cryptanalysis of a Pseudorandom Generator Based on Braid Groups*, EUROCRYPT 2002, LNCS **2332**, 1–13, 2002
11. D. Hofheinz and R. Steinwandt, *A Practical Attack on Some Braid Group based Cryptographic Primitives*, PKC 2003, LNCS **2567**, 187–198, 2003
12. J. Hughes, *A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem*, ACISP 2002, LNCS **2384**, 176–189, 2002
13. K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, and C. Park, *New Public-key Cryptosystem Using Braid Groups*, CRYPTO 2000, LNCS **1880**, 166–183, 2000

14. E. Lee, *Braid Groups in Cryptology*, IEICE Trans. Fund. **E87-A**, 986-992, 2004
15. E. Lee, S.J. Lee, and S.G. Hahn, *Pseudorandomness from Braid Groups*, CRYPTO 2001, LNCS **2139**, 486–502, 2001
16. E. Lee and J.H. Park, *Cryptanalysis of the Public-key Encryption based on Braid Groups*, EUROCRYPT 2003, LNCS **2565**, 477–490, 2003
17. S. J. Lee and E. Lee, *Potential Weaknesses of the Commutator Key Agreement Protocol based on Braid Groups*, EUROCRYPT 2002, LNCS **2332**, 14–28, 2002
18. I. Pak, *Random Walks on Finite Groups with Few Random Generators*, Electronic J. of Prob. **4**, 1–11, 1999
19. P.W. Shor, *Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a quantum Computer*, SIAM J. Comput. **26**, 1484–1509, 1997
20. H. Sibert, P. Dehornoy, and M. Girault, *Entity Authentication Schemes Using Braid Word Reduction*, Proceedings International Workshop on Coding and Cryptography, March 24-28 2003, Versailles (France), 153–164
21. R. Siromoney and L. Mathew, *A Public key Cryptosystem based on Lyndon Words*, Information Processing Letters **35**, 33–36, 1990
22. A. Yamamura, *Public-Key Cryptosystems Using the Modular Group*, PKC '98, LNCS **1431**, 203–216, 1998
23. A. Yamamura, *A Functional Cryptosystem Using a Group Action*, ACISP '99, LNCS **1587**, 314–325, 1999
24. N.R. Wagner and M.R. Magyarik, *A Public-key Cryptosystem based on the Word Problem*, CRYPTO '84, LNCS **196**, 19–36, 1984
25. W. Woess, *Cogrowth of groups and simple Random Walks*, Arch. Math. **41**, 363–370, 1983

Appendix: Cayley Graph

The *Cayley graph* $C(G, X)$ of a group G with a generating set X is a graph such that the vertices are in one-to-one correspondence with the group elements and there is a (directed) edge from the vertex labelled by v to the vertex labelled by vx for each $v \in G$ and $x \in X \cup X^{-1}$. So if G is an infinite group, its Cayley graph is also an infinite graph. The Cayley graph is a metric space by defining the length of each edge to be the unit length. The distance between two vertices v, w in the Cayley graph is exactly the shortest word-length of $v^{-1}w$ with respect to the given generating set.

Practical Two-Party Computation Based on the Conditional Gate

Berry Schoenmakers[1,⋆] and Pim Tuyls[2]

[1] Dept. of Mathematics and Computing Science, TU Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
berry@win.tue.nl
[2] Philips Research Labs,
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
pim.tuyls@philips.com

Abstract. We present new results in the framework of secure multiparty computation based on homomorphic threshold cryptosystems. We introduce the *conditional gate* as a special type of multiplication gate that can be realized in a surprisingly simple and efficient way using just standard homomorphic threshold ElGamal encryption. As addition gates are essentially for free, the conditional gate not only allows for building a circuit for any function, but actually yields efficient circuits for a wide range of tasks.

1 Introduction

Homomorphic threshold cryptosystems provide a basis for secure multiparty computation in the cryptographic model [FH96, JJ00, CDN01, DN03]. For a given n-ary function f, one composes a circuit C of elementary gates that given encryptions of $x_1, \ldots, x_n$ on its input wires, produces an encryption of $f(x_1, \ldots, x_n)$ on its output wire. The elementary gates operate in the same fashion. The wires of the entire circuit C are all encrypted under the same public key; the corresponding private key is shared among a group of parties. It is customary to distinguish addition gates and multiplication gates. Addition gates can be evaluated without having to decrypt any value, taking full advantage of the homomorphic property of the cryptosystem. Multiplication gates, however, require at least one threshold decryption to succeed even for an honest-but-curious (passive) adversary. To deal with a malicious (active) adversary, multiplication gates additionally require the use of zero-knowledge proofs.

While the result of [FH96] covers the case of a passive adversary only, an interesting feature is that it covers both the two-party case ($n = 2$) and the multiparty case ($n > 2$) in a *uniform* way. The later papers [JJ00, CDN01, DN03] do cover an active adversary, but only consider the multiparty case. In the present paper, we are particularly interested in extending the use of homomorphic threshold

⋆ Work done partly while visiting Philips Research Labs.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 119–136, 2004.

cryptosystems to the two-party case. We observe that solutions based on homomorphic threshold cryptosystems can be used just as well in the two-party case. To cover fairness, however, an additional protocol is needed that allows two parties to jointly decrypt the outputs in a gradual fashion. We present such a protocol by showing how to adapt the decryption step of a homomorphic threshold cryptosystem.

A major advantage of secure multiparty computation based on homomorphic threshold cryptosystems is the fact that it results in particularly efficient solutions, even for active adversaries. The communication complexity, which is the dominating complexity measure, is $O(nk|C|)$ bits for [JJ00, CDN01, DN03], where n is the number of parties, k is a security parameter, and $|C|$ is the number of gates of circuit C. A more detailed look at the performance of these solutions reveals, however, that there is considerable room for improvement in several respects.

It is assumed in [JJ00, CDN01, DN03] that the shared key for the homomorphic threshold cryptosystem used in the multiparty protocol is already given. As a consequence, the communication complexity of $O(nk|C|)$ bits does not include the communication needed for the distributed key generation (DKG) protocol of the underlying threshold cryptosystem. However, the performance of the DKG protocol is an issue since we envision a system supporting *ad hoc contacts* among a large group of peer users, where any pair of users may decide to engage in a secure two-party computation for a dynamically agreed upon function. For example, "profile matching" is an application in which two users jointly test whether some function of their (personal) profiles exceeds a given threshold, without divulging any further information on their profiles. In this scenario, it is unreasonable to assume that each pair of users shares a specific key pair for the underlying threshold cryptosystem. Instead, each time two users want to perform a two-party computation, they would need to run the DKG protocol first.

In this respect, an advantage of the Mix and Match approach of [JJ00] is its applicability to any discrete log setting, whereas [CDN01, DN03] depend critically on an RSA-like setting (e.g., using Paillier's cryptosystem). The advantage is that DKG protocols for discrete log based cryptosystems are efficient and relatively simple (see [Ped91, GJKR99]). In particular, DKG can be achieved practically for free in the two-party case. This contrasts sharply with the known protocols for distributed generation of a shared RSA modulus. Briefly, for the two-party case (without a helper party), Gilboa [Gil99] reports a communication complexity of about 42MB (or 29MB for a slightly optimized version) for generating a shared 1024-bit RSA modulus, while covering passive adversaries only. And, for the multiparty case, the results of [ACS02] show what is currently achievable, also covering passive adversaries only.

Interestingly, it is actually possible to combine the benefits of a discrete log setting and an RSA-like setting, as demonstrated recently in [DJ03]. To this end, one uses an amalgam of the ElGamal cryptosystem and the Paillier cryptosystem (such a combination has also been presented in the full version of [CS02]). A

system supporting ad hoc contacts may then be set up by jointly generating a single RSA modulus (between as many parties as deemed necessary, e.g., using a robust version of [ACS02]). A discrete log based DKG protocol will suffice to generate a shared key between any two users. We do note however that the security of the resulting system relies on both a discrete log related assumption and a factoring related assumption, which is undesirable from a theoretical point of view.

In this paper, we will focus on a solution for which the security depends on the standard decisional Diffie-Hellman (DDH) assumption. As a consequence our protocols can be implemented using elliptic curves, for which the security is assumed to be exponential as a function of the security parameter rather than sub-exponential (as for RSA, for example). The Mix and Match approach of [JJ00] is also secure under DDH, but we note that the resulting protocol for evaluating multiplication gates is—despite its conceptual simplicity—quite inefficient. We will show how to evaluate multiplication gates in a much simpler way, such that the computational effort decreases by at least one order of magnitude (that is, a ten-fold speed-up is achieved, see Section 3.3). On the other hand, a disadvantage of our approach is that, in general, the round complexity is $O(nd)$ for n parties and circuit depth $d = d(C)$, versus $O(n+d)$ for Mix and Match. For two-party computation, however, the round complexity is $O(d)$ in both cases, and more generally for small n the gain in computational efficiency outweighs the increased round complexity.

The basis of our approach is formed by the *conditional* gate, a special multiplication gate which we show to be efficiently implementable under DDH. Basically, a conditional gate allows us to efficiently multiply two encrypted values x and y, as long as x is restricted to a two-valued domain, e.g., $x \in \{0, 1\}$. We emphasize that the value of y is not restricted, e.g., we may have $y \in \mathbb{Z}_q$, where q is a large prime. This property can be exploited when designing circuits for specific functions. For example, from the formula $(y'_0, y'_1) = (y_0 - x(y_0 - y_1), y_1 + x(y_0 - y_1))$, with $x \in \{0, 1\}$, one sees that a conditional swap gate, swapping any two values in $\mathbb{Z}_q$ depending on the value of x, can be obtained using a single conditional gate. We will indicate that, using ElGamal, one cannot expect to achieve a multiplication gate for which both inputs are unrestricted. Note, however, that the result of [CDN01] shows that multiplication of two unrestricted values can be achieved efficiently under a factoring related assumption.

Overview. Throughout the paper we will describe the results in a general setting of n-party computation, $n \geq 2$, although we are mainly interested in the two-party case. In Section 2, we review the basics for homomorphic threshold ElGamal. In Section 3, we introduce the conditional gate as our elementary multiplication gate, and we show how it can be used to achieve xor-homomorphic ElGamal encryption efficiently. In Section 4, we then consider the secure evaluation of arbitrary circuits, following [CDN01], which we extend with a new, non-interactive protocol for achieving private outputs. Furthermore, we propose an efficient protocol for achieving fairness in the two-party case. In Section 5, we show that particularly efficient circuits can be built for basic operations such

as integer comparison, paying special attention to Yao's well-known millionaires problem in Section 5.2, for which we obtain a solution requiring $12m$ exponentiations for m-bit integers. Finally, in Section 6, we conclude with future work and give an example of a more advanced application which we call 'profile matching'.

2 Preliminaries on Homomorphic Threshold ElGamal

Discrete Log Setting. Let $G = \langle g \rangle$ denote a finite cyclic (multiplicative) group of prime order q for which the Decision Diffie-Hellman (DDH) problem is assumed to be infeasible: given $g^\alpha, g^\beta, g^\gamma \in_R G$, it is infeasible to decide whether $\alpha\beta \equiv \gamma \pmod q$. This implies that the Diffie-Hellman (DH) problem, which is to compute $g^{\alpha\beta}$ given $g^\alpha, g^\beta \in_R G$, is infeasible as well. In turn, this implies that the Discrete Log (DL) problem, which is to compute $\log_g h = \alpha$ given $g^\alpha \in_R G$, is infeasible.

Homomorphic ElGamal Encryption. For public key $h \in G$, a message $m \in \mathbb{Z}_q$ is encrypted as a pair $(a, b) = (g^r, g^m h^r)$, with $r \in_R \mathbb{Z}_q$. Encryption is *additively* homomorphic: given encryptions $(a, b), (a', b')$ of messages m, m', respectively, an encryption of $m + m'$ is obtained as $(a, b) \star (a', b') = (aa', bb') = (g^{r+r'}, g^{m+m'} h^{r+r'})$.

Given the private key $\alpha = \log_g h$, decryption of $(a, b) = (g^r, g^m h^r)$ is performed by first calculating $b/a^\alpha = g^m$, and then solving for $m \in \mathbb{Z}_q$. In general, this is exactly the DL problem, which we assume to be infeasible. The way out is to require that message m is constrained to a sufficiently small set $M \subseteq \mathbb{Z}_q$.[1] In this paper, the cardinality of M will be very small, often $|M| = 2$.

Homomorphic ElGamal encryption is semantically secure assuming the infeasibility of the DDH problem. Throughout the paper, we use $[\![m]\!]$ to denote the set of all ElGamal encryptions of m under some understood public key h, and, frequently, we also use $[\![m]\!]$ to denote one of its elements. More formally, using that $[\![0]\!]$ is a subgroup of $G \times G$, $[\![m]\!]$ is the coset of $[\![0]\!]$ in $G \times G$ containing encryption $(1, g^m)$. Hence, encryptions (a, b) and (a', b') belong to the same coset iff $\log_g(a/a') = \log_h(b/b')$. Lifting the operations on the direct product group $G \times G$ to the cosets, we thus have, for $x, y \in \mathbb{Z}_q$, that $[\![x]\!] \star [\![y]\!] = [\![x + y]\!]$, and $[\![x]\!]^y = [\![xy]\!]$, where $(a, b)^c = (a^c, b^c)$ for $c \in \mathbb{Z}_q$. Hence, $[\![x]\!] \star [\![y]\!]^{-1} = [\![x - y]\!]$. Addition and subtraction over $\mathbb{Z}_q$ and multiplication by a publicly known value in $\mathbb{Z}_q$ can thus be performed easily on encrypted values. These operations are deterministic. Another useful consequence is that any encryption in $[\![x]\!]$ can be transformed into a statistically independent encryption in $[\![x]\!]$ by multiplying it with a uniformly selected encryption in $[\![0]\!]$; this is often referred to as "random re-encryption."

Pedersen Commitment. Given $h' \in G$, a Pedersen commitment to $m \in \mathbb{Z}_q$ is a value $b = g^m h'^r$, with $r \in_R \mathbb{Z}_q$. The commitment is opened by revealing m and r.

[1] For intervals M, the Pollard-λ ("kangaroo") method runs in $O(\sqrt{|M|})$ time using $O(1)$ storage.

Pedersen's scheme is unconditionally hiding and computationally binding, under the assumption that $\log_g h'$ cannot be determined. The commitment scheme is also additively homomorphic, and we will sometimes use $\langle\langle m \rangle\rangle$ to denote a commitment to message m, where the randomization is suppressed.

Σ-Protocols. We briefly mention a few facts about Σ-protocols. A Σ-protocol for a relation $R = \{(v, w)\}$ is a three-move protocol between a prover and a verifier, where the prover does the first move. Both parties get a value v as common input, and the prover gets a "witness" w as private input, $(v, w) \in R$. A Σ-protocol is required to be a proof of knowledge for relation R satisfying special soundness and special honest-verifier zero-knowledge. See [CDS94] for details.

We need some well-known instances of Σ-protocols. The simplest case is Schnorr's protocol for proving knowledge of a discrete log α, on common input $a = g^\alpha$, and Okamoto's variant for proving knowledge of α, β, on common input $a = g^\alpha h^\beta$. Another basic case is Chaum-Pedersen's protocol for proving knowledge of α, on common input $(a, b) = (g^\alpha, h^\alpha)$, which is a way to prove that $(a, b) \in [\![0]\!]$ without revealing any information on α. Applying OR-composition [CDS94], these basic protocols can be combined into, for instance, a Σ-protocol for proving that $(a, b) \in [\![0]\!] \cup [\![1]\!]$, where the common input is an ElGamal encryption (a, b). The latter protocol thus proves that the message encrypted (which is an element of $\mathbb{Z}_q$) actually is a "bit", without divulging any further information on the message.

For simplicity, we will use the non-interactive versions of these Σ-protocols, which are obtained via the Fiat-Shamir heuristic, that is, by computing the challenge as a hash of the first message (and possibly other inputs). The resulting proofs are known to be secure in the random oracle model; in particular, we will use that these proofs can be simulated.

Threshold ElGamal Decryption. We use a $(t + 1, n)$-threshold ElGamal cryptosystem, $0 \leq t < n$, in which encryptions are computed using a common public key h (as above) while decryptions are done using a joint protocol between n parties $P_1, \ldots, P_n$. Each party P_i holds a share $\alpha_i \in \mathbb{Z}_q$ of the private key $\alpha = \log_g h$, where the corresponding value $h_i = g^{\alpha_i}$ is public. As long as more than t parties take part, decryption will succeed, whereas t or less parties are not able to decrypt successfully.

The parties initially obtain their shares α_i by running a secure distributed key generation protocol; see [Ped91, GJKR99] for details. We note that these protocols are practical (the communication complexity is $O(n^2 k)$ bits for security parameter k, where the hidden constant is small). For the two-party case ($t = 1$, $n = 2$), we briefly describe a (non-robust) distributed key generation protocol in the spirit of [GJKR99]. The protocol consists of two steps. In the first step, party P_i, $i = 1, 2$, broadcasts a Pedersen commitment $b_i = g^{\alpha_i} h'^{r_i}$, with $\alpha_i, r_i \in_R \mathbb{Z}_q$ along with a proof of knowledge for α_i, r_i. In the second step, party P_i, $i = 1, 2$, broadcasts r_i along with a proof of knowledge of $\log_g h_i$, where $h_i = b_i / h'^{r_i}$. The joint public key is $h = h_1 h_2$, with private key $\alpha = \alpha_1 + \alpha_2$. Clearly, this protocol is

very practical. In many cases, it may even be replaced by the trivial one-round protocol in which both parties broadcast $h_i = g^{\alpha_i}$ and a proof of knowledge of α_i. Although the trivial protocol allows one of the parties to influence the distribution of the public key h slightly, this need not be a problem for the application in which the key is used; see [GJKR03] for more details.

For decryption of (a, b), party P_i, $i = 1, \ldots, n$, produces a decryption share $d_i = a^{\alpha_i}$ along with a proof that $\log_a d_i = \log_g h_i$. Assuming w.l.o.g. that parties $P_1, \ldots, P_{t+1}$ produce correct decryption shares, the message can be recovered from $g^m = b/a^\alpha$, where a^α is obtained from $d_1, \ldots, d_{t+1}$ by Lagrange interpolation. Assuming homomorphic ElGamal, $m \in M$ will hold for some small set M; if such m cannot be found decryption fails. Also, if fewer than $t + 1$ parties provide a correct decryption share, decryption fails.

For later use in the proof of Theorem 1, we note that the threshold decryption protocol can be simulated for any input $(a, b) \in [\![m]\!]$, provided message $m \in \mathbb{Z}_q$ is given as well. Assume w.l.o.g. that parties $P_1, \ldots, P_t$ are corrupted, hence collectively form the adversary. The simulator first extracts the shares $\alpha_1, \ldots, \alpha_t$ of the adversary, by rewinding the proofs of knowledge based on these shares. (The parties prove knowledge of their shares during the distributed key generation protocol.) The simulator then computes $a^\alpha = b/g^m$ from b and m. The simulator then computes the correct decryption shares for the corrupted parties as $a^{\alpha_1}, \ldots, a^{\alpha_t}$, which enables the computation of the decryption shares for the honest parties by Lagrange interpolation on $a^\alpha, a^{\alpha_1}, \ldots, a^{\alpha_t}$. The corresponding proofs of correct decryption are simulated for the honest parties. For the corrupted parties, the decryption shares and the proofs of correct decryption are obtained from the adversary, running it as a black box; possibly some of these shares are wrong and/or some of the proofs fail, but these values are included in the output of the simulator anyway. The simulation is then completed by recovering the message as in the real protocol, possible ending with a decryption failure. As a result, the simulated transcript is consistent with the view of the adversary and statistically indistinguishable of real transcripts.

3 Special Multiplication Protocols

The results of the previous section imply that a function f can be evaluated securely in a multiparty setting if f can be represented as a circuit over $\mathbb{Z}_q$ consisting only of addition gates and simple multiplication gates. Here, an addition gate takes encryptions $[\![x]\!]$ and $[\![y]\!]$ as input and produces $[\![x]\!] \star [\![y]\!] = [\![x + y]\!]$ as output, and a simple multiplication gate takes $[\![x]\!]$ as input and produces $[\![x]\!]^c = [\![cx]\!]$ as output, for a publicly known value $c \in \mathbb{Z}_q$. To be able to handle any function f, however, we need more general multiplication gates for which both inputs are encrypted.

In this section, we consider two special multiplication gates. If no restrictions are put on x or y, a multiplication gate, taking $[\![x]\!]$ and $[\![y]\!]$ as input and producing $[\![xy]\!]$ as output efficiently, cannot exist assuming that the DH problem

is infeasible.[2] Therefore, we consider two special multiplication gates, putting some restrictions on the multiplier x. The first gate requires that the multiplier x is *private*, which means that it is known by a single party. The second gate, referred to as the *conditional gate*, requires that the multiplier x is from a *dichotomous* (two-valued) domain. As a direct application of the conditional gate, we also consider xor-homomorphic encryption based on ElGamal encryption.

3.1 Multiplication with a Private Multiplier

We present a multiplication protocol where the multiplier x is a *private* input rather than a shared input. That is, the value of x is known by a single party P. No restriction is put on the multiplicand y. Multiplication with a private multiplier occurs as a subprotocol in the protocol for the conditional gate and in other protocols further on in the paper.

Given encryptions $[\![x]\!] = (a, b) = (g^r, g^x h^r)$ and $[\![y]\!] = (c, d)$, where party P knows r, x, party P computes on its own a randomized encryption $[\![xy]\!] = (e, f) = (g^s, h^s) \star [\![y]\!]^x$, with $s \in_R \mathbb{Z}_q$, using the homomorphic properties. Party P then broadcasts $[\![xy]\!]$ along with a proof showing that this is the correct output, which means that it proves knowledge of witnesses $r, s, x \in \mathbb{Z}_q$ satisfying $a = g^r$, $b = g^x h^r$, $e = g^s c^x$, $f = h^s d^x$.

For later use, we need to be able to simulate the above protocol. The simulator gets as input $[\![x]\!]$ and $[\![y]\!]$, and a correct output encryption $[\![xy]\!]$, but it does not know x. As a result, the simulator only needs to add a simulated proof of knowledge. The simulated transcript is statistically indistinguishable from a real transcript.

Below, we will also use a variation of the above protocol, where the private multiplier x is multiplied with several multiplicands y_i at the same time. Furthermore, we note that often a slight optimization is possible by using a Pedersen commitment $\langle\!\langle x \rangle\!\rangle = g^x h'^r$ instead of an ElGamal encryption $[\![x]\!] = (g^r, g^x h^r)$ for the multiplier.

3.2 Conditional Gate

Next, we consider a multiplication gate for which the multiplier x is from a dichotomous (two-valued) domain, whereas the multiplicand y is unrestricted. We call it the *conditional gate*, and show how to implement it by an efficient protocol, using just homomorphic threshold ElGamal. We will formulate the conditional gate for the dichotomous domain $\{-1, 1\}$.[3]

[2] Given g^x, g^y we form encryptions $[\![x]\!]$, $[\![y]\!]$ and feed these into the multiplication gate. The gate would return an encryption $[\![xy]\!]$, which would give g^{xy} upon decryption.

[3] Domain $\{0, 1\}$ or any other domain $\{a, b\}$, $a \neq b$, can be used instead, as these domains can be transformed into each other by linear transformations: $x \mapsto a' + (b' - a')(x - a)/(b - a)$ maps $\{a, b\}$ to $\{a', b'\}$. These transformations can be applied directly to homomorphic encryptions, transforming $[\![x]\!]$ with $x \in \{a, b\}$ into $[\![x']\!]$ with $x' \in \{a', b'\}$.

Let $[\![x]\!], [\![y]\!]$ denote encryptions, with $x \in \{-1, 1\} \subseteq \mathbb{Z}_q$ and $y \in \mathbb{Z}_q$. The following protocol enables parties $P_1, \ldots, P_n$, $n \geq 2$, to compute an encryption $[\![xy]\!]$ securely. For simplicity, we assume that these parties also share the private key of the $(t+1, n)$-threshold scheme $[\![\cdot]\!]$, where $t < n$. The protocol consists of two phases.

1. Let $x_0 = x$ and $y_0 = y$. For $i = 1, \ldots, n$, party P_i in turn takes $[\![x_{i-1}]\!]$ and $[\![y_{i-1}]\!]$ as input, and broadcasts a commitment $\langle\!\langle s_i \rangle\!\rangle$, with $s_i \in_R \{-1, 1\}$. Then P_i applies the private-multiplier multiplication protocol to multiplier $\langle\!\langle s_i \rangle\!\rangle$ and multiplicands $[\![x_{i-1}]\!]$ and $[\![y_{i-1}]\!]$, yielding random encryptions $[\![x_i]\!]$ and $[\![y_i]\!]$, where $x_i = s_i x_{i-1}$ and $y_i = s_i y_{i-1}$. If P_i fails to complete this step successfully it is discarded immediately.
2. The parties jointly decrypt $[\![x_n]\!]$ to obtain x_n. If decryption fails because the number of correct shares is insufficient, the entire protocol is aborted. If decryption fails because $x_n \notin \{1, 1\}$, each party P_i is required to broadcast a proof that $s_i \in \{-1, 1\}$. Parties failing to do so are discarded, and the protocol is restarted (starting again at phase 1). Given x_n and $[\![y_n]\!]$, an encryption $[\![x_n y_n]\!]$ is computed publicly.

The output of the protocol is $[\![x_n y_n]\!]$. Clearly, if all parties are honest, $x_n y_n = (\prod_{i=1}^n s_i)^2 xy = xy$.

Any party may disrupt the protocol for at most one run of phase 1 by picking a value s_i outside the range $\{-1, 1\}$. Note that we do not need to require that each s_i is in $\{-1, 1\}$ in phase 1. For instance, parties P_1 and P_2 may cheat by setting $s_1 = 2$ and $s_2 = 1/2$. Since $s_1 s_2 = 1$, this type of "cheating" will go unnoticed in phase 2 if all other parties are honest. However, the security of the protocol is not affected by such "cheating." For $t < n/2$, the protocol is robust, allowing up to t failing parties in total (as the threshold decryption step tolerates up to t failing parties). For $n/2 \leq t < n$, the protocol is not robust, but we will see from Theorem 1 below that the adversary does not get an advantage in this case.

The protocol requires a single threshold decryption only. Since $x_n \in \{-1, 1\}$ is required to hold, decryption is feasible for the homomorphic ElGamal encryption scheme. As the value of x_n is statistically independent of x, the value of x_n does not reveal any information on x. This is stated in the following theorem, which holds for up to $t < n$ corrupting parties.

Theorem 1. *On input $[\![x]\!], [\![y]\!]$ with $x \in \{-1, 1\} \subseteq \mathbb{Z}_q$ and $y \in \mathbb{Z}_q$, the above protocol produces $[\![xy]\!]$, without leaking any additional information on x and y.*

Proof. The soundness of the proofs in phase 1 of the protocol ensures that $x_n = x \prod_{i=1}^n s_i$ and $y_n = y \prod_{i=1}^n s_i$ Since it is checked in phase 2 that $x_n \in \{-1, 1\}$, it follows from $x \in \{-1, 1\}$ that $\prod_{i=1}^n s_i \in \{-1, 1\}$ as well. Therefore, $x_n y_n = xy$.

To argue that no additional information on x and y is leaked we present the following simulation of the protocol. The simulation takes as input encryptions $[\![x]\!], [\![y]\!]$, and $[\![xy]\!]$. Given this information, the simulator is able to generate a complete transcript for the protocol, for which the distribution is exactly the same as in real executions of the protocol. If $[\![xy]\!]$ is not available, it may be

replaced by a random encryption in $[\![0]\!]$, as done in [CDN01–Theorem 1]. Since the simulator below does not use the shares of the honest parties to simulate decryptions, the simulated transcripts will be indistinguishable (under DDH) from real transcripts for any adversary controlling up to t parties.

Assume that parties $P_1, \ldots, P_t$ are corrupted, hence collectively form the adversary (the simulator is easily adapted for other sets of corrupted parties). The simulator lets the adversary run phase 1 of the protocol for parties $P_1, \ldots, P_t$, each time rewinding the proofs of knowledge used in the private-multiplier multiplication protocol to extract the values $s_1, \ldots, s_t$; if a party fails to provide a correct proof it is discarded. Subsequently, the simulator runs phase 1 for parties $P_{t+1}, \ldots, P_{n-1}$ as in the real protocol, leaving $[\![x_{n-1}]\!]$, with $x_{n-1} = s_1 \cdots s_{n-1} x$, as intermediate encryption. For party P_n, however, the simulator picks $s'_n \in_R \{-S, S\}$, where $S = \prod_{i=1}^{t} s_i$, and it computes a commitment $\langle\!\langle s'_n x_{n-1} \rangle\!\rangle$ and an encryption $[\![s'_n xy]\!]$, from $[\![x_{n-1}]\!]$ and $[\![xy]\!]$, respectively. Writing $s_n = s'_n x_{n-1}$, the simulator then simulates the private multiplier multiplication protocol for multiplier $\langle\!\langle s_n \rangle\!\rangle$ and multiplicands $[\![x_{n-1}]\!]$, $[\![y_{n-1}]\!]$ and outputs $[\![s'_n]\!]$, $[\![s'_n xy]\!]$, which are the correct outputs since $s'_n = s_n x_{n-1}$ and $s'_n xy = s'_n x_{n-1} y_{n-1} = s_n y_{n-1}$.

The output of phase 1 consists of encryptions $[\![s'_n]\!]$ and $[\![s'_n xy]\!]$. By construction, the simulator is able to perform the decryption in phase 2 itself, producing $s'_n \in \{-S, S\}$ as output. The simulator for the threshold decryption protocol is used for encryption $[\![s'_n]\!]$ using s'_n as an additional input (see Section 2). If decryption fails due to an insufficient number of correct decryption shares, the simulation stops, as in the real protocol. If $S \not\in \{\cancel{1}, 1\}$, decryption fails and a proof that $s_i \in \{-1, 1\}$ is generated for each party P_i, by letting the adversary do this for parties $P_1, \ldots, P_t$ (of which at least one fails), running the real protocol for parties $P_{t+1}, \ldots, P_{n-1}$, and using a simulation for P_n. After discarding the failing parties among $P_1, \ldots, P_t$, the simulation is continued by simulating another run of phase 1.

Finally, the simulator computes the encryption $[\![s'_n s'_n xy]\!] = [\![xy]\!]$, which is clearly the correct output. □

If the total number of parties is large compared to the total number of conditional gates to be evaluated, an alternative way to guarantee robustness is to let the parties use encryptions $[\![s_i]\!]$ instead of commitments $\langle\!\langle s_i \rangle\!\rangle$ in phase 1. Again, if $x_n \not\in \{\cancel{1}, 1\}$ in phase 2, all parties are required to prove that $s_i \in \{-1, 1\}$. Failing parties are discarded and their s_i values are decrypted to correct the value of x_n.

The performance of the protocol is as follows (analyzing the case that no party is cheating). The performance is determined by the communication complexity (in bits) and the round complexity. In phase 1 each party applies the private-multiplier multiplication protocol, broadcasting about 10 values. For decryption each party broadcasts 3 values at the most. Hence, the communication complexity is $O(nk)$ where the hidden constant is very small. In general, the round complexity is $O(n)$, which is high, but in case of two-party computation it is $O(1)$. Also, when many conditional gates are to be evaluated in parallel,

one may take advantage of the fact that the order in which parties $P_1, \ldots, P_n$ execute phase 1 of the conditional gate protocol can be chosen arbitrarily.

3.3 XOR-Homomorphic ElGamal Encryption

As a direct application of the conditional gate, we obtain an xor-homomorphic ElGamal encryption scheme. (The converse problem of constructing $(\mathbb{Z}_q, +)$-homomorphic schemes, $q > 2$, from xor-homomorphic schemes, such as the Goldwasser-Micali cryptosystem [GM84], is considered in [KMO01].)

Given $[\![x]\!], [\![y]\!]$ with $x, y \in \{0, 1\}$, $[\![x \oplus y]\!]$ is computed as follows, using one threshold decryption (cf. footnote 2):

1. Publicly convert $[\![x]\!]$ to $[\![x']\!]$ with $x' = 2x - 1 \in \{-1, 1\}$.
2. Apply the conditional gate to $[\![x']\!]$ and $[\![y]\!]$ to obtain $[\![x'y]\!]$.
3. Publicly compute $[\![x - x'y]\!]$, which is equal to $[\![x \oplus y]\!]$.

The work per party is very limited, about 13 exponentiations for each conditional gate. In contrast, the Mix and Match approach of [JJ00] would require each party to mix the 4 rows of a truth table for $x \oplus y$ in a verifiable way (Mix step, requiring 24 exponentiations for blinding the entries and, say, 6×12 exponentiations for the correctness proof, using the efficient protocol of [Gro03]), and perform on average 4 plaintext equality tests to find $[\![x \oplus y]\!]$ given $[\![x]\!]$ and $[\![y]\!]$ (Match step, requiring 4×7 exponentiations). Hence, the conditional gate provides approximately a ten-fold improvement, counting exponentations.

4 Circuit Evaluation

In this section, we briefly describe a protocol for evaluating a given circuit composed of elementary gates. Recall that our elementary gates operate over $\mathbb{Z}_q$, except that the first input of a conditional gate is required to belong to a two-valued domain. It is clear that these elementary gates suffice to emulate any Boolean circuit. Specifically, any operator on two bits $x, y \in \{0, 1\} \subseteq \mathbb{Z}_q$ can be expressed uniquely as a polynomial of the form $a_0 + a_1 x + a_2 y + a_3 xy$ with coefficients in $\mathbb{Z}_q$. Hence, any binary operator can be expressed using at most one conditional gate.

The protocol operates in much the same manner as the protocol for circuit evaluation of, for instance, [CDN01]. For convenience, we assume that the parties $P_1, \ldots, P_n$ evaluating the circuit are exactly the same as the parties for which the $(t + 1, n)$-threshold cryptosystem has been set-up, where $t < n$. The circuit is then evaluated in three phases:

1. The parties encrypt their inputs using the homomorphic cryptosystem $[\![\cdot]\!]$, and the parties are required to provide a proof of knowledge for their inputs, and possibly that the inputs belong to a dichotomous domain.
2. The parties then jointly evaluate the circuit gate-by-gate. Conditional gates at the same depth of the circuit are evaluated in parallel.
3. Finally, the parties jointly decrypt the outputs of the circuit.

As described in the previous section, parties failing at some stage in the protocol are discarded immediately. As long as no more than $t < n/2$ parties fail in total, the protocol will complete and all parties will learn the output. The case of $n/2 \leq t < n$ will be discussed below.

The formal security analysis of [CDN01] can be adapted to show that our protocol is secure against a static, active adversary corrupting at most $t < n$ parties, assuming the intractability of the DDH problem. This follows from the fact that we are able to simulate the multiplication protocols of Section 3 in a statistically indistinguishable manner, provided that the simulator for these protocols is given encryptions of the correct output values. We thus achieve the same level of security as [CDN01]. The important difference is that [CDN01] incorporates a general multiplication gate, for which they need an RSA-like cryptosystem such as Paillier's cryptosystem to get an efficient multiplication protocol, while we incorporate a restricted multiplication gate, for which we have presented an efficient multiplication protocol using the ElGamal cryptosystem.

4.1 Private Outputs

In this section, we propose a new non-interactive protocol for achieving private outputs in the context of secure multiparty computation based on homomorphic threshold cryptosystems. Previous methods require the receiving parties to perform some blinding step, as part of the decryption protocol. For our method, it suffices to know the public keys of the receiving parties.

We need a different method than [CDN01] to deal with private outputs anyway, since their method would require us to decrypt an ElGamal encryption of a random message in $\mathbb{Z}_q$. Suppose $[\![m]\!]$ is an encryption of a private output for party P_j, that is, output m is intended for party P_j only. Briefly, the method of [CDN01] is to let party P_j first blind encryption $[\![m]\!]$ by multiplying it with a random encryption $[\![r]\!]$ for some $r \in_R \mathbb{Z}_q$. The encryption $[\![m+r]\!]$ is then jointly decrypted, resulting in the value $m' = m + r$, from which (only) party P_j is able to compute $m = m' - r$. This method critically depends on the ability to decrypt arbitrary messages. Using an RSA-like cryptosystem, such as Paillier's cryptosystem, this is no problem. Using ElGamal encryption, however, we cannot decrypt $[\![m + r]\!]$ (see Section 2). A first way out is to adapt the ElGamal decryption step to output g^{m+r} instead of $m+r$; the receiving party may divide this value by g^r to obtain g^m from which m may be determined, assuming m is from a small set.

We note however that in general it is undesirable that interaction with the receiving parties is required to produce private outputs. Therefore, we present a protocol for which *no* interaction with the receiving party is required. The protocol runs as follows. Let $(a, b) \in [\![m]\!]$ be an output intended for party P_j and let $h_j = g^{\alpha_j}$ denote P_j's public key. Recall from Section 2 that threshold decryption requires each party P_i to produce the value a^{α_i} along with a proof of correctness. We modify this step as follows, by releasing a^{α_i} encrypted under P_j's public key and adapting the proof of correctness accordingly:

1. Each party P_i outputs an encryption $(c_i, d_i) = (g^{r_i}, h_j^{r_i} a^{\alpha_i})$ with $r_i \in_R \mathbb{Z}_q$ along with a proof that it knows r_i, α_i satisfying

$$h_i = g^{\alpha_i}, \quad c_i = g^{r_i}, \quad d_i = h_j^{r_i} a^{\alpha_i}.$$

2. For decryption, party P_j first uses Lagrange coefficients λ_i to compute the following product for a set of $t+1$ valid shares (c_i, d_i):

$$\prod_i (c_i, d_i)^{\lambda_i} = (g^{\sum_i \lambda_i r_i}, h_j^{\sum_i \lambda_i r_i} a^{\sum_i \lambda_i \alpha_i}).$$

   Then P_j decrypts this product using its private key α_j to obtain $a^{\sum_i \lambda_i \alpha_i} = a^{\alpha}$. Party P_j then proceeds to recover $g^m = b/a^{\alpha}$, from which it finds m assuming that m belongs to a relatively small, known subset of $\mathbb{Z}_q$.

The protocol requires only a small amount of additional work compared to the basic protocol for decrypting public outputs, where each party P_i outputs a^{α_i} along with a proof of correctness (cf. step 1), from which anyone is then able to recover $a^{\sum_i \lambda_i \alpha_i} = a^{\alpha}$ using $t+1$ valid shares (cf. step 2).

4.2 Fairness

Recall that t denotes the maximum number of corrupted parties tolerated by the circuit evaluation protocol. For $t < n/2$, that is, the case of a dishonest minority, the protocol achieves robustness. We now extend the protocol to handle the two-party case $t = 1, n = 2$ (which is a special case of a dishonest majority, $n/2 \leq t < n$).

For the two-party case we give up on robustness, since one cannot prevent one of the parties from quitting the protocol prematurely. If a party chooses to do so, however, it should not gain any advantage from it. If a protocol achieves this property, the protocol is said to be *fair*.

An important observation for the above circuit evaluation protocol is that neither party gains any advantage from quitting the protocol in phase 1 or phase 2 of the protocol. In particular, consider the case that party P_2, say, chooses to quit during the threshold decryption step of a conditional gate, for which party P_1 has already produced its decryption share. In that case, only P_2 learns the decrypted value x_n, but this value cannot possibly give P_2 an advantage, as follows from the simulation in the proof of Theorem 1.

Therefore, to achieve fairness, we only need to protect the decryption of the output values. For this purpose, we will apply a protocol similar to that of [BST01]. In [BST01], however, the protocol steps for achieving fairness are intertwined with the original protocol steps, while in our protocol the additional steps for achieving fairness are strictly limited to the decryption of the output values.

Let an encryption (a, b) be given. Recall that $(2, 2)$-threshold decryption, requires party P_i to provide $d_i = a^{\alpha_i}$, $i = 1, 2$, along with a proof that this value is correct w.r.t. the public key $h_i = g^{\alpha_i}$ of party P_i. Instead of directly revealing this value, we will release it gradually using the following protocol, where k is a

security parameter, $k < \log_2 q$, and $h \in \langle g \rangle$ denotes an additional generator for which $\log_g h$ is unknown to parties P_1, P_2:

1. For $i = 1, 2$, party P_i chooses $\epsilon_{ij} \in_R \{0, 1\}, \alpha_{ij} \in_R \mathbb{Z}_q$ for $j = 0, \ldots, k-1$ subject to the condition that $\alpha_i = \sum_{j=0}^{k-1} \alpha_{ij} 2^j$. Party P_i then broadcasts the values $d_{ij} = a^{\alpha_{ij}} h^{\epsilon_{ij}}$, $j = 0, \ldots, k-1$ along with a proof that each $\epsilon_{ij} \in \{0, 1\}$ and a proof that $\prod_{j=0}^{k-1} d_{ij}^{2^j} = a^{\alpha_i} h^{\epsilon}$, where $\alpha_i = \log_g h_i$, for some value ϵ.
2. Set $j = k - 1$. Parties P_1, P_2 repeatedly execute the following step. For $i = 1, 2$, party P_i broadcasts values $\alpha_{ij}, \epsilon_{ij}$. If these values verify correctly against d_{ij}, the value of j is decremented and the step is repeated if $j > 0$.
3. Once $j = 0$ both parties release ϵ_{i0} along with a proof of knowledge for a witness α_{i0} satisfying $d_{i0} h^{-\epsilon_{i0}} = a^{\alpha_{i0}}$.
4. Both parties are able to recover the missing value a^{α_i}, as follows:

$$a^{\alpha_i} = d_{i0} h^{-\epsilon_{i0}} a^{\sum_{j=1}^{k-1} \alpha_{ij} 2^j}.$$

At each stage of the protocol, either party is at most one bit ahead of the other party. If one sets $k = 80$, for instance, it is clearly infeasible for both parties to compute the missing value a^{α_i} at step 1, as it requires a search over 2^k possible values for $\epsilon_{i,k-1}, \ldots, \epsilon_{i0}$. At each later step, the search space is reduced in size by a factor of two.

The protocol does not leak any information on α_i beyond what is implied by the output values a^{α_i}. The protocol can be run in parallel for decrypting multiple outputs at the same time, and the protocol can be combined easily with our protocol for private outputs presented above.

The above protocol achieves a basic level of fairness. In [Pin03] a strengthened notion of fairness for two-party computation is considered, which also addresses the case where one party may be considerably more powerful than the other party; the timed commitments used to resolve this problem, however, critically depend on the hardness of factoring. (The recent paper [GMY04b] describes a way to cover fairness in a universally composable way, for static adversaries. In particular, the result of [CDN01] is extended to cover fairness as well, using a factoring related assumption to achieve timed commitments.) Apart from this difference, the result of [Pin03] is comparable to our result. The difference is that [Pin03] is based on Yao's garbled circuit approach, while our approach is based on homomorphic threshold cryptosystems. In both cases, however, the changes to make the protocol fair are limited to the output stage, where some form of gradual release is used in combination with a method to ensure that commitments opened during gradual release indeed contain the correct output of the computation.

5 Relational and Arithmetic Operators

In this section we apply our set of elementary gates over $\mathbb{Z}_q$ to obtain efficient circuits for basic operations such as integer comparison and integer addition. In

most cases, the inputs are required to be given by their binary representations. We consider the general case, in which the circuits operate on encrypted inputs, producing encrypted outputs, such that they can be used as building blocks in constructing circuits for more elaborate functions, either in a two-party setting or in a general multiparty setting.

5.1 $\mathbf{sgn}(x - y)$

Below, we present an efficient protocol for comparing two (non-negative) integer values x and y. The inputs are given as sequences of encrypted bits, $[\![x_{m-1}]\!], \ldots, [\![x_0]\!]$ and $[\![y_{m-1}]\!], \ldots, [\![y_0]\!]$, with $x = \sum_{i=0}^{m-1} x_i 2^i$, $y = \sum_{i=0}^{m-1} y_i 2^i$. The output of the protocol consists of an encryption $[\![\mathrm{sgn}(x - y)]\!]$, where sgn denotes the signum function:

$$\operatorname{sgn} z = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0. \end{cases}$$

Using that $x_i^2 = x_i$ and $y_i^2 = y_i$ for $x_i, y_i \in \{0, 1\}$, the general strategy is now to examine the unique multilinear polynomial p over $\mathbb{Z}_q$ satisfying $p(x_0, \ldots, x_{m-1}, y_0, \ldots, y_{m-1}) = \mathrm{sgn}(x - y)$ for all x, y, $0 \leq x, y < 2^m$. The problem that remains is to find an efficient circuit (or, equivalently, an oblivious evaluation order) for the polynomial p.

As a first step, we consider the evaluation $p = s_0$ with

$$s_{m-1} = 0, \qquad s_{i-1} = s_i + (1 - s_i^2)(x_i - y_i).$$

Clearly, this results in the correct output value. However, we cannot evaluate the term s_i^2 by means of a conditional gate since s_i is three-valued ($s_i \in \{-1, 0, 1\}$). This is easily resolved by introducing an auxiliary *binary* sequence v_i, with $v_i = 1 - s_i^2$:

$$\begin{aligned} s_{m-1} &= 0, & s_{i-1} &= s_i + v_i(x_i - y_i), \\ v_{m-1} &= 1, & v_{i-1} &= v_i - v_i(x_i - y_i)^2. \end{aligned}$$

Now, it is easy to draw up a circuit using $3m - 2$ conditional gates (using that $s_{m-1} = 0$ and $v_{m-1} = 1$ are publicly known values, hence need not be encrypted).

We note that the bits may also be traversed in the opposite direction, starting at the least significant bit. This results in the following sequence, with s'_m as output:

$$s'_0 = 0, \qquad s'_{i+1} = (1 - (x_i - y_i)^2)s'_i + x_i - y_i.$$

This method only needs $2m - 2$ conditional gates: per iteration, one conditional gate to compute $x_i y_i$ and one to subsequently compute $(1 - (x_i - y_i)^2)s'_i$ with $1 - (x_i - y_i)^2$ as dichotomous multiplier. Here we take full advantage of the fact that the conditional gate does not put any constraints on the multiplicand: whereas a Boolean circuit for $\mathrm{sgn}(x - y)$ requires all intermediate values to be binary, our circuit uses non-binary intermediate values, such as the ternary s'_i's.

5.2 $x > y$

The output of $x > y$ consists of one bit only, which is set to 1 if $x > y$ and to 0 otherwise. Starting at the least significant bit, the output is given by t_m, where

$$t_0 = 0, \qquad t_{i+1} = (1 - (x_i - y_i)^2)t_i + x_i(1 - y_i).$$

This method requires $2m - 1$ conditional gates. (For comparison we note that the best known circuit using only logical gates requires $5m$ binary gates, e.g. using the circuit for $Bigger_k(X, Y)$ of [KO02]. Similarly, for computing $Max(X, Y)$ given $Bigger_k(X, Y)$, $2m$ additional gates are required in [KO02], while we can compute the bits of $z = max(x, y)$ by setting $z_i = y_i - t_m(x_i + y_i)$, using only m additional conditional gates.)

We now specialize this solution for $x > y$ to obtain a solution for Yao's basic millionaires problem [Yao82]. In this case, the protocol is run by two parties, providing x and y respectively as *private* inputs. This allows for a much more efficient solution, as the conditional gates can all be replaced by the private-multiplier gates of Section 3.1. The private-multiplier gates can be even optimized slightly by using Pedersen commitments instead of ElGamal encryptions, and using that the multipliers are binary.

The total computational cost of our solution to Yao's millionaires problem, including the cost of the distributed key generation and the decryption of the result, is dominated by the cost of about $2m$ private-multiplier gates (computing $[\![y_i t_i]\!]$ and $[\![x_i(t_i - 2y_i t_i - y_i)]\!]$ as intermediate values), which require 6 exponentiations each, hence $12m$ exponentiations in total (starting at the least significant bit). To the best of our knowledge, this is the most efficient solution to date. Here, we cover the malicious case (unlike many other papers on the millionaires problem, that only deal with the semi-honest case, e.g., [Fis01, NN01, IG03]), but we do not cover fairness. We also note that we do not need an auxiliary trusted party, as in [Cac99], although that paper achieves fairness as well at a relatively low cost. Finally, while most other solutions rely on an RSA-like assumption, our solution is secure under the standard DDH assumption. This is also true for the solution of [KO02], but their solution is much less efficient because their circuits are evaluated using the expensive Mix and Match gates of [JJ00].

5.3 $x = y$

For testing equality of x and y, the following sequence can be used, where the output $u_m = 0$ iff $x = y$:

$$u_0 = 0, \qquad u_{i+1} = (1 - (x_i - y_i)^2)u_i + (x_i - y_i)^2.$$

The order in which the bits are processed is actually irrelevant. This method requires $2m - 1$ conditional gates, returning the output bit in encrypted form.

The socialist millionaires problem, a variant introduced by [JY96], is to evaluate $x = y$ for a two-party setting, where $x, y \in \mathbb{Z}_q$ are the respective *private* inputs, and the output may be public. The currently best solution is due to [BST01], using only $O(1)$ exponentiations, hence without using the binary representations of x and y. We obtain a solution in a similar vein as follows. The

parties broadcast $[\![x]\!]$ and $[\![y]\!]$, resp., and jointly form $[\![r]\!]$, where $r \in_R \mathbb{Z}_q$ and neither of the parties knows r. Using the private multiplier gate, the parties then compute $[\![(x-y)r]\!]$, which is jointly decrypted to obtain $g^{(x-y)r}$ (rather than obtaining $(x-y)r$). If $g^{(x-y)r} = 1$, then w.v.h.p. $x = y$, otherwise $x \not= y$.

5.4 $x + y$ and $x * y$

Given $[\![x]\!], [\![y]\!]$, one obtains $[\![x+y]\!]$ directly using the homomorphic property. A nice application of the conditional gate is that $[\![xy]\!]$ can also be computed efficiently, if we assume that x is given in binary form.

Given $[\![x_{m-1}]\!], \ldots, [\![x_0]\!]$ and $[\![y]\!]$, where $y \in \mathbb{Z}_q$, one computes $[\![xy]\!]$ using that $xy = \sum_{i=0}^{m-1} x_i(y2^i)$. This method requires only m conditional gates, whereas a standard Boolean circuit would require $O(m^2)$ bit multiplications.

6 Concluding Remarks

We envision a practical system supporting ad hoc contacts among a large group of peer users. Since efficient DKG protocols for $(2,2)$-threshold ElGamal are easily achieved, our results show that *any* pair of users is able to engage in a two-party computation for evaluating some dynamically agreed upon function. For example, the circuits of the previous section lead to solutions for tasks of practical interest, such as profile matching, allowing two users with profiles (length m bit vectors) x and y, resp., to evaluate $\Delta(x,y) > T$, where $\Delta(x,y) = \sum_{i=1}^{m} x_i y_i$ is an example similarity measure and T is a threshold.

Further research is required for a full comparison with some recent approaches to secure (two-party) computation. For instance, an interesting approach is presented in [GMY04a], which is based on committed oblivious transfer instead of homomorphic threshold encryption. The amount of work per gate is comparable to the work for a conditional gate, but the hidden constants for their approach are larger than in our case. This is partly due to the fact that their solution is designed to be universally composable, but remains true if their solution is 'downgraded' to a protocol for static adversaries; per gate, one party uses 5 bit commitments and proves a number of relations for these commitments, followed by a $\binom{4}{1}$ oblivious transfer. For a full comparison with [GMY04a], our solution needs to be 'upgraded' to a universally composable one, e.g., following the approach of [DN03]. This would provide an interesting alternative, as the extension of [GMY04a] to the multiparty case requires *each pair* of parties to run their basic two-party protocol for each multiplication gate, while with our approach the parties run a single joint protocol for each conditional gate.

A well-known alternative to the gate-by-gate approach, is Yao's garbled circuit approach for two-party computation. The Fairplay system is designed to evaluate the practical merits of the garbled circuit approach, including some optimizations that will pay off for sufficiently large circuits [MNPS04]. We expect a trade-off showing that the garbled circuit approach is best for large circuits whereas a gate-by-gate approach is best for small circuits, or rather circuits for which the number of inputs is proportional to the total number of gates.

Acknowledgements. We thank the anonymous referees for their helpful comments.

References

[ACS02] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology—CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432, Berlin, 2002. Springer-Verlag.

[BST01] F. Boudot, B. Schoenmakers, and J. Traoré. A fair and effcient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1–2):23–36, 2001. Special issue on Coding and Cryptology.

[Cac99] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *6th ACM Conference on Computer and Communications Security*, pages 120–127. ACM press, 1999.

[CDN01] R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300, Berlin, 2001. Springer-Verlag.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.

[CS02] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology—EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Berlin, 2002. Springer-Verlag.

[DJ03] I. Damgård and M. Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364, Berlin, 2003. Springer-Verlag.

[DN03] I. Damgård and J.B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264, Berlin, 2003. Springer-Verlag.

[FH96] M. Franklin and S. Haber. Joint encryption and message-efficient secure computation. *Journal of Cryptology*, 9(4):217–232, 1996.

[Fis01] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *Progress in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 457–471, Berlin, 2001. Springer-Verlag.

[Gil99] N. Gilboa. Two party RSA key generation. In *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 116–129, Berlin, 1999. Springer-Verlag.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310, Berlin, 1999. Springer-Verlag.

[GJKR03] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure applications of Pedersens distributed key generation protocol. In *Cryptographers' Track RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 373–390, Berlin, 2003. Springer-Verlag.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMY04a] G. Garay, P. MacKenzie, and K. Yang. Efficient and universally composable committed oblivious transfer and applications. In *Proc. 1st Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 297–316, Berlin, 2004. Springer-Verlag.

[GMY04b] J. Garay, P. MacKenzie, and K. Yang. Efficient and secure multi-party computation with faulty majority and complete fairness, 2004. Submitted. Available at http://eprint.iacr.org/2004/009/.

[Gro03] J. Groth. A verifable secret shuffle of homomorphic encryptions. In *Public Key Cryptography—PKC '03*, volume 2567 of *Lecture Notes in Computer Science*, pages 145–160, Berlin, 2003. Springer-Verlag.

[IG03] I. Ioannidis and A. Grama. An efficient protocol for Yao's millionaires' problem. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, page 6 pages. IEEE, 2003.

[JJ00] A. Juels and M. Jakobsson. Mix and match: Secure function evaluation via ciphertexts. In *Advances in Cryptology—ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177, Berlin, 2000. Springer-Verlag.

[JY96] M. Jakobsson and M. Yung. Proving without knowing: On oblivious, agnostic and blindfolded provers. In *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 186–200, Berlin, 1996. Springer-Verlag.

[KMO01] J. Katz, S. Myers, and R. Ostrovsky. Cryptographic counters and applications to electronic voting. In *Advances in Cryptology—EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 78–92, Berlin, 2001. Springer-Verlag.

[KO02] K. Kurosawa and W. Ogata. Bit-slice auction circuit. In *ESORICS 2002*, volume 2502 of *Lecture Notes in Computer Science*, pages 24–38, Berlin, 2002. Springer-Verlag.

[MNPS04] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay—a secure two-party computation system. In *Proceedings of Usenix Security 2004, August 9–13*, 2004. To appear.

[NN01] M. Naor and K. Nissim. Communication complexity and secure function evaluation. In *Mixer II, October 9, 2001, NEC Research Institute, Princeton, New Jersey*, DIMACS Mixer Series, 2001.

[Ped91] T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Berlin, 1991. Springer-Verlag.

[Pin03] B. Pinkas. Fair secure two-party computation. In *Advances in Cryptology—EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105, Berlin, 2003. Springer-Verlag.

[Yao82] A. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164. IEEE Computer Society, 1982.

Privacy in Non-private Environments*

Markus Bläser[1,**], Andreas Jakoby[2],
Maciej Liśkiewicz[2,***], and Bodo Manthey[2,†]

[1] Institut für Theoretische Informatik, ETH Zürich, Switzerland
mblaeser@inf.ethz.ch
[2] Institut für Theoretische Informatik, Universität zu Lübeck, Germany
{jakoby, liskiewi, manthey}@tcs.uni-luebeck.de

Abstract. We study private computations in information-theoretical settings on networks that are not 2-connected. Non-2-connected networks are "non-private" in the sense that most functions cannot privately be computed on them. We relax the notion of privacy by introducing lossy private protocols, which generalize private protocols. We measure the information each player gains during the computation. Good protocols should minimize the amount of information they lose to the players. Throughout this work, privacy always means 1-privacy, i.e. players are not allowed to share their knowledge. Furthermore, the players are honest but curious, thus they never deviate from the given protocol.

By use of randomness by the protocol the communication strings a certain player can observe on a particular input determine a probability distribution. We define the loss of a protocol to a player as the logarithm of the number of different probability distributions the player can observe. For optimal protocols, this is justified by the following result: For a particular content of any player's random tape, the distributions the player observes have pairwise fidelity zero. Thus the player can easily distinguish the distributions.

The simplest non-2-connected networks consists of two blocks that share one bridge node. We prove that on such networks, communication complexity and the loss of a private protocol are closely related: Up to constant factors, they are the same.

Then we study 1-phase protocols, an analogue of 1-round communication protocols. In such a protocol each bridge node may communicate with each block only once. We investigate in which order a bridge node should communicate with the blocks to minimize the loss of information. In particular, for symmetric functions it is optimal to sort the components by increasing size. Then we design a 1-phase protocol that for symmetric functions simultaneously minimizes the loss at all nodes where the minimum is taken over all 1-phase protocols.

Finally, we prove a phase hierarchy. For any k there is a function such that every $(k-1)$-phase protocol for this function has an information loss that is exponentially greater than that of the best k-phase protocol.

* The full version of this work appeared as Rev. 1 of Report 03-071, ECCC, 2003.
** Work done while at the Institut für Theoretische Informatik, Universität zuLübeck.
*** On leave from Instytut Informatyki, Uniwersytet Wrocławski, Poland.
† Supported by DFG research grant RE 672/3.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 137–151, 2004.

1 Introduction

Consider a set of players, each knowing an individual secret. They want to compute some function depending on their secrets. But after the computation, no player should know anything about the other secrets except for what he is able to deduce from his own secret and the function value. This is the aim of *private computation* (also called *secure multi-party computation*). To compute the function, the players can send messages to each other using secure links.

An example for such a computation is the "secret voting problem": The members of a committee wish to decide whether the majority votes for yes or no. But after the vote nobody should know anything about the opinions of the other members, not even about the exact number of yes and no votes, except for whether the majority voted for yes or no.

If no group of at most t players can infer anything about the input bits that cannot be inferred from the function value and their own input bits, we speak of t-privacy.

Any Boolean function can privately (in the following we identify privately with 1-privately) be computed on any 2-connected network. Unfortunately, there are many Boolean functions, even simple ones like parity or disjunction, that cannot privately be computed if the underlying network is not 2-connected [5].

However, many real-world networks are not 2-connected and private computation is not possible. If the players in the network have to compute something but do not trust each other, there is a natural interest of the players in privacy. What can we do? We relax the notion of privacy: One cannot require that any player learns only what he is able to deduce from his own secret and the function value. Instead we require that any player learns as little as possible about the secrets of the other players (in an information-theoretical sense) while it is still possible to compute the function.

Bridge nodes are important when considering non-2-connected networks. For all non-bridge players we can guarantee that they do not learn anything except for what they can deduce from their own bit and the function value. Thus, the bridge players are the only players that are able to learn something more. The question is now, how much the bridge players need to learn such that the function can be computed. The simplest setting is a network of two blocks with one bridge node in common. (A block is a maximal 2-connected subnetwork.) This reminds one of communication complexity with a man in the middle: Alice (one block) and Bob (another block) want to compute a function depending on their input while preventing Eve (the bridge node) from learning anything about their input. Unfortunately, Eve listens to the only communication channel between Alice and Bob. In terms of communication complexity, this problem had been examined by Modiano and Ephremedis [13, 14] and Orlitsky and El Gamal [17] under cryptographic security. In contrast, we deal with information-theoretical security, i.e. the computational power of the players is unrestricted. Furthermore, we are not interested in minimizing communication but in minimizing the information learned by any player. It turns out that there is a close relation between communication and privacy, at least in this special case.

1.1 Previous Results

Private computation was introduced by Yao [20]. He considered the problem under cryptographic assumptions. Private Computation with information-theoretical security has been introduced by Ben-Or et al. [3] and Chaum et al. [6]. Kushilevitz et al. [12] proved that the class of Boolean functions that have a circuit of linear size is exactly the class of functions that can privately be computed using only a constant number of random bits. Kushilevitz [10] and Chor et al. [7] considered private computations of integer-valued functions. They examined which functions can privately be computed by two players. Franklin and Yung [9] used directed hypergraphs for communication and described those networks on which every Boolean function can privately be computed.

While all Boolean functions can privately be computed on any undirected 2-connected network, Bläser et al. [5] completely characterized the class of Boolean functions that can still privately be computed, if the underlying network is connected but not 2-connected. In particular, no non-degenerate function can privately be computed if the network consists of three or more blocks. On networks with two blocks, only a small class of functions can privately be computed.

Chaum et al. [6] proved that any Boolean function can privately be computed, if at most one third of the participating players are dishonest, i.e. they are cheating. We consider the setting that all players are honest, i.e. they do not cheat actively but try to acquire knowledge about the input bits of the other players only by observing their communication. For this model, Ben-Or et al. [3] proved that any n-ary Boolean function can be computed $\lfloor \frac{n-1}{2} \rfloor$-private. Chor and Kushilevitz [8] showed that if a function can be computed at least $\frac{n}{2}$-private, then it can be computed n-private as well.

The idea of relaxing the privacy constraints has been studied to some extend in a cryptographic setting. Yao [20] examined the problem where it is allowed that the probability distributions of the messages seen by the players may differ slightly for different inputs, such that in practice the player should not be able to learn anything. Leakage of information in the information-theoretical sense has been considered only for two parties yet. Bar-Yehuda et al. [2] studied the minimum amount of information about the input that must be revealed for computing a given function in this setting.

1.2 Our Results

We study the leakage of information for *multi-party* protocols, where each player knows only a single bit of the input. Our first contribution is the definition of *lossy private protocols*, which is a generalization of private protocols in an information-theoretical sense (Section 2.2). Here and in the following, private always means 1-private. Throughout this work, we restrict ourselves to non-2-connected (in the sense of non-2-vertex-connected) networks that are still 2-edge-connected. Every block in such a network has size at least three and private computation within such a block is possible. We measure the information any particular player gains during the execution of the protocol in an information-theoretical sense. This is the *loss* of the protocol to the player. The players are assumed to be honest

but curios. This means that they always follow the protocol but try to derive as much information as possible.

We divide lossy protocols into phases. Within a phase, a bridge player may exchange messages only once with each block he belongs to. Phases correspond to rounds in communication complexity but they are locally defined for each bridge player.

In the definition of lossy protocols, the loss of a protocol to a player is merely the logarithm of the number of different probability distributions on the communication strings a player can observe. We justify this definition in Section 3: For a protocol with minimum loss to a player P and any particular content of P's random tape, the support of any two probability distributions is disjoint. Thus, in order to gain information, P can distinguish the distributions from the actual communication he observes and does not need to sample.

The simplest non-2-connected network consists of two blocks that share one bridge node. In Section 4 we show that the communication complexity of a function f and the loss of a private protocol for f are intimately connected: Up to constant factors, both quantities are equal.

Then we study 1-phase protocols. We start with networks that consist of d blocks that all share the same bridge player P. In a 1-phase protocol, P can communicate only once with each block he belongs to. However, the loss of the protocol may depend on the order in which P communicates with the blocks. In Section 5, we show that the order in which P should communicate with the blocks to minimize the loss equals the order in which d parties should be ordered on a directed line when they want to compute the function with minimum communication complexity. Particularly for symmetric functions, it is optimal to sort the components by increasing size. Then we design a 1-phase protocol (Theorem 4), which has the remarkable feature, that it achieves minimal loss at any node for symmetric functions. Hence, it simultaneously minimizes the loss for all nodes where the minimum is taken over all 1-phase protocols.

In Section 6, we prove a phase hierarchy. For any k there is a function for which every $(k-1)$-phase protocol has an exponentially greater information loss than that of the best k-phase protocol.

1.3 Comparison of Our Results with Previous Work

One of the important features of the two-party case is that at the beginning each party has knowledge about one half of the input. In the multi-party case each player knows only a single bit of the input.

Kushilevitz [10] examined which integer-valued functions can privately be computed by two players. He showed that requiring privacy can result in exponentially larger communication costs and that randomization does not help in this model. Chor et al. [7] considered multi-party computations of functions over the integers. They showed that the possibility of privately computing a function is closely related to its communication complexity, and they characterized the class of privately computable Boolean functions on countable domains. Neither Kushilevitz [10] nor Chor et al. [7] examined the problem how functions that

cannot privately be computed can still be computed while maintaining as much privacy as possible.

Leakage of information in the information-theoretical sense has been considered only for two parties, each holding one n-bit input of a two-variable function. Bar-Yehuda et al. [2] investigated this for functions that are not privately computable. They defined measures for the minimum amount of information about the individual inputs that must be learned during the computation and proved tight bounds on these costs for several functions. Finally, they showed that sacrificing some privacy can reduce the number of messages required during the computation and proved that at the costs of revealing k extra bits of information any function can be computed using $O(k \cdot 2^{(2n+1)/(k+1)})$ messages.

The counterpart of the two-party scenario in the distributed setting that we consider is a network that consists of two complete networks that share one node connecting them. Simulating any two-party protocol on such a network allows the common player to gain information depending on the deterministic communication complexity of the function that should be evaluated. Hence and in contrast to the two-party case, increasing the number of bits exchanged does not help to reduce the knowledge learned by the player that is part of either block. An important difference between the two-party scenario, where two parties share the complete input, and a network consisting of two 2-connected components connected via a common player (the bridge player) is that in the latter we have somewhat like a "man in the middle" (the bridge player) who can learn more than any other player, since he can observe the whole communication.

2 Preliminaries

For $i, j \in \mathbb{N}$, let $[i] := \{1, \ldots, i\}$ and $[i..j] := \{i, \ldots, j\}$. Let $x = x_1 x_2 \ldots x_n \in \{0,1\}^n$ be a string of length n. We often use the string operation $x_{i \leftarrow a}$ defined for any $i \in [n]$ and $a \in \{0,1\}$ by $x_1 \ldots x_{i-1}\, a\, x_{i+1} \ldots x_n$. For a function $f : \{0,1\}^n \to \{0,1\}$, an index $i \in [n]$, and $a \in \{0,1\}$, $f_{i \leftarrow \alpha} : \{0,1\}^{n-1} \to \{0,1\}$ denotes the function obtained from f by specialising the position i to the value given by a, i.e. for all $x = x_1 x_2 \ldots x_{n-1} \in \{0,1\}^{n-1}$,

$$f_{i \leftarrow a}(x) = f(x_1, \ldots, x_{i-1}, a, x_i, \ldots x_{n-1}) \,.$$

An undirected graph $G = (V, E)$ is called *2-connected*, if the graph obtained from G by deleting an arbitrary node is still connected. For a set $U \subseteq V$, let $G|_U := (U, E|_U)$ be the graph induced by U. A subgraph $G|_U$ is called a *block*, if $G|_U$ is 2-connected and no proper supergraph $G|_U$ is 2-connected. A block of size two is called an *isthmus*. A graph is called *2-edge-connected* if after removal of one edge, the graph is still connected. A graph is 2-edge-connected if it is connected and has no isthmi. A node belonging to more than one block is called a *bridge node*. The other nodes are called *internal nodes*. The blocks of a graph are arranged in a tree structure. For more details on graphs, see e.g. Berge [4].

A Boolean function is *symmetric*, if the function value depends only on the number of 1s in the input. See Wegener [19] for a survey on Boolean functions.

2.1 Private Computations

We consider the computation of Boolean functions $f : \{0,1\}^n \to \{0,1\}$ on a network of n players. In the beginning, each player knows a single bit of the input x. Each player has a random tape. The players can send messages to other players using secure links where the link topology is an undirected graph $G = (V, E)$. When the computation stops, all players should know the value $f(x)$. The goal is to compute $f(x)$ such that no player learns anything about the other input bits in an information-theoretical sense except for the information he can deduce from his own bit and the result. Such a protocol is called private.

Definition 1. *Let C_i be a random variable of the communication string seen by player P_i. A protocol $\mathcal{A}$ for computing a function f is* private with respect to player P_i *if for any pair of input vectors x and y with $f(x) = f(y)$ and $x_i = y_i$, for every c, and for every random string R_i provided to P_i,*

$$\Pr[C_i = c | R_i, x] = \Pr[C_i = c | R_i, y] \,,$$

where the probability is taken over the random strings of all other players. A protocol $\mathcal{A}$ is private *if it is private with respect to all players.*

In the following, we use a strengthened definition of privacy: We allow only one player, say P_i, to know the result. The protocol has to be private with respect to P_i according to Definition 1. Furthermore, for all players $P_j \neq P_i$, for all inputs x, y with $x_j = y_j$, and for all random strings R_j we require $\Pr[C_j = c | R_j, x] = \Pr[C_j = c | R_j, y]$. Thus, all other players do not learn anything. This definition does not restrict the class of functions computable by private protocols according to Definition 1. To achieve this additional restriction, P_i generates a random bit r. Then we use a private protocol for computing $r \oplus f(x)$.

2.2 Information Source

The definition of privacy basically states the following: The probability that a player P_i sees a specific communication string during the computation does not depend on the input of the other players. Thus, P_i cannot infer anything about the other inputs from the communication he observes.

If private computation is not possible since the graph is not 2-connected, it is natural to weaken the concept of privacy in the following way: We measure the information player P_i can infer from seeing a particular communication string. This leads to the concept of *lossy private protocols*. The less information any player can infer, the better the protocol is.

In the following, $c_1, c_2, c_3, \ldots$ denotes a fixed enumeration of all communication strings seen by any player during the execution of $\mathcal{A}$.

Definition 2. *Let C_i be a random variable of the communication string seen by player P_i while executing $\mathcal{A}$. Then for $a, b \in \{0,1\}$ and for every random string R_i provided to P_i, define the* information source *of P_i on a, b, and R_i as*

$$\mathcal{S}_{\mathcal{A}}(i, a, b, R_i) := \{(\mu_x(c_1), \mu_x(c_2), \ldots) \mid x \in \{0,1\}^n \wedge x_i = a \wedge f(x) = b\}$$

where $\mu_x(c_k) := \Pr[C_i = c_k | R_i, x]$ *and the probability is taken over the random strings of all other players.*

Basically $\mathcal{S}_{\mathcal{A}}(i, a, b, R_i)$ is the set of all different probability distributions on the communication strings observed by P_i when the input x of the players varies over all possible bit strings with $x_i = a$ and $f(x) = b$. The *loss* of a protocol $\mathcal{A}$ on a, b with respect to player P_i is

$$\ell = \max_{R_i} \log |\mathcal{S}_{\mathcal{A}}(i, a, b, R_i)| .$$

Thus the protocol looses ℓ bits of information to P_i. We call such a protocol ℓ-*lossy* on a, b with respect to P_i.

If a uniform distribution of the input bits is assumed, then the self-information of an assignment to the players $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ is $n - 1$ [18]. In this case the maximum number of bits of information that can be extracted by P_i is $n-1$. If $\mathcal{A}$ is 0-lossy for all $a, b \in \{0, 1\}$ with respect to P_i, then we say that $\mathcal{A}$ is *lossless* with respect to P_i. $\mathcal{A}$ is lossless to P_i iff $\mathcal{A}$ is private to P_i. Thus the notion of lossy private protocols generalizes the notion of private protocols.

Definition 3. *A protocol* $\mathcal{A}$ *computing a function* f *in a network* G *is* $\ell_{\mathcal{A}}$*-lossy, with* $\ell_{\mathcal{A}} : [n] \times \{0, 1\}^2 \to \mathbb{R}_0^+$*, if* $\ell_{\mathcal{A}}(i, a, b) = \max_{R_i} \log |\mathcal{S}_{\mathcal{A}}(i, a, b, R_i)|$*. Let* f *be an* n*-ary Boolean function. Then for every network* $G = (V, E)$ *with* $|V| = n$*, define* $\ell_G : [n] \times \{0, 1\}^2 \to \mathbb{R}_0^+$ *by*

$$\ell_G(i, a, b) := \min_{\mathcal{A}} \{ \ell_{\mathcal{A}}(i, a, b) \mid \mathcal{A} \text{ is an } \ell_{\mathcal{A}}\text{-lossy protocol for } f \text{ in } G \} .$$

The loss of a protocol $\mathcal{A}$ is *bounded by* $\lambda \in \mathbb{N}$, if $\ell_{\mathcal{A}}(i, a, b) \leq \lambda$ for all i, a, and b. $\ell_G(i, a, b)$ is obtained by locally minimizing the loss to each player P_i over all protocols. It is a priori not clear whether there is one protocol with $\ell_G(i, a, b) = \ell_{\mathcal{A}}(i, a, b)$ for all i, a, b. We show that this is the case for symmetric functions and 1-phase protocols (as defined in Section 2.3).

We also use the size of the information source, defined by $s_{\mathcal{A}}(i, a, b, R_i) = |\mathcal{S}_{\mathcal{A}}(i, a, b, R_i)|$ and $s_{\mathcal{A}}(i, a, b) = \max_{R_i} s_{\mathcal{A}}(i, a, b, R_i)$ for a given protocol $\mathcal{A}$. By definition, $\ell_{\mathcal{A}}(i, a, b) = \log s_{\mathcal{A}}(i, a, b)$. If the underlying protocol is clear from the context, we omit the subscript $\mathcal{A}$. Let f be an n-ary Boolean function. For a network $G = (V, E)$ with $|V| = n$, we define $s_G(i, a, b) := \min_{\mathcal{A}} s_{\mathcal{A}}(i, a, b)$. If a player P_i is an internal node of the network, then it is possible to design protocols that are lossless with respect to P_i (see Section 3). Players P_i that are bridge nodes are in general able to infer some information about the input.

2.3 Phases in a Protocol

We say that a player P_q who corresponds to a bridge node makes an *alternation* if he finishes the communication with one block and starts to communicate with another block. During such an alternation, information can flow from one block to another. We partition a communication sequence $c = d_1 d_2 \ldots$ of P_q into a minimal number of disjoint subsequences $(d_1, \ldots, d_{i_1}), (d_{i_1+1}, \ldots, d_{i_2}), \ldots$ such

that each subsequence is alternation-free (i.e. P_q makes no alternation during the corresponding interval). To make such a partition unique assume that each subsequence (maybe except for the first one) starts with a non-empty message. We call these subsequences *block sequences* of c and define $\text{block}_j(c) := (d_{i_{j-1}+1}, \ldots, d_{i_j})$ with $i_0 = 0$. Next we partition the work of P_q into phases as follows. P_q starts at the beginning of the first phase and it initiates a new phase when, after an alternation, it starts to communicate again with a block it already has communicated with previously in the phase.

A protocol $\mathcal{A}$ is a *k-phase protocol for a bridge node P_q* if for every input string and contents of all random tapes, P_q works in at most k phases. $\mathcal{A}$ is called a *k-phase protocol* if it is a k-phase protocol for every bridge node.

The start and end round of each phase does not need to be the same for each player. Of particular interest are 1-phase protocols. In such a protocol, each bridge player may only communicate once with each block he belongs to. Such protocols seem to be natural, since they have a local structure. Once the computation is finished in one block, the protocol will never communicate with this block again.

For k-phase protocols we define $\ell_G^k(i, a, b)$ and $s_G^k(i, a, b)$ in a similar way as $\ell_{\mathcal{A}}$ and s_G in the general case, but we minimize over all k-phase protocols.

During each phase a player communicates with at least two blocks. The order in which the player communicates within a phase can matter. The *communication order* σ_q of a bridge node P_q specifies the order in which P_q communicates with the blocks during the whole computation. Formally, σ_q is a finite sequence of (the indices of) blocks P_q belongs to and the length of σ_q is the total number of alternations made by P_q plus one. We say that a protocol is σ_q-ordered for P_q if for all inputs and all contents of the random tapes, the communication order of P_q is consistent with σ_q. Let $P_{q_1}, \ldots, P_{q_k}$ with $q_1 < q_2 < \ldots < q_k$ be an enumeration of all bridge players of a network G and $\sigma = (\sigma_{q_1}, \ldots, \sigma_{q_k})$ be a sequence of communication orders. We call a protocol *σ-ordered* if it is σ_{q_j}-ordered for every P_{q_j}. Finally, define $s_G(i, a, b, \sigma) := \min\{s_{\mathcal{A}}(i, a, b) \mid \mathcal{A} \text{ is } \sigma\text{-ordered for } f \text{ on } G\}$.

2.4 Communication Protocols

For comparing the communication complexity with the loss of private protocols, we need the following definitions. Let $f : \{0,1\}^{m_1} \times \{0,1\}^{m_2} \to \{0,1\}$ be a Boolean function and $\mathcal{B}$ be a two-party communication protocol for computing f. Let $y_1 \in \{0,1\}^{m_1}$ and $y_2 \in \{0,1\}^{m_2}$ be two strings as input for the two parties. Then $\text{CC}_{\mathcal{B}}(y_1, y_2)$ is the total number of bits exchanged by the two parties when executing $\mathcal{B}$.

$\text{CC}(\mathcal{B})$ is the maximum number of bits exchanged by executing $\mathcal{B}$ on any input. Analogously, $\text{CS}(\mathcal{B})$ is the number of different communication strings that occur. (We simply concatenate the messages sent.) Finally, we define $\text{CC}(f) = \min_{\mathcal{B} \text{ for } f} \text{CC}(\mathcal{B})$ and $\text{CS}(f) = \min_{\mathcal{B} \text{ for } f} \text{CS}(\mathcal{B})$.

$\text{CC}(f)$ and $\text{CS}(f)$ are the communication complexity and communication size, respectively, of the function f. $\text{CC}(\mathcal{B})$ and $\text{CS}(\mathcal{B})$ are the communication complexity and communication size for a certain protocol $\mathcal{B}$. The communication size is closely related to the number of leaves in a protocol tree, usually denoted

by $C^P(\mathcal{B})$. In the definition of CS, we do not care about who has sent any bit, since we concatenate all messages. In a protocol tree however, each edge is labeled by the bit sent and by its sender. The bits on a path from the root to a leaf form a communication string. Usually, the messages sent in a communication protocol are assumed to be prefix-free. In this case, we can reconstruct the sender of any bit from the communication string. If this is not the case, then we can make a particular communication protocol prefix-free by replacing the messages sent in each round by prefix-free code words. The complexity is at most doubled.

We also consider multi-party communication with a referee. Let $f : \{0,1\}^{m_1} \times \{0,1\}^{m_2} \times \ldots \times \{0,1\}^{m_k} \to \{0,1\}$ be a function. Let $A_1, \ldots, A_k$ be k parties and R be a referee, all with unlimited computational power. For computing f on input $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$, the referee cooperates with $A_1, \ldots, A_k$ as follows:

- Initially, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ are distributed among $A_1, \ldots, A_k$, i.e. A_i knows $\boldsymbol{x}_i$. The referee R does not have any knowledge about the inputs.
- In successive rounds, R exchanges messages with $A_1, \ldots, A_k$ according to a communication protocol. In each round R can communicate (i.e. receive or send a message) only with a single party.
- After finishing the communications, R eventually computes the result of f.

Let $\mathcal{B}$ be a communication protocol for computing f. Denote by $\mathrm{c}^R_{\mathcal{B}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ the whole communication string of R after protocol $\mathcal{B}$ has been finished. More precisely, $\mathrm{c}^R_{\mathcal{B}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ is a concatenation of messages sent (to or from R) on input $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ with additional stamps describing the sender and the receiver of each message. For $b \in \{0,1\}$ let

$$\begin{aligned}
\mathcal{CS}^R_{\mathcal{B}}(b) &= \{\mathrm{c}^R_{\mathcal{B}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \mid \forall i \in [k] : \boldsymbol{x}_i \in \{0,1\}^{m_i} \text{ and } f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = b\}\,, \\
\mathcal{CS}^R(\mathcal{B}) &= \mathcal{CS}^R_{\mathcal{B}}(0) \cup \mathcal{CS}^R_{\mathcal{B}}(1)\,, \\
\mathrm{CS}^R_{\mathcal{B}}(b) &= |\mathcal{CS}^R_{\mathcal{B}}(b)|\,, \qquad \mathrm{CS}^R(\mathcal{B}) = |\mathcal{CS}^R(\mathcal{B})|\,, \\
\mathrm{CS}^R(f,b) &= \min_{\mathcal{B} \text{ for } f} \mathrm{CS}^R_{\mathcal{B}}(b)\,, \text{ and } \mathrm{CS}^R(f) = \min_{\mathcal{B} \text{ for } f} \mathrm{CS}^R(\mathcal{B})\,.
\end{aligned}$$

3 The Suitability of the Model

We observe that it suffices to consider bridge players when talking about the loss of a protocol. More precisely, any protocol can be modified such that the loss to all internal players is zero, while the loss to any bridge player does not increase.

All Boolean functions can be computed by using only three players [3]. Thus, it is possible to compute functions privately within any block, since the networks we consider are isthmus-free. This holds even if some of the players know a subset of the input bits and the result consists of a binary string.

Finally, in optimal protocols, the probability distributions observed by any player have pairwise fidelity 0. Thus, any player can easily distinguish the different probability distributions he observes.

We consider arbitrary 1-connected networks. Let f be a Boolean function and $\mathcal{A}$ be a protocol for computing f on a 1-connected network G. Let P_q be a bridge player of G, $a, b \in \{0,1\}$, and R_q be the random string provided to P_q.

We define $X := \{x \in \{0,1\}^n \mid x_q = a \wedge f(x) = b\}$ and, for any communication string c, $\psi(c) := \{x \in X \mid \mu_x(c) > 0\}$, where $\mu_x(c) = \Pr[C_q = c | R_q, x]$. For every communication string c that can be observed by P_q on some input $x \in X$, P_q can deduce that $x \in \psi(c)$. If $s_{\mathcal{A}}(q,a,b) = s_G(q,a,b) = 1$, then we have either $\psi(c) = X$ or $\psi(c) = \emptyset$. Thus P_q does not learn anything in this case.

Theorem 1. *If $s_G(q,a,b) > 1$, then for any protocol $\mathcal{A}$ and every communication string c that can be observed by P_q on $x \in X$, $\psi(c)$ is a non-trivial subset of X, i.e. $\emptyset \neq \psi(c) \subsetneq X$, and there exist at least $s_G(q,a,b)$ different such sets.*

Hence, from seeing c on $x \in X$, P_q always gains some information and there are at least $s_G(q,a,b)$ different pieces of information that can be extracted by P_q on inputs from X. To prove this, we show that for each distribution we can find one representative string that can be used in the communication protocol.

The next result says that $s_G(q,a,b)$ is a tight lower bound on the number of pieces of information: the lower bound is achieved when performing an optimal protocol on G. Let μ and μ' be two probability distributions over the same set of elementary events. The *fidelity* is a measure for the similarity of μ and μ' (see e.g. Nielsen and Chuang [15]) and is defined by $F(\mu,\mu') = \sum_c \sqrt{\mu(c) \cdot \mu'(c)}$.

Theorem 2. *If $\mathcal{A}$ is an optimal protocol for P_q on a and b, i.e. $s_{\mathcal{A}}(q,a,b) = s_G(q,a,b)$, then for all random strings R_q and all probability distributions $\mu \neq \mu'$ in $\mathcal{S}_{\mathcal{A}}(q,a,b,R_q)$ we have $F(\mu,\mu') = 0$.*

4 Communication Complexity and Private Computation

In this section, we investigate the relations between deterministic communication complexity and the minimum size of an information source in a network with one bridge node. To distinguish protocols in terms of communication complexity and protocols in terms of private computation, we will call the former communication protocols. From the relation between C^P and CC (see e.g. Kushilevitz and Nisan [11–Sec. 2.2]), we get $\frac{1}{2}\log(\mathrm{CS}(f)) \leq \mathrm{CC}(f) \leq 3 \cdot \log(\mathrm{CS}(f))$. Making a communication protocol prefix-free yields the extra factor $\frac{1}{2}$.

Now we investigate the relations between communication size and the size of an information source on graphs that consist of two blocks sharing one bridge node P_q. In the model of private computation the input bits are distributed among n players whereas the input bits in a communication protocol are distributed among the two parties. Alice and Bob correspond to the first and second block, respectively, while both know the bridge player's bit.

Theorem 3. *If a function f has communication complexity c then there exists a protocol for computing f with loss bounded by $2c$. On the other hand, if f can be computed by a protocol with loss bounded by λ, then the communication complexity of f is bounded by $6\lambda + O(1)$.*

We can generalize the results obtained for the relation two-party communication and private computation to obtain similar results for the relation of multi-party communication with a referee and private computation as follows: For $a, b \in \{0,1\}$ we have $s_G(q,a,b) = \mathrm{CS}^R(f_{q \leftarrow a}, b)$.

5 1-Phase Protocols

We start our study of 1-phase protocols with considering networks that consist of one bridge player who is incident with d blocks. For the case that the order in which the bridge player communicates with the blocks is fixed for all inputs, we show a relationship between the size of the information source of 1-phase protocols and communication size of multi-party 1-way protocols. Furthermore, we prove that for every symmetric Boolean function 1-phase protocols can minimize the loss of information when the bridge player sorts the blocks by increasing size. Then we present a simple 1-phase protocol on arbitrarily connected networks that is optimal for every symmetric function.

5.1 Orderings

A natural extension of the two-party scenario for 1-way communication is a scenario in which the parties use a directed chain for communication: d parties $A_1, \ldots, A_d$ are connected by a directed chain, i.e. A_i can only send messages to A_{i+1}. For a communication protocol $\mathcal{B}$ on G and $i \in [d]$ let $S_i^{\rightarrow}(\mathcal{B})$ be the number of possible communication sequences on the subnetwork of $A_1, \ldots, A_i$. Each communication protocol $\mathcal{B}$ can be modified without increasing $S_i^{\rightarrow}(\mathcal{B})$ in the following way: Every party A_i first sends the messages it has received from A_{i-1} to A_{i+1} followed by the messages it has to send according to $\mathcal{B}$. In the following we restrict ourselves to communication protocols of this form.

If the network G consists of d blocks B_i with $i \in [d]$ and one bridge player P_q we consider a chain of d parties $A_1, \ldots, A_d$. For a σ-ordered 1-phase protocol $\mathcal{A}$, we assume that the enumeration of the blocks reflects the ordering σ. We have to determine the input bits of the parties in the chain according to the input bits of the players in the protocol. In the following we will assume that A_i knows the input bits of the players in B_i. Thus, each party A_i has to know the input bit x_q of the bridge player P_q. Therefore, we will investigate the restricted function $f_{q \leftarrow a}$ whenever we analyse the communication size of a communication protocol.

For a σ-ordered protocol $\mathcal{A}$ define $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, b, R_q) = \{\hat{\mu}_x \mid x_q = a \wedge f(x) = b\}$, where $\hat{\mu}_x(\hat{c}_k)$ denotes the sum of the probabilities $\Pr[C_q = c \mid R_q, x]$ over all c with $\widehat{c}_k = \mathrm{block}_1(c) \ldots \mathrm{block}_i(c)$ and $\widehat{c}_1, \widehat{c}_2, \widehat{c}_3, \ldots$ is a fixed enumeration of all strings describing the communication of P_q in the first i block sequences.

Lemma 1. *Let $\mathcal{A}$ be a σ-ordered 1-phase protocol for computing f on a network as described above. Then for every $a \in \{0,1\}$ and every content R_q of P_q's random tape there exists a 1-way communication protocol $\mathcal{B}$ for computing $f_{q \leftarrow a}$ such that for all $i \in [d-1]$, we have*

$$S_i^{\rightarrow}(\mathcal{B}) \leq |\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, R_q) \cup \mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, R_q)| \,.$$

Let us now focus on the structure of the possible communication sequences of an optimal communication protocol on a chain. In such a protocol, the message sent from A_i to A_{i+1} has to specify the subfunction obtained by specifying the input bits of the first i blocks according to their input. Hence, the number of

possible communication sequences on the network $A_1, \ldots, A_d$ is at least the number of different sequences of subfunctions that can be obtained in this way.

The knowledge about these sequences must also be provided to the bridge player. Hence, for every fixed R_q and $b \in \{0,1\}$ the number of distributions in $\mathcal{S}_{\mathcal{A}}^{[d-1]}(q,a,b,R_q)$ is at least the number of different sequences $f_{1,x}, \ldots, f_{d-1,x}$ for inputs x with $x_q = a$ and $f(x) = b$. This implies the following lemma.

Lemma 2. *For $a \in \{0,1\}$, let $\mathcal{B}$ be a communication protocol for computing $f_{q \leftarrow a}$ on a chain network. Then there exists a σ-ordered 1-phase protocol $\mathcal{A}$ for f such that for all $i \in [d-1]$ and every content R_q of P_q's random tape, we have*

$$S_i^{\rightarrow}(\mathcal{B}) = |\mathcal{S}_{\mathcal{A}}^{[i]}(q,a,0,R_q) \cup \mathcal{S}_{\mathcal{A}}^{[i]}(q,a,1,R_q)| \,.$$

Furthermore, for any $b \in \{0,1\}$: If we restrict the inputs to $x \in \{0,1\}^{n-1}$ with $f_{q \leftarrow a}(x) = b$, the number of possible communication sequences on the subnetwork $A_1, \ldots, A_{i+1}$ is $|\mathcal{S}_{\mathcal{A}}^{[i]}(q,a,b,R_q)|$.

We can show that there exist functions, for which no ordered 1-phase protocol minimizes the size of the bridge players' information source. Thus, we generalize the class of ordering that we consider to achieve such a property.

We call a protocol $\mathcal{A}$ *quasi-ordered* if for every $a, b \in \{0,1\}$, for every content R_q for P_q's random tape, and for every distribution $\mu \in \mathcal{S}_{\mathcal{A}}(q,a,b,R_q)$ there exists a 1-phase ordering σ such that every communication string c with $\mu(c) > 0$ the string c is σ-ordered. Note that this ordering is not necessarily the same for all inputs. However, given any input, the ordering is fixed.

We can prove that among all 1-phase protocols for a given function, there always exists a quasi-ordered protocol that minimizes the loss to P_q.

5.2 Orderings for Symmetric Functions

For symmetric Boolean functions, we can show even more. Arpe et al. [1] have proved the following for symmetric Boolean functions with a fixed partition of the input bits: for all i, $S_i^{\rightarrow}(\mathcal{B})$ is minimal, if the number of bits known by the parties in the chain corresponds to the position of the party, i.e. the first party knows the smallest number of input bits, the second party knows the second smallest number, and so on. This observation also holds, if we count the number of communication sequences in a chain network for inputs x with $f(x) = 1$ and the number of communication sequences in a chain network for inputs x with $f(x) = 0$. Together with Lemma 2, we obtain the following: Let G be a connected network with one bridge player P_q and d blocks. Let σ be a one phase ordering that enumerates the blocks of G according to their size. Then for every ordered 1-phase protocol $\mathcal{A}'$ there exists a σ-ordered 1-phase protocol $\mathcal{A}$ such that for all $a, b \in \{0,1\}$, for all $i \leq d-1$, and every content R_q of P_q' random tape $|\mathcal{S}_{\mathcal{A}}^{[i]}(q,a,b,R_q)| \leq |\mathcal{S}_{\mathcal{A}'}^{[i]}(q,a,b,R_q)|$.

This result can be generalized to networks with more than one bridge player. Let $G_1, \ldots, G_k$ be the connected subgraphs obtained by deleting the bridge player P_q with $|G_i| \leq |G_{i+1}|$. We say that P_q works in increasing order, if

it starts communicating with G_1, then with G_2 and so on. We call a 1-phase protocol $\mathcal{A}$ *increasing-ordered*, if every bridge player works in increasing order.

For a graph G let $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_h = (V_h, E_h)\}$ be the set of blocks and $\mathcal{Q} = \{q_1, \ldots, q_k\}$ be the set of bridge nodes of G. Every graph G induces a tree $T_G = (V_G, E_G)$ defined as follows: $V_G = V_\mathcal{Q} \cup V_\mathcal{G}$ with $V_\mathcal{Q} = \{u_1, \ldots, u_k\}$ and $V_\mathcal{G} = \{v_1, \ldots, v_h\}$ and $E_G = \{\{u_i, v_j\} \mid q_i \in V_j\}$.

For every 1-phase communication order $\sigma = (\sigma_{q_1}, \ldots, \sigma_{q_k})$ and every bridge node q_i the order σ_{q_i} defines an ordering of the nodes $v_j \in V_\mathcal{G}$ adjacent to the tree-node u_i. Let $G_{\sigma_{q_i}(1)}, \ldots, G_{\sigma_{q_i}(k_i)}$ denote the ordering of blocks adjacent to q_i with respect to σ_{q_i} and $\mathrm{root}_\sigma(u_i) := v_{\sigma_{q_i}(k_i)}$. If σ is an increasing communication order, then there exists a single tree-node $v_j \in V_\mathcal{G}$, such that $v_j = \mathrm{root}_\sigma(u_i)$ for all $u_i \in V_\mathcal{Q}$ adjacent to v_j. Let us call this node the root of T_G. For a tree-node $w \in V_G$ let $T_G[w]$ denote the subtree of T_G rooted by w and let $V[w]$ denote the nodes of G located in the blocks G_j with $v_j \in T_G[w]$.

For computing a symmetric function f we use the following protocol. Let σ be an increasing communication order. Then for an input x every bridge player q_i computes a sequence of strings $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k_i-1}$ as follows: Let $X_j = \bigcup_{e \in [j]} V[v_{\sigma_{q_i}(e)}]$ and $\ell_j = |X_j|$. Then $\boldsymbol{y}_j \in \{0,1\}^{\ell_j}$ such that for all $j \leq k_i - 1$ the function obtained from f by specialising the positions in X_j to $\boldsymbol{y}_j$ is equal to the function obtained from f by specialising the positions to x_{X_j}, where x_I for $I \subseteq [n]$ denotes the input bits with indices in I. Finally, a node of the block that corresponds to the root of T_G computes the result $f(x)$. This can be implemented such that no player gains any additional information except for $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k_i-1}$ learned by the bridge nodes q_i.

Theorem 4. *Let G be a 2-edge-connected network and f be a symmetric Boolean function. Then for every 1-phase protocol $\mathcal{A}'$ computing f on G there exists an increasing-ordered 1-phase protocol $\mathcal{A}$ for f on G such that for every player P_i and for all $a, b \in \{0,1\}$, we have $s_\mathcal{A}(i,a,b) \leq s_{\mathcal{A}'}(i,a,b)$.*

Thus, the protocol presented in this section is optimal for 1-phase computations of symmetric functions with respect to the size of the information source.

6 A Phase Hierarchy

In this section we show that there are functions for which the size of the information source of some player for a $(k-1)$-phase protocol is exponentially larger than for a k-phase protocol. The natural candidate for proving such results is the pointer jumping function p_j: Our network G has two blocks A and B, one of size $n \log n$ and the other of size $n \log n + 1$, sharing one bridge player P_i. For simplicity we assume that A and B are complete subgraphs. The input bits represent two lists of n pointers, each of length $\log n$ bits. The input bit of P_i belongs to the list of the smaller component. Starting with some predetermined pointer of A, the task is to follow these pointers, find the jth pointer and output the parity of the bits of the jth pointer. Define CS^j and CC^j in the same manner

as CS and CC, but by minimizing over j-round communication protocols instead of arbitrary communication protocols.

Theorem 5. *For any protocol $\mathcal{A}$ for computing p_{2k-1}, we have $s_{\mathcal{A}}^{k-1}(i,a,b) = 2^{\Omega(n/(k\log k))}$ for all a,b. For p_{2k-1}, $s_G^k(i,a,b) = 2^{O(k\log n)}$ for all a,b.*

The lower bound follows from work by Nisan and Wigderson [16].

7 Conclusions and Open Problems

We have considered distributed protocols in "non-private" environments: networks that are connected but not 2-connected. Since private computation of arbitrary Boolean functions is impossible on such networks, we have introduced a measure for the information that can be inferred by any player and discussed some general properties of protocols with respect to this measure. A natural question is finding optimal protocols for some concrete functions.

For threshold ($f_{n_0}(x_1,\ldots,x_n) = 1$ iff $\sum_{i=1}^n x_i \geq n_0$) and counting modulo p ($g_p(x_1,\ldots,x_n) = 1$ iff $\sum_{i=1}^n x_i \equiv 0 \pmod p$), the information loss to any player does not depend on the ordering in which a 1-phase protocol computes any of these functions, if each block has size at least n_0 and p, respectively. If we have blocks of less than $p-1$ nodes, there can be a slight difference in the size of P_q's information source depending on the order.

In general, the size of the information source while communicating in one order can be exponentially larger than the size obtained by communication in another order. This holds even in case of symmetric functions.

For 1-phase protocols for symmetric Boolean functions, we have been able to minimize the number of bits a player learns for all players simultaneously. An obvious question concerns minimizing the loss of more than one bridge player simultaneously for general functions. For 1-phase protocols, the answer is negative: There are functions, for which no protocol exists that minimizes the loss to all players simultaneously.

It is open whether there exist functions and networks that do not allow to minimize the loss to each bridge player simultaneously. For such functions, we have to generalize our measure. Two simple examples one might want to examine is the sum of the loss to each player and the maximum loss to any player.

References

1. Jan Arpe, Andreas Jakoby, and Maciej Liśkiewicz. One-way communication complexity of symmetric boolean functions. In A. Lingas and B. J. Nilsson, editors, *Proc. of the 14th Int. Symp. on Fundamentals of Computation Theory (FCT)*, volume 2751 of *Lecture Notes in Computer Science*, pages 158–170. Springer, 2003.
2. Reuven Bar-Yehuda, Benny Chor, Eyal Kushilevitz, and Alon Orlitsky. Privacy, additional information, and communication. *IEEE Transactions on Information Theory*, 39(6):1930–1943, 1993.

3. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 1–10. ACM Press, 1988.
4. Claude Berge. *Graphs*. North-Holland, 1991.
5. Markus Bläser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Siebert. Private computation — k-connected versus 1-connected networks. In M. Yung, editor, *Proc. of the 22nd Ann. Int. Cryptology Conf. (CRYPTO)*, volume 2442 of *Lecture Notes in Computer Science*, pages 194–209. IACR, Springer, 2002.
6. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 11–19. ACM Press, 1988.
7. Benny Chor, Mihály Geréb-Graus, and Eyal Kushilevitz. Private computations over the integers. *SIAM Journal on Computing*, 24(2):376–386, 1995.
8. Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy. *SIAM Journal on Discrete Mathematics*, 4(1):36–47, 1991.
9. Matthew Franklin and Moti Yung. Secure hypergraphs: Privacy from partial broadcast. In *Proc. of the 27th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 36–44. ACM Press, 1995.
10. Eyal Kushilevitz. Privacy and communication complexity. *SIAM Journal on Discrete Mathematics*, 5(2):273–284, 1992.
11. Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.
12. Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. *Journal of Computer and System Sciences*, 58(1):129–136, 1999.
13. Eytan H. Modiano and Anthony Ephremides. Communication complexity of secure distributed computation in the presence of noise. *IEEE Transactions on Information Theory*, 38(4):1193–1202, 1992.
14. Eytan H. Modiano and Anthony Ephremides. Communication protocols for secure distributed computation of binary functions. *Information and Computation*, 158(2):71–97, 2000.
15. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*, chapter 9. Cambridge University Press, 2000.
16. Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, 1993.
17. Alon Orlitsky and Abbas El Gamal. Communication with secrecy constraints. In *Proc. of the 16th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 217–224. ACM Press, 1984.
18. Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3, 4):379–423 & 623–656, 1948.
19. Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.
20. Andrew Chi-Chih Yao. Protocols for secure computations. In *Proc. of the 23rd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE Computer Society, 1982.

Asynchronous Proactive Cryptosystems Without Agreement
(Extended Abstract)*

Bartosz Przydatek[1] and Reto Strobl[2]

[1] Department of Computer Science, ETH Zürich, Switzerland
[2] IBM Research, Zurich Research Laboratory, Switzerland

Abstract. In this paper, we present efficient asynchronous protocols that allow to build proactive cryptosystems secure against a mobile fail-stop adversary. Such systems distribute the power of a public-key cryptosystem among a set of servers, so that the security and functionality of the overall system is preserved against an adversary that crashes and/or eavesdrops every server repeatedly and transiently, but no more than a certain fraction of the servers at a given time. The building blocks of proactive cryptosystems — to which we present novel solutions — are protocols for *joint random secret sharing* and for *proactive secret sharing*.

The first protocol provides every server with a share of a random value unpredictable by the adversary, and the second allows to change the shared representation of a secret value. Synchronous protocols for these tasks are well-known, but the standard method for adapting them to the asynchronous model requires an asynchronous agreement sub-protocol. Our solutions are more efficient as they go without such an agreement sub-protocol. Moreover, they are the first solutions for such protocols having a bounded *worst-case* complexity, as opposed to only a bounded *average-case* complexity.

1 Introduction

Threshold cryptography addresses the task of distributing a cryptosystem among n servers such that the security and functionality of this distributed system is guaranteed even if an adversary corrupts up to t servers [2] (see [3] for a survey). Threshold cryptosystems are realized by sharing the key of the underlying cryptosystem among all servers using a $(t+1)$-out-n sharing scheme [4], and by accomplishing the cryptographic task through a distributed protocol. If this task involves the choice of secret random values, then the distribution of the task involves so-called *joint random secret sharing* (JRSS) [5], which allow the servers to jointly generate a $(t+1)$-out-n sharing of a random value unpredictable by the adversary.

Proactive cryptosystems use threshold cryptosystems as the basis, but drastically reduce the assumption concerning failures [6] (see [7] for a survey). They operate in a sequence of time periods called *phases* and tolerate a *mobile adversary*, which corrupts the servers transiently and repeatedly, and is only restricted to corrupt at most t servers

* The full version of this paper is available as an IBM Technical Report [1].

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 152–169, 2004.

during every phase. Technically, proactive cryptosystems are threshold cryptosystems that change the representation of the shared secret key from one phase to another using *proactive secret sharing* (PSS) [8], so that the representations are independent; the old representation has to be *erased*.

The key to efficient proactivization of many public key cryptosystems for signing and encryption lies in efficient solutions for JRSS and for PSS. In the synchronous network model with broadcast channels, such solutions exist [5, 8]. Although such synchrony assumptions are justified in principle by the existence of clock synchronization and broadcast protocols, this approach may lead to rather expensive solutions in practice, for example when deployed in wide-area distributed systems with only loosely synchronized clocks. Furthermore, such systems are vulnerable to timing attacks.

These issues can be eliminated by considering an asynchronous network in the first place. However, the standard approach to building asynchronous protocols for JRSS and PSS requires an asynchronous agreement sub-protocol, which substantially contributes to the overall complexity of such solutions; see for example [9].

Contributions. In this paper, we provide the first solutions for asynchronous JRSS and for asynchronous PSS, which do not rely on an agreement sub-protocol. Avoiding agreement results in two main advantages. On one hand, we are able to bound the *worst-case* complexity of our protocols. For previous protocols, one could only bound their *average case* complexity; such protocols therefore could (at least theoretically) run forever. On the other hand, our protocols have a worst-case latency of only six rounds, whereas the best known previous solution of Cachin et al. [9] has an *expected* latency of 17 rounds (this comparison takes into account that [9] can be optimized in our model).

Our protocols tolerate a fail-stop adversary who may adaptively and repeatedly eavesdrop and crash up to t servers in every two subsequent phases, where $t < n/3$. We stress that assuming a fail-stop adversary (as opposed to a fully Byzantine adversary) does not make the problem of avoiding agreement trivial: the main reason why the standard solutions for asynchronous JRSS and PSS require agreement is the fact that a crashed server cannot be distinguished from a slow server, and this problem also occurs for a fail-stop adversary. Note that in principle our protocols can be extended to tolerate Byzantine adversaries without affecting the resilience of $t < n/3$, using known techniques for asynchronous verifiable secret sharing [9] and zero-knowledge proofs [10]. Furthermore, as shown in [11–Chapter 7], our protocols remain secure even under arbitrary composition.

The cost of our approach is a higher communication complexity. Specifically, if k is the security parameter of the system, our protocols transmit a total of $O(kn^4)$ bits across the network using $O(n^3)$ messages, whereas the (optimized) solution of Cachin et al. [9] transmits only $O(kn^3)$ bits using also $O(n^3)$ messages. However, in a practical setting, this additional overhead is of little concern as the size of n is typically very small relative to k (e.g. 10 vs. 1024).

Technically, the key to our solutions is a novel *proactive pseudorandomness* (PPR) scheme [12], with an additional property that we call *constructibility*. Such a scheme provides at every phase to every server P_i a random value pr_i which remains hidden from the adversary. Additionally, it enables the honest servers to jointly reconstruct any such value pr_i. We then build our JRSS and PSS schemes such that a server P_i derives all its

random choices from its value pr_i by using it as a seed to a pseudorandom function [13]. This allows the honest servers to reproduce the steps of a (possibly) faulty server in public, instead of agreeing on a set of such servers and then excluding them from the computation (as it is done by previous work).

Related Work. As mentioned previously, Cachin *et al.* [9] implemented asynchronous proactive protocols using an agreement subprotocol as a building block, which results in a relatively high round complexity. Zhou [14] proposed to build proactive cryptosystems on a weaker notion of PSS, which can be implemented without agreement. In this weaker PSS protocol, every server computes in every phase a list of candidate shares such that one of these candidates is the fresh share of the secret. Zhou shows that this suffices to implement a proactive version of RSA signatures exploiting the fact that RSA signatures are unique in the sense that for any public key and any message, there exists only *one* signature on the given message valid under the given public key. Unfortunately, the approach of Zhou [14] cannot be applied to proactivize discrete-logarithm signature schemes such as ElGamal [15] or DSS [16], as these schemes are not unique in the above sense. The only known technique for proactivizing these signature schemes are protocols for JRSS and for PSS in the sense we introduced them before.

Organization. In the next section we introduce our system model, and recall the definitions of cryptographic tools we use in the proposed solutions. In Section 3 we give an overview of our constructions. Section 4 presents an efficient secret sharing protocol, which will be useful in our constructions. In Section 5 we present our solution for an asynchronous proactive pseudorandomness scheme. In Sections 6 and 7, we describe our solutions to asynchronous proactive secret sharing, and to asynchronous joint random secret sharing, respectively. In Section 8 we sketch how these protocols can be used to proactivize public-key signature schemes, considering Schnorr's signature scheme [17] as an example. Finally, in Section 9 we conclude the paper.

2 Asynchronous Proactive System Model

Motivation. Proactive cryptosystems are threshold cryptosystems that operate in a sequence of phases. At the beginning of every phase, the servers *refresh* the shares of the underlying threshold system such that the new shares are independent of the old shares (except for the fact that they define the same secret). This prevents an adversary from learning the shared key, assuming that she corrupts no more than t servers in every phase. Such an assumption can be justified if every phase lasts some limited amount of real time, the idea being that it takes the adversary a certain amount of real time to corrupt a server, and that corruptions are transient, i.e., do not last forever [6].

This idea maps onto a synchronous network in a straightforward way: one can define phases with respect to a common clock accessible to every server and implement refresh using a synchronous protocol [8]. The drawback of this approach is that synchronous protocols proceed in rounds, i.e., messages are sent on a clock "tick", and are received at the next "tick". This may lead to slow protocols in practice, as the duration of a communication round must account for maximal message delays and maximal shifts

among local clocks of the servers. Moreover, as the security of synchronous protocols relies on the timely delivery of messages, this approach is also vulnerable to timing attacks, which are often easy to launch.

Cachin et. al [9] suggest to avoid these issues by implementing refresh using an asynchronous protocol. Such protocols are message-driven, i.e., proceed as soon as messages arrive. This allows a server to terminate a refresh and proceed with the next phase as soon as it has received enough information. Moreover, such protocols do *not* rely on upper bounds on message delays or clock shifts, i.e., they are as fast as the network. Timing attacks will only slow down such protocols, but not affect their security.

However, in a purely asynchronous network servers would not have access to a common clock for defining phases. Therefore, Cachin et al. [9] suggest to define phases *locally* to every server in terms of a single time signal, or *clock tick*, that occurs locally for a server and only indicates the start of a phase. The idea is to model systems where the time signals come from a local clock, say every day at 0:00 UTC, and where the local clocks are loosely synchronized, say they agree on which day and hour it is. Hence, the model is partially synchronous with long stretches of asynchrony. Such a setting implies an upper bound on the real time available to an adversary for corrupting servers in a certain phase, which justifies the assumption that an adversary corrupts only t servers in the same local phase [6].

The formal model of [9] does not further constrain the synchronization of phases, i.e., it leaves the scheduling of phases up to the adversary. This is to ensure that the security of a protocol does not rely on any synchrony assumptions, and hence, is not affected by timing attacks.

Network and Adversary. We adopt the basic system model from [9], which is parameterized by a security parameter k; a function $\epsilon(k)$ is called *negligible* if for all $c > 0$ there exists a k_0 such that $\epsilon(k) < \frac{1}{k^c}$ for all $k > k_0$. The network consists of n servers $P_1, \ldots, P_n$ and an adversary which are all probabilistic interactive Turing machines (PITM) [10] that run in polynomial time in k. The random tape of a server is initialized at the beginning of the computation, and we assume that the servers can *erase* information. There is also an initialization algorithm run by a trusted dealer before the system starts. On input k, n, t, and further parameters, it generates the state information used to initialize the servers.

Every server operates in a sequence of $m(k)$ *local phases*, where $m(k)$ is a polynomial. The phases are defined with respect to dedicated *input actions* of the form $(\mathsf{in}, \mathsf{clock\_tick})$, scheduled by the adversary. The local phase of a server is defined as the number of such input actions it has received.

The servers are connected by a *proactive secure asynchronous network* that allows every pair of servers to communicate authentically and privately whenever they are in the same local phase. The scheduling of the communication is determined by the adversary. Formally, we model such a network as follows. There exists a global set of messages $\mathcal{M}$, whose elements are identified by a *label* (s, r, l, τ) denoting the sender s, the receiver r, the length l of the message, and the phase τ when the message has been sent. The adversary sees the labels of all messages in $\mathcal{M}$, but not their contents. All communication is driven by the adversary, and proceeds in steps as follows. Initially, $\mathcal{M}$ is empty. At each step, the adversary performs some computation, chooses a server P_i,

and selects some message $m \in \mathcal{M}$ with label (s, i, l, τ), where P_i must be currently in local phase τ. The message m is then removed from $\mathcal{M}$, and P_i is *activated* with m on its communication input tape. When activated, P_i reads the contents of its communication input tape, performs some computation, and generates one or more response messages, which it writes to its communication output tape. Then, the response messages are added to $\mathcal{M}$, and control returns to the adversary. This step is repeated arbitrarily often until the adversary halts. We view this sequence of steps as logical time, and sometimes use the phrase "at a certain point in time" to refer to such a step. Such proactive secure asynchronous networks can be implemented based on a secure co-processor [18], or on the assumption that the network itself is authentic during short periods of time, allowing the exchange of fresh communication keys [19].

We assume an *adaptive mobile fail-stop* adversary. The adversary may corrupt a server P_i at any point in time by activating it on a special input action. After such an event, she may read the entire internal state of P_i, which includes its random tape but not previously erased information. Furthermore, she may observe all messages being received, until she *leaves* the server. During such a period of time, we call a server *corrupted*; at every other point in time, a server is called *honest*. The adversary may also cause a corrupted server to stop executing a protocol. We call an adversary *t-limited* if for any phase τ, she corrupts at most t servers that are in a local phase τ or $\tau + 1$.

Protocol Execution and Notation. In our model, protocols are invoked by the adversary. Every protocol *instance* is identified by a unique string ID, which is chosen by the adversary when it invokes the instance. For a protocol instance ID, we model the specific input and output *actions* of a server in terms of messages of the form $(ID, \mathsf{in}, \dots)$ and $(ID, \mathsf{out}, \dots)$ that a server may receive and produce, respectively. Messages that servers send to each other over the network on behalf of an instance ID have the form $(ID, \mathsf{type}, \dots)$, where type is defined by the protocol. We call a message *associated* with a protocol instance ID if it is of the form $(ID, \dots)$.

We describe a protocol in terms of *transition rules* that are executed in parallel. Such a transition rule consists of a condition on received messages and other state variables, and of a sequence of statements to be executed in case the condition is satisfied. We define *parallel* execution of transition rules as follows. When a server is activated and the condition of one or more transition rule is satisfied, one such rule is chosen arbitrarily and the corresponding statements are executed. This is repeated until no more conditions of transition rules are satisfied. Then, the activation of the server is terminated.

A protocol instance may also invoke another protocol instance by sending it a suitable input action and obtain its output via an output action. We assume that there is an appropriate server-internal mechanism which creates the instance for the sub-protocol, delivers the input message, and passes the produced output message to the calling protocol. Furthermore, we assume that upon termination of a protocol instance, all internal variables associated with this instance are erased.

Efficiency Measures and Termination. We define the *message complexity* of a protocol instance as the number of all associated messages produced by honest servers. It is a family of random variables that depend on the adversary and on k. Similarly, the *communication complexity* of a protocol instance is defined as the bit length of all

associated messages, and is also a family of such random variables. To define the *latency* (round complexity) of a protocol, we follow the approach of [20], where informally speaking the latency of an execution is the absolute duration of the execution divided by a longest message delay in this execution, where both times are as measured by an imaginary external clock. The latency of a protocol is a latency of a worst-case execution. (For details see the full version of [20], page 6.)

These quantities define a *protocol statistic* X, i.e., a family of real-valued, non-negative random variables $\{X_A(k)\}$, parameterized by the adversary A and the security parameter k, where each $X_A(k)$ is a random variable induced by running the system with A. We call a protocol statistic *uniformly bounded* if there exists a fixed polynomial $p(k)$ such that for all adversaries A, the probability $\Pr[X_A(k) > p(k)]$ is negligible.

As usual in asynchronous networks, we require *liveness* of a protocol, i.e., that "something good" eventually happens, only to the extent that the adversary delivers in every phase all associated messages among the servers that remain honest during this phase. As in [21], we define termination as the combination of liveness and an efficiency condition, which requires a protocol to have *uniformly bounded* message complexity, i.e., the number of messages produced by the protocol is independent of the adversary.

Cryptographic Assumptions. Our constructions are based on the assumption that there exists pseudo-random functions [13] defined as follows (sketch): Let $\mathcal{F}_k$ denote the set of functions from $\{0,1\}^k \to \{0,1\}^k$, and let $e \in_R Dom$ denote the process of choosing an element e uniformly at random from domain Dom. Finally, let D^f denote the execution of an algorithm D when given oracle access to f, where f is a random variable over $\mathcal{F}_k$. We say that D with oracle access distinguishes between two random variables ψ and g over $\mathcal{F}_k$ with gap $s(k)$, if $|\Pr[D^{\psi}(1^k) = 1] - \Pr[D^{g}(1^k) = 1]| = s(k)$. We say a random variable ψ over $\mathcal{F}_k$ is $s(k)$-pseudorandom, if no polynomial time in k algorithm D with oracle access distinguishes ψ from $g \in_R \mathcal{F}_k$ with gap $s(k)$.

A function family $\Psi_k = \{\psi_l\}_{l \in \{0,1\}^k}$ (with $\psi_l \in \mathcal{F}_k$) is called $s(k)$-*pseudorandom*, if the random variable ψ_l for $l \in_R \{0,1\}^k$ is $s(k)$-pseudorandom. If $s(k)$ is negligible, the collection $\{\Psi_k\}_{k \in \mathbb{N}}$ is called pseudorandom. We consider pseudorandom collections which are efficiently constructible, i.e., there exists a polynomial time algorithm that on input $l, x \in \{0,1\}^k$ outputs $\psi_l(x)$.

Pseudorandom function families can be constructed from any pseudorandom generator [13], which in turn could be constructed from any one-way function [22]. Alternatively, one could trust and use much simpler constructions based on AES or other widely available cryptographic functions.

In our protocols we make use also of *distributed* pseudorandom functions (DPRF), as introduced by Naor *et al.* [23]. In a DPRF the ability to evaluate the function is distributed among the servers, such that any authorized subset of the servers can evaluate the function, while no unauthorized subset gets any information about the function. For example, in a *threshold* DPRF the authorization to the evaluation of the functions is determined by the cardinality of the subset of the servers. In the sequel, we denote by $\Phi_k = \{\varphi_l\}_{l \in \{0,1\}^k}$ a family of efficiently constructible distributed pseudorandom functions. Moreover, we assume that if Φ_k denotes a DPRF *with threshold* κ, and if every server holds a polynomial κ-out-n share r_i of a seed r (where all r_i's are from the same domain as r) then $\varphi_r(x)$ can be efficiently computed from any set of κ values

$\varphi_{r_i}(x)$ for any position $x \in \{0,1\}^k$. Threshold DPRFs with this property are also called *non-interactive*. Nielsen [24] showed how to construct efficiently such non-interactive threshold DPRFs based on the decisional Diffie-Hellman assumption [25].

3 Technical Roadmap

Hybrid Secret Sharing. A basic tool we need is a *κ-out-n hybrid secret sharing* scheme: it allows a dealer to share a secret value among all other servers, such that every server receives an *additive* n-out-n share of the secret, as well as a κ-out-n backup share of every other server's additive share ($t+1 \leq \kappa \leq n$). Moreover, it guarantees to terminate for any server if the dealer is honest; otherwise, it either terminates for none or for all honest servers. Details of our scheme are given in Section 4.

Reconstructible Proactive Pseudorandomness (PPR). The key to our solutions for proactive secret sharing and for joint random secret sharing is a reconstructible PPR scheme. Such a scheme provides at every phase τ to every server P_i a secret value $pr_{\tau,i}$ which looks completely random to the adversary. Furthermore, any set of $n-t$ servers must be able to reconstruct the value $pr_{\tau,j}$ of any other server P_j without affecting the secrecy of the random value $pr_{\tau',j}$ computed by this server in another phase $\tau' \neq \tau$.

Our implementation assumes a trusted dealer that provides in the first phase every server P_i with a random key r_i, and with a $(n-t)$-out-n backup share r_{ji} of every other server's key r_j. The idea is to compute $pr_{\tau,i}$ as $\varphi_{r_i}(\mathtt{c})$, where $\{\varphi_l\}$ is a DPRF with threshold $(n-t)$, and $\mathtt{c}$ is some constant (pseudorandomness and constructibility of $pr_{\tau,i}$ then follows by the properties of DPRFs). This approach requires the servers to refresh in every phase their keys r_i (and shares $r_{1i}, \ldots, r_{ni}$) such that the fresh keys of honest servers are unknown to the adversary. This can be done as follows.

In a first step, P_i shares the pseudorandom value $\psi_{r_i}(\mathtt{a})$ (where $\mathtt{a}$ denotes some public constant) among all other servers using a $(n-t)$-out-n hybrid secret sharing scheme, where it derives *all* random choices using its current key r_i as a seed to a pseudorandom function. It then computes its new key r'_i as the sum of the additive shares provided by *all* these hybrid secret sharing schemes (the new shares $r'_{1i}, \ldots, r'_{ni}$ are computed as the sum of all provided backup shares). To do this, P_i waits until $n-t$ servers have completed their sharing scheme as a dealer; for every other server P_j, it reveals the share r_{ji}. It can now simply wait until either P_j's sharing scheme terminates, or until it receives enough shares r_{jl} from other servers P_l to reconstruct r_j and derive the missing shares thereof; since a sharing scheme terminates either for none or for all servers, one of the two cases eventually happens.

Notice that the servers need *not* agree on whether to derive the missing shares from the sharing schemes, or from the reconstructed key r_j, as both ways provide the *same* values. Our protocol ensures that there is at least one honest server whose sharing scheme is *not* reconstructed. This ensures secrecy of the new keys r'_i.

Proactive Secret Sharing (PSS). Suppose that at the beginning of the computation, a trusted dealer shares a secret s among the servers. To prevent a mobile adversary from learning s, the servers have to compute *fresh* shares of s whenever they enter a new phase. This can be done using a proactive secret sharing scheme.

Our implementation for PSS relies on an underlying PPR scheme (initialized by the dealer). Furthermore, it assumes that the trusted dealer initially provides every server with an *additive* share of the secret s, and with a $(t+1)$-out-n *backup* share of every other server's additive share.

In an epoch τ, the servers refresh their shares of the secret by first re-sharing their additive share of s using a $(t+1)$-out-n hybrid sharing scheme; in this step, every server P_i derives *all* its random choices by using the current random value $pr_{\tau,i}$ (provided by the PPR scheme) as a seed to a pseudorandom function.

As in the PPR scheme, every server then computes its fresh additive share of s as the sum of the additive shares provided by *all* re-sharing protocols (the backup shares are computed analogously). It therefore waits for $n-t$ re-sharing schemes to terminate, and reconstructs the remaining schemes in public. This can be done by reconstructing for every corresponding dealer P_j the random value $pr_{\tau,j}$ as well as P_j's current additive share of the secret. Reconstructing $pr_{\tau,j}$ can be done using the reconstruction mechanism of the PPR scheme, whereas P_j's additive share can be reconstructed by revealing the corresponding backup shares.

Joint Random Secret Sharing (JRSS). The goal of a JRSS protocol is to provide every server with a $(t+1)$-out-n share of a random value e unknown by the adversary. It can be executed repeatedly during the phases. Our implementation works exactly as the above protocol for refreshing a sharing, except for the following differences. In an instance with tag ID of protocol JRSS, a server P_i derives its random choices from the (pseudo)random value $\varphi_{r_i}(ID)$ (as opposed to $pr_{\tau,i} = \varphi_{r_i}(\mathsf{c})$), where r_i and $\{\varphi_l\}$ is the current key of P_i and the DPRF, respectively, used by the underlying PPR scheme. It then shares a *random* value e_i and proceeds as above. If the sharing scheme of a server P_j needs to be reconstructed, the servers reconstruct only the corresponding randomness $\varphi_{r_j}(ID)$. Adding up all backup shares provided by the sharing schemes yields the desired $(t+1)$-out-n share of the random value $e = e_1 + \cdots + e_n$.

Building Proactive Cryptosystems. Our protocols for PSS and for JRSS allow to build proactive versions of a large class of discrete logarithm-based cryptosystems without the use of expensive agreement sub-protocols. The idea is to share the key of the cryptosystem using our PSS protocol, and to accomplish the cryptographic operation using a distributed protocol. Such a protocol can be derived by combining our JRSS protocol with known techniques from threshold cryptography. We illustrate this idea in Section 8, considering Schnorr's signature scheme [17] as example.

4 Hybrid Secret Sharing

In this section, we describe the syntax and security properties of our protocol for hybrid secret sharing, $\mathsf{HybridShare}_\kappa$, which will serve as a basic tool in our subsequent constructions. A description and analysis of the protocol is given in [1].

Intuitively, our hybrid secret sharing protocol allows a dealer to share a secret s among n servers in such a way that every server P_i computes an *additive* share s_i of the secret, and a κ-out-n *backup* share s_{ji} of every other server's additive share, where $t+1 \leq \kappa \leq n$ (the idea of backing up additive shares is inspired by [26]).

Our specification treats the randomness r used by the dealer as an explicit parameter, and requires that the share of every server is a deterministic function of s and r. This constructibility of the shares will be essential for our purposes.

Formally, our sharing protocol $\mathsf{HybridShare}_\kappa$ has the following syntax. Let $\mathbb{F}_q$ be an arbitrary finite field, denoting the domain of secrets. There is a distinguished server P_d called the *dealer* which *activates* an instance $ID.d$ of $\mathsf{HybridShare}_\kappa$ upon receiving an input of the form $(ID.d, \mathsf{in}, \mathsf{share}, s, r)$, where $s \in \mathbb{F}_q$ and $r \in \{0,1\}^k$; if this happens, we also say the dealer *shares* s *over* $\mathbb{F}_q$ *using randomness* r *through* $ID.d$. Every other server activates $ID.d$ upon receiving a message $(ID.d, \mathsf{in}, \mathsf{share})$. A server *terminates* $ID.d$ when it produces an output of the form $(ID.d, \mathsf{out}, \mathsf{shared}, s_i, s_{1i}, \ldots, s_{ni})$, where $s_i, s_{1i}, \ldots, s_{ni} \in \mathbb{F}_q$.

Our protocol $\mathsf{HybridShare}_\kappa$ has message complexity of $\mathcal{O}(n^2)$, communication complexity of $\mathcal{O}(kn^3)$ bits, and round complexity equal four. Furthermore, for any t-limited adversary where $t < \frac{n}{3}$, the following holds: Whenever a dealer shares a secret s over $\mathbb{F}_q$ using randomness r through an instance $ID.d$ of $\mathsf{HybridShare}_\kappa$, it holds that:

LIVENESS: If the dealer is honest throughout $ID.d$, then all honest servers terminate $ID.d$, provided all servers activate $ID.d$ in the same phase τ, and the adversary delivers all messages among servers honest during phase τ.

AGREEMENT: If one honest server terminates $ID.d$, then all honest servers terminate $ID.d$, provided all servers activate $ID.d$ in the same phase τ, and the adversary delivers all messages among servers honest during phase τ.

CORRECTNESS: The values s and r uniquely define n polynomials $f_j(x) \in \mathbb{F}_q[x]$ for $j \in [1,n]$ of degree κ, such that $s = \sum_{j=1}^{n} f_j(0)$, and the following holds: If a server P_i outputs $s_i, s_{1i}, \ldots, s_{ni}$, then $f_i(0) = s_i$ and $f_j(i) = s_{ji}$ for $j \in [1,n]$.

PRIVACY: If the dealer is honest throughout $ID.d$, and s and r are uniformly distributed in $\mathbb{F}_q$ and $\{0,1\}^k$, respectively, then the adversary cannot guess s with probability significantly better than $1/|\mathbb{F}_q|$.

EFFICIENCY: The message complexity of $ID.d$ is uniformly bounded.

5 Asynchronous Reconstructible Proactive Pseudorandomness

In this section we give a definition for an asynchronous reconstructible PPR scheme along the lines of [12], and describe our implementation. The security proof of the scheme is contained in the full version of the paper.

5.1 Definition

Let $l(k)$ be a fixed polynomial. An *asynchronous reconstructible proactive pseudorandomness* scheme consists of a probabilistic setup algorithm σ, a proactive pseudorandomness protocol π, and a reconstruction protocol ρ. An instance of such a scheme has an associated tag ID and works as follows.

The setup algorithm σ produces the initial state information $state_{0,i}$ and the initial random value $pr_{0,i}$ of every server P_i. It is executed at the beginning of the computation by a trusted dealer. At the beginning of every phase $\tau \in [1, m(k)]$, the servers execute an instance $ID|\mathsf{ppr}.\tau$ of π to compute a fresh pseudorandom value for phase

τ. The input action for server P_i carries the state information $state_{\tau-1,i}$ of the previous phase, and has the form $(ID|\mathsf{ppr}.\tau, \mathsf{in}, state_{\tau-1,i})$. The output action comprises the pseudorandom value $pr_{\tau,i}$ and the updated state information $state_{\tau,i}$. It has the form $(ID|\mathsf{ppr}.\tau, \mathsf{out}, pr_{\tau,i}, state_{\tau,i})$. If P_i does not produce an output in phase τ (which could be the case if the server was corrupted and halted in the previous phase) then its input $state_{\tau,i}$ to the subsequent instance of π is the *empty input* $\perp$.

In every phase $\tau \in [1, m(k)]$, the servers may execute an instance $ID|\mathsf{rec}_j.\tau$ of protocol ρ to reconstruct the current pseudorandom value of server P_j. The corresponding input and output actions have the form $(ID|\mathsf{rec}_j.\tau, \mathsf{in}, state_{\tau,i})$, and $(ID|\mathsf{rec}_j.\tau, \mathsf{out}, z_i)$, respectively, where $state_{\tau,i}$ denotes the current state information of P_i. We say a server *reconstructs a value z_i for P_j*, if it outputs a message $(ID|\mathsf{rec}_j.\tau, \mathsf{out}, z_i)$.

As in [12], we define the security requirements with respect to the following *on-line attack*: The scheme is run in the presence of a t-limited adversary for $m(k)$ phases. At every phase τ, the adversary may also instruct the servers to reconstruct the value $pr_{\tau,i}$ of *any* server. At a certain phase l (chosen adaptively by the adversary), the adversary chooses an honest server P_j whose value $pr_{\tau,j}$ is not reconstructed at that phase. She is then given a test value v, and the execution of the scheme is resumed for phases $l+1, \ldots, m(k)$. (Our definition will require that the adversary is unable to say whether v is P_j's output at phase l, or a random value.)

For an instance ID of a PPR scheme and an adversary A, let $A(ID, PR)$ denote the output of A after an on-line attack on ID, when v is indeed the output of P_j; similarly, let $A(ID, R)$ denote the corresponding output when v is a random value.

Definition 1. *Let σ, π, and ρ be given as above. We call (σ, π, ρ) a* t-resilient asynchronous reconstructible proactive pseudorandomness scheme *if for every instance ID, and every t-limited adversary A the following properties hold:*

LIVENESS: *Every server P_i honest throughout a phase $\tau \in [1, m(k)]$ terminates instance $ID|\mathsf{ppr}.\tau$ in phase τ, provided that in every phase $\tau' \in [1, \tau]$, the adversary activates each server honest throughout τ' on $ID|\mathsf{ppr}.\tau'$, and delivers all associated messages among servers honest during phase τ'. Furthermore, if every such server P_i subsequently activates $ID|\mathsf{rec}_j.\tau$ for some $j \in [1, n]$, it reconstructs some value z_i for P_j, provided the adversary delivers all associated messages among servers honest during phase τ.*

CORRECTNESS: *If a server P_j outputs $(ID, \mathsf{out}, pr_{\tau,j}, state_{\tau,j})$ in some phase $\tau \in [1, m(k)]$, and another server P_i reconstructs z_i for P_j in phase τ, then $z_i = pr_{\tau,j}$.*

PSEUDORANDOMNESS: $|\Pr[A(ID, PR) = 1] - \Pr[A(ID, R) = 1]|$ *is negligible.*

EFFICIENCY: *The message complexity of an instance of π is uniformly bounded.*

5.2 Implementation

Let $\Phi_k = \{\varphi_i\}_{i \in \{0,1\}^k}$ denote a DPRF with threshold $n - t$, and $\mathtt{a}, \mathtt{b}, \mathtt{c}$ denote distinct arbitrary constants in the domain of Φ_k. For convenience, we view elements from $\{0,1\}^k$ as elements from $\mathbb{F}_{2^k}$ (and conversely), according to some fixed bijective map from $\{0,1\}^k$ to $\mathbb{F}_{2^k}$. All computations are done over $\mathbb{F}_{2^k}$.

The Setup Algorithm σ_{ppr}. The setup algorithm provides to every server P_i a random value $r_i \in \mathbb{F}_{2^k}$, and a $(t+1)$-out-n share $r_{ji} \in \mathbb{F}_{2^k}$ of the random value

of every other server. It therefore chooses n random polynomials $f_i(x) \in \mathbb{F}_{2^k}[x]$ of degree t for $i \in [1,n]$. The initial state information of a server P_i is defined as $state_{0,i} \triangleq (f_i(0), f_1(i), \ldots, f_n(i))$. The initial pseudorandom value is computed as $pr_{0,i} \leftarrow \varphi_{f_i(0)}(\mathtt{c})$.

The Reconstruction Protocol ρ_{ppr}. Let $r_i, r_{1i}, \ldots, r_{ni}$ denote P_i's local input to an instance $ID|\mathsf{rec}_j.\tau$ of protocol ρ_{ppr}. Reconstructing the pseudorandom value $pr_{\tau,j}$ of server P_j is straightforward. Every server P_i computes $pr_{\tau,ji} \leftarrow \varphi_{r_{ji}}(\mathtt{c})$, and sends it to every other server. Using the reconstruction mechanism of Φ_k, every server can compute $pr_{\tau,j}$ upon receiving $n-t$ "shares" $pr_{\tau,jm}$ from other servers P_m.

The Asynchronous Proactive Pseudorandomness Protocol π_{ppr}. Let $r_i, r_{1i}, \ldots, r_{ni}$ denote server P_i's local input $state_{\tau-1,i}$ to instance $ID|\mathsf{ppr}.\tau$ of π_{ppr}. To refresh this sharing, and to compute fresh pseudorandom values $\{pr_{\tau,i}\}$, every server P_i executes the following transition rules in parallel.

SHARE: When P_i invokes the protocol with non-empty input, it shares the pseudorandom value $\varphi_{r_i}(\mathtt{a})$ over $\mathbb{F}_{2^k}$ using randomness $\varphi_{r_i}(\mathtt{b})$ through an instance $ID|\mathsf{ppr}.\tau|\mathsf{share}.i$ of protocol $\mathsf{HybridShare}_{n-t}$.

SHARE-TERMINATION: Whenever P_i terminates a sharing protocol $ID|\mathsf{ppr}.\tau|\mathsf{share}.j$, it stores the corresponding output in the local variables $\bar{d}_{ji}, \bar{d}_{j1i}, \ldots, \bar{d}_{jni}$. If the $(n-t)$'th such sharing protocol has terminated and P_i has received non-empty input before, it sends to all servers a reveal message containing the values $\varphi_{r_{mi}}(\mathtt{a})$ and $\varphi_{r_{mi}}(\mathtt{b})$ for servers P_m whose sharing protocol *did not* terminate yet.

RECONSTRUCT: Whenever P_i receives $n-t$ values $\varphi_{r_{mi}}(\mathtt{a}), \varphi_{r_{mi}}(\mathtt{b})$ for a server P_m, it reconstructs $\varphi_{r_m}(\mathtt{a})$ and $\varphi_{r_m}(\mathtt{b})$ using the reconstruction mechanism of Φ_k. It then computes the values $\bar{d}_{mi}, \bar{d}_{m1i}, \ldots, \bar{d}_{mni}$ as the i'th share when sharing a secret $\varphi_{r_m}(\mathtt{a})$ using randomness $\varphi_{r_m}(\mathtt{b})$ according to protocol $\mathsf{HybridShare}_{n-t}$.

COMBINE: When P_i has computed values $\bar{d}_{ji}, \bar{d}_{j1i}, \ldots, \bar{d}_{jni}$ for every $j \in [1,n]$, it computes its local output values $pr_{\tau,i}$ and $state_{\tau,i} \triangleq (r'_i, r'_{1i}, \ldots, r'_{ni})$ as $r'_i \leftarrow \sum_{j=1}^{n} \bar{d}_{ji}$, $r'_{mi} \leftarrow \sum_{j=1}^{n} \bar{d}_{jmi}$ for $m \in [1,n]$, and $pr_{\tau,i} \leftarrow \varphi_{r'_i}(\mathtt{c})$.

The scheme guarantees pseudorandomness because the pseudorandom values $\varphi_{r_h}(\mathtt{a})$ and $\varphi_{r_h}(\mathtt{b})$ of at least one honest server remain hidden from the adversary. This is guaranteed because all honest servers together reveal at most $(n-t)t$ "shares" $\varphi_{r_{ij}}(\mathtt{a})$ and $\varphi_{r_{ij}}(\mathtt{b})$. But to reconstruct $\varphi_{r_i}(\mathtt{a})$ and $\varphi_{r_i}(\mathtt{b})$ of *all* $(n-t)$ honest servers, the adversary needs at least $(n-t)(n-2t) \geq (n-t)(t+1)$ such shares, as the threshold of Ψ_k is $(n-t)$.

The reason why the scheme avoids an agreement (while preserving constructibility) is the following: if an honest server P_i terminates the protocol $ID|\mathsf{ppr}.\tau|\mathsf{share}.j$ and computes the tuple $(\bar{d}_{ji}, \bar{d}_{j1i}, \ldots, \bar{d}_{jni})$, then this is *the same tuple* it would compute by first reconstructing the randomness r_j of P_j from backup shares, and then reproducing the computations of P_j in the sharing protocol $ID|\mathsf{ppr}.\tau|\mathsf{share}.j$. Hence, the servers *do not have to agree* whether to compute their share of P_j's sharing protocol by the SHARE-TERMINATION or the RECONSTRUCT transition rule, respectively, as both rules provide the *same* share. We prove the following theorem in [1].

Theorem 1. *$(\sigma_{\mathsf{ppr}}, \pi_{\mathsf{ppr}}, \rho_{\mathsf{ppr}})$ is a t-resilient asynchronous reconstructible proactive pseudorandomness scheme for $t < n/3$. It has a latency of five rounds, uses $\mathcal{O}(n^3)$ messages, and has a communication complexity of $\mathcal{O}(kn^4)$ bits.*

6 Refreshing a Sharing

In this section we define an asynchronous PSS scheme along the lines of [9], and sketch our implementation. The security proof of the scheme can be found in [1].

6.1 Definition

Let K denote the domain of possible secrets, S denote the domain of possible shares, and $l(k)$ a fixed polynomial. An *asynchronous proactive secret sharing* scheme consists of a setup algorithm σ, a proactive refresh protocol π, and a reconstruction protocol ρ. An instance of a PSS has a tag ID and works as follows.

The setup algorithm produces for each server P_i the initial state information $state_{0,i}$ and the initial share $s_{0,i} \in S$ of the secret. It is executed at the beginning of the computation by the trusted dealer. At the beginning of every phase $\tau \in [1, m(k)]$ the servers execute an instance $ID|\mathsf{ref}.\tau$ of protocol π to refresh the old share $s_{\tau-1,i}$, and to update the state $state_{\tau-1,i}$. The corresponding input and output actions of server P_i have the form $(ID|\mathsf{ref}.\tau, \mathsf{in}, s_{\tau-1,i}, state_{\tau-1,i})$ and $(ID|\mathsf{ref}.\tau, \mathsf{out}, s_{\tau,i}, state_{\tau,i})$, respectively, where $s_{\tau-1,i}$ and $state_{\tau-1,i}$ equal $\perp$ in case P_i did not produce an output in phase $\tau - 1$.

In every phase $\tau \in [1, m(k)]$, the servers may execute an instance $ID|\mathsf{rec}.\tau$ of protocol ρ to reconstruct the secret. The input and output actions for server P_i have the form $(ID|\mathsf{rec}.\tau, \mathsf{in}, s_{\tau,i})$, and $(ID|\mathsf{rec}.\tau, \mathsf{out}, z_i)$, respectively, where $s_{\tau,i}$ denotes the current share as computed by the instance $ID|\mathsf{ref}.\tau$. We say that a server *reconstructs* a value z_i, when it outputs a message $(ID|\mathsf{rec}.\tau, \mathsf{out}, z_i)$.

Definition 2. *Let σ, π, and ρ be given as above. We call (σ, π, ρ) a* t-resilient asynchronous proactive secret sharing scheme*, if for every instance ID, and every t-limited adversary the following properties hold:*

LIVENESS: *Every server P_i honest throughout a phase $\tau \in [1, m(k)]$ terminates instance $ID|\mathsf{ref}.\tau$ in phase τ, provided that in every phase $\tau' \in [1, \tau]$, the adversary activates every server honest throughout phase τ' on $ID|\mathsf{ref}.\tau'$, and delivers all associated messages among servers honest during phase τ'. Further, if every such P_i subsequently activates $ID|\mathsf{rec}.\tau$, it reconstructs some value z_i, provided the adversary delivers all associated messages among servers honest during phase τ.*

CORRECTNESS: *After initialization, there exists a fixed value $s \in K$. Moreover, if an honest server reconstructs a value z_i, then $z_i = s$.*

PRIVACY: *As long as no honest server activates an instance of ρ, the adversary cannot guess s with probability significantly better than $1/|K|$.*

EFFICIENCY: *The message complexity of π and ρ is uniformly bounded.*

We stress that the security of the sharing does not depend on the timely delivery of messages. Even if the adversary fails to deliver the messages within prescribed phase, the privacy of the shared secret is not compromised.

6.2 Implementation

Our implementation of the PSS scheme is a suitable example to illustrate how the PPR scheme introduced in the previous section can be used to avoid the need for agreement even if it seems to be inherently necessary. We therefore briefly recall the standard solution [9] for PSS that depends on agreement. Here, every server initially receives a $(t+1)$-out-n share of the secret. To refresh the shares, every server provides every other server with a $(t+1)$-out-n sub-share of its own share, using a suitable sharing scheme. The servers then agree on a set of at least $t+1$ servers whose re-sharing scheme terminates for all honest servers, and compute the new share as the linear combination of the received sub-shares (with Lagrange coefficients).

We follow the same approach (see Section 3), but avoid agreement by reconstructing the re-sharing schemes of the slowest (possibly crashed) servers in public. However, this approach only works if the publicly reconstructed sub-shares are *identical* to the ones which the re-sharing scheme would produce. Otherwise, the servers would again have to agree on which sub-shares to reconstruct, and which to take from the re-sharing schemes. This is where the PPR scheme comes in handy, as it allows to reconstruct the random choices made by a server when it is re-sharing its share. The technical details are given below. Let the domain of possible secrets be a field $\mathbb{F}_q$ where $q \leq 2^k$. All computations are in $\mathbb{F}_q$ or $\mathbb{F}_{2^k}$, as is clear from the context.

The Setup Algorithm σ_{pss}. The setup algorithm provides every server with an additive share s_i of a randomly chosen secret, and with a $(t+1)$-out-n share s_{ji} of every other server's additive share. It therefore chooses n random polynomials $f_i(x) \in \mathbb{Z}_q[x]$ of degree t for $i \in [1,n]$ (the secret is defined as $s = \sum_{i=1}^{n} s_i$). The initial share of server P_i is defined as $s_{0,i} \triangleq (s_i, s_{1i}, \ldots, s_{ni})$, where $s_i = f_i(0)$ and $s_{ji} = f_j(i)$.

Additionally, the setup algorithm provides every server with the initial state information needed to initialize a PPR scheme. It therefore runs the setup algorithm σ_{ppr}, and computes the initial state information $state_{0,i}$ of server P_i as the tuple $(state^{\mathsf{ppr}}_{0,i}, pr_{0,i})$

The Reconstruction Protocol ρ_{pss}. The reconstruction protocol is straight forward. Every server P_i sends its input $s_{\tau,i} \triangleq (s_i, s_{1i}, \ldots, s_{ni})$ to every other server. Upon receiving $t+1$ such values the server interpolates all missing shares s_j from the received sub-shares s_{ji} by Lagrange interpolation, and computes the secret as $s = \sum_{j=1}^{n} s_j$.

The Refresh Protocol π_{pss}. Let $(s_i, s_{1i}, \ldots, s_{ni})$ and $(state^{\mathsf{ppr}}_{\tau-1,i}, pr_{\tau-1,i})$ denote server P_i's local input $s_{\tau-1,i}$ and $state_{\tau-1,i}$, respectively, to instance $ID|\mathsf{ref}.\tau$. To compute a fresh share $(s'_i, s'_{1i}, \ldots, s'_{ni})$ and updated state information $(state^{\mathsf{ppr}}_{\tau,i}, pr_{\tau,i})$, every server P_i executes the following transition rules in parallel:

SHARE: When P_i invokes the protocol, it activates an instance $ID|\mathsf{ppr}.\tau$ of protocol π_{ppr} with input $state^{\mathsf{ppr}}_{\tau-1,i}$ to compute $(state^{\mathsf{ppr}}_{\tau,i}, pr_{\tau,i})$. Furthermore, if P_i received non-empty input, it shares its share s_i over $\mathbb{F}_q$ using randomness $pr_{\tau-1,i}$ through an instance $ID|\mathsf{ref}.\tau|\mathsf{share}.i$ of protocol $\mathsf{HybridShare}_{t+1}$.

SHARE-TERMINATION: Whenever P_i terminates an instance $ID|\mathsf{ref}.\tau|\mathsf{share}.j$ of a sharing protocol , it stores the corresponding output in the local variables $\bar{e}_{ji}$, $\bar{e}_{j1i}, \ldots, \bar{e}_{jni}$. If for $n-t$ servers P_j the corresponding protocols $ID|\mathsf{ref}.\tau|\mathsf{share}.j$

have terminated, it sends the indices of all servers whose sharing protocol *did not* terminate yet to every other server in a missing message.

REVEAL: If for some index m, P_i receives $(n-t)$ missing messages from other servers containing this index and has received non-empty input before, it sends a reveal message to every other server containing the backup share s_{mi} and the index m. Next, it activates the instance $ID|\mathsf{rec}_m.\tau$ of protocol ρ_{ppr} with input $state^{\mathsf{ppr}}_{\tau-1,i}$ to reconstruct the randomness $pr_{\tau-1,m}$ of P_m.

RECONSTRUCT: Whenever P_i receives $(t+1)$ reveal messages for the same index m *and* reconstructs the value $pr_{\tau-1,m}$ for P_m, it computes the share s_m from the received backup shares by Lagrange interpolation. It then computes the tuple $(\bar{e}_{mi}, \bar{e}_{m1i}, \ldots, \bar{e}_{mni})$ as the i'th share when sharing s_m using randomness $pr_{\tau-1,m}$.

COMBINE: When P_i has computed values $(\bar{e}_{mi}, \bar{e}_{m1i}, \ldots, \bar{e}_{mni})$ for all $m \in [1,n]$, it computes the new share $(s'_i, s'_{1i}, \ldots, s'_{ni})$ as follows: $s'_i \leftarrow \sum_{j=1}^{n} \bar{e}_{ji}$, $s'_{mi} \leftarrow \sum_{j=1}^{n} \bar{e}_{jmi}$ for $m \in [1,n]$.

Notice that the protocol has the same message flow as the pseudorandomness protocol π_{ppr}, except for the additional missing messages. They ensure the secrecy of the share s_h of at least one honest server P_h, and are needed because the servers hold a $(t+1)$-out-n hybrid sharing of the secret s. We remark that for refreshing a $(n-t)$-out-n hybrid sharing, the servers could omit waiting for $t+1$ such messages, and could execute the REVEAL rule directly at the end of the SHARE-TERMINATION rule. This would save one communication round. The proof of the following theorem can be found in [1].

Theorem 2. *The tuple $(\sigma_{\mathsf{pss}}, \pi_{\mathsf{pss}}, \rho_{\mathsf{pss}})$ is a t-resilient asynchronous proactive secret sharing scheme for $t < n/3$. The refresh protocol π_{pss}uses $\mathcal{O}(n^3)$ messages, has latency of six rounds and communication complexity of $\mathcal{O}(kn^4)$.*

7 Asynchronous Proactive Joint Random Secret Sharing

The goal of an asynchronous proactive joint random secret sharing scheme is to enable the servers to repeatedly generate $(t+1)$-out-n sharings of random values, such that the random values remain hidden from the adversary. Due to lack of space, we only sketch the definition and implementation.

Definition. An asynchronous proactive joint random secret sharing (JRSS) scheme consists of a setup algorithm σ, a proactive update protocol π, a joint random secret sharing protocol γ, and a reconstruction protocol ρ. An instance of such a scheme has a tag ID and works as follows.

At the beginning of the computation, a trusted dealer executes the setup algorithm σ and provides every server with its initial state information $state_{0,i}$. At the beginning of every phase $\tau \in [1, m(k)]$, the servers execute protocol π to update the state information $\{state_{\tau-1,i}\}$. During every phase $\tau \in [1, m(k)]$, the servers can repeatedly execute protocol γ to generate a sharing of a random value z_c in a domain K. Every such instance has a unique tag $ID|\mathsf{gen}_c$. For every server P_i, it takes the current state information $state_{\tau,i}$ as input, and produces as output a share $s_{c,i}$ of the random value z_c. These

shares may serve as input to the reconstruction protocol ρ with tag $ID|\mathsf{rec}_c$, which produces for every server P_i a value $z_{c,i}$ as output.

For a JRSS scheme to be secure, we require that when the first server completes an instance $ID|\mathsf{gen}_c$, there is a fixed value z_c such that the following holds: (Correctness) If a server P_i terminates $ID|\mathsf{rec}_c$ and outputs $z_{c,i}$, then $z_{c,i} = z_c$. Furthermore, (Privacy) as long as no honest server activates $ID|\mathsf{rec}_c$, the adversary cannot guess z_c with probability significantly better than $1/|K|$.

Implementation. Our implementation builds on our PPR scheme $(\sigma_{\mathsf{ppr}}, \pi_{\mathsf{ppr}}, \rho_{\mathsf{ppr}})$. Let $\Phi_k = \{\varphi_i\}$ denote the DPRF family used by the PPR scheme, $\mathtt{a}$ and $\mathtt{b}$ denote two distinct constants, and $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ denote a collision resistant hash function (it is well-known how to construct such functions from standard computational assumptions such as the hardness of the discrete-logarithm problem).

The state information $\{state_{\tau,i}\}$ of our JRSS scheme comprises only the state information of our PPR scheme, i.e., $state_{\tau,i} \triangleq (r_i, r_{1i}, \ldots, r_{ni})$. Protocols σ_{jrss} and π_{jrss} for setting up and refreshing this state, respectively, consist only of calling the protocols σ_{ppr} and π_{ppr}. The protocol γ_{jrss} for generating sharings of random values in $\{0,1\}^k$ works as follows. Given input $state_{\tau,i} \triangleq (r_i, r_{1i}, \ldots, r_{ni})$ to an instance $ID|\mathsf{gen}_c$ of γ_{jrss}, every server P_i performs the following steps (all computations are done in $\mathbb{F}_{2^k}$).

SHARE: When P_i invokes the protocol with non-empty input, it shares $\varphi_{r_i}(\mathcal{H}(ID|\mathsf{gen}_c|\mathtt{a}))$ over $\mathbb{F}_{2^k}$ through an instance of protocol $\mathsf{HybridShare}_{t+1}$ with tag $ID|\mathsf{gen}_c|\mathsf{share}.i$ using randomness $\varphi_{r_i}(\mathcal{H}(ID|\mathsf{gen}_c|\mathtt{b}))$.

SHARE-TERMINATION: Whenever P_i terminates a sharing protocol $ID|\mathsf{gen}_c|\mathsf{share}.j$, it stores the corresponding output in local variables $\bar{e}_{ji}, \bar{e}_{j1i}, \ldots, \bar{e}_{jni}$. Once $n-t$ sharing protocols have terminated and P_i has received non-empty input before, it sends to all servers a reveal message containing values $\varphi_{r_{mi}}(\mathcal{H}(ID|\mathsf{gen}_c|\mathtt{a}))$ and $\varphi_{r_{mi}}(\mathcal{H}(ID|\mathsf{gen}_c|\mathtt{b}))$ for servers P_m whose sharing protocol *did not* terminate yet.

RECONSTRUCT: Upon receiving $n-t$ reveal messages for the same index m, P_i reconstructs values $\varphi_{r_m}(\mathcal{H}(ID|\mathsf{gen}_c|\mathtt{a}))$ and $\varphi_{r_m}(\mathcal{H}(ID|\mathsf{gen}_c|\mathtt{b}))$ (using the threshold evaluation property of Φ_k) and derives the missing sub-share $\bar{e}_{mi}, \bar{e}_{m1i}, \ldots, \bar{e}_{mni}$.

COMBINE: When P_i has computed values $(\bar{e}_{mi}, \bar{e}_{m1i}, \ldots, \bar{e}_{mni})$ for every $m \in [1,n]$, it computes $s_{c,i} \triangleq (s_i, s_{1i}, \ldots, s_{ni})$ as follows: $s_i \leftarrow \sum_{j=1}^{n} \bar{e}_{ji}$, $s_{mi} \leftarrow \sum_{j=1}^{n} \bar{e}_{jmi}$ for $m \in [1,n]$.

The shared secret value z_c is never reconstructed but equals $\sum_{i=1}^{n} s_i$. The protocol has a latency of five rounds, a message complexity of $O(n^3)$, and a communication complexity of $O(kn^4)$ bits.

An instance $ID|\mathsf{rec}_c$ of the reconstruction protocol ρ_{jrss} works as follows. Every server i sends its share $s_{c,i} \triangleq (s_i, s_{1i}, \ldots, s_{ni})$ — which it receives as input — to every other server. Upon receiving $t+1$ such values, P_i derives all values s_j from the received sub-shares s_{jm} by Lagrange interpolation and computes the secret as $z_c = \sum_{j=1}^{n} s_j$.

8 A Simple Proactive Secure Signature Scheme

Our protocols for PSS and JRSS can be used to proactivize a large class of discrete-logarithm based public-key cryptosystems for signing and encryption. In this section, we sketch how this can be done considering Schnorr's signature scheme as an example.

Let p denote a large prime, and $\langle g \rangle$ denote a multiplicative subgroup of $\mathbb{Z}_p^*$ of prime order q such that $q|p-1$. In the regular centralized Schnorr signature scheme, the secret key x of the signer is a random element from $\mathbb{Z}_q$, and the public key is $y = g^x$. To sign a message $m \in \{0,1\}^*$, the signer picks a random number $r \in \mathbb{Z}_q$, and computes the signature (ρ, σ) as $\rho \leftarrow g^r \mod p$ and $\sigma \leftarrow r + \mathcal{H}(m||\rho)x \mod q$. A signature (ρ, σ) on a message m can then be verified by checking that $g^\sigma = \rho y^{\mathcal{H}(m||\rho)} \mod p$.

In a proactive signature scheme, the power to sign a message is distributed among the servers such that in every epoch, only a set of at least $t+1$ servers can generate valid signatures, whereas any smaller set can neither compute a signature nor prevent the overall system from operating correctly. For a formal treatment of proactive signature schemes we refer to [12].

Proactivizing Schnorr's signature scheme in the above sense can be done as follows. First, a trusted dealer chooses the values p, q, g, x as in the standard Schnorr scheme, and initializes a PSS scheme with a sharing of x. It also initializes a JRSS scheme, and announces the public parameters p, q, g and y. To compute a signature (ρ, σ) on a message m, every server i performs the following steps:

generate $\rho = g^r$:

(1) Use the underlying JRSS scheme to compute a $(t+1)$-out-n share r_i of a random value $r \in \mathbb{Z}_q$.

(2) Reveal the value $\rho_i = g^{r_i} \mod p$ to all other servers.

(3) Upon receiving $t+1$ values ρ_j, compute ρ from the values ρ_j by using Lagrange interpolation in the exponent, i.e., $\rho \leftarrow \prod_{i \in Q} \rho_j^{\lambda_j} \mod p$. Here, Q denotes the indices of the received values ρ_i, and λ_i the Lagrange interpolation coefficient for the set Q and position 0.

generate $\sigma = r + \mathcal{H}(m||\rho)x$:

(1) Reveal the value $\sigma_i = r_i + \mathcal{H}(m||\rho)x_i \mod q$ to all other servers; here, x_i denotes server i's current share of x as computed by the underlying PSS scheme.

(2) Upon receiving $t+1$ values σ_j, compute σ by using Lagrange interpolation, i.e., $\sigma \leftarrow \sum_{j \in S} \lambda_j \sigma_j \mod q$. Here, S denotes the indices of the received values σ_j, and λ_j the Lagrange coefficients for the set S and position 0.

Verification of the computed signature can be done exactly as in the centralized Schnorr scheme. One can show that this proactive signature scheme is as secure as the centralized Schnorr scheme in the following sense: If there exists a t-limited mobile adversary against the proactive signature scheme that can forge a signature (under an adaptively chosen message attack), then there exists an adversary against the centralized Schnorr scheme that can forge signatures (under an adaptively chosen message attack).

Proactivizing other discrete-logarithm signature schemes such as ElGamal [15] or DSS [16] can be done in a similar way (to solve the inversion problem that occurs in DSS, one can use the approach of [27]).

9 Conclusions and Open Problems

In this paper, we have presented the first asynchronous schemes for proactive secret sharing and proactive joint random secret sharing with a bounded *worst case* complexity. Moreover, our solutions run three times faster (in terms of latency) than the best known previous solutions.

The technical novelty of our schemes is that they do not rely on an agreement subprotocol. The fact that agreement can be avoided is surprising on its own, as all known previous techniques for implementing such schemes require the servers to have at some point a common view of which servers have been crashed.

A natural open problem is to enhance our techniques to tolerate a Byzantine adversary. Here, the main difficulty lies in designing a *verifiable* version of our hybrid secret sharing scheme. In such a scheme, the dealer must be committed to a random value (of the same size as the secret), such that every server can verify that the dealer has indeed computed the shares by using this random value as a seed to a pseudorandom function. In principle, this can be done using the technique of general zero-knowledge proofs [10]. We suggest it as an open research problem to construct a pseudorandom function together with *efficient* zero-knowledge proofs for this task.

References

1. Przydatek, B., Strobl, R.: Asynchronous proactive cryptosystems without agreement. Technical Report RZ 3551, IBM Research (2004)
2. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Proc. CRYPTO '87. (1987)
3. Desmedt, Y.: Threshold cryptography. European Transactions on Telecommunications **5** (1994) 449–457
4. Shamir, A.: How to share a secret. Communications of the ACM **22** (1979) 612–613
5. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure key generation for discrete-log based cryptosystems. In: Proc. EUROCRYPT '99. (1999) 295–310
6. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks. In: Proc. 10th ACM Symposium on Principles of Distributed Computing (PODC). (1991) 51–59
7. Canetti, R., Gennaro, R., Herzberg, A., Naor, D.: Proactive security: Long-term protection against break-ins. RSA Laboratories' CryptoBytes **3** (1997)
8. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or how to cope with perpetual leakage. In: Proc. CRYPTO '95. (1995) 339–352
9. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Aysnchronous verifiable secret sharing and proactive cryptosystems. In: Proc. 9th ACM CCS. (2002)
10. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proc. 17th ACM STOC. (1985) 291–304
11. Strobl, R.: Distributed Cryptographic Protocols for Asynchronous Networks with Universal Composability. PhD thesis, ETH (2004)
12. Canetti, R., Herzberg, A.: Maintaining security in the presence of transient faults. In: Proc. CRYPTO '94. (1994) 425–438
13. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM **33** (1986) 792–807
14. Zhou, L.: Towards fault-tolerant and secure on-line services. PhD thesis, Cornell Univ. (2001)

15. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Info. Theory **IT 31** (1985)
16. National Institute for Standards, Technology: Digital signature standard (DSS). Technical Report 169 (1991)
17. Schnorr, C.P.: Efficient signature generation by smart cards. J. of Cryptology **4** (1991) 161–174
18. Backes, M., Cachin, C., Strobl, R.: Proactive secure message transmission in asynchronous networks. In: Proc. 21th ACM PODC. (2003)
19. Canetti, R., Halevi, S., Herzberg, A.: Maintaining authenticated communication in the presence of break-ins. J. of Cryptology **13** (2000) 61–106
20. Canetti, R., Rabin, T.: Fast asynchronous Byzantine agreement with optimal resilience. In: Proc. 25th ACM STOC. (1993) full version at www.research.ibm.com/security/cr-ba.ps.
21. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols (extended abstract). In: Proc. CRYPTO 01. (2001) 524–541
22. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing **28** (1999) 1364–1396
23. Naor, M., Pinkas, B., Reingold, O.: Distributed pseudo-random functions and KDCs. In: Proc. EUROCRYPT '99. (1999) 327–346
24. Nielsen, J.B.: A threshold pseudorandom function construction and its applications. In: Proc. CRYPTO '02. (2002) 401–416
25. Boneh, D.: The decision Diffie-Hellman problem. In: Third Algorithmic Number Theory Symposium. Volume 1423 of LNCS. (1998) 48–63
26. Rabin, T.: A simplified approach to threshold and proactive RSA. In: Proc. CRYPTO '98. (1998)
27. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Proc. CRYPTO '99. (1999)

Lattice-Based Threshold-Changeability for Standard Shamir Secret-Sharing Schemes

Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk

Dept. of Computing, Macquarie University, North Ryde, Australia
{rons, hwang, josef}@ics.mq.edu.au
http://www.ics.mq.edu.au/acac/

Abstract. We consider the problem of increasing the threshold parameter of a secret-sharing scheme after the setup (share distribution) phase, without further communication between the dealer and the shareholders. Previous solutions to this problem require one to start off with a non-standard scheme designed specifically for this purpose, or to have communication between shareholders. In contrast, we show how to increase the threshold parameter of the *standard* Shamir secret-sharing scheme without communication between the shareholders. Our technique can thus be applied to existing Shamir schemes even if they were set up without consideration to future threshold increases.

Our method is a new positive cryptographic application for lattice reduction algorithms, inspired by recent work on lattice-based list decoding of Reed-Solomon codes with noise bounded in the Lee norm. We use fundamental results from the theory of lattices (Geometry of Numbers) to prove quantitative statements about the information-theoretic security of our construction. These lattice-based security proof techniques may be of independent interest.

Keywords: Shamir secret-sharing, changeable threshold, lattice reduction, geometry of numbers.

1 Introduction

Background. A (t, n)-threshold secret-sharing scheme is a fundamental cryptographic scheme, which allows a *dealer* owning a secret to distribute this secret among a group of n *shareholders* in such a way that any t shareholders can reconstruct the secret, but no subset of less than t shareholders can gain information on the secret. Classical constructions for (t, n) secret-sharing schemes include the polynomial-based Shamir scheme [18] and the integer-based Chinese Remainder Theorem (CRT) scheme [2].

A common application for (t, n) secret-sharing schemes is for achieving *robustness* of distributed security systems. A distributed system is called robust if system security is maintained even against an attacker who manages to break into/eavesdrop up to a certain number of components of the distributed system. For example, access control to a system can be enforced using a secret shared

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 170–186, 2004.

among n system servers using a (t, n)-threshold secret-sharing scheme, while maintaining security if less than t servers are compromised. In such applications, the threshold parameter t must be determined by a security policy, based on an assessment which is a compromise between the value of the protected system and attacker capabilities on the one hand (which require as high a threshold as possible) and user convenience and cost on the other hand (which require as low a threshold as possible). In many settings, the system value and attacker capabilities are likely to change over time, thus requiring the security policy and hence threshold parameter t to *vary over time*. In particular, an increase in system value or attacker capabilities after the initial setup with a relatively low threshold parameter t, will require an increase in the threshold parameter to a higher value $t' > t$. The longer the lifetime of the system, the more likely that such a change will be needed. Note that we assume that shareholders will cooperate honestly in making the transition to the larger threshold $t' > t$, since the attacker in our setting is an *outsider*.

Previous Solutions. A trivial solution to the problem of increasing the threshold parameter of a (t, n)-threshold secret-sharing scheme to $t' > t$ is for the shareholders to discard their old shares and for the dealer to distribute new shares of a (t', n) secret-sharing scheme to all shareholders. However, this solution is not very attractive, since it requires the dealer to be involved after the setup stage and moreover requires communication between the dealer and each shareholder (such communication may be difficult to establish after the initial setup stage).

A much better solution would allow the threshold to be changed at any time without any communication between the dealer and shareholders after the setup stage. We say that such schemes allow *dealer-free* threshold changeability. A trivial dealer-free threshold changeable scheme can be constructed as follows: the dealer initially sets up $n - t + 1$ threshold schemes for each possible future threshold $t' \in \{t, t + 1, \ldots, n\}$, and gives to each shareholder $n - t + 1$ shares of the secret. Namely, for each $t' \in \{t, \ldots, n\}$, the shareholder receives a share of the secret for a (t', n)-threshold scheme. Such a trivial scheme may not be applicable because of the following drawbacks:

(1) *Non-standard Initial Scheme:* The dealer must plan ahead for future threshold increases by initially setting up a non-standard (t, n)-threshold scheme designed specifically for threshold-changeability, whose shares consist of $n - t+1$ shares corresponding to the $n-t+1$ underlying (t', n)-threshold schemes. Hence the trivial scheme cannot be applied to increase the threshold of existing *standard* Shamir (t, n)-schemes which were not originally designed for threshold changeability and in which each shareholder has only a single share of *one* Shamir (t, n)-scheme.

(2) *Large Storage/Communication Requirements for Shareholders:* Each shareholder must receive and store $n - t + 1$ shares, where each share is as long as the secret (assuming that perfect security is desired). Hence the trivial scheme cannot be applied when storage or communication costs for $n-t+1$ shares are prohibitive.

Other 'dealer-free' solutions to the threshold increase problem have been proposed in the literature (see related work below), but they all suffer from at least one of the two drawbacks above, or they require communication *between the shareholders*.

Our Contributions. In this paper, we present a new method for increasing the threshold of the *standard* Shamir (t, n)-threshold secret-sharing scheme[18], which does not have any of the drawbacks discussed above. In particular, and in contrast to previous solutions, our method does not require communication between the dealer and shareholders after the initial setup stage nor between shareholders, and can be applied to existing Shamir schemes even if they were set up without consideration to future threshold increase. Storage and communication costs are the same as for the standard Shamir scheme.

The basic idea of our method is the following: to increase the threshold from t to $t' > t$, the shareholders add an appropriate amount of random noise to their shares (or delete a certain fraction of the bits of their share) to compute *subshares* which contain *partial* information about (e.g. half the most-significant bits of) the original shares. Since the subshares contain only partial information about the original shares, a set of t subshares may no longer be sufficient to reconstruct the secret uniquely, but if one observes a sufficiently larger number $t' > t$ of subshares then one can expect the secret to be uniquely determined by these t' subshares (e.g. if the subshares contain only half the information in the original shares then one can expect that $t' = 2t$ subshares will uniquely determine the secret)[1]. By replacing the share *combiner* algorithm of the original (t, n)-threshold secret-sharing with an appropriate 'error-correction' algorithm which can uniquely recover the secret from any t' subshares, we obtain the desired threshold increase from t to t', leaving the secret unchanged.

Our efficient 'error-correction' combiner algorithm for the Shamir secret-sharing scheme is constructed using lattice basis reduction techniques. Thus, our method is a new positive cryptographic application for lattice reduction algorithms. Furthermore, we make use of fundamental tools from the theory of lattices (Geometry of Numbers) to prove quantitative statements about the information-theoretic security and correctness of our construction. These lattice-based security proof techniques may be of independent interest.

Although our threshold-increase method does not yield a perfect (t', n) secret-sharing scheme, we obtain a useful result about the information-theoretic security of our method, which we believe suffices for many applications. Roughly speaking, we prove that for any desired $\epsilon > 0$, our method can be used to change the threshold to $t' > t$ (meaning that any t' subshares can be used to recover the secret) such that any $t_s < t' - (t'/t)$ observed subshares leak to the attacker at most a fraction ϵ of the entropy of the secret, where ϵ can be made as small as we wish by an appropriate choice of security parameter.

[1] We remark that this intuitive reasoning is not rigorous, and indeed there exist examples for which it is incorrect. However, our results show that it is approximately true for the Shamir scheme.

Interestingly, our lattice-based methods can be adapted also to change the threshold of the standard integer-based Chinese Remainder Theorem (CRT) secret-sharing scheme[2]. We provide full details of this result in a companion paper [22].

Related Work. Several approaches to changing the parameters of a threshold scheme in the absence of the dealer have been proposed in the literature. The technique of *secret redistribution*[5, 16] involves communication among the shareholders to 'redistribute' the secret with the a threshold parameter. Although this technique can be applied to standard secret-sharing schemes, its disadvantage is the need for secure channels for communication between shareholders. Methods for changing threshold which do not require secure channels have been studied in [4, 14, 15, 13], but they all require the initial secret-sharing scheme to be a non-standard one, specially designed for threshold increase (as a simple example of such a non-standard scheme, the dealer could provide each shareholder with two shares of the secret: one share for a (t, n) scheme and one share for a (t', n) scheme).

Our scheme uses a lattice-based 'error-correction' algorithm which is a slight variant of an algorithm for 'Noisy Polynomial Approximation' with noise bounded in the Lee norm [20]. This algorithm in turn is one of a large of body of recent work on 'list decoding' of Reed-Solomon and Chinese Remainder codes [9, 19, 6, 21]. We remark also that although the *correctness* proof of our scheme is based on the work of [20], our *security* proof is new and the lattice-based techniques used may be of independent interest.

Organization of This Paper. Section 2 presents notations, known results on lattices, and a counting lemma that we use. In Section 3, we provide definitions of changeable-threshold secret-sharing schemes and their correctness/security notions. In Section 4 we present the original Shamir (t, n)-threshold secret sharing scheme, and our threshold-changing algorithms to increase the threshold to $t' > t$. We then provide concrete proofs of the correctness and security properties of our scheme. Section 5 concludes the paper. Due to page limitations, some proofs have been omitted. They are included in the full version of this paper, available on the authors' web page.

2 Preliminaries

2.1 Notation

Vectors and Polynomials. For a vector $\mathbf{v} \in \mathbb{R}^n$, we write $\mathbf{v} = (\mathbf{v}[0], \ldots, \mathbf{v}[n-1])$, where, for $i = 0, \ldots, n-1$, $\mathbf{v}[i]$ denotes the ith coordinate of $\mathbf{v}$. Similarly for a polynomial $a(x) = a[0] + a[1]x + \ldots + a[t-1]x^{t-1}$, we let $a[i]$ denote the coefficient of x^i. For a ring R, we denote the set of all polynomial of degree at most t with coefficients in the ring R by $R[x; t]$.

Lee and Infinity Norm. For a prime p and an integer z we denote *Lee norm of z modulo p* as $\|z\|_{L,p} = \min_{k \in \mathbb{Z}} |z - kp|$. Similarly, for a vector $\mathbf{v} \in \mathbb{Z}^n$, we

define the Lee norm of $\mathbf{v}$ modulo p by $\|\mathbf{v}\|_{L,p} = \max_{0\le i\le n-1} \|\mathbf{v}[i]\|_{L,p}$. For a vector $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$, we denote the infinity norm of $\mathbf{z}$ by $\|\mathbf{z}\|_\infty = \max_{1\le i\le n} |z_i|$. For integers a and p, we denote $a \bmod p$ by $\lfloor a \rfloor_p$. For real z we define $Int(z) = \lceil z \rceil - 1$ as the largest integer strictly less than z.

Sets. For a set S, we denote by $\#S$ the size of S. For any set S and integer n, we denote by S^n the set of all n-tuples of elements from S and by $D(S^n)$ the set of all n-tuples of *distinct* elements from S. For integer n, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

Entropy. We denote by $\log(.)$ the logarithm function with base 2. For a discrete random variable s with probability distribution $P_s(.)$ on a set S, we denote by $H(s \in S) = \sum_{x\in S} P_s(x) \log(1/P_s(x))$ the Shannon entropy of s. Let t be any other random variable on a set T, and let u denote any element of T. Let $P_s(.|u)$ denote the conditional probability distribution of s given the event $t = u$. We denote by $H(s \in S|u) = \sum_{x\in S} P_s(x|u) \log(1/P_s(x|u))$ the conditional entropy of s given the event $t = u$.

2.2 Lattices

Here we collect several known results that we use about lattices, which can be found in [8, 10, 7]. Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a set of n linearly independent vectors in $\mathbb{R}^n$. The set

$$\mathcal{L} = \{\mathbf{z}\colon \mathbf{z} = c_1\mathbf{b}_1 + \ldots + c_n\mathbf{b}_n,\ c_1, \ldots, c_n \in \mathbb{Z}\}$$

is called an *n-dimensional (full-rank) lattice* with *basis* $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Given a basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n) \in \mathbb{R}^n$ for a lattice $\mathcal{L}$, we define the associated *basis matrix* $M_{\mathcal{L},\mathbf{B}}$ to be the (full-rank) $n \times n$ matrix whose ith row is the ith basis vector $\mathbf{b}_i$ for $i = 1, \ldots, n$. The quantity $|\det(M_{\mathcal{L},\mathbf{B}})|$ is independent of $\mathbf{B}$. It is called the *determinant* of the lattice $\mathcal{L}$ and denoted by $\det(\mathcal{L})$.

Given a basis for lattice $\mathcal{L}$, the problem of finding a shortest non-zero vector in $\mathcal{L}$ is known as the *shortest vector problem*, or SVP. An algorithm is called an *SVP approximation algorithm with $\|\cdot\|_\infty$-approximation factor* γ_{SVP} if it is guaranteed to find a non-zero lattice vector $\mathbf{c}$ such that $\|\mathbf{c}\|_\infty \le \gamma_{SVP} \min_{\mathbf{v}\in\mathcal{L}\setminus\mathbf{0}} \|\mathbf{v}\|_\infty$. The celebrated *LLL algorithm* of Lenstra, Lenstra and Lovász [12] is a *fully* polynomial time SVP approximation algorithm with $\|\cdot\|_\infty$-approximation factor $\gamma_{LLL} = n^{1/2}2^{n/2}$. Also, as shown in[1, 11], there exists an SVP approximation algorithm with $\|\cdot\|_\infty$-approximation factor $\gamma_{ex} = n^{1/2}$ which polynomial time in the size of elements of M_{cL} but not in dimension of $\mathcal{L}$.

In this paper we actually need to solve a variation of SVP called the *closest vector problem* (CVP): given a basis of a lattice $\mathcal{L}$ in $\mathbb{R}^n$ and a "target" vector $\mathbf{t} \in \mathbb{R}^n$, find a lattice vector $\mathbf{c}$ such that $\|\mathbf{c}-\mathbf{t}\|_\infty$ is minimized. An algorithm is called a *CVP approximation algorithm with $\|\cdot\|_\infty$-approximation factor* γ_{CVP} if it is guaranteed to find a lattice vector $\mathbf{c}$ such that $\|\mathbf{c}-\mathbf{t}\|_\infty \le \gamma_{CVP} \min_{\mathbf{v}\in\mathcal{L}} \|\mathbf{v}-\mathbf{t}\|_\infty$. Babai[3] has shown how to convert the LLL algorithm into a *fully* polynomial CVP approximation algorithm with $\|\cdot\|_\infty$-approximation factor $\gamma_{Bab} = n^{1/2}2^{n/2}$.

In our proof of security we use several fundamental theorems from the theory of lattices ('Geometry of Numbers'). The original theorems are quite general, but the restricted versions stated below suffice for our purposes. First, we need the following definition of *successive Minkowski minima* of a lattice.

Definition 1 (Minkowski Minima). *Let $\mathcal{L}$ be a lattice in $\mathbb{R}^n$. For $i = 1, \ldots, n$, the ith succesive Minkowski minimum of $\mathcal{L}$, denoted $\lambda_i(\mathcal{L})$, is the smallest real number such that there exists a set $\{\mathbf{b}_1, \ldots, \mathbf{b}_i\}$ of i linearly-independent vectors in $\mathcal{L}$ with $\|\mathbf{b}_j\|_\infty \leq \lambda_i(\mathcal{L})$ for all $j = 1, \ldots, i$.*

Note that $\lambda_1(\mathcal{L})$ is just the shortest infinity-norm over all non-zero vectors in $\mathcal{L}$. Next, we state Minkowski's 'first theorem' in the geometry of numbers.

Theorem 1 (Minkowski's First Theorem). *Let $\mathcal{L}$ be a lattice in $\mathbb{R}^n$ and let $\lambda_1(\mathcal{L})$ denote the first Minkowski minimum of $\mathcal{L}$ (see Def. 1). Then $\lambda_1(\mathcal{L}) \leq \det(\mathcal{L})^{\frac{1}{n}}$.*

We will use the following point-counting variant of Minkowski's 'first theorem', which is due to Blichfeldt and van der Corput(see [8]).

Theorem 2 (Blichfeldt-Corput). *Let $\mathcal{L}$ be a lattice in $\mathbb{R}^n$ and let K denote the origin-centered box $\{\mathbf{v} \in \mathbb{R}^n : \|\mathbf{v}\|_\infty < H\}$ of volume $Vol(K) = (2H)^n$. Then the number of points of the lattice $\mathcal{L}$ contained in the box K is at least $2 \cdot Int\left(\frac{Vol(K)}{2^n \det(\mathcal{L})}\right) + 1$, where for any $z \in \mathbb{R}$, $Int(z)$ denotes the largest integer which is strictly less than z.*

Finally, we will also make use of Minkowski's 'second theorem' [8].

Theorem 3 (Minkowski's Second Theorem). *Let $\mathcal{L}$ be a full-rank lattice in $\mathbb{R}^n$ and let $\lambda_1(\mathcal{L}),\ldots,\lambda_n(\mathcal{L})$ denote the n Minkowski minima of $\mathcal{L}$ (see Definition 1). Then $\lambda_1(\mathcal{L}) \cdots \lambda_n(\mathcal{L}) \leq 2^n \det(\mathcal{L})$.*

2.3 An Algebraic Counting Lemma

The following is a fundamental lemma that we use, interestingly, for *both* the correctness and security proofs of our construction. Fix a prime p defining the finite field $\mathbb{Z}_p$, positive integer parameters $(\widehat{n}, \widehat{t}, \widehat{H})$, and an arbitrary set $\widehat{A}$ of polynomials of degree at least 1 and at most $\widehat{t}$ over $\mathbb{Z}_p$. The lemma gives us an upper bound on the probability that, for $\widehat{n}$ randomly chosen elements $\alpha_1, \ldots, \alpha_{\widehat{n}}$ of $\mathbb{Z}_p$, there will exist a polynomial $a(x) \in \widehat{A}$ which has 'small' absolute value modulo p (less than $\widehat{H}$) at all the points $\alpha_1, \ldots, \alpha_{\widehat{n}}$. We remark that a similar (and more general) lemma was used in the analysis of a polynomial approximation algorithm [20]. Note that the lemma does not hold in general if we allow $\widehat{A}$ to contain constant polynomials, since these polynomials may have constant coefficient smaller than $\widehat{H}$.

Lemma 1. *Fix a prime p, positive integers $(\widehat{n}, \widehat{t}, \widehat{H})$, and a non-empty set $\widehat{A}$ of polynomials of degree at least 1 and at most $\widehat{t}$ with coefficients in $\mathbb{Z}_p$. Let*

$\mathcal{E}(\widehat{n},\widehat{t},\widehat{H},\widehat{A}) \subseteq \mathbb{Z}_p^{\widehat{n}}$ denote the set of vectors $\alpha = (\alpha_1,\ldots,\alpha_{\widehat{n}}) \in \mathbb{Z}_p^{\widehat{n}}$ for which there exists a polynomial $a \in \widehat{A}$ such that $\|a(\alpha_i)\|_{L,p} < \widehat{H}$ for all $i = 1,\ldots,\widehat{n}$. The size of the set $\mathcal{E}(\widehat{n},\widehat{t},\widehat{H},\widehat{A})$ is upper bounded as follows:

$$\#\mathcal{E}(\widehat{n},\widehat{t},\widehat{H},\widehat{A}) \leq \#\widehat{A} \cdot (2\widehat{H}\widehat{t})^{\widehat{n}}.$$

Proof. Suppose that $\alpha = (\alpha_1,\ldots,\alpha_{\widehat{n}}) \in \mathbb{Z}_p^{\widehat{n}}$ is such that there exists a polynomial $a \in \widehat{A}$ such that

$$\|a(\alpha_i)\|_{L,p} < \widehat{H} \text{ for } i = 1,\ldots,\widehat{n}. \tag{1}$$

It follows that there exist $\widehat{n}$ integers $r_1,\ldots,r_{\widehat{n}}$ such that, for each $i = 1,\ldots,\widehat{n}$, we have $a(\alpha_i) - r_i \equiv 0 \pmod{p}$ with $|r_i| < \widehat{H}$ and hence α_i is a zero of the polynomial $g_i(x) = a(x) - r_i$ over $\mathbb{Z}_p$. But for each i, g_i is a polynomial of degree at least 1 and at most $\widehat{t}$ over $\mathbb{Z}_p$ and hence has at most $\widehat{t}$ zeros in $\mathbb{Z}_p$. So for each possible value for $(r_1,\ldots,r_n) \in (-\widehat{H},\widehat{H})^{\widehat{n}}$ and $a \in \widehat{A}$, there are at most $\widehat{t}^{\widehat{n}}$ 'bad' values for $\alpha = (\alpha_1,\ldots,\alpha_{\widehat{n}})$ in $(\mathbb{Z}_p)^{\widehat{n}}$ such that (1) holds. Using the fact that there are less than $(2\widehat{H})^{\widehat{n}}$ possible values for $(r_1,\ldots,r_{\widehat{n}})$ and less than $\#\widehat{A}$ possible values for a, the claimed bound follows. □

3 Definition of Changeable-Threshold Secret-Sharing Schemes

We will use the following definition of a threshold secret-sharing scheme, which is a slight modification of the definition in [17].

Definition 2 (Threshold Scheme). *A (t,n)-threshold secret-sharing scheme $\mathsf{TSS} = (\mathsf{GC},\mathsf{D},\mathsf{C})$ consists of three efficient algorithms:*

1. GC *(Public Parameter Generation): Takes as input a security parameter $k \in \mathcal{N}$ and returns a string $x \in \mathcal{X}$ of public parameters.*
2. D *(Dealer Setup): Takes as input a security/public parameter pair (k,x) and a secret s from the secret space $\mathcal{S}(k,x) \subseteq \{0,1\}^{k+1}$ and returns a list of n shares $\mathbf{s} = (s_1,\ldots,s_n)$, where s_i is in the ith share space $\mathcal{S}_i(k,x)$ for $i = 1,\ldots,n$. We denote by*

$$\mathsf{D}_{k,x}(.,.) : \mathcal{S}(k,x) \times \mathcal{R}(k,x) \rightarrow \mathcal{S}_1(k,x) \times \cdots \times \mathcal{S}_n(k,x)$$

the mapping induced by algorithm D (here $\mathcal{R}(k,x)$ denotes the space of random inputs to the probabilistic algorithm D).
3. C *(Share Combiner): Takes as input a security/public parameter pair (k,x) and any subset $\mathbf{s}_I = (s_i : i \in I)$ of t out of the n shares, and returns a recovered secret $s \in \mathcal{S}(k,x)$. (here I denotes a subset of $[n]$ of size $\#I = t$).*

The correctness and security properties of a (t, n)-threshold secret-sharing scheme can be quantified by the following definitions, which are modifications of those in [17].

Definition 3 (Correctness, Security). *A (t, n) threshold secret-sharing scheme $\mathsf{TSS} = (\mathsf{GC}, \mathsf{D}, \mathsf{C})$ is said to be:*

1. δ_c-correct*: If the secret recovery fails for a 'bad' set of public parameters of probability p_f at most δ_c. Precisely, p_f is the probability (over $x = \mathsf{GC}(k) \in \mathcal{X}$) that there exist $(s, r) \in \mathcal{S}(k, x) \times \mathcal{R}(k, x)$ and $I \subseteq [n]$ with $\#I = t$ such that $\mathsf{C}_{k,x}(\mathbf{s}_I) \not= s$, where $\mathbf{s} = \mathsf{D}_{k,x}(s, r)$ and $\mathbf{s}_I \stackrel{\text{def}}{=} \{s_i : i \in I\}$.*
   We say that TSS is asymptotically correct if, for any $\delta > 0$, there exists $k_0 \in \mathcal{N}$ such that TSS is δ-correct for all $k > k_0$.
2. $(t_s, \delta_s, \epsilon_s)$-secure *with respect to the secret probability distribution $P_{k,x}$ on $\mathcal{S}(k, x)$: If, with probability at least $1-\delta_s$ over the choice of public parameters $x = \mathsf{GC}(k)$, the worst-case secret entropy loss for any t_s observed shares is at most ϵ_s, that is*

$$|L_{k,x}(\mathbf{s}_I)| \stackrel{\text{def}}{=} |H(s \in \mathcal{S}(k, x)) - H(s \in \mathcal{S}(k, x)|\mathbf{s}_I)| \leq \epsilon_s,$$

   for all $\mathbf{s} \in \mathcal{S}_1(k, x) \times \cdots \times \mathcal{S}_n(k, x)$ and $I \subseteq [n]$ with $\#I \leq t_s$. We say that TSS is asymptotically t_s-secure with respect to $P_{k,x}$ if, for any $\delta > 0$ and $\epsilon > 0$ there exists $k_0 \in \mathcal{N}$ such that TSS is $(t_s, \delta, \epsilon \cdot k)$-secure with respect to $P_{k,x}$ for all $k > k_0$.

The following definition of the *Threshold Changeability* without dealer assistance for a secret sharing scheme is a modification of the definition in [15].

Definition 4 (Threshold-Changeability). *A (t, n)-threshold secret-sharing scheme $\mathsf{TSS} = (\mathsf{GC}, \mathsf{D}, \mathsf{C})$ is called* threshold-changeable *to t' with δ_c-correctness and $(t_s, \delta_s, \epsilon_s)$-security with respect to secret distribution $P_{k,x}$, if there exist n efficient subshare generation algorithms $\mathsf{H}_i : \mathcal{S}_i(x, k) \to \mathcal{T}_i(x, k)$ for $i = 1, \ldots, n$, and an efficient subshare combiner algorithm C' such that the modified (t', n)-threshold scheme $\mathsf{TSS}' = (\mathsf{GC}, \mathsf{D}', \mathsf{C}')$, with modified shares*

$$\mathsf{D}'_{k,x}(s, r) \stackrel{\text{def}}{=} (\mathsf{H}_1(s_1), \ldots, \mathsf{H}_n(s_n)) \in \mathcal{T}_1(k, x) \times \cdots \mathcal{T}_n(k, x),$$

where $(s_1, \ldots, s_n) = \mathsf{D}_{k,x}(s; r)$, is δ_c-correct and $(t_s, \delta_s, \epsilon_s)$-secure with respect to $P_{k,x}$. TSS is called asymptotically threshold-changeable to (t_s, t') with respect to $P_{k,x}$ *if there exist algorithms $\mathsf{H}_i : \mathcal{S}_i(k, x) \to \mathcal{T}_i(k, x)$ $(i = 1, \ldots, n)$ and C' such that the (t', n)-threshold scheme TSS' defined above is asymptotically correct and asymptotically t_s-secure with respect to $P_{k,x}$.*

The idea captured by the above definition is that the change of threshold from t to t' is implemented by getting each shareholder to replace his original share s_i by the subshare $\mathsf{H}_i(s_i)$ output by the subshare generation algorithm H_i (the original share s_i is then discarded).

4 Threshold-Changeability for Shamir Secret-Sharing

4.1 The Standard Shamir Scheme

The standard Shamir (t, n)-threshold secret sharing scheme is defined as follows.

Scheme $\mathsf{ShaTSS} = (\mathsf{GC}, \mathsf{D}, \mathsf{C})$**: Shamir** (t, n)**-Threshold Secret-Sharing**

1. $\mathsf{GC}(k)$ (Public Parameter Generation):
   (a) Pick a (not necessarily random) prime $p \in [2^k, 2^{k+1}]$ with $p > n$.
   (b) Pick uniformly at random n distinct non-zero elements $\alpha = (\alpha_1, \ldots, \alpha_n) \in D((\mathbb{Z}_p^*)^n)$. Return $x = (p, \alpha)$.
2. $\mathsf{D}_{k,x}(s, \mathbf{a})$ (Dealer Setup): To share secret $s \in \mathbb{Z}_p$ using $t-1$ uniformly random elements $\mathbf{a} = (a_1, \ldots, a_{t-1}) \in \mathbb{Z}_p^{t-1}$, build the polynomial $a_{s,\mathbf{a}}(x) = s + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1} \in \mathbb{Z}_p[x; t-1]$. The ith share is $s_i = a(\alpha_i) \bmod p$ for $i = 1, \ldots, n$.
3. $\mathsf{C}_{k,x}(\mathbf{s}_I)$ (Share Combiner): To combine shares $\mathbf{s}_I = (s_i : i \in I)$ for some $I \subseteq [n]$ with $\#I = t$, compute by Lagrange interpolation the unique polynomial $b \in \mathbb{Z}_p[x; t-1]$ such that $b(\alpha_i) \equiv s_i \pmod p$ for all $i \in I$. The recovered secret is $s = b(0) \bmod p$.

4.2 Threshold-Changing Algorithms

Our threshold-changing subshare generation and combiner algorithms to change the (t, n)-threshold scheme $\mathsf{ShaTSS} = (\mathsf{GC}, \mathsf{D}, \mathsf{C})$ into a (t', n)-threshold scheme $\mathsf{ShaTSS}' = (\mathsf{GC}, \mathsf{D}', \mathsf{C}')$ are defined as follows. Note that the subshare combiner algorithm uses an efficient CVP approximation algorithm $\mathsf{A}_{\mathsf{CVP}}$ with $\|\cdot\|_\infty$-approximation factor γ_{CVP}. We define $\Gamma_{CVP} = \log(\lceil \gamma_{CVP} + 1 \rceil)$ (if we use the Babai poly-time CVP algorithm, we have $\Gamma_{CVP} \leq 1 + 0.5(t' + t + \log(t' + t))$).

Scheme ShaTSS'**: Changing Threshold to** $t' > t$

1. $\mathsf{H}_i(s_i)$ (ith Subshare Generation): To transform share $s_i \in \mathbb{Z}_p$ of original (t, n)-threshold scheme into subshare $t_i \in \mathbb{Z}_p$ of desired (t', n)-threshold scheme $(t' > t)$ the ith shareholder does the following (for all $i = 1, \ldots, n$):
   (a) Determine noise bound H which guarantees δ_c-correctness (typically, we set $\delta_c = k^{-t'}$):
      i. Set $H = \max(\lfloor p^\alpha / 2 \rfloor, 1)$ with
      ii. $\alpha = 1 - \frac{1 + \delta_F}{(t'/t)} > 0$ (noise bitlength fraction) and
      iii. $\delta_F = \frac{(t'/t)}{k} \left(\log(\delta_c^{-1/t'} nt) + \Gamma_{CVP} + 1 \right)$.
   (b) Compute $t_i = \alpha_i \cdot s_i + r_i \bmod p$ for a uniformly random integer r_i with $|r_i| < H$.
2. $\mathsf{C}'_{k,x}(\mathbf{t}_I)$ (Subshare Combiner): To combine subshares $\mathbf{t}_I = (t_i : i \in I)$ for some $I = \{i[1], \ldots, i[t']\}$ with $\#I = t'$ (and guaranteed δ_c-correctness), do the following:

(a) Build the following $(t'+t) \times (t'+t)$ matrix $M_{Sha}(\alpha_I, H, p)$, whose rows form a basis for a full-rank lattice $\mathcal{L}_{Sha}(\alpha_I, H, p)$ in $\mathbb{Q}^{t'+t}$:

$$M_{Sha}(\alpha_I, H, p) = \begin{pmatrix} p & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & p & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & p & 0 & 0 & \dots & 0 \\ \alpha_{i[1]} & \alpha_{i[2]} & \dots & \alpha_{i[t']} & H/p & 0 & \dots & 0 \\ \alpha_{i[1]}^2 & \alpha_{i[2]}^2 & \dots & \alpha_{i[t']}^2 & 0 & H/p & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{i[1]}^t & \alpha_{i[2]}^t & \dots & \alpha_{i[t']}^t & 0 & 0 & \dots & H/p \end{pmatrix}.$$

Here $H = \lfloor p^{\alpha}/2 \rfloor$, $\alpha = 1 - \frac{1+\delta_F}{(t'/t)}$, $\delta_F = \frac{(t'/t)}{k}\left(\log(\delta_c^{-1/t'} nt) + \Gamma_{CVP} + 1\right)$.

(b) Define $\mathbf{t}' = (t_{i[1]}, \dots, t_{i[t']}, 0, 0, \dots, 0) \in \mathbb{Z}^{t'+t}$.

(c) Run CVP Approx. alg. $\mathsf{A_{CVP}}$ on lattice $\mathcal{L}_{Sha}(\alpha_I, H, p)$ given by $M_{Sha}(\alpha_I, H, p)$ with target vector $\mathbf{t}'$. Let $\mathbf{c} = (c_1, \dots, c_{t'}, c_{t'+1}, \dots, c_{t'+t}) \in \mathbb{Q}^{t'+t}$ denote the output vector returned by $\mathsf{A_{CVP}}$, approximating the closest vector in $\mathcal{L}_{Sha}$ to $\mathbf{t}'$.

(d) Compute the recovered secret $\widehat{s} = (p/H) \cdot c_{t'+1} \bmod p$.

Remark 1. The reason for multiplying the shares s_i by α_i before adding the noise r_i, is that otherwise, the secret may not be uniquely recoverable from the noisy subshares (indeed, $a(\alpha_i) + r_i = a(\alpha_i) + 1 + (r_i - 1)$, and typically $|r_i - 1| < H$, so secrets s and $s+1$ would be indistinguishable).

Remark 2. It is not difficult to see that our method of adding a 'small' random noise integer r_i with $|r_i| < H$ to the share multiple $\alpha_i \cdot s_i$ modulo p, is essentially equivalent (in the sense of information on the secret) to passing the residues $\alpha_i \cdot s_i \bmod p$ through a deterministic function which chops off the $\log(2H) \approx (1 - 1/(t'/t)) \cdot k$ least-significant bits of the k-bit residues $\alpha_i \cdot s_i \bmod p$, and this also yields shorter subshares than in our method above. However, since reducing the length of the original shares is not our main goal, we have chosen to present our scheme as above since it slightly simplifies our scheme and its analysis. Similar results can be obtained, however, for the 'deterministic' approach of chopping off least-significant bits.

Remark 3. Some special variants of the Shamir scheme use special values for the points α_i, such as $\alpha_i = i$ for $i = 1, \dots, n$, to which the above method does not apply, because of its reliance on the random choice of the α_i's. However, it turns out that our method can be modified to work even for these special Shamir variants. The idea is to make up for the loss of randomness in the α_i's by getting the shareholders to multiply their shares by additional random integers (say $B_i \in \mathbb{Z}_p$) prior to adding the random noise r_i. The B_i's are then sent along to the combiner with the noisy subshares. We do not analyze this variant of our scheme in this paper.

4.3 Correctness

The following theorem shows that the choice of the parameter δ_F used in our threshold changing algorithm is sufficient to guarantee the δ_c-correctness of our scheme for all sufficiently large security parameters.

Theorem 4 (Correctness). *The scheme* ShaTSS′ *(with parameter choice* $\delta_c = k^{-t'}$*) is asymptotically correct. Concretely, for any choice of parameter* δ_c *(*$0 < \delta_c < 1$*), the* (t', n) *scheme* ShaTSS′ *is* δ_c*-correct for all security parameters* k *satisfying the inequality* $k \geq k'_0$*, where*

$$k'_0 = \left(\frac{t'/t}{t'/t-1}\right)\left(\log(\delta_c^{-1/t'} nt) + \Gamma_{CVP} + 2\right).$$

Proof. Fix a subshare subset $I \subseteq [n]$ with $\#I = t'$. We know by construction of lattice $\mathcal{L}_{Sha}(\alpha_I)$, that the dealer's secret polynomial $a_{s,\mathbf{a}}(x) = s + a_1 x + \ldots + a_{t-1}x^{t-1} \in \mathbb{Z}_p[x; t-1]$ gives rise to the lattice vector

$$\mathbf{a}' = (\alpha_{i[1]} a_{s,\mathbf{a}}(\alpha_{i[1]}) - k_1 p, \ldots, \alpha_{i[t']} a_{s,\mathbf{a}}(\alpha_{i[t']}) - k_{t'} p, (\frac{s}{p}H), \frac{a_1}{p}H, \ldots, \frac{a_{t-1}}{p}H),$$

which is "close" to the target vector

$$\mathbf{t}' = (\alpha_{i[1]} a_{s,\mathbf{a}}(\alpha_{i[1]}) - k_1 p + r_{i[1]}, \ldots, \alpha_{i[t']} a_{s,\mathbf{a}}(\alpha_{i[t']}) - k_{t'} p + r_{i[t']}, 0, 0, \ldots, 0),$$

where $k_j = \lfloor \frac{\alpha_{i[j]} a(\alpha_{i[j]}) + r_{i[j]}}{p} \rfloor \in \mathbb{Z}$ for all $j = 1, \ldots, t'$. In particular we have, using $|r_{i[j]}| < H$ for all $j = 1, \ldots, t'$, that $\|\mathbf{a}' - \mathbf{t}'\|_\infty < H$. Consequently, since $\mathsf{A}_{\mathsf{CVP}}$ is a CVP approximation algorithm with $\|.\|_\infty$ approximation factor γ_{CVP}, its output lattice vector $\mathbf{c}$ will also be "close" to the target vector, namely we have $\|\mathbf{c} - \mathbf{t}'\|_\infty < \gamma_{CVP} \cdot H$. Applying the triangle inequality, we conclude that the lattice vector $\mathbf{z} = \mathbf{c} - \mathbf{a}'$ satisfies

$$\|\mathbf{z}\|_\infty = \|\mathbf{c} - \mathbf{a}'\|_\infty < (\gamma_{CVP} + 1)H. \tag{2}$$

Now, either $\frac{p}{H}\mathbf{c}[t'+1] \equiv \frac{p}{H}\mathbf{a}'[t'+1] \equiv s \pmod p$ in which case the combiner succeeds to recover secret s, or otherwise we have the 'bad' case that

$$\frac{p}{H}\mathbf{z}[t'+1] = \frac{p}{H}\mathbf{c}[t'+1] - \frac{p}{H}\mathbf{a}'[t'+1] \not\equiv 0 \pmod p. \tag{3}$$

Hence, for fixed I, the combiner succeeds except for a fraction δ_I of 'bad' choices of $\alpha_I \in D((\mathbb{Z}_p^*)^{t'})$, for which $\mathcal{L}_{Sha}(\alpha_I)$ contains a 'short' and 'bad' vector $\mathbf{z}$ satisfying (2) and (3). To upper bound δ_I, consider the polynomial $f(x) = \frac{p}{H}\mathbf{z}[t'+1]x + \cdots + \frac{p}{H}\mathbf{z}[t'+t]x^t$. Note that, since $\mathbf{z} \in \mathcal{L}_{Sha}$, we have $f(\alpha_{i[j]}) \equiv \mathbf{z}[j] \pmod p$ and hence $\|f(\alpha_{i[j]})\|_{L,p} < (\gamma_{CVP}+1)H$ for all $j \in [t']$ using (2). Also, $f(x) \bmod p$ has zero constant coefficient and degree at least 1 and at most t over $\mathbb{Z}_p$ using (3). Applying Lemma 1 (with parameters $\widehat{n} = t', \widehat{t} = t, \widehat{H} = 2^{\Gamma_{CVP}} H$, $\#\widehat{A} \leq p^t$) we conclude that such a 'bad' polynomial f exists for at most a fraction $\delta_I \leq p^t (2\widehat{H}t)^{t'} / \#D((\mathbb{Z}_p^*)^{t'})$ of $\alpha_I \in D((\mathbb{Z}_p^*)^{t'})$, for each

fixed I. Hence, the probability δ that a uniformly chosen $\alpha \in D((\mathbb{Z}_p^*)^n)$ is 'bad' for *some* $I \subseteq [n]$ with $\#I = t'$ is upper bounded as

$$\delta \leq \frac{\binom{n}{t'} p^t (2\widehat{H}t)^{t'}}{\#D((\mathbb{Z}_p^*)^{t'})}, \tag{4}$$

and a straightforward calculation (see full paper) shows that the right-hand side of (4) is upper bounded by δ_c for all $k \geq \left(\frac{t'/t}{t'/t-1}\right)\left(\log(\delta_c^{-1/t'} nt) + \Gamma_{CVP} + 2\right)$. This completes the proof. □

4.4 Security

The concrete security of our scheme is given by the following result. It shows that, for fixed (t', n) and with parameter choice $\delta_c = k^{-t'}$, the (t', n) scheme ShaTSS′ leaks at most fraction $\epsilon_s/k = O(\log k/k) = o(1)$ of the entropy of the secret to an attacker observing less than $t' - (t'/t)$ subshares (for all except a fraction $\delta_s \leq \delta_c = o(1)$ of public parameters, and assuming the security parameter k is sufficiently large).

Theorem 5 (Security). *The scheme* ShaTSS′ *(with parameter choice* $\delta_c = k^{-t'}$*) is asymptotically* $Int(t' - (t'/t))$*-secure with respect to the uniform secret distribution on* $\mathbb{Z}_p$*. Concretely, for any parameter choice* $\delta_c > 0$*, the* (t', n) *scheme* ShaTSS′ *is* $(t_s, \delta_s, \epsilon_s)$*-secure with:*

$$t_s = \frac{t' - (t'/t)}{1 + \frac{(t'/t)}{k}\left(\log(\delta_c^{-1/t'} nt) + \Gamma_{CVP} + 1\right)},$$

$$\delta_s = \delta_c,\ \epsilon_s = (\beta + 7)(t_s + t) + t_s \log t + 1,\ \beta = \frac{\log(2\delta_c^{-1}\binom{n}{t_s})}{t_s + t - 1},$$

for all security parameters $k \geq k_0$*, where, letting* $m = t_s + t + 1$ *and* k_0' *as defined in Theorem 4,*

$$k_0 = \max\left(k_0' + \frac{(t'/t+1)^2}{t'/t-1}(\beta + \log t + 3), (\beta+4)m^2 + 5t_s m \log m\right).$$

Proof. (Sketch) Fix an observed subshare subset $I \subseteq [n]$ with $\#I = t_s$. Assuming the secret is uniformly distributed on $\mathbb{Z}_p$ it is easy to show (see full paper) that the conditional probability $P_{k,x}(s|\mathbf{s}_I)$ of the secret taking the value $s \in \mathbb{Z}_p$ given that the observed sub-share vector takes the value $\mathbf{s}_I$ is given by:

$$P_{k,x}(s|\mathbf{s}_I) = \frac{\#S_{s,p}(\alpha_I, t, p, H, \mathbf{s}_I)}{\#S_{0,1}(\alpha_I, t, p, H, \mathbf{s}_I)}, \tag{5}$$

where, for any integers $\widehat{s} \geq 0$ and $\widehat{p} \geq 1$, we define the set

$$S_{\widehat{s},\widehat{p}}(\alpha_I, t, p, H, \mathbf{s}_I) \stackrel{\text{def}}{=} \{a \in \mathbb{Z}_p[x; t-1] : \|\alpha_{i[j]} a(\alpha_{i[j]}) - s_{i[j]}\|_{L,p} < H \forall j \in [t_s] \text{ and } a(0) \equiv \widehat{s} \pmod{\widehat{p}}\}.$$

We will derive a probabilistic lower bound on $\#S_{0,1}$ and upper bound on $\#S_{s,p}$ which both hold for all except a fraction $\delta_I \leq \delta_s / \binom{n}{t_s}$ of 'bad' choices for $\alpha_I \in D((\mathbb{Z}_p^*)^{t_s})$ assuming $k \geq k_0$ (with t_s, δ_s and k_0 defined in the theorem statement). We then apply these bounds to (5) to get a bound $P_{k,x}(s|\mathbf{s}_I) \leq 2^{\epsilon_s}/p$ for all s (with ϵ_s defined in the theorem statement) so that for fixed I, entropy loss is bounded as $L_{k,x}(\mathbf{s}_I) \leq \epsilon_s$, except for fraction δ_I of $\alpha_I \in D((\mathbb{Z}_p^*)^{t_s})$. It then follows that $L_{k,x}(\mathbf{s}_I) \leq \epsilon_s$ for *all* $I \subseteq [n]$ with $\#I = t_s$ except for a fraction $\delta \leq \binom{n}{t_s}\delta_I \leq \delta_s$ of $\alpha \in D((\mathbb{Z}_p^*)^n)$ assuming that $k \geq k_0$, which proves the theorem.

Reduction to Lattice Point Counting. It remains to derive the desired probabilistic upper and lower bounds on $\#S_{\widehat{s},\widehat{p}}$. The following lemma shows that $\#S_{\widehat{s},\widehat{p}}$ is equal to the number of points of a certain lattice $\mathcal{L}_{Sha}$ (closely related to the lattice used in our subshare combiner algoritm) contained in a (t_s+t)-dimensional box of side length $2H$, centered on a certain non-lattice vector $\widehat{\mathbf{s}}_I$.

Lemma 2. *Fix positive integers $(t, t_s, p, H, \widehat{p})$ such that $p \geq 2H$ and $\widehat{p}$ is a divisor of p. Let $\widehat{s} \in \mathbb{Z}_{\widehat{p}}$, $\alpha_I = (\alpha_{i[1]}, \ldots, \alpha_{i[t_s]}) \in \mathbb{Z}_p^n$ and $\mathbf{s}_I = (s_{i[1]}, \ldots, s_{i[t_s]}) \in \mathbb{Z}_p^{t_s}$. Define $\mathcal{L}_{Sha}(\alpha_I, t, p, H, \widehat{p})$ as the full-rank lattice in $\mathbb{Q}^{t_s+t}$ with basis consisting of the rows of the matrix*

$$M_{Sha}(\alpha_I, t, p, H, \widehat{p}) = \begin{pmatrix} p & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & p & \ldots & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & p & 0 & 0 & \ldots & 0 \\ \widehat{p}\alpha_{i[1]} & \widehat{p}\alpha_{i[2]} & \ldots & \widehat{p}\alpha_{i[t_s]} & 2H/(p/\widehat{p}) & 0 & \ldots & 0 \\ \alpha_{i[1]}^2 & \alpha_{i[2]}^2 & \ldots & \alpha_{i[t_s]}^2 & 0 & 2H/p & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{i[1]}^t & \alpha_{i[2]}^t & \ldots & \alpha_{i[t_s]}^t & 0 & 0 & \ldots & 2H/p \end{pmatrix},$$

and define the vector $\widehat{\mathbf{s}}_I \in \mathbb{Q}_{t_s+t}$ by

$$\widehat{\mathbf{s}}_I \stackrel{\text{def}}{=} \left(s_{i[1]} - \widehat{s}\alpha_{i[1]}, \ldots, s_{i[t_s]} - \widehat{s}\alpha_{i[t_s]}, H(1 - \frac{1+2\widehat{s}}{p}), H(1-\frac{1}{p}), \ldots, H(1-\frac{1}{p}) \right).$$

Then the sizes of the following two sets are equal:

$$S_{\widehat{s},\widehat{p}}(\alpha_I, t, p, H, \mathbf{s}_I) \stackrel{\text{def}}{=} \{a \in \mathbb{Z}_p[x; t-1] : \|\alpha_{i[j]} a(\alpha_{i[j]}) - s_{i[j]}\|_{L,p} < H \forall j \in [t_s] \text{ and } a(0) \equiv \widehat{s} \pmod{\widehat{p}}\},$$

and

$$V_{\widehat{s},\widehat{p}}(\alpha_I, t, p, H, \widehat{\mathbf{s}}_I) \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathcal{L}_{Sha}(\alpha_I, t, p, H, \widehat{p}) : \|\mathbf{v} - \widehat{\mathbf{s}}_I\|_\infty < H\}.$$

Finding a Lower Bound on $\#V_{0,1}$. Lower bounding the number $\#V_{0,1}$ of points of the lattice $\mathcal{L}_{Sha}$ in a symmetric box $T_{\mathbf{s}_I}(H) \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathbb{Q}^{t_s+1} :$

$\|\mathbf{v}-\widehat{\mathbf{s}}_I\|_\infty < H\}$ centered on vector $\widehat{\mathbf{s}}_I$ seems a difficult 'non-homogenous' problem because $\widehat{\mathbf{s}}_I$ is in general not a lattice vector. But by 'rounding' $\widehat{\mathbf{s}}_I$ to a nearby lattice vector $\widehat{\mathbf{s}}'_I$ (with rounding error $\epsilon = \|\widehat{\mathbf{s}}'_I - \widehat{\mathbf{s}}_I\|_\infty$), we reduce the problem to two simpler problems: (1) The 'homogenous' problem of lower bounding the number of lattice points in an *origin-centered* box $T_{\mathbf{0}} \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathbb{Q}^{t_s+t} : \|\mathbf{v}\|_\infty < H-\epsilon\}$, and (2) Upper bounding the largest Minkowski minimum $\lambda_{t_s+t}(\mathcal{L}_{Sha})$ of the lattice. This general reduction is stated precisely as follows.

Lemma 3. *For any full-rank lattice $\mathcal{L}$ in $\mathbb{R}^n$, vector $\mathbf{s} \in \mathbb{R}^n$, and $H > 0$, we have*

$$\#\{\mathbf{v} \in \mathcal{L} : \|\mathbf{v}-\mathbf{s}\|_\infty < H\} \geq \#\{\mathbf{v} \in \mathcal{L} : \|\mathbf{v}\|_\infty < H-\epsilon\},$$

where $\epsilon = \frac{n}{2} \cdot \lambda_n(\mathcal{L})$.

To solve the 'homogenous' counting problem (1) above we directly apply the Blichfeldt-Corput theorem (Theorem 2 in Sec. 2). To solve the problem (2) above of upper bounding $\lambda_{t_s+t}(\mathcal{L}_{Sha})$, we apply Minkowski's "second theorem" (Theorem 3 in Sec. 2) to reduce this problem further to the problem of *lower bounding* the *first* Minkowski minimum $\lambda_1(\mathcal{L}_{Sha})$. Namely, since $\lambda_i(\mathcal{L}_{Sha}) \geq \lambda_1(\mathcal{L}_{Sha})$ for all $i \in [t_s]$, then Minkowski's second theorem gives $\lambda_{t_s+t}(\mathcal{L}_{Sha}) \leq \frac{2^{t_s+t}\det(\mathcal{L}_{Sha})}{\lambda_1(\mathcal{L}_{Sha})^{t_s+t-1}}$. Finally, to lower bound $\lambda_1(\mathcal{L}_{Sha})$ (i.e. the infinity norm of the shortest non-zero vector in $\mathcal{L}_{Sha}$), we use a probabilistic argument based on the algebraic counting lemma 1 (similar to the argument used in proving Theorem 4), to obtain the following result.

Lemma 4. *Fix positive integers $(t, t_s, p, H, \widehat{p})$ and a positive real number β, such that $p \geq \max(2H, 2t_s)$ is prime and $\widehat{p} \in \{1, p\}$. For each $\alpha_I \in D((\mathbb{Z}_p^*)^{t_s})$, let $\mathcal{L}_{Sha}(\alpha_I, \widehat{p})$ denote the lattice in $\mathbb{Q}^{t_s+t}$ with basis matrix $M_{Sha}(\alpha_I, \widehat{p})$ defined in Lemma 2, and let $\mathcal{L}'_{Sha}(\alpha_I)$ denote the lattice in $\mathbb{Q}^{t_s+t-1}$ with basis matrix $M'_{Sha}(\alpha_I)$ obtained from $M_{Sha}(\alpha_I, \widehat{p})$ by removing the (t_s+1)th row and column. In the case $\widehat{p} = 1$, if*

$$1 \leq 2^{-[\beta+3+\frac{t_s \log t}{t_s+t}]} \det\left(\mathcal{L}_{Sha}(\alpha_I, 1)\right)^{\frac{1}{t_s+t}} \leq H$$

then, for at least a fraction $1 - 2^{-\beta(t_s+t)}$ of $\alpha_I \in D((\mathbb{Z}_p^*)^{t_s})$ we have

$$\lambda_1(\mathcal{L}_{Sha}(\alpha_I, 1)) \geq 2^{-[\beta+3+\frac{t_s \log t}{t_s+t}]} \det(\mathcal{L}_{Sha}(\alpha_I, 1))^{\frac{1}{t_s+t}}.$$

In the case $\widehat{p} = p$, if

$$1 \leq 2^{-[\beta+3+\frac{t_s \log t}{t_s+t-1}]} \det(\mathcal{L}'_{Sha}(\alpha_I))^{\frac{1}{t_s+t-1}} \leq H$$

then, for at least a fraction $1 - 2^{-\beta(t_s+t-1)}$ of $\alpha_I \in D((\mathbb{Z}_p^*)^{t_s})$ we have

$$\lambda_1(\mathcal{L}'_{Sha}(\alpha_I)) \geq \lambda_1(\mathcal{L}_{Sha}(\alpha_I, p)) \geq 2^{-[\beta+3+\frac{t_s \log t}{t_s+t-1}]} \det(\mathcal{L}'_{Sha}(\alpha_I))^{\frac{1}{t_s+t-1}}.$$

Combining the above results (for $(\widehat{s},\widehat{p}) = (0,1)$) we obtain the desired lower bound on $\#V_{0,1}$.

Finding an Upper Bound on $\#V_{s,p}$. We first reduce the point counting problem in $\mathcal{L}_{Sha}(\alpha_I, p)$ to a point counting problem in the lower-dimensional lattice $\mathcal{L}'_{Sha}(\alpha_I)$ defined in Lemma 4. This is possible because all the vectors of $\mathcal{L}_{Sha}(\alpha_I, p)$ in the desired box have their (t_s+1)th coordinate equal to 0.

Lemma 5. *Let $\mathcal{L}_{Sha}(\alpha_I, p) \subseteq \mathbb{Q}^{t_s+t}$ and $\mathcal{L}'_{Sha}(\alpha_I) \subseteq \mathbb{Q}^{t_s+t-1}$ be the lattices defined in Lemma 4, let $\widehat{\mathbf{s}}_I$ be the vector in $\mathbb{Q}^{t_s+t}$ defined in Lemma 2, and let $\widehat{\mathbf{s}}'_I$ be the vector in $\mathbb{Q}^{t_s+t-1}$ obtained from $\widehat{\mathbf{s}}_I$ by removing the (t_s+1)th coordinate. Then $\#V_{s,p} \leq \#V'_{s,p}$, where $V_{s,p} \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathcal{L}_{Sha}(\alpha_I, p) : \|\mathbf{v} - \widehat{\mathbf{s}}_I\|_\infty < H\}$ and $V'_{s,p} \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathcal{L}'_{Sha}(\alpha_I) : \|\mathbf{v} - \widehat{\mathbf{s}}'_I\|_\infty < H\}$.*

By comparing the total volume of the $\#V_{s,p}$ disjoint boxes of sidelength $\lambda_1(\mathcal{L}'_{Sha})$ centered on the lattice points in $T_{\widehat{\mathbf{s}}'_I}(H) \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathbb{Q}^{t_s+t-1} : \|\mathbf{v} - \widehat{\mathbf{s}}'_I\|_\infty < H\}$, to the volume of $\widehat{T}_{\widehat{\mathbf{s}}'_I}(H) \stackrel{\text{def}}{=} \{\mathbf{v} \in \mathbb{Q}^{t_s+t-1} : \|\mathbf{v} - \widehat{\mathbf{s}}'_I\|_\infty < H + \lambda_1(\mathcal{L}'_{Sha})/2\}$ which contains those disjoint boxes, we reduce the problem of upper bounding $\#V_{s,p}$ to the problem of lower bounding the $\lambda_1(\mathcal{L}'_{Sha})$. This general reduction can be stated as follows.

Lemma 6. *For any lattice $\mathcal{L}$ in $\mathbb{R}^n$, vector $\mathbf{s} \in \mathbb{R}^n$, and $H > 0$, we have*

$$\#\{\mathbf{v} \in \mathcal{L} : \|\mathbf{v} - \mathbf{s}\|_\infty < H\} \leq \left[\frac{2H}{\lambda_1(\mathcal{L})} + 1\right]^n .$$

Now we apply the probabilistic lower bound on $\lambda_1(\mathcal{L}'_{Sha})$ from Lemma 4 in Lemma 6 (with $(\widehat{s},\widehat{p}) = (s,p)$) to get the desired upper bound on $\#V_{s,p}$.

After some straightforward calculation (see full paper), we find that the probabilistic lower and upper bounds on $\#V_{\widehat{s},\widehat{p}}$ obtained above hold for all except a fraction $\delta_I \leq \delta_s / \binom{n}{t_s}$ of 'bad' choices for $\alpha_I \in D((\mathbb{Z}_p^*)^{t_s})$ assuming $k \geq k_0$ (with t_s, δ_s and k_0 defined in the theorem statement), and plugging the bounds in (5) gives the desired bound $P_{k,x}(s|\mathbf{s}_I) \leq 2^{\epsilon_s}/p$ for all s (with ϵ_s defined in the theorem statement). This completes the proof sketch. □

An immediate consequence of the above results is the following.

Corollary 1. *For any (t,n) and $t' > t$, the standard Shamir (t,n)-threshold secret-sharing scheme* ShaTSS *is asymptotically threshold-changeable to $(Int(t' - t'/t), t')$ with respect to the uniform secret distribution.*

5 Conclusions

We presented a new cryptographic application of lattice reduction techniques to achieve threshold-changeability for the standard Shamir (t,n)-threshold scheme. We proved concrete bounds on the correctness and security of our method, making use of fundamental results from lattice theory in our analysis.

Acknowledgements. We would like to thank Scott Contini and Igor Shparlinski for helpful discussions and encouragement to work on this problem. This work was supported by ARC Discovery Grants DP0345366 and DP0451484.

References

1. M. Ajtai, R. Kumar, and D. Sivakumar. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *Proc. 33-rd ACM Symp. on Theory of Comput.*, pages 601–610, New York, 2001. ACM Press.
2. C. Asmuth and J. Bloom. A Modular Approach to Key Safeguarding. *IEEE Trans. on Information Theory*, 29:208–210, 1983.
3. L. Babai. On Lovasz' Lattice Reduction and the Nearest Lattice Point Problem. *Combinatorica*, 6, 1986.
4. C. Blundo, A. Cresti, A. De Santis, and U. Vaccaro. Fully Dynamic Secret Sharing Schemes. In *CRYPTO '93*, volume 773 of *LNCS*, pages 110–125, Berlin, 1993. Springer-Verlag.
5. Y. Desmedt and S. Jajodia. Redistributing Secret Shares to New Access Structures and Its Application. Technical Report ISSE TR-97-01, George Mason University, 1997.
6. O. Goldreich, D. Ron, and M. Sudan. Chinese Remaindering with Errors. *IEEE Transactions on Information Theory*, 46:1330–1338, 2000.
7. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1993.
8. P. Gruber and C. Lekkerkerker. *Geometry of Numbers*. Elsevier Science Publishers, 1987.
9. V. Guruswami and M. Sudan. Improved Decoding of Reed-Solomon Codes and Algebraic-Geometric Codes. *IEEE Trans. Inf. Th.*, 45:1757–1767, Sep. 1999.
10. E. Hlawka, J. Schoiβengeier, and R. Taschner. *Geometric and Analytic Number Theory*. Springer-Verlag, 1991.
11. R. Kannan. Algorithmic Geometry of Numbers. *Annual Review of Comp. Sci.*, 2:231–267, 1987.
12. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261:515–534, 1982.
13. A. Maeda, A. Miyaji, and M. Tada. Efficient and Unconditionally Secure Verifiable Threshold Changeable Scheme. In *ACISP 2001*, volume 2119 of *LNCS*, pages 402–416, Berlin, 2001. Springer-Verlag.
14. K. Martin. Untrustworthy Participants in Secret Sharing Schemes. In *Cryptography and Coding III*, pages 255–264. Oxford University Press, 1993.
15. K. Martin, J. Pieprzyk, R. Safavi-Naini, and H. Wang. Changing Thresholds in the Absence of Secure Channels. *Australian Computer Journal*, 31:34–43, 1999.
16. K. Martin, R. Safavi-Naini, and H. Wang. Bounds and Techniques for Efficient Redistribution of Secret Shares to New Access Structures. *The Computer Journal*, 8, 1999.
17. M. Quisquater, B. Preneel, and J. Vandewalle. On the Security of the Threshold Scheme Based on the Chinese Remainder Theorem. In *PKC 2002*, volume 2274 of *LNCS*, pages 199–210, Berlin, 2002. Springer-Verlag.
18. A. Shamir. How To Share a Secret. *Comm. of the ACM*, 22:612–613, 1979.

19. M.A. Shokrollahi and H. Wasserman. List Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 45:432–437, March 1999.
20. I.E. Shparlinski. Sparse Polynomial Approximation in Finite Fields. In *Proc. 33rd STOC*, pages 209–215, New York, 2001. ACM Press.
21. I.E. Shparlinski and R. Steinfeld. Noisy Chinese Remaindering in the Lee Norm. *Journal of Complexity*, 20:423–437, 2004.
22. R. Steinfeld, J. Pieprzyk, and H. Wang. Dealer-Free Threshold Changeability for Standard CRT Secret-Sharing Schemes. Preprint, 2004.

Masking Based Domain Extenders for UOWHFs: Bounds and Constructions

Palash Sarkar

Cryptology Research Group,
Applied Statistics Unit,
Indian Statistical Institute,
203, B.T. Road,
Kolkata 700108, India
palash@isical.ac.in

Abstract. We study the class of masking based domain extenders for UOWHFs. Our first contribution is to show that any correct masking based domain extender for UOWHF which invokes the compression UOWHF s times must use at least $\lceil \log_2 s \rceil$ masks. As a consequence, we obtain the key expansion optimality of several known algorithms among the class of *all* masking based domain extending algorithms. Our second contribution is to present a new parallel domain extender for UOWHF. The new algorithm achieves asymptotically optimal speed-up over the sequential algorithm and the key expansion is almost everywhere optimal, i.e., it is optimal for almost all possible number of invocations of the compression UOWHF. Our algorithm compares favourably with all previously known masking based domain extending algorithms.

Keywords: UOWHF, domain extender, parallel algorithm.

1 Introduction

A universal one-way hash function (UOWHF) is a function family $\{h_k\}_{k \in \mathcal{K}}$ with $h_k : \{0,1\}^n \rightarrow \{0,1\}^m$, for which the following task of the adversary is computationally infeasible: the adversary chooses an n-bit string x, is then given a k chosen uniformly at random from $\mathcal{K}$ and has to find a x' such that $x \not= x'$ and $h_k(x) = h_k(x')$. The notion of UOWHF was introduced in [9].

Intuitively, a UOWHF is a weaker primitive than a collision resistant hash function (CRHF), since the adversary has to commit to the string x before knowing the actual hash function h_k for which a collision has to be found. In fact, Simon [15] has shown that there is an oracle relative to which UOWHFs exist but CRHFs do not exist. Further, as pointed out in [1], the birthday paradox does not apply to the UOWHF and hence the message digest can be smaller. Thus a construction for UOWHF may be faster than a construction for CRHF.

There is a second and perhaps a more important reason to prefer UOWHF over CRHF. A protocol built using a UOWHF maybe "more" secure than a protocol built using CRHF. The intuitive reason being that even if it is possible

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 187–200, 2004.

to find a collision for a hash function, it might still be difficult to find a collision for it when considered as a UOWHF. This situation is nicely summed up in [1]: "Ask less of a hash function and it is less likely to disappoint!"

The important paper by Bellare and Rogaway [1] provides the foundation for the recent studies on UOWHFs. They introduce the notion of domain extender for UOWHF; show that the classical Merkle-Damgård algorithm does not work for UOWHFs; provide several new constructions for UOWHF domain extenders and finally provide a secure digital signature scheme based on UOWHF in the hash-then-sign paradigm.

The study in [1] shows that extending the domain usually requires an associated increase in key length. One of the major new ideas behind their domain extending algorithm is "masking" the outputs of intermediate invocations by random strings. This idea of masking based algorithms have been later pursued by several authors [14, 3, 13, 12, 8, 7]. We would like to point out that [1] also presents other (i.e., non-masking type) techniques for domain extension. However, the key expansion for these techniques is more than the masking type techniques. Consequently, subsequent work, including the current one, have concentrated only on masking type domain extenders.

Our Contributions: The contribution of this paper is twofold.

We start by formalizing the class $\boldsymbol{\mathcal{A}}$ of *all* masking based domain extending algorithms. This class includes all known efficient domain extending algorithms [14, 3, 13, 12, 8, 7]. Any masking based algorithm in $\boldsymbol{\mathcal{A}}$ proceeds by XORing the output of any intermediate invocation of the compression UOWHF by a fixed length bit string called a mask.

Suppose $\{h_k\}_{k\in\mathcal{K}}$, $h_k : \{0,1\}^n \rightarrow \{0,1\}^m$ is a compression UOWHF whose domain is to be extended. Further, suppose that an algorithm in $\boldsymbol{\mathcal{A}}$ makes s invocations of h_k (for some $k \in \mathcal{K}$) and uses a total of ρ masks. In Proposition 1, we show that the length of any string in the extended domain is $n+(s-1)(n-m)$. The resulting amount of key expansion is ρm and hence the key expansion is totally determined by the number of masks.

Our main result on class $\boldsymbol{\mathcal{A}}$ is to obtain a necessary condition for any algorithm in $\boldsymbol{\mathcal{A}}$ to be a correct domain extending algorithm. Using this necessary condition, we obtain a non-trivial lower bound on the number of masks used by any correct algorithm in $\boldsymbol{\mathcal{A}}$. More precisely, in Theorem 1, we show that $\rho \geq \lceil \log_2 s \rceil$. Based on this lower bound, we define the masking efficiency, ME of an algorithm which uses ρ masks and makes s invocations of the compression UOWHF to be $\mathsf{ME} = \rho - \lceil \log_2 s \rceil$. In the case $\mathsf{ME} = 0$, we say that the algorithm achieves optimal masking. Our lower bound immediately shows the masking optimality of the sequential algorithm of Shoup [14] and the parallel algorithms of [3, 7].

The basic unit of operation of a domain extending algorithm is one invocation of the compression UOWHF. The number of operations made by any sequential algorithm is equal to the number of invocations of the compression UOWHF. On the other hand, in a parallel algorithm, several invocations of the compression

UOWHF is done in parallel and thus the number of parallel rounds will be lower. Suppose an algorithm makes s invocations of the compression UOWHF and uses N_p processors to complete the computation in N_r rounds. Since there are s invocations and N_p processors, at least $\lceil s/N_p \rceil$ parallel rounds will be required and hence $N_r \geq \lceil s/N_p \rceil$. We define the parellelism efficiency, PE to be equal to s/N_r. In general, $\mathsf{PE} \leq N_p$ and in the case $\mathsf{PE} = N_p$ we say that the algorithm achieves optimal parallelism.

Our second contribution is to obtain a parallel domain extending algorithm. The basic idea of the algorithm is to divide the input message into several parts, hash each part separately and then combine the different parts using a binary tree. This idea has already been suggested for collision resistant hash functions by Damgård in [2]. Our contribution is to add a suitable masking strategy. The result is a simple and efficient parallel domain extending algorithm for UOWHF. The masking efficiency ME is almost always zero and in very few cases it is one. Hence we say that the masking efficiency of our algorithm is almost always optimal. Further, the parallelism efficiency PE is asymptotically optimal. Thus our algorithm provides a satisfactory parallel domain extender for UOWHF and to a certain extent completes the line of research on obtaining efficient domain extenders for UOWHFs which was started in [1].

Related Work: We have already mentioned that UOWHF was introduced by Naor and Yung [9] and the important work done by Bellare and Rogaway [1]. There are several direct constructions of UOWHFs based on general assumptions [10, 4]. However, as noted in [1] these are not very efficient. Subsequent to the work in [1], Shoup [14] described a nice domain extending algorithm which is a modification of the Merkle-Damgård construction. Shoup's algorithm is a sequential algorithm and Mironov [6] proved that the algorithm achieves minimal key length expansion among all *sequential* masking based domain extending algorithms. (As opposed to this, our lower bound shows that Shoup's algorithm is optimal among *all* masking based domain extending algorithms.) Later work [13, 8, 3, 12, 7] have provided different parallel constructions of domain extending algorithms with varying trade-off between degree of parallelism and key length expansion. These are summarized in Tables 1 and 2.

We note that none of the previous constructions simultaneously achieve optimal parallelism and optimal key expansion. (In [7], it is claimed that their algorithm achieves optimal parallelism. This claim is not correct: In [7], $s = 2^T$ and the number of parallel rounds is $N_r = T + 1$. This requires $N_p = 2^{T-1}$ and hence $\mathsf{PE} \approx N_p / \log_2 N_p$; as explained above, for optimal parallelism we should have $\mathsf{PE} = N_p$.)

Note that the algorithms in [1, 13, 8, 7, 3] can also be executed with a fixed number of processors by a level-by-level simulation of the large binary tree. However, this simulation will require storing the results of all the invocations at one level and hence will push up the memory requirement. In contrast, for our algorithm, the required number of buffers is exactly equal to the number of processors.

Table 1. Comparison of masking efficiency. Here s is the number of invocations of the compression UOWHF

construction	[1]	[13]	[8]	[12]	[14, 3, 7]	ours
ME	$\approx \log_2 s$	$\approx \log_2 \log_2 s$	$O(\log^* s)$	const	0	0 or $1^\dagger$

†: the value is almost always 0.

Table 2. Comparison of parallelism efficiency. Here N_p is the number of processors

construction	[1, 13, 8, 7]	[3]	[12], ours
PE	$\approx \frac{N_p}{\log N_p}$	$\approx N_p^{1-1/l}$, l const.	$\approx N_p$

In [7], a sufficient condition for the correctness of any algorithm in $\boldsymbol{\mathcal{A}}$ is presented. Essentially, this condition states that, if, for any subtree, there is at least one mask which occurs *exactly once* in that subtree, then the construction is correct. In contrast, our necessary condition states that for any correct construction, for any subtree, there must be at least one mask which occurs an *odd number of times*. Though these two combinatorial conditions are close, they are not the same and they have not yet been proved to be equivalent. In fact, it is also possible that they *cannot* be proved to be equivalent.

Our necessary condition yields a tight lower bound on the number of masks, whereas the sufficient condition in [7] is used to verify the correctness of some previous constructions. However, it is not easy to apply the sufficient condition of [7] to prove the correctness of the construction in [12] and the construction presented here. On the other hand, for small examples, it is possible to verify that both the construction in [12] and the one presented here satisfy the sufficient condition of [7]. Thus our necessary condition and the sufficient condition of [7] are actually different and are of separate interest. It could be an interesting research problem to obtain a single necessary and sufficient condition for correct domain extension for any algorithm in $\boldsymbol{\mathcal{A}}$.

The rest of the paper is organized as follows. In Section 2, we describe the necessary preliminaries. In Section 3, we describe the formal model for masking based domain extenders and study its properties. In this section, we also obtain the necessary condition and the lower bound on the number of masks. The new construction of a parallel domain extending algorithm is described in Section 4. Finally, Section 5 concludes the paper. Due to lack of space, most of the proofs are omitted. These can be found in the full version of the paper and in the technical report [11].

2 Preliminaries

All logarithms in this paper are to the base 2. The notation $x \in_r A$ denotes the (uniformly at) random choice of the element x from the set A. Also $\boldsymbol{\lambda}$

denotes the empty string. By an (n, m) function f we will mean a function $f : \{0,1\}^n \to \{0,1\}^m$. A formal definition for UOWHF is given in [9]. In this paper we will be interested in "securely" extending the domain of a given UOWHF. Our proof technique will essentially be a reduction. We formalize this as a reduction between two suitably defined problems.

Let $\mathbf{F} = \{h_k\}_{k \in \mathcal{K}}$ be a keyed family of hash functions, where each h_k is an (n, m) function, with $n > m$. Consider the following adversarial game $\mathcal{G}(\mathbf{F})$ for the family $\mathbf{F}$.

1. Adversary chooses an $x \in \{0,1\}^n$.
2. Adversary is given a k which is chosen uniformly at random from $\mathcal{K}$.
3. Adversary has to find x' such that $x \neq x'$ and $h_k(x) = h_k(x')$.

The problem $\mathcal{G}(\mathbf{F})$-UOWHF for the family $\mathbf{F}$ is to win the game $\mathcal{G}(\mathbf{F})$.

A strategy $\mathcal{A}$ for the adversary runs in two stages. In the first stage $\mathcal{A}^{\text{guess}}$, the adversary finds the x to which he has to commit in Step 1. It also produces some auxiliary state information state. In the second stage $\mathcal{A}^{\text{find}}(x, k, \mathsf{state})$, the adversary either finds a x' which provides a collision for h_k or it reports failure. Both $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(x, k, \mathsf{state})$ are probabilistic algorithms. The success probability of the strategy is measured over the random choices made by $\mathcal{A}^{\text{guess}}$ and $\mathcal{A}^{\text{find}}(x, k, \mathsf{state})$ and the random choice of k in Step 2 of the game. We say that $\mathcal{A}$ is an (ϵ, q)-strategy for $\mathcal{G}(\mathbf{F})$-UOWHF if the success probability of $\mathcal{A}$ is at least ϵ and it invokes some hash function from the family $\mathbf{F}$ at most q times. Informally, we say that $\mathbf{F}$ is a UOWHF if there is no "good" winning strategy for the game $\mathcal{G}(\mathbf{F})$.

In this paper, we are interested in extending the domain of a UOWHF. Let $\mathbf{F} = \{h_k\}_{k \in \mathcal{K}}$, where each h_k is an (n, m) function. For $i \geq 1$, let $n_i = n + (i - 1)(n - m)$. Define $\mathbf{F}_0 = \mathbf{F}$ and for $i > 0$, define $\mathbf{F}_i = \{H_{p_i}\}_{p_i \in \mathcal{P}_i}$, where each H_{p_i} is an (n_i, m) function. The family $\mathbf{F}_i$ is built from the family $\mathbf{F}$. In fact, as shown in Proposition 1, a function in $\mathbf{F}_i$ is built using exactly i invocations of some function in $\mathbf{F}$.

We consider the problem $\mathcal{G}(\mathbf{F}_i)$-UOWHF. We say that the adversary has an (ϵ, q)-strategy for $\mathcal{G}(\mathbf{F}_i)$-UOWHF if there is a strategy $\mathcal{B}$ for the adversary with probability of success at least ϵ and which invokes some hash function from the family $\mathbf{F}$ at most q times. Note that $\mathbf{F}_i$ is built using $\mathbf{F}$ and hence while studying strategies for $\mathcal{G}(\mathbf{F}_i)$ we are interested in the number of invocations of hash functions from the family $\mathbf{F}$.

The correctness of our construction will essentially be a Turing reduction. We will show that if there is an (ϵ, q)-strategy for $\mathcal{G}(\mathbf{F}_i)$, then there is an (ϵ_1, q_1)-strategy for $\mathbf{F}_i$, where ϵ_1 is not "significantly" less than ϵ and q_1 is not "significantly" more than q. In fact, we will have $\epsilon_1 = \epsilon/i$ and $q_1 = q + 2i$. Since $\mathbf{F}_i$ invokes a hash function from $\mathbf{F}$ a total of i times, we "tolerate" a reduction in success probability by a factor of $1/i$. (This is also true for other constructions such as in [1].) The intuitive interpretation of the reduction is that if $\mathbf{F}$ is a UOWHF then so is $\mathbf{F}_i$ for each $i \geq 1$.

The key length for the base hash family $\mathbf{F}$ is $\lceil \log |\mathcal{K}| \rceil$. On the other hand, the key length for the family $\mathbf{F}_i$ is $\lceil \log |\mathcal{P}_i| \rceil$. Thus increasing the size of the

input from n bits to n_i bits results in an increase of the key size by an amount $\lceil \log |\mathcal{P}_i| \rceil - \lceil \log |\mathcal{K}| \rceil$.

3 Lower Bound on Key Expansion

In this section, we consider the problem of minimising the key expansion while securely extending the domain of a UOWHF. More precisely, we are interested in obtaining a lower bound on key length expansion. Obtaining a complete answer to this problem is in general difficult. Thus we adopt a simpler approach to the problem. We fix a class of possible domain extending algorithms and obtain a lower bound on key expansion for any algorithm in this class. (Note that in computer science, this is the usual approach for proving lower bounds on algorithmic problems. For example, the lower bound of $O(n \log n)$ for sorting n elements is obtained for the class of all *comparison based algorithms.*)

The usefulness of our lower bound depends on the class of algorithms that we consider. The class that we consider consists of all masking based domain extending algorithms. (We make this more precise later.) All previously known masking based algorithms [1, 14, 13, 12, 8, 3, 7] belong to this class. We consider this to be ample evidence for the usefulness of our lower bound. However, we would like to point out that our lower bound does not hold for *any* domain extending algorithm. Thus it might be possible to achieve lower key expansions. However, any such algorithm must adopt an approach which is different from masking based algorithms. One possible approach could be to develop the technique of using separate keys for the compression functions (see [1]).

Let $\mathbf{F} = \{h_k\}_{k \in \mathcal{K}}$, where each h_k is an (n, m) function. We are interested in the class $\boldsymbol{\mathcal{A}}$ of masking based domain extension algorithms. We do not want the algorithm to be dependent on the structure of the UOWHF; in fact it should work for all UOWHF's which can "fit" into the structure of the algorithm. Any algorithm $A \in \boldsymbol{\mathcal{A}}$ behaves in the following manner.

1. It invokes some function $h_k \in \mathbf{F}$ a finite number of times.
2. The outputs of all but one invocation of $h_k()$ is masked by XORing with an m-bit string selected from the set $\{\mu_0, \ldots, \mu_{\rho-1}\}$.
3. The invocations of $h_k()$ whose outputs are masked are called *intermediate* invocations and the invocation whose output is not masked is called the *final* invocation.
4. The entire output of any intermediate invocation is fed into the input of exactly one separate invocation of $h_k()$.
5. Each bit of the message x is fed into exactly one invocation of $h_k()$.
6. The output of the final invocation is the output of A.

We emphasize that all previously known masking based algorithms [1, 14, 13, 12, 8, 3, 7] belong to $\boldsymbol{\mathcal{A}}$. In the following we make a general study of any algorithm in $\boldsymbol{\mathcal{A}}$, with particular emphasis on obtaining a lower bound on key expansion made by any algorithm in $\boldsymbol{\mathcal{A}}$.

Proposition 1. *Let $A \in \boldsymbol{\mathcal{A}}$ be such that A invokes $h_k()$ a total of s times. Then the length of the message which is hashed is equal to $n + (s-1)(n-m)$.*

Thus the number of invocations of $h_k()$ and the parameters n and m determine the length of the message to be hashed irrespective of the actual structure of the algorithm. Hence any algorithm $A \in \boldsymbol{\mathcal{A}}$ which invokes $h_k()$ a total of s times defines a family $\mathbf{F}^{(A,s)} = \{H_p^{(A,s)}\}_{p \in \mathcal{P}}$, where $\mathcal{P} = \{0,1\}^{|k|+m\rho}$ and each $H_p^{(A,s)}$ is an $(n + (n-m)(s-1), m)$ function. The structure of any algorithm $A \in \boldsymbol{\mathcal{A}}$ which makes s invocations of $h_k()$ is described by a labelled directed graph $D_s^A = (V_s, E_s, \psi_s)$, where

1. $V_s = \{v_1, \ldots, v_s\}$, i.e., there is a node for each invocation of $h_k()$.
2. $(v_i, v_j) \in E_s$ if and only if the output of the ith invocation is fed into the input of the jth invocation.
3. ψ_s is a map $\psi : E_s \rightarrow \{\mu_0, \ldots, \mu_{\rho-1}\}$, where $\psi(v_{i_1}, v_{i_2}) = \mu_j$ if the output of the i_1-th invocation of $h_k()$ is masked using μ_j.

The nodes corresponding to the intermediate invocations are called intermediate nodes and the node corresponding to the final node is called the final node. Without loss of generality we assume the final node to be v_s. Nodes with indegree zero are called leaf nodes and the others are called internal nodes. Define $\delta(D_s^A) = \max\{\mathsf{indeg}(v) : v \in V_s\}$. We call $\delta(D_s^A)$ to be the fan-in of algorithm A for s invocations.

Proposition 2. *The outdegree of any intermediate node in D_s^A is 1 and the outdegree of the final node is 0. Hence there are exactly $(s-1)$ arcs in D_s^A. Consequently, D_s^A is a rooted directed tree where the final node is the root of D_s^A.*

Proposition 3. *If $\delta = \delta(D_s^A)$, then $n \geq \delta m$.*

Thus an algorithm A with fan-in δ cannot be used with all UOWHFs. The value of fan-in places a restriction on the values of n and m. However, given this restriction the actual structure of D_s^A does not depend on the particular family $\mathbf{F}$.

Let T be a non-trivial subtree of D_s^A. Denote by $\mathsf{vec}_\psi(T)$ the ρ-tuple

$$(\mathsf{num}_{\mu_0}(T) \bmod 2, \ldots, \mathsf{num}_{\mu_{\rho-1}}(T) \bmod 2),$$

where $\mathsf{num}_{\mu_i}(T)$ is the number of times the mask μ_i occurs in the tree T. We say that D_s^A is *null-free* if $\mathsf{vec}\psi(T) \not= (0 \ldots, 0)$ for each non-trivial subtree of D_s^A.

We now turn to the task of obtaining a lower bound on key expansion made by any algorithm A in $\boldsymbol{\mathcal{A}}$. This consists of two tasks. Firstly, we show that for any "correct UOWHF preserving domain extender" A which invokes some function from the compression UOWHF exactly s times, the DAG D_s^A must be null-free. This translates the problem into a combinatorial one. Our second task is to use use this combinatorial property to obtain the required lower bound.

The intuitive idea behind the first part is as follows. Given D_s^A and a family $\mathbf{F}'$ with suitable parameters, we construct a family $\mathbf{F}$ such that if $\mathbf{F}'$ is a UOWHF,

then so is $\mathbf{F}$. Then we extend the domain of $\mathbf{F}$ using D_s^A to obtain the family $\mathbf{F}^{(A,s)}$ and show that if D_s^A is not null-free then it is possible to exhibit a collision for every function in $\mathbf{F}^{(A,s)}$. Now we argue as follows. If $\mathbf{F}'$ is a UOWHF and A is correct for s invocations, then $\mathbf{F}^{(A,s)}$ must also be a UOWHF and hence D_s^A must be null-free. This intuitive argument is now formalized in terms of reductions.

Let $A \in \boldsymbol{\mathcal{A}}$ and D_s^A be the DAG corresponding to s invocations of the compression family by A. We set $\delta = \delta(D_s^A)$. Let $\mathbf{F}' = \{h'_k\}_{k\in\mathcal{K}}$ where each h' is an (n, m') function with $\mathcal{K} = \{0,1\}^K$, $m = m' + K$ and $n = \delta m + \delta + 1$. For $z \in \{0,1\}^n$ write

$$z = z_{1,1}||z_{1,2}||z_{2,1}||z_{2,2}||\ldots||z_{i,1}||z_{i,2}||\ldots||z_{\delta,1}||z_{\delta,2}||y||b$$

where $|z_{i,1}| = m$, $|z_{i,2}| = K$ for $1 \leq i \leq \delta$, $|y| = \delta$ and $b \in \{0,1\}$. We write $y = y(z)$ and $b = b(z)$ to show the dependence of y and b on z. Given $z \in \{0,1\}^n$, define $\mathsf{KLst} = \{z_{1,2}, z_{2,2}, z_{3,2}, \ldots, z_{\delta,2}\}$. Given $z \in \{0,1\}^n$ and $k \in \mathcal{K}$, define a Boolean function $\phi(z,k)$ to be true (T) if and only if $k = \bigoplus_{w\in S} w$ for some $\emptyset \not= S \subseteq \mathsf{KLst}(z)$. We define the family of functions $\mathbf{F} = \{h_k\}_{k\in\mathcal{K}}$, where each h_k is an (n, m) function in the following manner.

$$\left.\begin{array}{lll} h_k(z) = h'_k(z)||k & \text{if } b = 1 \text{ and } \phi(z,k) = \mathsf{F}; \\ \quad = h'_k(z)||0^K & \text{if } b = 0, y = 0^\delta \text{ and } \phi(z,k) = \mathsf{F}; \\ \quad = h'_k(z)||S_y & \text{if } b = 0, y \not= 0^\delta \text{ and } \phi(z,k) = \mathsf{F}; \\ \quad = 1^m & \phi(z,k) = \mathsf{T}. \end{array}\right\} \quad (1)$$

Here $y = y(z)$ and $S_y = \oplus_{y_i=1} z_{i,2}$, i.e., the XOR's of the $z_{i,2}$'s for which the ith bit of y is 1.

Proposition 4. *Suppose there is an (ϵ, q)-strategy for $\mathcal{G}(\mathbf{F})$. Then there is an $(\epsilon - \frac{1}{2^K}, q)$-strategy for $\mathcal{G}(\mathbf{F}')$.*

Intuitively, this means that if $\mathbf{F}'$ is a UOWHF, then so is $\mathbf{F}$. In the next result we show that if D_s^A is not null-free, then it is possible to exhibit a collision for each function in $\mathbf{F}^{(A,s)}$.

Lemma 1. *Let $A \in \boldsymbol{\mathcal{A}}$ and $\mathbf{F}$ be defined as in (1). For $s > 0$, let $\mathbf{F}^{(A,s)}$ be the family obtained by extending the domain of $\mathbf{F}$ using D_s^A. If D_s^A is not null-free, then it is possible to define two strings x, x' such that $x \not= x'$ and $H_p^{(A,s)}(x) = H_p^{(A,s)}(x')$ for any $H_p^{(A,s)} \in \mathbf{F}^{(A,s)}$,*

We now translate Lemma 1 into a lower bound on the number of masks.

Definition 1. *Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be two subtrees of D_s^A. We denote by $T_1 \Delta T_2$ the subtree of D_s^A induced by the set of arcs $E_1 \Delta E_2$, where $E_1 \Delta E_2$ is the symmetric difference between E_1 and E_2.*

Definition 2. *Let $\mathcal{F}$ be a family of non-trivial subsets of D_s^A such that for any $T_1, T_2 \in \mathcal{F}$, the tree $T_1 \Delta T_2$ is also a non-trivial subtree of D_s^A. We call $\mathcal{F}$ a connected family of D_s^A.*

Lemma 2. *Let D_s^A be null-free and let $\mathcal{F}$ be a connected family of D_s^A. Then*

1. *For any $T \in \mathcal{F}$, $\mathsf{vec}_\psi(T) \not= (0, \ldots, 0)$.*
2. *For any $T_1, T_2 \in \mathcal{F}$, $\mathsf{vec}_\psi(T_1) \not= \mathsf{vec}_\psi(T_2)$.*

Consequently, $2^\rho - 1 \geq |\mathcal{F}|$ or equivalently $\rho \geq \lceil \log_2(|\mathcal{F}|+1) \rceil$, where ρ is the number of masks used by A for s invocations.

Lemma 2 provides a lower bound on the number of masks in terms of sizes of connected families. Thus the task is to find a connected family of maximum size in D_s^A. We show the existence of a connected family of size $(s-1)$ in D_s^A. For each intermediate node $v \in D_s^A$, let P_v be the path from v to the final node of D_s^A. Define $\mathcal{F} = \{P_v : v \text{ is an intermediate node in } D_s^A\}$. It is easy to check that $\mathcal{F}$ is a connected family of size $(s-1)$. Hence we have the following result.

Theorem 1. *Let $s > 0$ and $A \in \boldsymbol{\mathcal{A}}$ be correct for s invocations. Then the number of masks required by A is at least $\lceil \log_2 s \rceil$.*

The bound in Theorem 1 is tight since Shoup's algorithm [14] meets this bound with equality. This also shows that Shoup's algorithm is optimal for the class $\boldsymbol{\mathcal{A}}$. Also we would like to point out that the lower bound of Theorem 1 can be improved for particular algorithms.

Lemma 3. *Suppose D_s^A is the full binary tree on $s = 2^t - 1$ nodes. If $t = 2$, there is a connected family of size 3 in D_s^A and for $t \geq 3$, there is a connected family of size $5 \times 2^{t-2} - 2$ in D_s^A. Consequently, $\rho \geq 2$ for $t = 2$ and $\rho \geq t + 1$ for $t \geq 3$.*

4 New Construction

For $t > 0$, let $\mathcal{T}_t$ be the binary tree defined as $\mathcal{T}_t = (V_t = \{P_0, \ldots, P_{2^t-2}\}, A_t)$, where $A_t = \{(P_{2j+1}, P_j), (P_{2j+2}, P_j) : 0 \leq j \leq 2^{t-1}-2\}$. The underlying digraph for our algorithm is a binary tree with sequential paths terminating at the leaf nodes of the tree. We define a digraph $\mathcal{G}_{t,i}$ which consists of the full binary tree $\mathcal{T}_t$ alongwith a total of i nodes on the sequential paths. The precise definition of $\mathcal{G}_{t,i} = (V_{t,i}, A_{t,i})$ is

$$\left.\begin{array}{rl} V_{t,i} &= V_t \cup \{Q_0, \ldots, Q_{i-1}\} \\ A_{t,i} &= A_t \cup \{(Q_j, P_{2^{t-1}+j-1}) : 0 \leq j \leq 2^{t-1}-1\} \\ & \quad \cup \{(Q_j, Q_{j-2^{t-1}}) : 2^{t-1} \leq j \leq i-1\}. \end{array}\right\} \tag{2}$$

The total number of nodes in $\mathcal{G}_{t,i}$ is equal to $2^t - 1 + i$, where $2^t - 1$ nodes are in the binary tree part and i nodes are in the sequential part. We define parameters $r_{t,i}$ and $s_{t,i}$ (or simply r and s) in the following manner: If $i = 0$, then $r = s = 0$; if $i > 0$, then r and s are defined by the equation:

$$i = r2^{t-1} + s \tag{3}$$

where s is a unique integer from the set $\{1, \ldots, 2^{t-1}\}$. For $i > 0$, we can write $i = (r+1) \times s + (2^{t-1} - s)r$. Thus in $\mathcal{G}_{t,i}$ there are s sequential paths of length

$(r+1)$ each and these terminate on the left most s leaf nodes of $\mathcal{T}_t$. There are also $(2^{t-1}-s)$ sequential paths of length r each and these terminate on the other $(2^{t-1}-s)$ leaf nodes of $\mathcal{T}_t$. Figure 1 shows $\mathcal{G}_{4,19}$.

We define $\rho_{t,i}$ or (simply ρ) to be the maximum length (counting only Q nodes) of a path from a Q-node to a P-node. Hence $\rho = 0$ if $i = 0$ and $\rho = r+1$ if $i > 0$.

When $i = 0$, $\mathcal{G}_{t,i}$ is simply the full binary tree $\mathcal{T}_t$ and when $t = 1$, $\mathcal{G}_{t,i}$ is a dipath of length $r+1$. These are the two extreme cases – one leading to a full binary tree and the other leading to a single dipath. In practical applications, t will be fixed and there will be "long" dipaths terminating on the leaf nodes of $\mathcal{T}_t$. For implementation purpose, the number of processors required is 2^{t-1}. Hence for practical applications, the value of $t \leq 5$.

Remark: The idea of breaking a message into parts, hashing them independently and finally combining the outputs is present in Damgård [2] in the context of collision resistant hash functions. The current construction can be seen as a development of the "UOWHF version" of this idea.

4.1 Notation

We define a few notation for future reference.

1. t is the number of levels in the binary tree $\mathcal{T}_t$.
2. i is the total number of nodes in the sequential part of the algorithm.
3. r and s are as defined in (3).
4. $\rho = 0$ if $i = 0$ and $\rho = r+1$ if $i > 0$.
5. $N = 2^t - 1 + i$ is the total number of nodes in $\mathcal{G}_{t,i}$.
6. For $U \in V_{t,i}$, define $\mathsf{nodenum}(U) = j$ if $U = P_j$ and $\mathsf{nodenum}(U) = j + 2^t - 1$ if $U = Q_j$.
7. For $U \in V_{t,i}$, we say that U is a P-node (resp. Q-node) if $U = P_j$ (resp. $U = Q_j$) for some j.

For $U \in V_{t,i}$, we define $\mathsf{indeg}(U)$ (resp. $\mathsf{outdeg}(U)$) to be the indegree (resp. outdegree) of U. Note that other than P_0 each node U has $\mathsf{outdeg}(U) = 1$. Thus for each node $U \not= P_0$ there is a unique out neighbour.

The concept of level is defined in the following manner. There are $L = \rho + t$ levels in $\mathcal{G}_{t,i}$ and the level number of each node is defined as follows.

$$\left.\begin{array}{ll} \mathsf{level}(P_j) = L - 1 - j_1 & \text{if } 2^{j_1} - 1 \leq j \leq 2^{j_1+1} - 2 \text{ and } 0 \leq j_1 \leq t-1; \\ \mathsf{level}(Q_j) = \rho - j_1 - 1 & \text{if } j_1 2^{t-1} \leq j \leq (j_1+1)2^{t-1} - 1 \text{ and } 0 \leq j_1 \leq r-1; \\ \mathsf{level}(Q_j) = 0 & \text{if } r2^{t-1} \leq j \leq r2^{t-1} + s. \end{array}\right\} \quad (4)$$

Note that if $\rho = 0$, there are no Q-nodes and hence the level numbers of Q-nodes are not defined. The root node of $\mathcal{T}_t$ has the highest level. Nodes with indegree zero can be at levels zero and one. Let $U \in V_{t,i}$ and $j = \mathsf{nodenum}(U)$: If $0 \leq j \leq 2^{t-1} - 2$ then we define $\mathsf{lchild}(U) = P_{2j+1}$ and $\mathsf{rchild}(U) = P_{2j+2}$; if $2^{t-1} - 1 \leq j \leq N - 1$, then we define predecessor of U in the following manner:

$$\left.\begin{array}{ll} \mathsf{pred}(U) = Q_{j+2^{t-1}} & \text{if } 2^{t-1} - 1 \leq j \leq 2^{t-1} + i - 2; \\ = \mathsf{NULL} & \text{if } 2^{t-1} + i - 1 \leq j \leq N - 1; \end{array}\right\} \quad (5)$$

For a node U, $\mathsf{pred}(U) = \mathsf{NULL}$ implies that the indegree of U is zero.

4.2 Mask Assignment Algorithm

There are two disjoint sets of masks $\{\alpha_0, \ldots, \alpha_{l-1}\}$ and $\{\beta_0, \ldots, \beta_{t-2}\}$ where $l = \lceil \log(\rho + t) \rceil$. The mask assignment

$$\psi : A_{t,i} \rightarrow \{\alpha_0, \ldots, \alpha_{l-1}\} \cup \{\beta_0, \ldots, \beta_{t-2}\}.$$

is a function from the set of arcs of $\mathcal{G}_{t,i}$ to the set of masks. The definition of ψ is as follows: Let $(U, V) \in A_{t,i}$ with $\mathsf{level}(U) = j - 1$ and $\mathsf{level}(V) = j$ for some $j \in \{1, \ldots, L-1\}$.

- If ((U is a Q-node) or (U is a P-node and $U = \mathsf{lchild}(V)$)),

then $\psi(U, V) = \alpha_{\nu(j)}$.

- If (U is a P-node and $U = \mathsf{rchild}(V)$) then $\psi(U, V) = \beta_{j-(\rho+1)}$.

Here $\nu(j)$ is defined to be the non negative integer j_1 such that $2^{j_1} | j$ and $2^{j_1+1} \not| j$. Also for the convenience of notation we write $\psi(U, V)$ instead of $\psi((U, V))$. The mask assignment for $\mathcal{G}_{4,19}$ is shown in Figure 1.

4.3 Optimality of Mask Assignment

The total number of masks used is equal to $t - 1 + \lceil \log(\rho + t) \rceil$. The total number of nodes in $\mathcal{G}_{t,i}$ is equal to $N = 2^t - 1 + i$. Using Theorem 1, at least $\mathcal{L}_{t,i} = \lceil \log(2^t - 1 + i) \rceil$ masks are required by any algorithm in class $\boldsymbol{\mathcal{A}}$. Our algorithm requires $\mathcal{R}_{t,i} = t - 1 + \lceil \log(\rho + t) \rceil$ masks. Define $\mathcal{D}_{t,i} = \mathcal{R}_{t,i} - \mathcal{L}_{t,i}$. We study $\mathcal{D}_{t,i}$.

Proposition 5.

$$\left.\begin{array}{rl} \mathcal{D}_{t,i} = 0 & \text{if } i = 0 \text{ and } t = 1; \\ = \lceil \log t \rceil - 1 & \text{if } i = 0 \text{ and } t > 1; \\ = \lceil \log(r + 1 + t) \rceil - \left\lceil \log\left(r + 2 + \frac{s-1}{2^{t-1}}\right) \right\rceil & \text{if } i > 0. \end{array}\right\} \quad (6)$$

Furthermore, $\mathcal{D}_{t,i} = 0$ if and only if either $t = 1$; or ($t = 2$ and $i = 0$); or $2^j - 1 - \lceil (s-1)/2^{t-1} \rceil \leq r \leq 2^{j+1} - t - 1$ for some $j > 0$.

For $t = 1$, the mask assignment algorithm reduces to the mask assignment algorithm of Shoup [14] and for $i = 0$, the mask assignment algorithm reduces to the mask assignment algorithm of Sarkar [13]. Hence we concentrate on the case $t > 1$ and $i > 0$. For practical parallel implementation, the value of t will determine the number of processors and will be fixed whereas the value of i can grow in an unbounded manner.

Suppose $2^{\tau-1} < t - 1 \leq 2^\tau$. For $j \geq 0$, define two intervals of integers in the following manner:

$$\left.\begin{array}{l} I_j = \{2^{\tau+j} - 1 - \left\lceil \frac{s-1}{2^{t-1}} \right\rceil, 2^{\tau+j} - \left\lceil \frac{s-1}{2^{t-1}} \right\rceil, \ldots, 2^{\tau+j+1} - t - 1\}; \\ J_j = \{2^{\tau+j+1} - t, 2^{\tau+j+1} - t + 1, \ldots, 2^{\tau+j+1} - 2 - \left\lceil \frac{s-1}{2^{t-1}} \right\rceil\}. \end{array}\right\} \quad (7)$$

Clearly, $|I_j| = 2^{\tau+j} - t + 1 + \lceil (s-1)/2^{t-1} \rceil$, $|J_j| = t - 1 - \lceil (s-1)/2^{t-1} \rceil$ and $|I_j| + |J_j| = 2^{\tau+j}$.

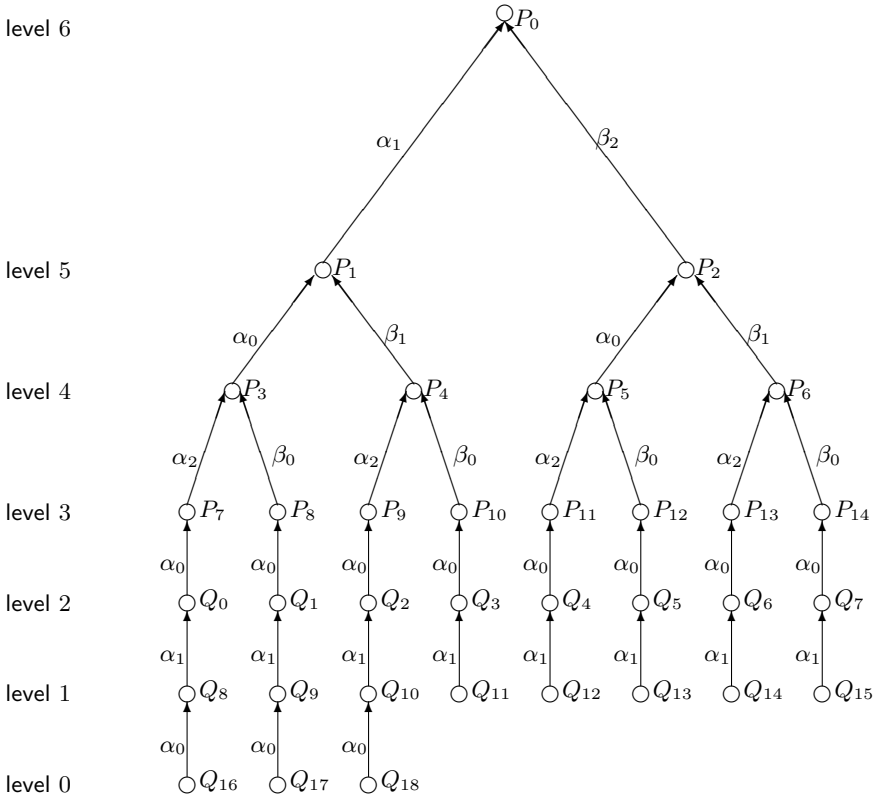


Fig. 1. Example of mask assignment for $t = 4$ and $i = 19$

Theorem 2. *Suppose* $i > 0$, $2^{\tau-1} < t - 1 \leq 2^{\tau}$ *and for* $j \geq 0$, I_j *and* J_j *are as defined in 7. Then* $\mathcal{D}_{t,i} = 0$ *if* $r \in I_j$*; and* $\mathcal{D}_{t,i} = 1$ *if* $r \in J_j$*.*

From Theorem 2, it follows that for $j \geq \tau$, in any interval $2^j + 1 \leq r \leq 2^{j+1}$, there are exactly $t - 1 - \lceil (s-1)/2^{t-1} \rceil$ points where the algorithm is suboptimal with respect to the lower bound. Moreover, at these points it requires exactly one extra mask over the lower bound. In any practical parallel implementation, the value of t will be fixed, whereas the value of r will grow. In such a situation, the ratio $\frac{1}{2^j}(t - 1 - \lceil (s-1)/2^{t-1} \rceil)$ approaches zero very fast and hence we can say that for $t \geq 2$, the algorithm achieves optimal key length expansion *almost everywhere*. (Note that for $t = 1$, the algorithm reduces to Shoup's algorithm and hence achieves optimal key length expansion.)

4.4 Computation of Message Digest

Let $\{h_k\}_{k \in \mathcal{K}}$, where each h_k is an (n, m) function, be the compression UOWHF whose domain is to be extended. *For* $t > 1$, *we require* $n \geq 2m$. The nodes of $\mathcal{G}_{t,i}$ represent the invocations of h_k. Thus h_k is invoked a total of N times.

The output of P_0 is provided as output digest, whereas the outputs of all the other nodes are used as inputs to other invocations as defined by the arcs of $\mathcal{G}_{t,i}$. Using Proposition 1 of [12] we obtain the following: Suppose a message x is hashed using $\mathcal{G}_{t,i}$ and the compression UOWHF $\{h_k\}_{k\in\mathcal{K}}$, where each h_k is an (n, m) function. Then $|x| = N(n-m)+m$.

Thus $\{h_k\}_{k\in\mathcal{K}}$ is extended to $\{H_p^{(t,i)}\}_{p\in\mathcal{P}}$ where each $H_p^{(t,i)}$ is an $(N(n-m)+m, m)$ function and

$$p = k||\alpha_0||\ldots||\alpha_{l-1}||\beta_0||\ldots||\beta_{t-2}.$$

The message x of length $N(n-m)+m$ has to be formatted into small substrings and provided as input to the different invocations of h_k. Write $x = x_0||\ldots||x_{N-1}$, where the lengths of the x_j's are as follows.

$$\left.\begin{array}{ll} |x_j| = n-2m & \text{if } 0 \le j \le 2^{t-1}-2; \\ = n-m & \text{if } 2^{t-1}-1 \le j \le 2^{t-1}+i-2; \\ = n & \text{if } 2^{t-1}+i-1 \le j \le 2^t+i-2. \end{array}\right\} \quad (8)$$

The substring x_j is provided as input to node U with $\mathsf{nodenum}(U) = j$ and the m-bit output of U is denoted by z_j. The outputs $z_1, \ldots, z_{N-1}$ are masked using the α and β masks to obtain m-bit strings $y_1, \ldots, y_{N-1}$ in the following manner.

$$y_j = z_j \oplus \psi(U, V) \text{ if } \mathsf{nodenum}(U) = j. \quad (9)$$

The inputs to the invocations of h_k are formed from the x's and the y's in the following manner. There are N invocations whose inputs are denoted by $w_0, \ldots, w_{N-1}$ and are defined as follows.

$$\left.\begin{array}{ll} w_j = x_j||y_{2j+1}||y_{2j+2} & \text{if } 1 \le j \le 2^{t-1}-2; \\ = x_j||y_{j+2^{t-1}} & \text{if } r > 0 \text{ and } 2^{t-1}-1 \le j \le 2^{t-1}+i-2; \\ = x_j & \text{if } 2^{t-1}+i-1 \le j \le 2^t+i-2. \end{array}\right\} \quad (10)$$

Note that the length of each w_j is n and hence we can invoke h_k on w_j for all $j \in \{0, \ldots, N-1\}$. For any node $U \in V_{t,i}$ we define $x(U), y(U)$ and $w(U)$ to be the x, y and w strings associated to the node U as defined respectively in (8), (9) and (10). Similarly the output of node U will be denoted by $z(U)$.

Now we are ready to describe the digest computation algorithm. Most of the work has already been done, so that the description of the algorithm becomes simple. Suppose the compression UOWHF is $\{h_k\}_{k\in\mathcal{K}}$. We describe the digest computation of $H_p^{(t,i)}(x)$.

Algorithm to compute $H^{(t,i)}(x)$

1. for $j = 0$ to $L-1$ do
2. for all U with $\mathsf{level}(U) = j$ do in parallel
3. compute $z(U) = h_k(w(U))$;
4. end do;
5. end do;
6. return z_0.

5 Conclusion

In this paper, we have formalized the model for masking based domain extending algorithms. Using this formal model, we obtained a lower bound on the minimum amount of key expansion required by any masking based algorithm. Our second contribution has been to develop a simple and efficient parallel domain extender. The key expansion of our algorithm is almost everywhere optimal whereas the efficiency of parallelism is asymptotically optimal.

References

1. M. Bellare and P. Rogaway. Collision-resistant hashing: towards making UOWHFs practical. *Proceedings of Crypto 1997*, pp 470–484.
2. I. B. Damgård. A design principle for hash functions. *Proceedings of Crypto 1989*, Lecture Notes in Computer Science, volume 435 (1990), 416–427.
3. W. Lee, D. Chang, S. Lee, S. Sung and M. Nandi. New Parallel Domain Extenders for UOWHF. *Proceedings of Asiacrypt 2003*, Lecture Notes in Computer Science, pp 208–227.
4. R. Impagliazzo and M. Naor. Efficient Cryptographic Schemes provably as secure as subset sum. *Journal of Cryptology*, volume 9, number 4, 1996.
5. R. C. Merkle. One way hash functions and DES. *Proceedings of Crypto 1989*, Lecture Notes in Computer Science, volume 435, 1990, pp 428–226.
6. I. Mironov. Hash functions: from Merkle-Damgård to Shoup. *Proceedings of Eurocrypt 2001*, Lecture Notes in Computer Science, volume 2045 (2001), Lecture Notes in Computer Science, pp 166–181.
7. M. Nandi. Optimal Domain Extension of UOWHF and a Sufficient Condition. *Proceedings of SAC 2004*, Lecture Notes in Computer Science, to appear.
8. M. Nandi. A New Tree based Domain Extension of UOWHF, Cryptology e-print archive, Report No. `http://eprint.iacr.org, 2003/142`.
9. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic aplications. *Proceedings of the 21st Annual Symposium on Theory of Computing*, ACM, 1989, pp. 33–43.
10. J. Rompel. One-way functions are necessary and sufficient for digital signatures. *Proceedings of the 22nd Annual Symposium on Theory of Computing*, ACM, 1990.
11. P. Sarkar. Masking Based Domain Extenders for UOWHFs: Bounds and Constructions, Cryptology e-print archive, `http://eprint.iacr.org`, Report No. 2003/225.
12. P. Sarkar. Domain Extenders for UOWHFs: A Finite Binary Tree Algorithm, Cryptology e-print archive, `http://eprint.iacr.org`, Report No. 2003/009.
13. P. Sarkar. Construction of UOWHF: Tree Hashing Revisited, Cryptology e-print archive, `http://eprint.iacr.org`, Report No. 2002/058.
14. V. Shoup. A composition theorem for universal one-way hash functions. *Proceedings of Eurocrypt 2000*, pp 445–452, 2000.
15. D. Simon. Finding collisions on a one-way street: Can secure hash function be based on general assumptions?, *Proceedings of Eurocrypt 1998*, Lecture Notes in Computer Science, pp 334–345, 1998.
16. D. R. Stinson. Some observations on the theory of cryptographic hash functions. `http://www.cacr.math.uwaterloo.ca/~dstinson/papers/newhash.ps`.

Higher Order Universal One-Way Hash Functions*

Deukjo Hong[1,**], Bart Preneel[2], and Sangjin Lee[1,***]

[1] Center for Information Security Technologies (CIST),
Korea University, Seoul, Korea
{hongdj, sangjin}@cist.korea.ac.kr
[2] ESAT/SCD-COSIC, Katholieke Universiteit Leuven, Belgium
bart.preneel@esat.kuleuven.ac.be

Abstract. Universal One-Way Hash Functions (UOWHFs) are families of cryptographic hash functions for which first a target input is chosen and subsequently a key which selects a member from the family. Their main security property is that it should be hard to find a second input that collides with the target input. This paper generalizes the concept of UOWHFs to UOWHFs of order r. We demonstrate that it is possible to build UOWHFs with much shorter keys than existing constructions from fixed-size UOWHFs of order r. UOWHFs of order r can be used both in the linear $(r+1)$-round Merkle-Damgård construction and in a tree construction.

Keywords: Hash Function, Collision Resistant Hash Function (CRHF), Universal One-Way Hash Function (UOWHF), Higher Order Universal One-Way Hash Function.

1 Introduction

Since the introduction of the notion of UOWHFs by Naor and Yung in 1989 [5], it is widely believed that UOWHFs form an attractive alternative to CRHFs (Collision Resistant Hash Functions). The main requirement for a UOWHF is that it is hard to find a second preimage. First a challenge input is selected by the opponent, subsequently a key is chosen which selects a member of the class of functions and only after this choice the opponent has to produce a second preimage with the same hash value (for this key) as the challenge input. This should be contrasted to CRHFs, where first a key is selected and subsequently a two colliding inputs need to be found; due to the birthday paradox, a black box approach for a CRHF with an n-bit result takes on average about $2^{n/2}$ queries.

* Supported by Korea University Grant in 2003 year.
** A guest researcher at ESAT/SCD-COSIC, K.U.Leuven from 2003 to 2004.
*** Supported (in part) by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 201–213, 2004.

Simon [10] has demonstrated that a UOWHF is a strictly weaker concept than a CRHF. UOWHFs can replace CRHFs in many applications; even for digital signatures this is feasible, but it should be noted that one becomes vulnerable to attacks by the signers (who can cheat and choose the key before the target message). The concept of UOWHFs has been generalized by Zheng *et al.* [12] and by Mironov [4].

A standard approach to construct hash functions that take input strings of arbitrary length is to start from a compression function that compresses input strings of fixed length. For CRHFs, the Merkle-Damgård construction is a widely used and efficient method [2, 3]. Both authors showed independently that it is sufficient for the hash function to be collision resistant that the compression function is. Damgård also proposed a tree construction. Naor and Yung showed that it is possible in principle to build UOWHFs by composition [5]. However, Bellare and Rogaway showed that even if the compression function is a UOWHF, a 2-round Merkle-Damgård iteration of this function may not be a UOWHF.

Subsequently, provably secure constructions have been developed based on compression functions at the cost of an increase in key length. Bellare and Rogaway [1] propose two types of constructions.

- The first type has a linear structure; two variants of the Merkle-Damgård construction were shown to be secure: the basic linear hash and the XOR linear hash. Later, Shoup improved the XOR linear hash construction. He shows that if one has a fixed size UOWHF which maps n bits to m bits (with $n > m$), one can construct a UOWHF that can hash messages of bit-length $2^t(m-n)+m$ bits to m using a key of bit-length $t \cdot m$ and 2^t applications of the compression function. Mironov has proved that this construction is optimal in terms of key size among linear constructions [4].
- The second type has a tree structure. Here the two constructions with a security proof are the basic tree hash and the XOR tree hash (they extend the work of [5]). XOR tree hash has subsequently been improved further, a.o. by Sarkar [7, 8] and by Lee *et al.* [11], who reduce the key size and extend these structures to higher dimensional parallel constructions.

1.1 Motivation

The special UOWHF made by Bellare and Rogaway [1] loses its universal one-wayness when it is extended to 2-round Merkle-Damgård construction. This example motivated us to study general constructions that work for any UOWHF compression function. It means that the Merkle-Damgård construction cannot be used for extending a universal one-way compression function in general. However, this property does not applied to all UOWHFs. The compression functions of certain UOWHFs may not lose their universal one-wayness until they are extended to 3-round Merkle-Damgård construction. In this case, a 2-round Merkle-Damgård construction based on the compression function can be used as another compression function and so the key size of the whole scheme is reduced by a factor of 2. This lead to promising results, since an important goal of research

on constructions extending UOWHFs has been optimalization of the key size. We began with the Merkle-Damgård construction, but we found that the tree construction has the same problem as the Merkle-Damgård construction.

Intuitively, a UOWHF which does not lose its universal one-wayness until it is extended to 3-round Merkle-Damgård construction is a slightly stronger primitive than Bellare-Rogaway's special UOWHF. More generally, a UOWHF which does not lose its universal one-wayness until it is extended to more round Merkle-Damgård construction is stronger. So, we need new security notions to classify UOWHFs.

1.2 Our Contribution

We define the order of a UOWHF. We can classify UOWHFs according to the order. The classes of UOWHFs of same order form a chain between CRHF and UOWHF classes.

We show in Theorem 1 that if a UOWHF has a higher order, a Merkle-Damgård construction with more rounds based on it becomes a UOWHF. Theorem 3 states that if a UOWHF has a higher order, a tree construction with more levels becomes a UOWHFs. Theorems 1 and 3 are our main results. They consider collisions of the same length only, since we want to use our Merkle-Damgård and tree constructions only as a compression function which plays the role of a building block in the known constructions. Theorems 2 and 4 are generalizations of Theorem 1 and 3 which are mainly of theoretical interest.

1.3 Organization of This Paper

This paper is organized as follows. Section 2 introduces our notation and definitions and presents the counterexample of Bellare and Rogaway. In Sect. 3, our new definition of higher order UOWHFs is introduced. Section 4 and 5 present respectively the Merkle-Damgård and the tree construction based on higher order UOWHFs. Some concluding remarks are made in Sect. 6.

2 Preliminaries

We will follow the notation and computation models in [1].

2.1 Notation

We denote the concatenation of strings x and x' by $x||x'$ or xx'. Σ^n is the set of all strings of bit-length n. We use the notation Σ_n^m instead of Σ^{nm} when we want to stress that each string consists of m blocks of bit-length n. The set of all strings whose lengths are multiple of n is denoted by Σ_n^+.

A hash function family is a function $H : \Sigma^k \times \Sigma^m \to \Sigma^c$, where Σ^k is the key space, Σ^m is the message space, and Σ^c is the set of hash values. We often need to change Σ^m to describe different hash function families.

We write $x \xleftarrow{R} \Sigma^n$ for choosing a string of n-bit length uniformly at random. For a string x, $|x|$ is its bit-length. When A is an algorithm, program or adversary,

$A(x) \rightarrow y$ means that A gets an information of x to output y. When we want to address that A has no information to outputs y, we write $A(null) \rightarrow y$ with the null string $null$.

We take the RAM (Random Access Machine) model of computation which is also used in [1], and measure the running time of a program with respect to that model. If $H : \Sigma^k \times \Sigma^m \rightarrow \Sigma^c$ is a hash function family, we let T_H indicate the worst-case time to compute $H(K, x)$, in the underlying model of computation, when $K \in \Sigma^k$ and $x \in \Sigma^m$.

2.2 Definitions of CRHF and UOWHF

Recently, Rogaway and Shrimpton suggested seven simple and nice definitions of hash functions including CRHF and UOWHF [6], but we prefer to use some games to define our objects and to describe our work.

Definition 1 (CRHF). *A hash function family $H : \Sigma^k \times \Sigma^m \rightarrow \Sigma^c, m \geq c$, is (t, ε)-CRHF if no adversary A wins in the following game with the probability ε and within the time t:*

Game(CRHF,A,H)
$K \xleftarrow{R} \Sigma^k$
$A(K) \rightarrow (x, x')$
A wins if $x \neq x'$ and $H(K, x) = H(K, x')$.

In the game of Definition 1, the adversary gets the key K of H. This implies that the adversary knows everything about $H(K, \cdot)$ and so it can try any experiments until it produces its output within the time t. However, the behavior of the adversary is more restricted in Definition 2.

Definition 2 (UOWHF). *A hash function family $H : \Sigma^k \times \Sigma^m \rightarrow \Sigma^c, m \geq c$, is (t, ε)-UOWHF if no adversary $A = (A_1, A_2)$ wins in the following game with the probability ε and within the time t:*

Game(UOWHF,A,H)
$A_1(null) \rightarrow (x, State)$
$K \xleftarrow{R} \Sigma^k$
$A_2(K, x, State) \rightarrow x'$
$A = (A_1, A_2)$ wins if $x \neq x'$ and $H(K, x) = H(K, x')$.

In Definition 2, algorithm A_1 outputs the target message x. The only information which the adversary has before producing the target message is H. *State*, the other output of A_1, is some extra state information which helps A_2 to find a collision. Algorithm A_2 outputs the sibling message x' on input $(x, State)$.

Strictly speaking, when we are given $H : \Sigma^k \times \Sigma^m \rightarrow \Sigma^c$, we should call it a *CRHF family* or a *UOWHF family*, but for simplicity we often just call it *CRHF* or *UOWHF*.

3 Higher Order UOWHFs

Let us revisit Definition 2. No access to any oracles is given to A_1. A_1 outputs $(x, State)$ with no information. So, random selection of K from $\mathcal{K}$ is independent of A_1's behavior. Consequently, changing the order of steps 1 and 2 in the game doesn't effect the success probability of the adversary. So, the following game is essentially equivalent to the game in Definition 2.

Definition 3. *A hash function family $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, is (t, ε)-UOWHF′ if no adversary $A = (A_1, A_2)$ wins in the following game with the probability ε and within the time t:*

Game(UOWHF′, A, H)
$K \xleftarrow{R} \Sigma^k$
$A_1(null) \to (x, State)$
$A_2(K, x, State) \to x'$
$A = (A_1, A_2)$ *wins if* $x \neq x'$ *and* $H(K, x) = H(K, x')$.

However, unlike the game in Definition 2, we can add an oracle $\mathcal{O}^{H(K,\cdot)}$ to the game in Definition 3, which gets a query x and returns $y = H(K, x)$. We can then allow the adversary to access the oracle before he chooses the target message. Now we give the following definition. Let Q be a set of adaptive query-answer pairs associated with the oracle $\mathcal{O}^{H(K,\cdot)}$ which is initialized to the empty set $\emptyset$ in the game.

Definition 4 (r-th Order UOWHF). *A hash function family $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, is (t, ε)-UOWHF(r) if no adversary $A = (A_1, A_2)$ wins in the following game with the probability ε and within the time t:*

Game(UOWHF(r), A)
$K \xleftarrow{R} \Sigma^k$; $Q \leftarrow \emptyset$
if $r > 0$ **do**:
  for $i = 1, ..., r$ **do**:
    $A_1(Q) \to x_i$
    $y_i \leftarrow \mathcal{O}^{H(K,x_i)}$
    $Q \leftarrow \{(x_i, y_i)\} \cup Q$
$A_1(Q) \to (x, State)$
$A_2(K, x, State) \to x'$
$A = (A_1, A_2)$ *wins if* $x \neq x'$ *and* $H(K, x) = H(K, x')$.

Indeed, the hash function families which satisfy Definition 3 can be regarded as UOWHF(0) families. The relationships among Definitions 1, 2, 3 and 4 can be summarized as follows.

Proposition 1. *Let $H : \Sigma^k \times \Sigma^m \to \Sigma^c, m \geq c$, be a hash function family. Then,*

1. *H is a (t,ε)-UOWHF $\Leftrightarrow$ H is a (t,ε)-UOWHF(0).*
2. *For any $r \geq 0$, H is a (t',ε)-UOWHF$(r+1)$ $\Rightarrow$ H is a (t,ε)-UOWHF(r), where $t = t' - \Theta(T_H + m + c)$.*
3. *For any $r \geq 0$, H is a (t',ε)-CRHF $\Rightarrow$ H is a (t,ε)-UOWHF(r), where $t = t' - \Theta(r)(T_H + m + c)$.*

Proof. The proofs of 1 and 2 are trivial. So, we only prove 3. Suppose that $A = (A_1, A_2)$ is an adversary for H in the UOWHF(r) sense. We use it to make the adversary B who works in **Game**(CRHF,B, H) as follows.

Game(CRHF,B, H)
$K \xleftarrow{R} \Sigma^k$
$B(K)$ **do**:
  $Q \leftarrow \emptyset$
  if $r > 0$ **do**:
    for $i = 1, ..., r$ **do**:
      $A_1(Q) \rightarrow x_i$
      $y_i \leftarrow H(K, x_i)$
      $Q \leftarrow \{(x_i, y_i)\} \cup Q$
  $A_1(Q) \rightarrow (x, State)$
  $A_2(K, x, State) \rightarrow x'$
  output (x, x')

In the above game, B simulates $\mathcal{O}^{H(K,\cdot)}$ for A_1. Since B just outputs a collision which A found, the probability that B wins the game is same as A. The running time of B is at most $t + \Theta(r)(T_H + m + c)$. □

We claim that H is not a UOWHF(1) in Bellare and Rogaway's example [1]. If the adversary asks any query x and get the answer $y = H(K, x)$, then he would obtain the key K and can make a collision easily.

4 Merkle-Damgård Construction Based on Higher Order UOWHF

Suppose we have a hash function family $H : \Sigma^k \times \Sigma^{c+m} \rightarrow \Sigma^c$, where m is a positive integer. The Merkle-Damgård construction of H with variable initial value gives a hash function family MD$[H] : \Sigma^k \times (\Sigma^c \times \Sigma_m^+) \rightarrow \Sigma^c$. For each key $K \in \Sigma^k$ and any message $x = x_0 x_1 ... x_n$, where $x_0 \in \Sigma^c$ and $x_i \in \Sigma^m$ for $i = 1, ..., n$, MD$[H]$ is defined as follows.

Algorithm MD$[H](K, x)$
$n \leftarrow (|x| - c)/m$
$y_0 \leftarrow x_0$
for $i = 1, ..., n$ **do**:
  $y_i \leftarrow H_K(y_{i-1} || x_i)$
return y_n

If $\mathrm{MD}[H]$ only takes $(c+nm)$-bit messages for a fixed n, it would always have n rounds. In that case we use the notation $\mathrm{MD}_n[H]$ instead of $\mathrm{MD}[H]$. In the following theorem, we say that if H is a UOWHF(r), the $(r+1)$-round $\mathrm{MD}_{r+1}[H]$ is a UOWHF.

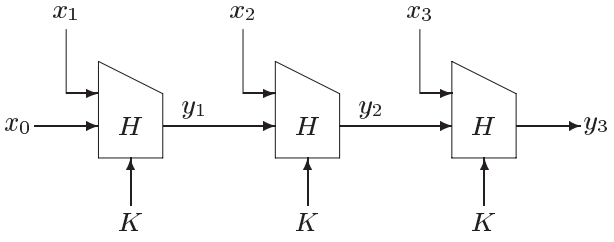


Fig. 1. 3-round Merkle-Damgård construction $\mathrm{MD}_3[H]$

Theorem 1. *Let $H : \Sigma^k \times \Sigma^{c+m} \to \Sigma^c$ be a (t', ε')-UOWHF(r). Then, $\mathrm{MD}_{r+1}[H] : \Sigma^k \times \Sigma^{c+rm} \to \Sigma^c$ is a (t, ε)-UOWHF, where $\varepsilon = (r+1)\varepsilon'$ and $t = t' - \Theta(r)(T_H + m + c)$.*

Proof. Let $x, x' \in \Sigma^{c+(r+1)m}$ be a collision for $\mathrm{MD}_{r+1}[H](K, \cdot)$. We observe that there exists an index $j \in \{1, ..., r+1\}$ such that

$$\begin{aligned} \mathrm{MD}[H](K, x_0 x_1 \cdots x_j) &= \mathrm{MD}[H](K, x'_0 x'_1 \cdots x'_j) \\ \mathrm{MD}[H](K, x_0 x_1 \cdots x_{j-1}) || x_j \;/=\; \mathrm{MD}[H](K, x'_0 x'_j \cdots x'_{j-1}) || x'_j. \end{aligned} \tag{1}$$

We will exploit this below.

Assume that $A = (A_1, A_2)$ is an adversary who breaks $\mathrm{MD}_{r+1}[H]$ with inputs of equal-length in the UOWHF sense. We use it to make the adversary $B = (B_1, B_2)$ who works in the **Game**(UOWHF(r),B, H) as follows.

Game(UOWHF(r),B, H)
$K \xleftarrow{R} \Sigma^k$; $Q \leftarrow \emptyset$
if $r > 0$ **do**:
  for $j = 1, ..., r$ **do**:
    $B_1(Q)$ **do**:
      if $j = 1$ **do**:
        $A_1(null) \to (x, State_A)$
        $y_0 \leftarrow x_0$
        query $y_0 || x_1$ **to** $\mathcal{O}^{H(K,\cdot)}$
      if $j > 1$ **do**:
        query $y_{j-1} || x_j$ **to** $\mathcal{O}^{H(K,\cdot)}$
    $y_j \leftarrow \mathcal{O}^{H(K, y_{j-1}||x_j)}$
    $Q \leftarrow \{(y_{j-1} || x_j, y_j)\} \cup Q$

$B_1(Q)$ **do**:
  $i \xleftarrow{R} \{1, ..., r+1\}$
  output $(y_{i-1}||x_i, State_B)$
$B_2(K, y_{i-1}||x_i, State_B)$ **do**:
  $A_2(K, x, State_A) \to x'$
  $y'_{i-1} \leftarrow \mathrm{MD}[H](K, x'_0 x'_1 \cdots x'_{i-1})$
  output $y'_{i-1}||x'_i$

The adversary B works in the game as follows. When $j = 1$, Q is empty and B_1 runs A_1 to obtain $(x, State_A)$ where $x = x_0 x_1 \cdots x_{r+1}$. Then B_1 sets y_0 to x_0 and sends a query $y_0||x_1$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. When $j \neq 1$, Q is nonempty and B_1 sends a query $y_{j-1}||x_j$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. After collecting r adaptive query-answer pairs, B_1 selects $i \in \{1, ..., r+1\}$ at random, and then outputs $y_{i-1}||x_i$ and $State_B = (i, x, State_A)$ as the target message and an additional state information for B_2, respectively. On input $(K, y_{i-1}||x_i, State_B)$, B_2 runs A_2 by giving $(K, x, State_A)$. Once A_2 outputs its sibling message x', B_2 computes y'_{i-1} and outputs $y'_{i-1}||x'_i$ as its sibling message.

Now we must bound the probability that $(y_{i-1}||x_i, y'_{i-1}||x'_i)$ is a collision for $H(K, \cdot)$ in the game. Note that i was chosen at random, so if (x, x') is a collision for $\mathrm{MD}_{r+1}[H](K, \cdot)$ then we have $i = j$ with probability $1/(r+1)$, where j is the value of Equation (1). So, $\varepsilon' > \varepsilon/(r+1)$.

The running time of B is that of A plus the overhead. This overhead is $\Theta(r)(T_H + m + c)$. The choice of t in the theorem statement makes all this at most t', from which we conclude the result. □

Assume that a domain $\mathcal{D}$ and a range $\mathcal{R}$ are fixed. We regard the notion of *UOWHF(r)* as the class of all UOWHF(r). We also consider the class of all the hash functions which do not lose universal one-wayness (upto equal-length collisions) until they are extended to $(r+1)$-round Merkle-Damgård construction, *UOW-MD(r)*. From Proposition 1 and Theorem 1, it is easy to see that these classes forms two different chains between the classes *CRHF* and *UOWHF* with the same domain $\mathcal{D}$ and the same range $\mathcal{R}$, and that for each integer $r > 0$, *UOWHF(r)* implies *UOW-MD(r+1)* (see Fig. 2).

We can generalize Theorem 1 to MD$[H]$ taking inputs of variable length. We assume that the message is always padded such that its length is a multiple of m. There are many padding methods but we don't mention any specific one. We use the notation $(t, \mu_1, \mu_2, \varepsilon)$-UOWHF instead of (t, ε)-UOWHF. μ_1 is the bound on the length of the target message and μ_2 is the bound on the length of the sibling message. Note that the only restriction on μ_2 is that the algorithms on the sibling message should be computable in polynomial time.

Theorem 2. *Suppose* $H : \Sigma^k \times \Sigma^{c+m} \to \Sigma^c$ *be a* (t', ε')*-UOWHF*(r)*. Suppose* $\mu_1 - c$ *and* $\mu_2 - c$ *are multiples of* m*. Then, for* $\mu_1 \leq c + (r+1)m$ *and a proper* μ_2, $\mathrm{MD}[H] : \Sigma^k \times (\Sigma^c \times \Sigma_m^+) \to \Sigma^c$ *is a* $(t, \mu_1, \mu_2, \varepsilon)$*-UOWHF, where* $\varepsilon = \sigma_{\min}\varepsilon'$ *and* $t = t' - \Theta(\sigma_{\max})(T_H + m + c)$ *for* $\sigma_{\min} = \min\{(\mu_1 - c)/m, (\mu_2 - c)/m\}$ *and* $\sigma_{\max} = \max\{(\mu_1 - c)/m, (\mu_2 - c)/m\}$.

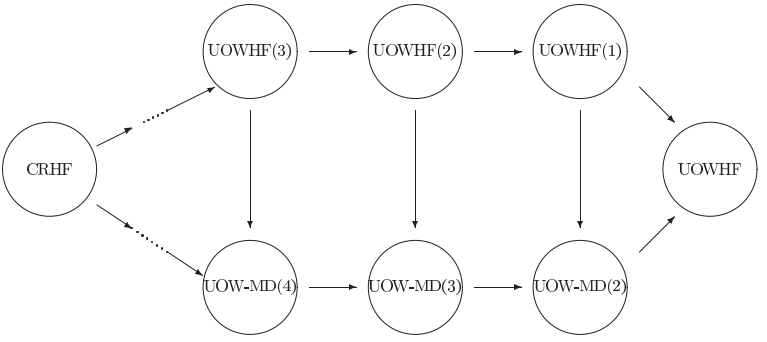


Fig. 2. Two chains between CRHF and UOWHF. Each arrow means the implication

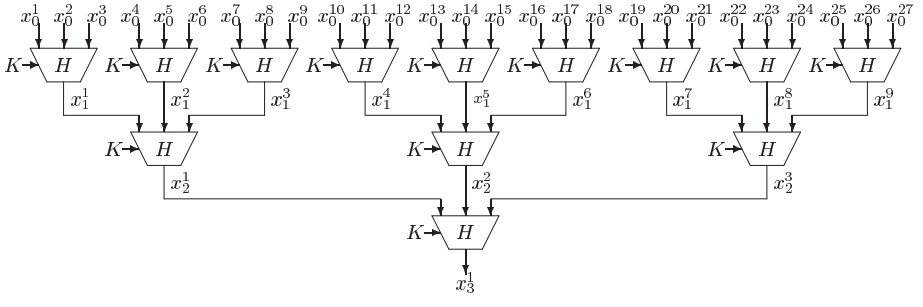


Fig. 3. 3-level tree construction $\mathrm{TR}_3[H]$ for the case of $l = 3$

Proof. The proof is similar to that of Theorem 1.

5 Tree Construction Based on Higher Order UOWHF

If we are given a UOWHF family $H : \Sigma^k \times \Sigma^m \to \Sigma^c$ and m is a multiple of c, we can extend it more efficiently by using a tree structure. If $m = dc$ for a positive integer d, we can use a d-ary tree structure. Since a tree construction consists of parallel procedures, it can be more efficient than the Merkle-Damgård construction if multiple processors are available.

Firstly, we define the parallel construction $\mathrm{PA}[H]$ on H. $\mathrm{PA}_n[H]$ consists of n components $\mathrm{PA}_n[H]_1, ..., \mathrm{PA}_n[H]_n$. For $K \in \Sigma^k$, each component function $\mathrm{PA}_n[H]_i$ is

$$\mathrm{PA}_n[H]_i(K, x) = \begin{cases} H(K, x_i) & \text{if } |x| = dc \\ x_i & \text{if } |x| = c \end{cases}$$

That is, the domain of $\mathrm{PA}_n[H]_i(K, \cdot)$ is $\Sigma^c \cup \Sigma^{dc}$, while the domain of $H(K, \cdot)$ is Σ^c.

We assume that we are given $x = x_1 \cdots x_n$ for each $x_i \in \Sigma^c \cup \Sigma^{dc}$. For a key $k \in \Sigma^k$, $\mathrm{PA}_n[H](K, x)$ is defined as follows.

Algorithm $\mathrm{PA}_n[H](K, x)$
$n \leftarrow \log_{dc} |x|$
for $i = 1, ..., n$ **do**:
  $y_i \leftarrow \mathrm{PA}_n[H]_i(K, x_i)$
return $y_1 || \cdots || y_n$

Now we define a tree construction $\mathrm{TR}[H]$ based on H. We begin with the message space $\Sigma_c^{d^l}$. We denote the tree construction on H to hash only messages in $\Sigma_c^{d^l}$ as $\mathrm{TR}_l[H]$. For each key $K \in \Sigma^k$ and any message $x \in \Sigma_c^{d^l}$, $\mathrm{TR}_l[H]$ is defined according to:

Algorithm $\mathrm{TR}_l[H](K, x)$
$Level[0] \leftarrow x$
for $i = 1, ..., l$ **do**
  $Level[i] \leftarrow \mathrm{PA}_{d^{l-i}}[H](K, Level[i-1])$
return $Level[l]$

Write $Level[0] = x$ as $x_0 = x_{0,1} x_{0,2} \cdots x_{0,d^l}$ where $x_{0,i} \in \Sigma^c$ for $i = 1, ..., d^l$. Then, we can see $\mathrm{TR}_l[H](K, x)$ is computed as follows.

$$\begin{aligned} Level[0] &= x_0^1 x_0^2 \ \cdots \ \cdots \ \cdots \ x_0^{d^l} \\ Level[1] &= x_1^1 x_1^2 \ \cdots \ \cdots \ x_1^{d^{l-1}} \\ &\vdots \\ Level[l-1] &= x_{l-1}^1 \ \cdots \ x_{l-1}^d \\ Level[l] &= x_l^1 \end{aligned}$$

where $x_i^j = H(K, x_{i-1}^{(j-1)d+1} x_{i-1}^{(j-1)d+2} \cdots x_{i-1}^{jd})$.

The following theorem states that if H is a UOWHF(r) and $r = (d^l - 1)/(d - 1)$, then $\mathrm{TR}_l[H]$ is a UOWHF.

Theorem 3. *Let* $H : \Sigma^k \times \Sigma_c^d \to \Sigma^c$ *be a* (t', ε')*-UOWHF*(r) *and* $r = (d^l - d)/(d-1)$. *Then* $\mathrm{TR}_l[H] : \Sigma^k \times \Sigma_c^{d^l} \to \Sigma^c$ *is a* (t, ε)*-UOWHF, where* $\varepsilon = (r+1)\varepsilon'$, *and* $t' = t + \Theta(d^l)(T_H + dc)$.

Proof. Assume that $x, y \in \Sigma_c^{d^l}$ is a collision for $\mathrm{TR}_l[H](K, \cdot)$. We observe that there exist $\alpha \in \{1, ..., l\}$ and $\beta \in \{1, ..., d^{l-\alpha}\}$ such that

$$\begin{gathered} x_\alpha^\beta = y_\alpha^\beta \\ x_{\alpha-1}^{(\beta-1)d+1} || \cdots || x_{\alpha-1}^{\beta d} \ / \ \not{=}_{\alpha-1}^{(\beta-1)d+1} || \cdots || y_{\alpha-1}^{\beta d}. \end{gathered} \tag{2}$$

We will exploit this below.

Assume that $A = (A_1, A_2)$ is an adversary who breaks $\mathrm{TR}_l[H]$ with inputs of equal-length in the UOWHF sense. We use it to make the adversary $B = (B_1, B_2)$ who works in **Game**(UOWHF(r),B, H) as follows.

Game(UOWHF(r),B, H)
$K \xleftarrow{R} \Sigma^k$; $Q \leftarrow \emptyset$
if $r > 0$ **do**:
  for $u = 1, ..., l-1$ **do**:
    for $v = 1, ..., d^{l-u}$ **do**:
      $B_1(Q)$ **do**:
        if $(u, v) = (1, 1)$ **do**:
          $A_1(null) \rightarrow (x, State_A)$
          query $x_0^1 || \cdots || x_0^d$ **to** $\mathcal{O}^{H(K,\cdot)}$
        if $(u, v) \neq (1)$ **do**:
          query $x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd}$ **to** $\mathcal{O}^{H(K,\cdot)}$
        $x_u^v \leftarrow \mathcal{O}^{H(K, x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd})}$
        $Q \leftarrow \{(x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd}, x_u^v)\} \cup Q$
$B_1(Q)$ **do**:
  $i \xleftarrow{R} \{1, ..., l\}$; $j \xleftarrow{R} \{1, ..., d^{l-i}\}$
  output $(x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, State_B)$
$B_2(K, x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, State_B)$ **do**:
  $A_2(K, x, State_A) \rightarrow y$
  output $y_{i-1}^{(j-1)d+1} || \cdots || y_{i-1}^{jd}$

The adversary B works in the game as follows. When $(u, v) = (1, 1)$, Q is empty and B_1 runs A_1 to obtain $(x, State_A)$ where $x = x_0^1 || \cdots || x_0^{d^l} \in \Sigma_c^{d^l}$. Then, B_1 sends a query $x_0^1 || \cdots || x_0^d$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. When $(u, v) \neq (1)$, Q is nonempty. B_1 sends a query $x_{u-1}^{(v-1)d+1} || \cdots || x_{u-1}^{vd}$ to the oracle $\mathcal{O}^{H(K,\cdot)}$. After collecting r adaptive query-answer pairs, B_1 randomly selects i and j from $\{1, ..., l\}$ and $\{1, ..., d^{l-i}\}$, respectively. The B_1 outputs $x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}$ as the target message and $State_B = (i, j, x, State_A)$ as an additional state information for B_2. On the input $(x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, State_B)$, B_2 runs A_2 by giving $(K, x, State_A)$. Once A_2 outputs its sibling message x', B_2 computes and outputs $y_{i-1}^{(j-1)d+1} || \cdots || y_{i-1}^{jd}$ as its sibling message.

Now we must bound the probability that $x_{i-1}^{(j-1)d+1} || \cdots || x_{i-1}^{jd}, y_{i-1}^{(j-1)d+1} || \cdots || y_{i-1}^{jd}$ is a collision for $H(K, \cdot)$. The number of possibilities for (i, j) is at most $d^0 + \cdots + d^{l-1} = (d^l - 1)/(d - 1)$. Note that i and j were chosen randomly and independently, so if x, y is a collision for $\mathrm{TR}_l[H](K, \cdot)$ then we have $(i, j) = (\alpha, \beta)$ with probability $(d - 1)/(d^l - 1)$, where (α, β) is the pair in Equation (2). So, $\varepsilon' > \varepsilon(d - 1)/(d^l - 1)$.

The running time of B is that of A plus the overhead, which is equal to $\Theta(d^l)(T_H + dc)$. □

The tree construction can hash the messages of variable lengths like the Merkle-Damgård construction. We assume that we are given a message x. When $\lceil \log_d \frac{|x|}{c} \rceil = l$, we pad x such that the length of the padded message x^* is the smallest value larger than $|x|$ of the form $|x^*| = (d^l - qd + q)c$ for some integer

$0 < q < d^{l-1}$. Then, the number of applications of the underlying hash function is $1 + d + d^2 + \cdots + d^{l-1} - q = \frac{d^l - 1}{d-1} - q$. See Fig. 4, 5, 6 and 7 for the case of $l = 2, d = 4$. We denote the set of the padded messages in such way by

$$\mathcal{S}(c,d) = \{x \in \Sigma^* || |x| = (d^l - qd + q)c \text{ for some integers } l > 0 \text{ and } 0 \le q < d^{l-1}\}$$

Now we generalize Theorem 3.

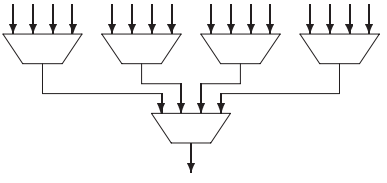


Fig. 4. $|x|/c = 16 = 4^2$

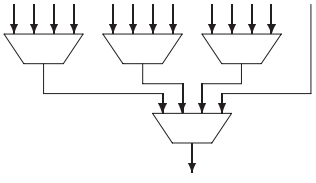


Fig. 5. $|x|/c = 13 = 4^2 - 1 \cdot 4 + 1$

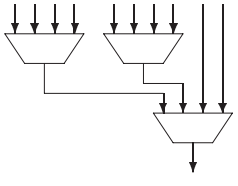


Fig. 6. $|x|/c = 10 = 4^2 - 2 \cdot 4 + 2$

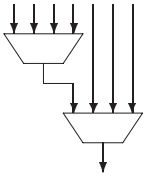


Fig. 7. $|x|/c = 7 = 4^2 - 3 \cdot 4 + 3$

Theorem 4. *Suppose* $H : \Sigma^k \times \Sigma_c^d \to \Sigma^c$ *be a* (t', ε')*-UOWHF*(r)*. Suppose* $\mu_i = d^{l_i} - q_i d + q_i$ *for* $i = 1, 2$*. Then for* $\mu_1 \le c(r(d-1) + d)$ *and a proper* μ_2*,* $\mathrm{TR}[H] : \Sigma^k \times \mathcal{S}(c,d) \to \Sigma^c$ *is a* (t, ε)*-UOWHF, where* $\varepsilon = \sigma_{\min}\varepsilon'$ *and* $t' = t + \Theta(\sigma_{\max})(T_H + dc)$ *for* $\sigma_{\min} = \min\{\frac{d^{l_1}-1}{d-1} - q_1, \frac{d^{l_2}-1}{d-1} - q_2\}$ *and* $\sigma_{\max} = \max\{\frac{d^{l_1}-1}{d-1} - q_1, \frac{d^{l_2}-1}{d-1} - q_2\}$.

Proof. The proof is similar to that of Theorem 3. □

6 Conclusion

We defined the order of a UOWHF family and showed how much the efficiency of known constructions for UOWHFs is improved by the notion of the order. Our main results are as follows.

- If the order of the underlying UOWHF H is r, then the $(r+1)$-round Merkle-Damgård construction $MD_{r+1}[H]$ is also a UOWHF. If the resulting function $MD_{r+1}[H]$ is used as a building block in existing constructions with linear structure, the key size can be reduced with at most a factor of $(r+1)$.
- If the order of the underlying UOWHF $H : \Sigma^k \times \Sigma_c^d \rightarrow \Sigma^c$ is $r = \frac{d^l - d}{d-1}$, then the l-level tree construction $TR_l[H]$ is also a UOWHF. If the resulting function $TR_l[H]$ is used as a building block in existing constructions with tree structure, the key size can be reduced with at most a factor of l.

References

1. M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," In B.S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, LNCS 1294, Springer-Verlag, pages 470–484, 1997.
2. I. Damgård, "A Design Priciple for Hash Functions," In G. Brassard, editor, *Advances in Cryptology – Crypto'89*, LNCS 435, Springer-Verlag, pages 416–427, 1989.
3. R. Merkle, "One way hash functions and DES," In G. Brassard, editors, *Advances in Cryptology – Crypto'89*, LNCS 435, Springer-Verlag, pages 428–446, 1989.
4. I. Mironov, "Hash Functions: From Merkle-Damgård to Shoup," In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt 2001*, LNCS 2045, Springer-Verlag, pages 166-181, 2001.
5. M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proceedings of the Twenty-first ACM Symposium on Theory of Computing*, pages 33–43, 1989.
6. P. Rogaway and T. Shrimpton, "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance," In B. Roy and W. Meier, editors, *Fast Software Encryption 2004*, LNCS 3017, Springer-Verlag, pages 371–388, 2004.
7. P. Sarkar, "Constuction of UOWHF: Tree Hashing Revisited," Cryptology ePrint Achive, http://eprint.iacr.org/2002/058.
8. P. Sarkar, "Domain Extenders for UOWHF: A Generic Lower Bound om Key Expansion and a Finite Binary Tree Algorithm," Cryptology ePrint Archive, http://eprint.iacr.org/2003/009.
9. V. Shoup, "A composite theorem for universal one-way hash functions," In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, LNCS 1807, Springer-Verlag, pages 445–452, 2000.
10. D. Simon, "Finding collisions on a one-way street: can secure hash functions be based on general assumptions?," In K. Nyberg, editor, *Advances in Cryptology – Eurocrypt'98*, LNCS 1403, Springer-Verlag, pages 334–345, 1998.
11. W. Lee, D. Chang, S. Lee, S. Sung, and M. Nandi, "New Parallel Domain Extenders for UOWHF," *Advances in Cryptology – ASIACRYPT 2003*, LNCS 2894, Springer-Verlag, pages 208–227, 2003.
12. Y. Zheng, T. Matsumoto, H. Imai, "Connections between several versions of one-way hash functions," *Trans. IEICE E*, Vol. E73, No. 7, pp. 1092–1099, July 1990.

The MD2 Hash Function Is Not One-Way

Frédéric Muller

DCSSI Crypto Lab 51, Boulevard de Latour-Maubourg,
75700 Paris 07 SP France
Frederic.Muller@sgdn.pm.gouv.fr

Abstract. MD2 is an early hash function developed by Ron Rivest for RSA Security, that produces message digests of 128 bits. In this paper, we show that MD2 does not reach the ideal security level of 2^{128}. We describe preimage attacks against the underlying compression function, the best of which has complexity of 2^{73}. As a result, the full MD2 hash can be attacked in preimage with complexity of 2^{104}.

1 Introduction

Cryptographic hash functions are an important primitive used in various situations. The main fields of applications are message authentication codes, digital signatures, and therefore certificates. Hash functions are also used as a building tool in many protocols and advanced constructions.

By definition, a hash function H is a function mapping an input message m of arbitrary length to an output h of fixed length (typically this length ranges from 128 to 512 bits)

$$h = H(m)$$

The main properties expected from a cryptographic hash function are:

- **Collision Resistance:** it should be hard to find two inputs m and m' that map to the same output by H.
- **Second Preimage Resistance:** for a given m, it should be hard to find a second input m' such that m and m' map to the same output by H.
- **Preimage Resistance:** for a given challenge h, it should be hard to find an input m which maps to h by H.

More can be found on the theory of hash functions in [9, 10]. Most of the hash functions used in practice belong to the so-called "MD family". This family of hash functions was initially developed by Ron Rivest for RSA Security. The first proposal was MD2 [7], an early, non-conventional, byte-oriented design. It was quickly followed by MD4 [11] and MD5 [12], two hash functions with a more modern, 32-bit-oriented design. Despite not being collision-resistant [3], MD4 has inspired most modern hash functions designs, like the RIPEMD family or the SHA family. Over the last years, the effort on attacking hash functions has mostly concerned collision resistance [2-4, 15], since this property is essential for many applications. However, few results have been reported regarding (second) preimage attacks for these hash functions (see [5, 9]).

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 214–229, 2004.

In this paper, we focus on the MD2 hash function [7]. Despite being the oldest hash functions from its family, and despite using an old-fashioned architecture, MD2 is still used in several contexts. For instance, if we look at the recent PKCS #1 v2.1, a cryptographic standard from RSA Security [17], the MD2 hash is still given as an example of one-way, collision-resistant hash function, while MD4 has been removed, presumably because of Dobbertin's collision attack [3]. In addition, it is precised that "MD2 (is) recommended only for compatibility with existing applications based on PKCS #1 v1.5". The underlying explanation is that the use of MD2 was highly encouraged in the previous version from 1993 [16] where MD2 was recommended as a "conservative design". This confidence in MD2 is not surprising because, despite being quite inefficient and based on an older design philosophy, MD2 has surprisingly well resisted to cryptanalysis. The only attack known is a collision attack against the compression function [14]. This attacks works with the correct IV, however it no longer works when a checksum is appended to the message, as imposed in the specifications [7]. For the full hash function, no attack is known.

Consequently MD2 still appears in various applications and even some proposed standards [1]. However, the crucial security point regarding MD2 is now its use in public-key infrastructures. Many certificates have been generated with RSA-MD2 in the past and many of them are still widely used (like Verisign certificates for instance). Actually, anyone can easily verify that recent versions of Windows are delivered with those MD2 certificates. Therefore millions of users are probably using MD2-based certificates on a regular basis. The security of certificates is a particular problem. Indeed, collision attacks do not threat the security of the scheme, because the input of the signature primitive (typically the usual primitive used with MD2 is the RSA signature) is fixed. An attacker needs to find a collision between two inputs of MD2, one of them being the data part of the certificate. If he succeeds, he will manage to forge a new valid certificate. Hence what is required here is exactly second preimage resistance of MD2. This is an important motivation to analyze the security of MD2 regarding preimage and second preimage attacks, which is the focus of this paper. We obtained interesting new results and theoretical attacks. Since our best attack against MD2 is more efficient than a naive guessing attack in 2^{128}, MD2 can no longer be considered a secure one-way hash function.

First, we describe briefly the MD2 algorithm. Then, we focus on the compression function and describe several attacks. The best is a pseudo-preimage attack with complexity 2^{73}. Finally, we show how to turn these attacks into an attack for the full hash, which is not straightforward because of the checksum bytes.

2 The MD2 Hash Function

2.1 Generalities

The MD2 Message-Digest algorithm was developed in 1989 by Ron Rivest. The actual specifications can be found in RFC 1319 [7]. This algorithm belongs, together with MD4 and MD5, to the family of hash functions developed by Ron

Rivest for RSA Security. However, compared to the other algorithms of the family (and to most actual hash functions), MD2 has several interesting particularities

- MD2 is a **byte-oriented** hash function. Indeed all instructions handle 8 bits of data. While this was useful for old architectures, today's processors can manipulate words of (at least) 32 bits. Consequently all modern hash functions use 32-bit instructions. This is the case of MD4, MD5 and also for the hash functions of the RIPEMD and SHA families.
- MD2 uses a **checksum** of 128 bits computed from the whole message and appended as the last input block of the compression function. Hence MD2 does not follow the Merkle-Damgård construction, contrarily to most actual hash functions. Consequently classical results [9] on how to turn collisions on the compression function to collisions for the whole hash function do not apply here. This is the reason why the collision attack described in [14] does not extend to the full MD2 hash.
- the compression function of MD2 has a different architecture from most modern hash functions. Indeed it does not look like a block cipher. Instead, a fixed "scrambling" function is iterated on a 384 bits long internal state. The initial state is derived linearly from a message block of length 128 bits and an intermediate hash of 128 bits. The final state is truncated to 128 bits. This function uses simple instructions like XOR and a nonlinear S-box.

Therefore MD2 is a very early design of hash function and differs significantly from modern hash functions. In terms of efficiency, it compares quite bad to its challengers (mostly because of the byte-oriented structure).

2.2 Description of MD2

In this section, we describe more precisely the mechanisms used by the MD2 hash function (see [7] for the full specifications). The general description of MD2 is found in Figure 1.

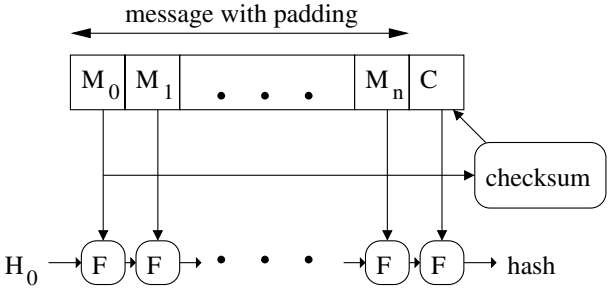


Fig. 1. The MD2 Hash Function

All blocks manipulated have length 128 bits. We refer to the blocks of the message by $M_0, \ldots, M_n$. The first step of MD2 is to append a padding to the

initial message, then to compute a checksum block (that we call C). This increases the length of the message by 1 block. Finally the compression function (referred to as F) is applied iteratively to produce the hash value. If we call H_i the i-th intermediate hash,

$$H_{i+1} = F(H_i, M_i)$$

The IV of the hash function is H_0 and is set by default to 0.

The Compression Function. A precise representation of the compression function F is given in Figure 2. Each box in this figure contains one byte. F is decomposed into 3 matrices - denoted by A, B and C - with 16 columns and 19 rows each. The first row of each matrix is initialized respectively with H_i, M_i and $H_i \oplus M_i$. Then the rows of each matrix are computed recursively from top to bottom. The last rows of B and C are not used. The '+' symbol denotes addition modulo 256.

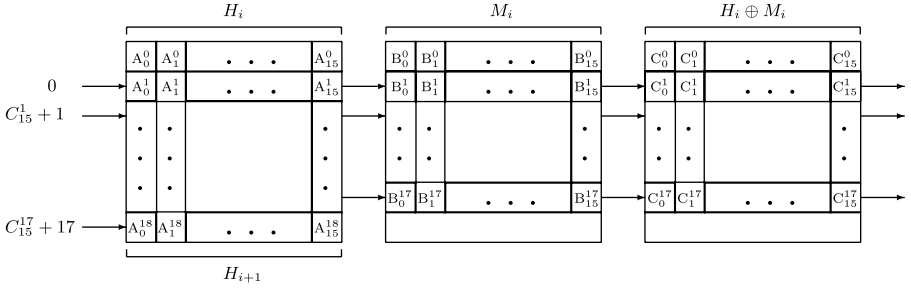


Fig. 2. The Compression Function of MD2

The computations are based on a function ϕ from 16 bits to 8 bits. In the case of the matrix A, this can be described by the equations:

$$\begin{aligned} A_i^t &= \phi(A_i^{t-1}, A_{i-1}^t) \\ &= A_i^{t-1} \oplus \mathrm{S}(A_{i-1}^t) \end{aligned}$$

where S is a fixed S-box of size 8 bits. Equations for matrices B and C are exactly the same. This function ϕ is represented in Figure 3. For the particular case $i = 0$, a byte extracted from another matrix is used instead of A_{i-1}^t (see Figure 2).

The Checksum Function. The checksum C is computed from the blocks of message by iterating a non-linear checksum function, that we call G. Details on G are not relevant for our attacks. Basically G uses only basic operations like XOR and the S-box S. At a high-level, the following equations describe this mechanism:

$$\begin{aligned} \mathrm{IC}_0 &= 0 \\ \mathrm{IC}_{i+1} &= G(\mathrm{IC}_i, M_i) \end{aligned}$$

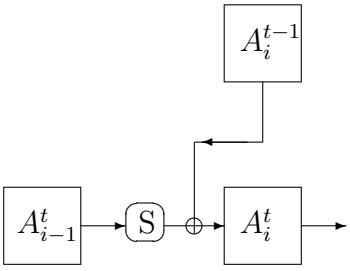


Fig. 3. The ϕ function

G is a complicated function, however it is straightforward to compute the intermediate checksum IC_i from IC_{i+1} and M_i. The final value IC_{n+1} is the appended checksum C. A precise description of G is available in [7].

3 Collision Attacks Against MD2

The only known cryptanalytic result against MD2 is the paper by Rogier and Chauvaud [14]. In this paper, collisions on the compression function of F are described. This attack works very well because the IV used in MD2 is O (although a variant is proposed for other IV's with an increased complexity). Details of this attack are not essential here. The key idea is to use the symmetry between matrices B and C when $H_i = 0$. (the first rows are equal in this case). Unfortunately collisions cannot be extended to the full MD2 because of the checksum bytes.

Although collision attacks may be of interest in many contexts, there are several arguments why researching efficient collision attacks for MD2 is no longer a major concern.

- First, one has to take into account the dimension of MD2. The produced hashed values have length only 128 bits. Therefore birthday paradox attacks have complexity of the order of 2^{64}. This is not a satisfying level of security for modern applications. As an example, the MD5 hash function (whose output have also a length of 128 bits) is actually the subject of a distributed attack to find collisions [8]. It is clear that the interest of finding complicated shortcut attacks diminishes when efficient attacks using a large computational power are possible [18].
- Secondly, MD2 is no longer widely used in practice. For instance, in MAC or signatures, the collision resistance of a hash function is generally a requirement, but MD2 is no longer recommended for such applications. However, as we mentioned previously, MD2 is still used in some certificates. In this context, collision resistance is not really a concern but preimage and second preimage resistance are required.

4 Preimage Attacks Against MD2 Compression Function

A large variety of definitions for preimage and second preimage attacks exist in the literature, depending on what is a fixed challenge for the attacker and what can be freely chosen. A classical reference is [9], however a new classification of these notions has been recently given in [13].

In this section, we focus only on (preimage) attacks against the compression function of MD2. It is well known that these attacks can generally be extended to attacks against the whole hash (see [9]).

4.1 Three Scenarios

According to the previous notations, the compression function F operates by:

$$H_{i+1} = F(H_i, M_i)$$

where the H_i's are intermediate hash values and M_i is a message block (see Section 2.2). Basically we can consider 3 attack scenarios at this point:

1. H_{i+1} and H_i are given and the attacker must find an appropriate M_i.
2. H_{i+1} and M_i are given and the attacker must find an appropriate H_i.
3. H_{i+1} is given and the attacker must find appropriate H_i and M_i.

Any of these attacks may be of interest to attack the whole hash. Obviously, the 1st and 2nd attack are very similar because the roles of H_i and M_i in F are almost symmetric.

These 3 attack scenarios have received different names in the literature. Recently the names "aPre" ("a" stands for "always"), "ePre" ("e" stands for "everywhere") and "Pre" have been given to these 3 notions [13]. In [9], the terminology of "preimage resistance" and "pseudo-preimage resistance" is used. In the following sections, we envisage each scenario separately and propose new attacks.

4.2 Attacking Scenario 1

In this scenario, we suppose that H_i and H_{i+1} are a fixed challenge and our goal is to find an appropriate M_i such that

$$H_{i+1} = F(H_i, M_i)$$

First, we notice that a solution does not necessarily exist. Indeed all variables have length 128 bits, so in average only one solution M_i is expected, but there is no guarantee. We propose an attack that recovers all solutions corresponding to a given challenge (H_i, H_{i+1}). Basically our attack is a sophisticated combination of exhaustive search and meet-in-the-middle attacks. It proceeds with two distinct steps. In the following, we call $(m_0, \ldots, m_{15})$ the 16 bytes of M_i.

First Step. The first step of the attack is to derive all possible information from the challenge (H_i, H_{i+1}). These two objects are stored at the first and last row of matrix A (see Figure 4 where dashed cells correspond to the known bytes).

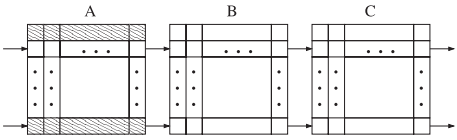


Fig. 4. Initial knowledge when H_{i+1} and H_i are fixed

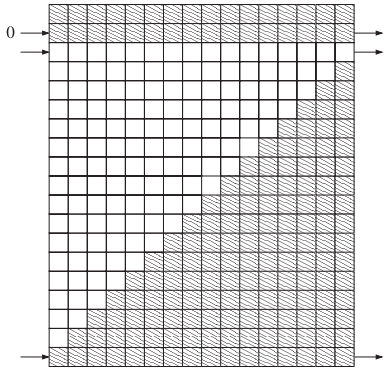


Fig. 5. Known values in the matrix A

Because of the structure of ϕ (this function is used to compute the contents of the matrices, see Section 2.2), more information can be derived directly from the challenge. For instance, when A_{i-1}^t and A_i^t are known, we can obtain A_i^{t-1} since:

$$\begin{aligned} A_i^t &= \phi(A_i^{t-1}, A_{i-1}^t) \\ &= A_i^{t-1} \oplus \mathrm{S}(A_{i-1}^t) \end{aligned}$$

In Figure 5, we represented by dashed boxes the large portion of A that can be directly derived this way. The second row is known because the byte introduced on the left hand side is known and always equal to 0.

In addition, if we **guess the byte introduced on the left hand side** of the 3rd row in A (*i.e.* $C_{15}^1 + 1$), then we can derive the **full content** of matrix A by similar considerations. In particular the bytes A_{15}^i's are known, and also the bytes C_{15}^i's for $i > 0$.

Second Step. Then, the second step of the attack is to perform a meet-in-the-middle attack on the matrices B and C to find an appropriate value of M_i. Basically at this point, we know what enters on the left hand side of B and what exits on the right hand side of C. Hence, we apply the following "meet-in-the-middle" algorithm:

– Guess the 4 bytes $(B_{15}^1, \ldots, B_{15}^4)$
  - for all values of the 8 bytes $(m_0, \ldots, m_7)$,
    * compute the 4 bytes $(B_7^1, \ldots, B_7^4)$ (this is possible because the sequence of A_{15}^i's is known)
    * compute the 4 bytes $(C_7^1, \ldots, C_7^4)$ (this is possible because H_i is known)
    * store these $4 + 4 = 8$ bytes in a table T_1
  - sort T_1 (which has 2^{64} entries of 64 bits each)
  - Repeat the same process with $(m_8, \ldots, m_{15})$ to obtain a table T_2 that contains also the bytes $(B_7^1, \ldots, B_7^4, C_7^1, \ldots, C_7^4)$.
  - Find all collisions between T_1 and T_2. This can be done efficiently by computing the joint product $T = T_1 \bowtie T_2$ (see [19]) with complexity of the order of 2^{64}
  - The resulting table T contains on average 2^{64} candidate values for $M_i = (m_0, \ldots, m_{15})$
  - Loop over all these candidates to find all valid M_i's

One can also refer to Figure 6 for the general philosophy of this attack. Dashed boxes represent the 8 bytes stored in tables T_1 and T_2, where we look for collisions.

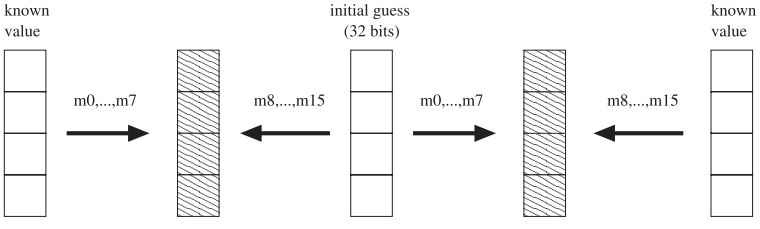


Fig. 6. The general philosophy of the attack

Analysis. In this attack, there are two outside loops. A loop of size 2^8 comes from the First Step of the attack (we need to guess one byte in order to find the full content of A). Besides an outside loop of length 2^{32} is required in the "meet-in-the-middle" algorithm. Inside these loops we need to create and to sort the tables T_1 and T_2. Those are tables with 2^{64} entrees, sorted using a key of 64 bits. Sorting the tables can be done efficiently with an appropriate "bucket-sort" algorithm so the cost is above 2^{64} instructions. Creating the tables has also a cost of the order of 2^{64} instructions. Since these two operations are performed twice (once for T_1 and once for T_2), the complexity is of the order of

$$\text{Complexity } = 2^8 \times 2^{32} \times (4 \times 2^{64}) = 2^{106}$$

basic instructions. This corresponds approximatively to 2^{95} applications of the compression function (a quick estimation shows that about 2^{11} instructions are needed for the compression function).

This should be compared to the complexity of an exhaustive search to find a preimage which would cost 2^{128} applications of the compression function. However, our attack requires about 2^{71} bits of memory. High memory requirements are known to increase the "real" cost of attacks [20]. Nevertheless this complexity is of the order of $2^{3n/4}$ while 2^n would be expected for a good compression function on n bits. An improved attack is also proposed in Appendix A to reduce these memory requirements. Further improvements have been investigated but no attack with complexity below $2^{3n/4}$ was found.

4.3 Attacking Scenario 2

In the second scenario, the message block M_i is fixed and we search an appropriate H_i. Attacking this scenario is very similar to attacking scenario 1 because there is an important symmetry in the compression function.

In the previous attack we managed to reconstruct the content of A from the initial challenge, and then applied a "meet-in-the-middle" attack to B and C. In Scenario 2, we can reconstruct the content of B from the challenge (M_i, H_{i+1}) and then attack by the middle the matrices A and C. Details of this attack are not very helpful to break the full MD2 hash, so we decided not to explore further this scenario.

4.4 Attacking Scenario 3

Finally, we suppose that only H_{i+1} is fixed, and the problem is to find any pair (H_i, M_i) solution of the equation

$$H_{i+1} = F(H_i, M_i)$$

This type of attack is often referred to as a **pseudo-preimage attack** on the compression function [9]. Of course, it is easier to find such a solution because we have more degrees of freedom. Therefore we wish to find an attack with complexity better than the previous 2^{95}. In this section, we describe an attack with complexity of the order of 2^{73} against this scenario.

The Attack. First, one should notice that many solutions exist to this problem. Indeed, we expect

$$\frac{2^{128} \times 2^{128}}{2^{128}} = 2^{128}$$

solutions in average. Therefore it is reasonable to impose some additional constraints.

Like for the previous attacks, we first derive all possible information from the given challenge (H_{i+1} here). In addition, we impose the constraint that $A^1_{15} = A^2_{15} = c$, where c is some constant, say $c = 0$ for instance. Figure 7 represents the resulting known values in the matrix A.

We observe that the complete rightmost column of A is known, which helps when considering the behavior of matrix B. At this point, a 6 bytes constant $(k_0, \ldots, k_5)$ is chosen at random. Then we apply the following algorithm:

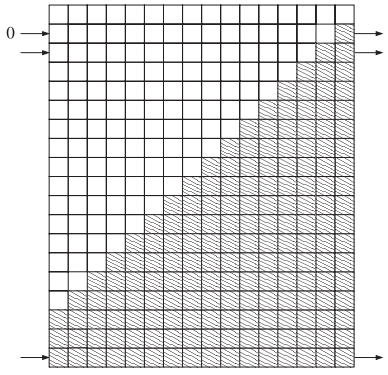


Fig. 7. Known values in the matrix A

– Pick 2^{72} messages M_i of the form

$$M_i = (m_0, \ldots, m_9, k_0, \ldots, k_5)$$

where the m_i's are chosen at random. It is straightforward to compute the matrix B for each M_i since the rightmost column of A is known. Hence we build a table T (with 2^{72} entries) where we store the rightmost column of B, i.e. the B_{15}^i's.

– Pick 2^{64} intermediate hashes H_i of the form

$$H_i = (h_0, \ldots, h_9, k_0, \ldots, k_5)$$

where the h_i's are chosen at random[1]. It is straightforward to compute the complete matrix A for each H_i. Therefore all values C_{15}^i for $i > 0$ are also known. Besides

$$H_i \oplus M_i = (*, \ldots, *, k_0 \oplus k_0, \ldots, k_5 \oplus k_5)$$

thus the 6 rightmost boxes of the first row of C are known and equal to 0. Hence a lot of information about C can be derived (see Figure 8). By the way, the bytes B_{15}^i for $11 \leq i \leq 17$ are also known at this point. We store these elements in a table T'.

The final step of the attack is to find collisions on the objects of 56 bits

$$(B_{15}^{11}, \ldots, B_{15}^{17})$$

that have been computed by two different means and stored in tables T and T'.

Using the birthday paradox, we expect 2^{80} collisions because

$$|T| \times |T'| \times 2^{-56} = 2^{72} \times 2^{64} \times 2^{-56} = 2^{80}$$

[1] Actually there is an extra constraint, that $\phi(A_0^0) = A_0^1$. Thus only 1 out of 256 values of H_i are valid. Once $(h_1, \ldots, h_9)$ are chosen, the value of h_0 is fully determined. This induce no extra cost but must be taken into account when choosing the H_i's.

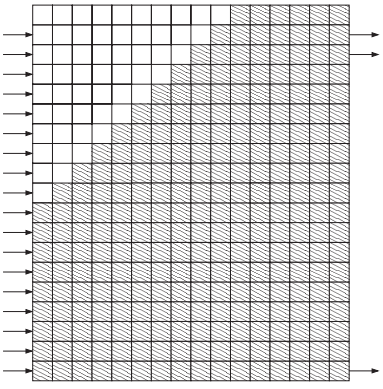


Fig. 8. Known values in the matrix C

All these collisions can be found efficiently by computing $T \bowtie T'$ (see [19]). Each collision corresponds to some pair (H_i, M_i). In order for this pair to solve the initial problem, we need an additional equality between

- the bytes $(B_{15}^1, \ldots, B_{15}^{10})$ stored in table T
- the value of the same bytes obtained when we fill up all the content of matrix C (which is possible for each candidate since $H_i \oplus M_i$ is now known).

Hence a little extra processing is required to find a real solution and a condition on 80 bits must be verified. However, we have 2^{80} candidates from the joint product of T and T' so one "real" solution should be found among them. The probability of failure (*i.e.* that no solution exists) can be roughly approximated to $\frac{1}{e} \simeq 0.368$. Otherwise, we can pick a little more candidates for M_i and H_i or choose other constants.

Analysis. The bottleneck in the previous attack is the time spent analyzing each of the 2^{80} candidates (H_i, M_i). However, using an "early-abort" strategy, most candidates can be eliminated after the first check for the value B_{15}^1. Therefore, only half a row of matrix C must be computed in average. To compute the compression function, $3 \times 18 = 54$ rows are computed. So we have a speedup by a factor

$$2 \times 54 \simeq 2^{6.75}$$

compared to a full computation of F.

Therefore this pseudo-preimage attack has complexity of about $2^{73.25}$ computations of the compression function, and requires about 2^{78} bits of memory. This is much faster than the expected value of 2^{128}. All attacks against the compression function are summarized in Table 1.

Table 1. Summary of the attacks against the compression function

Attack	Fixed Challenge	Variable	Time	Memory
Simple	H_{i+1} and H_i	M_i	2^{95}	2^{71}
Improved	H_{i+1} and H_i	M_i	2^{95}	2^{38}
Pseudo-Preimage	H_{i+1}	H_i and M_i	2^{73}	2^{78}

5 Preimage Attacks for the Full MD2 Hash

The objective of a preimage attack is, for a given challenge h, to find a message m such that hashing m with MD2 gives h:

$$\text{MD2}(m) = h$$

Classical techniques exist to turn attacks against the compression function into attacks against the full hash. However they apply to classical iterated hash functions, like those based on the Merkle-Damgård paradigm. The use of an additional checksum in MD2 make things slightly more complicated.

5.1 Attacking MD2 Without the Checksum

If we omit the checksum, it is straightforward to apply the previous attacks directly to MD2. For instance, the attack described in Section 4.2 is immediately useful. Indeed, for a given (H_i, H_{i+1}), we are able to find M_i such that:

$$H_{i+1} = F(H_i, M_i)$$

faster than exhaustive search. If we take $H_i = 0$ (*i.e.* the IV of the MD2 specifications) and $H_{i+1} = h$ (the target value), the message of 1 block $m = M_i$ basically solves the preimage problem (some extra work might be necessary to ensure the padding is correct). Anyway, this clearly no longer works when the checksum block is appended at the end.

Preimage attacks against the full hash can also be found based on a pseudo-preimage attack (like the one described in Section 4.4, with complexity 2^{73}). For instance, a general meet-in-the-middle technique is:

- Pick 2^{100} random values of the first block of message M_1, and store all intermediate hashes H_1 in a table T_1.
- Apply 2^{28} times the pseudo-preimage attack and, for each solution (H_2, M_2), store the intermediate hash H_2 in a table T_2.
- Search for a collision between some H_1 in table T_1 and some H_2 in table T_2. The corresponding message $m = (M_1, M_2)$ is a solution.

Since $2^{100} \times 2^{28} = 2^{128}$, a collision is indeed expected. Hence this attack builds a solution m of length two blocks and has complexity of the order of 2^{101}, which is faster than exhaustive search. However when the checksum is used, this input message is likely to be invalid. Indeed, we need a collision on the intermediate hash values and the intermediate checksums simultaneously.

5.2 A Chaining Attack

The principle of chaining attacks is to iterate an attack against the compression function, while chaining the intermediate variables used in each attack. Here, we first choose at random a sequence of intermediate hashes of the form:

$$0 = H_0, H_1, \ldots, H_{127}, H_{128} = h$$

For each pair (H_i, H_{i+1}), we apply the attack of Section 4.2 to find all solutions of:

$$H_{i+1} = F(H_i, M_i)$$

A constraint we add is that at least two solutions M_i and M_i' must be found, for all i. Assuming F is a random function, this should happen with a reasonable probability (called p). It can roughly be approximated by 1 minus the probability to have exactly 0 or 1 solution:

$$\begin{aligned} p &\simeq 1 - (1 - 2^{-128})^{2^{128}-1} - (1 - 2^{-128})^{2^{128}} \\ &\simeq 1 - 2\, e^{-1} \\ &\simeq 0.264 \end{aligned}$$

If there are less than 2 solutions, we throw away H_{i+1} and pick another value. In average, we need to apply $128 \times p^{-1} \simeq 2^9$ times the attack of Section 4.2 to find an appropriate pair of solutions (M_i, M_i') for all i.

Then, we have 2^{128} possible messages that are solution of the preimage problem for MD2 with challenge h (there are 2 possible blocks of message for all i). Among them, one of the message is likely to satisfy the checksum constraint, *i.e.* its last block should be the checksum of the 127 previous blocks. To find this message, a simple meet-in-the-middle attack applies:

- Compute the 2^{64} intermediate checksums IC_{64} by testing the two possible blocks of message at all positions $i, 0 \leq i \leq 63$.
- Compute the 2^{64} intermediate checksums IC_{64} by inverting the checksum function G, starting for both values M_{127} and M_{127}', and for all blocks of message at positions $i, 64 \leq i < 127$.
- Search for a collision between these 2 lists of 2^{64} elements

This technique is similar to the one used in [6]. The resulting attack against the full hash is only marginally slower than the attack against the compression function, since the deterioration corresponds to a factor 2^9. Therefore it will cost about $2^{95} \times 2^9 = 2^{104}$ applications of the compression function. In addition, a memory of 2^{71} bits is required (or 2^{38} using the improved algorithm of Appendix A). This is much faster than a naive exhaustive search.

6 Second Preimage Attacks

A second preimage attack consists, on the challenge of a message m, to provide a second message m' which gives the same MD2 hash:

$$\mathrm{MD2}(m) = \mathrm{MD2}(m')$$

The resistance of MD2 against this type of attack is critical for the security of existing certificates. Indeed a certificate generally consists in a data part m and a signature of the data part. To compute this signature, a hash of the data part is generally computed. If an attacker is able to replace m with an other data part m' mapping to the same hash, he is able to forge a new certificate.

If we omit the checksum blocks for MD2, it is straightforward to find a second preimage, based on the previous attacks. For any of the intermediate steps

$$H_{i+1} = F(H_i, M_i)$$

in the original message m, we apply the attack described in Section 4.2. With probability $p \simeq 0.26$, another message block M_i', mapping H_i to H_{i+1} is found. Then we can simply substitute M_i' to M_i to forge a new certificate.

Unfortunately, when the checksum is used, this attack no longer works because the checksum is altered by the previous substitution. Therefore the last block of message is no longer valid.

We could not find a dedicated second preimage attack against the full MD2, including the checksum bytes. An attack is still possible by applying a preimage attack on $h = \text{MD2}(m)$. The result m' is a preimage of h and is very likely to be different from m. Unfortunately m' is very constrained:

- its length is at least 128 blocks (including the checksum block), so the message m' is of length > 2 Kbytes. Some variants of the attack can increase this message length but it is not possible to reduce it. This is slightly larger than a typical certificate, however a trade-off between the size of the forged certificate and the probability of success could also be envisaged.
- at least 128 blocks in the forged certificates are random and therefore cannot be chosen by the attacker.

All together, it seems difficult for the moment to forge new certificates that respect the required format. However we are not far from it and we think it is an interesting topic for further research. We encourage a deeper analysis of the MD2 hash function whose security, especially regarding (second) preimage attacks is important for many existing certificates.

7 Conclusion

In this paper, we described preimage and pseudo-preimage attacks against the compression function of MD2, the best of which has complexity 2^{73}. The resulting attack against the full hash (including the checksum) costs about 2^{104} applications of the compression function. As a consequence, MD2 can no longer be considered a secure one-way hash function.

These results are also very interesting from a theoretical point of view, because preimage attacks against hash functions are quite rare. Most of the research in recent years has focused on finding collisions for hash functions.

References

1. D. Balenson. RFC 1423 - Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers, february 1993. RSA Laboratories.
2. F. Chabaud and A. Joux. Differential Collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lectures Notes in Computer Science*, pages 56–71. Springer, 1998.
3. H. Dobbertin. Cryptanalysis of MD4. In D. Gollmann, editor, *Fast Software Encryption – 1996*, volume 1039 of *Lectures Notes in Computer Science*, pages 53–69. Springer, 1996.
4. H. Dobbertin. The Status of MD5 after a Recent Attack. *CryptoBytes*, 2(2):1–6, 1996.
5. H. Dobbertin. The First Two Rounds of MD4 are Not One-Way. In S. Vaudenay, editor, *Fast Software Encryption – 1998*, volume 1372 of *Lectures Notes in Computer Science*, pages 284–292. Springer, 1998.
6. A. Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In *Advances in Cryptology – CRYPTO'04*, To appear.
7. B. Kaliski. RFC 1319 - The MD2 Message-Digest Algorithm, april 1992. RSA Laboratories.
8. MD5CRK, a new distributed computing project. See `http://www.md5crk.com/`.
9. A. Menezes, P. van 0orschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
10. B. Preneel. *Analysis and design of cryptographic hash functions.* PhD thesis, Katholieke Universiteit Leuven, 1993.
11. R. Rivest. The MD4 Message Digest Algorithm. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lectures Notes in Computer Science*, pages 303–311. Springer, 1991.
12. R. Rivest. RFC 1321 - The MD5 Message-Digest Algorithm, april 1992. RSA Laboratories.
13. P. Rogaway and T. Shrimpton. Cryptographic Hash Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In B. Roy and W. Meier, editors, *Fast Software Encryption – 2004*, pages 349–366, 2004. Pre-proceedings Version.
14. N. Rogier and P. Chauvaud. MD2 Is not Secure without the Checksum Byte. *Designs, Codes and Cryptography*, 12(3):245–251, november 1997. An early version of this paper was presented at the 2nd SAC Workshop in 1995.
15. B. Van Rompay, A. Biryukov, and B. Preneel. Cryptanalysis of 3-Pass HAVAL. In C. Laih, editor, *Advances in Cryptology – ASIACRYPT'03*, volume 2894 of *Lectures Notes in Computer Science*, pages 228–245. Springer, 2003.
16. RSA Laboratories. PKCS #1 v1.5 : RSA Encryption Standard, 1993. Available at `http://www.rsalabs.com/pkcs/pkcs-1`.
17. RSA Laboratories. PKCS #1 v2.1 : RSA Encryption Standard, 2002. Available at `http://www.rsalabs.com/pkcs/pkcs-1`.
18. P. van Oorschot and M. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, 12(1):1–28, 1999.
19. D. Wagner. A Generalized Birthday Problem. In M. Yung, editor, *Advances in Cryptology – Crypto'02*, volume 2442 of *Lectures Notes in Computer Science*, pages 288–303. Springer, 2002. Extended Abstract.
20. A. Wiemers. The Full Cost of Cryptanalytic Attacks. *Journal of Cryptology*, 17(2):105–124, March 2004.

A A Memory-Efficient Attack

The attack described in Section 4.2 is much faster than an exhaustive search, however the large memory requirements make it highly unpractical and probably contributes to under-estimate the "real" complexity. Here, we propose an improved attack regarding the data complexity.

The general idea of the attack of Section 4.2 is to split the target M_i in two halves $(m_0, \ldots, m_7)$ and $(m_8, \ldots, m_{15})$ of 64 bits each. The improved attack consists in **splitting** M_i **in** 4 **parts instead of** 2 using the following algorithm:

- Guess the 6 bytes $\{(B_7^1, B_7^2), (B_{15}^1, B_{15}^2), (C_7^1, C_7^2)\}$
  - guess the 4 bytes $m_0, \ldots, m_3$
    - compute and store in table T_1 the bytes $B_3^1, B_3^2, C_3^1, C_3^2$
  - guess the 4 bytes $m_4, \ldots, m_7$
    - compute and store in table T_2 the bytes $B_3^1, B_3^2, C_3^1, C_3^2$
  - guess the 4 bytes $m_8, \ldots, m_{11}$
    - compute and store in table T_3 the bytes $B_{11}^1, B_{11}^2, C_{11}^1, C_{11}^2$
  - guess the 4 bytes $m_{12}, \ldots, m_{15}$
    - compute and store in table T_4 the bytes $B_{11}^1, B_{11}^2, C_{11}^1, C_{11}^2$
  - Compute the joint product $T = T_1 \bowtie T_2$ of size 2^{32}. It contains candidate values for $(m_0, \ldots, m_7)$.
  - Compute the joint product $T' = T_3 \bowtie T_4$ of size 2^{32}. It contains candidate values for $(m_8, \ldots, m_{15})$.
  - Guess 2 additional bytes B_{15}^3 and B_{15}^4
    - For each element of T compute the 4 bytes $B_7^3, B_7^4, C_7^3, C_7^4$
    - Compute similarly these 4 bytes for each element of T'
    - Search for a collision in the two resulting lists.
  - This results in a list of 2^{32} candidates for $(m_0, \ldots, m_{15})$.

This slightly more complex attack has complexity of the order of

$$2^8 \times 2^{48} \times 2^{16} \times 2^{32} \simeq 2^{104}$$

instructions, like previously. However the largest tables we handle have 2^{32} entries of 32 bits. The philosophy of this improved attack is described in Figure 9.

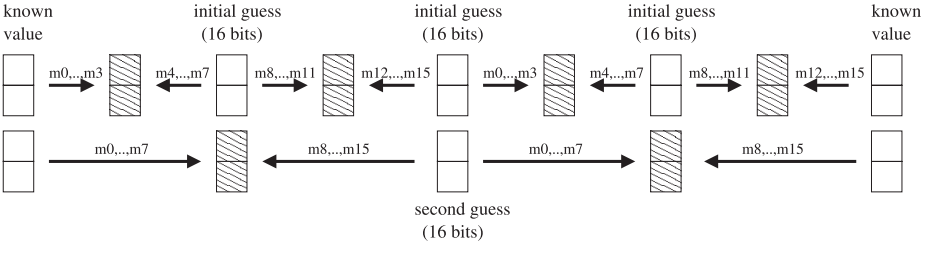


Fig. 9. The general philosophy of the improved attack

New Approaches to Password Authenticated Key Exchange Based on RSA

Muxiang Zhang

Verizon Communications Inc.,
40 Sylvan Road, Waltham, MA 02451, USA
muxiang.zhang@verizon.com

Abstract. We investigate efficient protocols for password-authenticated key exchange based on the RSA public-key cryptosystem. To date, most of the published protocols for password-authenticated key exchange were based on Diffie-Hellman key exchange. It seems difficult to design efficient password-authenticated key exchange protocols using RSA and other public-key cryptographic techniques. In fact, many of the proposed protocols for password-authenticated key exchange based on RSA have been shown to be insecure; the only one that remains secure is the SNAPI protocol. Unfortunately, the SNAPI protocol has to use a prime public exponent e larger than the RSA modulus n. In this paper, we present a new password-authenticated key exchange protocol, called *PEKEP*, which allows using both large and small prime numbers as RSA public exponent. Based on number-theoretic techniques, we show that the new protocol is secure against the *e-residue attack*, a special type of off-line dictionary attack against RSA-based password-authenticated key exchange protocols. We also provide a formal security analysis of PEKEP under the RSA assumption and the random oracle model. On the basis of PEKEP, we present a computationally-efficient key exchange protocol to mitigate the burden on communication entities.

1 Introduction

The design of authentication and key exchange protocol is usually based on the assumption that entities either share or own some secret data (called keys) which are drawn from a space so large that an adversary can not enumerate, either on-line or off-line, all possible keys in the space. In practice, however, cryptographic keys may often be substituted by human-memorable passwords consisting of only six to ten characters. The consequence is the proliferation of the so-called exhaustive guessing or dictionary attacks against many password-based systems [26]. Password guessing attacks have been around for so long, it seems paradoxical that strong authentication using only small passwords would be possible. In 1992, Bellovin and Merritt [5] showed that such paradoxical protocols did exist. They presented a family of protocols known as *Encrypted Key Exchange*, or *EKE*. By using a combination of symmetric and asymmetric (public-key) cryptographic techniques, EKE provides insufficient information for an adversary to verify a

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 230–244, 2004.

guessed password and thus defeats off-line dictionary attacks. Following EKE, a number of protocols for password-based authentication and key exchange have been proposed, e.g., [2, 6, 8, 10, 11, 13, 15–17, 21, 25] A comprehensive list of such protocols can be found in Jablon's research link [14].

Password-authenticated key exchange protocols are attractive for their simplicity and convenience and have received much interest in the research community. Over the last decade, many researchers have investigated the feasibility of implementing EKE using different public-key cryptosystems such as RSA, ElGamal, and Diffie-Hellman key exchange. Nonetheless, most of the well-known and secure variants of EKE are based on Diffie-Hellman key exchange. It seems that EKE works well with Diffie-Hellman key exchange, but presents subtleties when implemented with RSA and other public-key cryptosystems. In their original paper [5], Bellovin and Merritt pointed out that the RSA-based EKE variant is subject to a special type of dictionary attack, called *e-residue attack*. Based on number-theoretic techniques, Patel [22] further investigated the security of the RSA-based EKE variant and concluded that simple modifications of EKE would not prevent the RSA-based EKE variant from off-line dictionary attacks. In 1997, Lucks [18] proposed an RSA-based password-authenticated key exchange protocol (called OKE) which was claimed to be secure against the e-residue attack. Later, Mackenzie et al. [19] found that the OKE protocol is still subject to the e-residue attack. In [19], Mackenzie et al. proposed an RSA-based password-authenticated key exchange protocol (SNAPI) and provided a formal security proof in the random oracle model. The SNAPI protocol mandates that the public exponent e be a prime number larger than the RSA modulus n. This ensures that e is relatively prime to $\phi(n)$.

To avoid using large public exponents, Zhu et al. [27] proposed an "interactive" protocol which is revised from an idea of [5]. In the interactive protocol, Bob sends to Alice a number of messages encrypted using Alice's public key. If Alice can successfully decrypt the encrypted messages, then Bob is ensured that the encryption based on Alice's public key is a permutation. In [24], Wong et al. revised the interactive protocol of [27] to reduce the message size involved in the interactive protocol. Recently, Catalano et al. [9] proposed an interactive protocol similar to that of [24] and provided a security proof in the random oracle model. A drawback of the interactive protocols [27, 24, 9] is the large communication overhead involved in the verification of RSA public key.

In this paper, we investigate RSA-based password-authenticated key exchange protocols that can use both large and small primes as RSA public exponent, but without inducing large communication overhead on communication entities. For this purpose, we develop a new protocol for password-authenticated key exchange based on RSA. The new protocol, called *PEKEP*, involves two entities, Alice and Bob, who share a short password and Alice possesses a pair of RSA keys, n, e and d, where $ed \equiv 1 \pmod{\phi(n)}$. Unlike the protocol SNAPI, however, the new protocol PEKEP allows Alice to select both large and small primes for the RSA public exponent e. In the protocol PEKEP, Bob does not need to verify if e is relatively prime to $\phi(n)$, and furthermore, Bob does not

have to test the primality of a large public exponent selected by Alice. Thus, the protocol PEKEP improves on SNAPI by reducing the cost of primality test of RSA public exponents. The protocol PEKEP also improves on the interactive protocols of [27, 24, 9] by reducing the size of messages communicated between Alice and Bob. Based on number-theoretic techniques, we prove that the protocol PEKEP is secure against the e-residue attack as described in [5, 22]. We also provide a formal security analysis of PEKEP under the RSA assumption and the random oracle model.

To further reduce the computational load on entities, we present a computationally efficient key exchange protocol (called *CEKEP*) in this paper. The protocol CEKEP is derived from PEKEP by adding two additional flows between Alice and Bob. With the two additional flows, we show that the probability for an adversary to launch a successful e-residue attack against CEKEP is less than or equal to ε, where ε is a small number (e.g., $0 < \varepsilon \leq 2^{-80}$) selected by Bob. In the protocol CEKEP, the computational burden on Bob includes two modular exponentiations, each having an exponent of $\mathcal{O}(\lceil \log_2 \varepsilon^{-1} \rceil)$ bits. When $\varepsilon = 2^{-80}$, for example, the computational burden on Bob is lighter than that in a Diffie-Hellman based password-authenticated key exchange protocol. In the Diffie-Hellman based EKE variant, Bob needs to compute two modular exponentiations, each having an exponent of at least 160 bits.

The rest of the paper is organized as follows. We provide an overview of the security model for password-authenticated key exchange in Section 2. In Section 3, we present the protocol PEKEP and investigate its security against e-residue attack. We describe the protocol CEKEP in Section 4 and pursue formal security analysis of PEKEP and CEKEP in Section 5. We conclude in Section 6.

2 Security Model

We consider two-party protocols for authenticated key-exchange using human-memorable passwords. In its simplest form, such a protocol involves two entities, say *Alice* and *Bob* (denoted by A and B), both possessing a secret password drawn from a small password space $\mathcal{D}$. Based on the password, Alice and Bob can authenticate each other and upon a successful authentication, establish a session key which is known to nobody but the two of them. There is present an active adversary, denoted by $\mathcal{A}$, who intends to defeat the goal for the protocol. The adversary has full control of the communications between Alice and Bob. She can deliver messages out of order and to unintended recipients, concoct messages of her own choosing, and create multiple instances of entities and communicate with these instances in parallel sessions. She can also enumerate, *off-line*, all the passwords in the password space $\mathcal{D}$. She can even acquire session keys of accepted entity instances. Our formal model of security for password-authenticated key exchange protocols is based on that of [2]. In the following, we review the operations of the adversary and formulate the definition of security. For details of the security model, we refer the readers to [2].

Initialization. Let I denote the identities of the protocol participants. Elements of I will often be denoted A and B (Alice and Bob). We emphasis that A and B are variables ranging over I and not fixed members of I. Each pair of entities, $A, B \in I$, are assigned a password w which is randomly selected from the password space $\mathcal{D}$. The initialization process may also specify a set of cryptographic functions (e.g., hash functions) and establish a number of cryptographic parameters.

Running the Protocol. Mathematically, a protocol Π is a probabilistic polynomial-time algorithm which determines how entities behave in response to received message. For each entity, there may be multiple instances running the protocol in parallel. We denote the i-th instance of entity A as Π_A^i. The adversary $\mathcal{A}$ can make queries to any instance; she has an endless supply of Π_A^i oracles ($A \in I$ and $i \in \mathbb{N}$). In response to each query, an instance updates its internal state and gives its output to the adversary. At any point in time, the instance may accept and possesses a session key sk, a session id sid, and a partner id pid. The query types, as defined in [2], include:

- $\mathsf{Send}(A, i, M)$: This sends message M to instance Π_A^i. The instance executes as specified by the protocol and sends back its response to the adversary. Should the instance accept, this fact, as well as the session id and partner id will be made visible to the adversary.
- $\mathsf{Execute}(A, i, B, j)$: This call carries out an honest execution between two instances Π_A^i and Π_B^j, where $A, B \in I, A \neq B$ and instances Π_A^i and Π_B^j were not used before. At the end of the execution, a transcript is given to the adversary, which logs everything an adversary could see during the execution (for details, see [2]).
- $\mathsf{Reveal}(A, i)$: The session key sk_A^i of Π_A^i is given to the adversary.
- $\mathsf{Test}(A, i)$: The instance Π_A^i generates a random bit b and outputs its session key sk_A^i to the adversary if $b = 1$, or else a random session key if $b = 0$. This query is allowed only once, at any time during the adversary's execution.
- $\mathsf{Oracle}(M)$: This gives the adversary oracle access to a function h, which is selected at random from some probability space Ω. The choice of Ω determines whether we are working in the standard model, or in the random-oracle model (see [2] for further explanations).

Note that the Execute query type can be implemented by using the Send query type. The Execute query type reflects the adversary's ability to passively eavesdrop on protocol execution. As well shall see, the adversary shall learn nothing about the password or the session key from such oracle calls. The Send query type allows the adversary to interact with entity instances and to carry out an active man-in-the-middle attack on the protocol execution.

Definition. Let Π_A^i and Π_B^i, $A \neq B$, be a pair of instances. We say that Π_A^i and Π_B^i are *partnered* if both instances have accepted and hold the same session id sid and the same session key sk. Here, we define the sid of Π_A^i (or Π_B^i) as the concatenation of all the messages sent and received by Π_A^i (or Π_B^i). We say that Π_A^i is *fresh* if: i) it has accepted; and ii) a Reveal query has not been called either

on Π_A^i or on its partner (if there is one). With these notions, we now define the advantage of the adversary $\mathcal{A}$ in attacking the protocol. Let Succ denote the event that $\mathcal{A}$ asks a single Test query on a fresh instance, outputs a bit b', and $b' = b$, where b is the bit selected during the Test query. The advantage of the adversary $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{ake} = 2Pr(\mathsf{Succ}) - 1$.

It is clear that a polynomial-time adversary $\mathcal{A}$ can always gain an advantage close to 1 if we do not limit her capability to perform on-line password-guessing attacks. In such an attack, the adversary picks a password π as her guess and then impersonates as an instance Π_A^i to start the protocol towards another instance Π_B^j. By observing the decision of Π_B^j (i.e., accepts or rejects), the adversary can test the correctness of the guessed password π. Furthermore, by analyzing, off-line, the transcript of the execution, the adversary may be able to test passwords other than π. For a secure protocol, we expect that the adversary can only test a single password in each on-line password-guessing attack. As suggested in [10], we use the Send query type to count the number of on-line guesses performed by the adversary. We only count *one* Send query for each entity instance, that is, if the adversary sends two Send queries to an entity instance, it should still count as a single password guess. Based on this idea, we have the following definition of secure password-authenticated key exchange protocol, which is the same as in [10].

Definition 1. *A protocol Π is called a secure password-authenticated key exchange protocol if for every polynomial-time adversary $\mathcal{A}$ that makes at most Q_{send} ($Q_{send} \leq |\mathcal{D}|$) queries of* Send *type to different instances, the following two conditions are satisfied:*

(1) *Except with negligible probability, each oracle call* $\mathsf{Execute}(A, i, B, j)$ *produces a pair of partnered instances Π_A^i and Π_B^j.*
(2) $\mathsf{Adv}_{\mathcal{A}}^{ake} \leq Q_{send}/|\mathcal{D}| + \epsilon$, *where $|\mathcal{D}|$ denotes the size of the password space and ϵ is a negligible function of security parameters.*

The first condition basically says that when the adversary carries out an honest execution between two instances Π_A^i and Π_B^j (A / $\not{B}$), both instances accept and hold the same session key and the same session id. The second condition means that the advantage of the adversary is at most negligibly more than $Q_{send}/|\mathcal{D}|$ if she sends at most Q_{send} queries of Send type to different entity instances, or equivalently, if she interacts on-line with at most Q_{send} entity instances using the Send query type.

3 Password Enabled Key Exchange Protocol

In this section, we present a new protocol, called *Password Enabled Key Exchange Protocol*, or simply, *PEKEP*. In the protocol PEKEP, there are two entities, Alice and Bob, who share a password w drawn at random from the password space $\mathcal{D}$ and Alice has also generated a pair of RSA keys n, e and d, where n is a large positive integer (e.g., $n > 2^{1023}$) equal to the product of two primes of

(roughly) the same size, e is a positive integer relatively prime to $\phi(n)$, and d is a positive integer such that $ed \equiv 1 \pmod{\phi(n)}$. The encryption function is define by $E(x) = x^e \bmod n$, $x \in \mathbb{Z}_n$. The decryption function is $D(x) = x^d \bmod n$. For a positive integer m, we define E^m recursively as $E^m(x) = E(E^{m-1}(x)) = x^{e^m} \bmod n$. Likewise, $D^m(x) = D(D^{m-1}(x)) = x^{d^m} \bmod n$. Before describing the protocol, let's review some of the facts of number theory.

Let a be a positive integer relatively prime to n, we say that a is an *e-th power residue of* n if the congruence $x^e \equiv a \pmod{n}$ has a solution in $\mathbb{Z}_n^*$. Let g be a positive integer relatively prime to n. The least positive integer i such that $g^i \equiv 1 \pmod{n}$ is called the *order of* g *modulo* n. If the order of g is equal to $\phi(n)$, then g is called a *primitive root of* n. It is known (see [1, 23]) that a positive integer n, $n > 1$, possesses a primitive root if and only if $n = 2, 4, p^t$ or $2p^t$, where p is an odd prime and t is a positive integer. If n has a primitive root g, then a positive integer a relatively prime to n can be represented as $a = g^i$

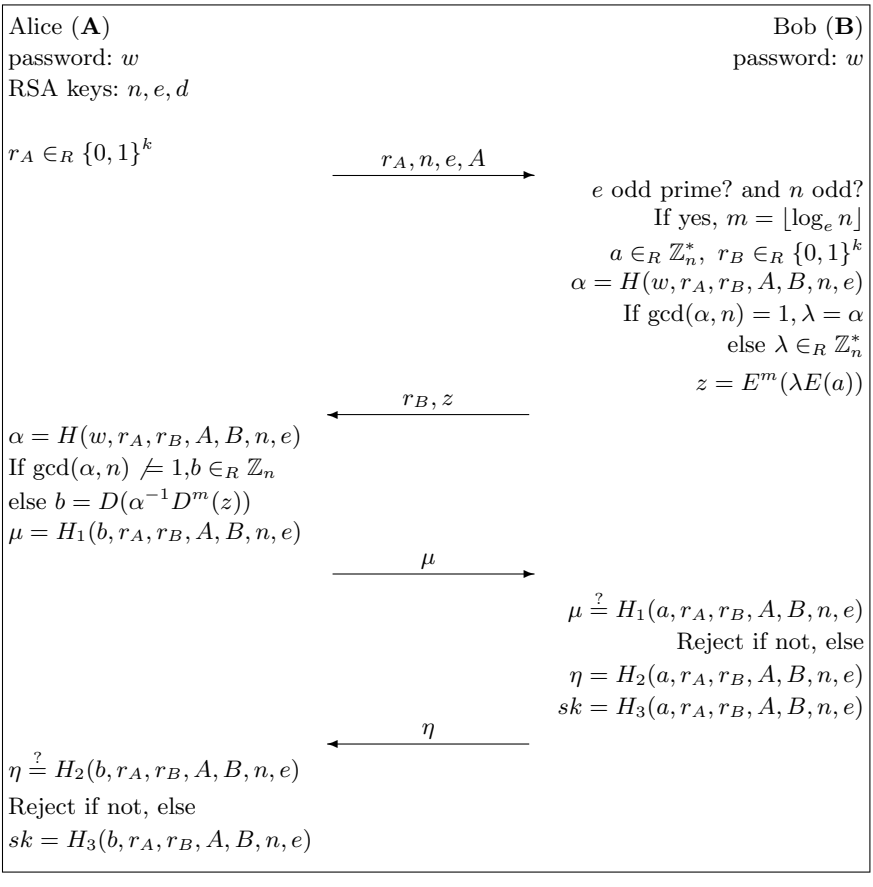


Fig. 1. Password Enabled Key Exchange Protocol (PEKEP)

mod n, $0 \leq i \leq \phi(n) - 1$. The integer i is called the *index of a to the base g modulo n*, and is denoted by $\text{ind}_g a$.

Define hash functions $H_1, H_2, H_3 : \{0,1\}^* \rightarrow \{0,1\}^k$ and $H : \{0,1\}^* \rightarrow \mathbb{Z}_n$, where k is a security parameter, e.g., $k = 160$. Note that H can be implemented using a standard hash function $h : \{0,1\}^* \rightarrow \{0,1\}^\ell$, where ℓ is the length of n, i.e., $\ell = \lceil \log_2 n \rceil$. On input x, $H(x) = h(x)$, if $h(x) < n$, and $H(x) = h(x) - \lceil n/2 \rceil$ if else. The protocol PEKEP is described in Fig. 1. Alice starts the protocol by sending her public key (n, e) and a random number $r_A \in_R \{0,1\}^k$ to Bob. Bob verifies if e is an odd prime and n is an odd integer. Bob may also verify that the integer n is large enough, e.g., $n > 2^{1023}$. If e is not an odd prime or n is not an odd integer, Bob rejects; otherwise, Bob computers an integer $m = \lfloor \log_e n \rfloor$ and selects two random numbers $a \in_R \mathbb{Z}_n^*$, and $r_B \in_R \{0,1\}^k$. Bob then computes $\alpha = H(w, r_A, r_B, A, B, n, e)$ and checks if $\gcd(\alpha, n) = 1$. If $\gcd(\alpha, n) \; / = 1$, Bob assigns a random number $\in \mathbb{Z}_n^*$ to λ; otherwise, Bob assigns α to λ. Next, Bob computes $z = E^m(\lambda E(a))$ and sends r_B and z to Alice. Subsequently, Alice computes α using her password w and checks if α and n are relatively prime. If $\gcd(\alpha, n) \; / = 1$, Alice assigns a random number $\in \mathbb{Z}_n$ to the variable b. If $\gcd(\alpha, n) = 1$ and z is an e^m-th power residue of n, Alice sets $b = D(\alpha^{-1} D^m(z))$. Next, Alice and Bob authenticate each other using a and b and generate a session key sk upon successful authentication.

Note that, when e is a prime number larger than n, Bob sets $m = 0$. In this case, the run of PEKEP is nearly identical to that of SNAPI. In the protocol PEKEP, Bob only verifies if the public exponent e is an odd prime and the RSA modulus n is an odd integer; Bob does not verify if e is relatively prime to $\phi(n)$. This may foster the so-called e-residue attack as described in [5, 22]. In the e-residue attack, an adversary, say, *Eva*, selects $\pi_0 \in \mathcal{D}$ as her guess of Alice's password. She also selects an odd prime number e and an odd integer n such that $e \mid \phi(n)$, i.e., (n, e) is not a valid RSA public key. Then Eva impersonates as Alice and starts the protocol PEKEP by sending r_E, n, e, A to Bob, where $r_E \in \{0,1\}^k$ is a random number generated by Eva. After receiving r_B and z from Bob, Eva computes μ and sends it back to Bob. If Bob accepts, then Eva has a successful guess of Alice's password (i.e., π_0). If Bob rejects, on the other hand, Eva excludes her guess π_0 from the password space $\mathcal{D}$. Furthermore, Eva may exclude more passwords by repeating, *off-line*, the following three steps:

1) Eva selects a password π from $\mathcal{D}$.
2) Eva computes $\alpha = H(\pi, r_E, r_B, A, B, n, e)$.
3) Eva tests if $\gcd(\alpha, n) = 1$. If not, Eva returns to step 1; otherwise, Eva verifies if the congruence $(\alpha x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$. If the congruence has a solution, Eva returns to step 1. If the congruence has no solution in $\mathbb{Z}_n^*$, then Eva knows that π is not the password of Alice. Next, Eva excludes π from $\mathcal{D}$ and returns to step 1.

We say that Eva succeeds if she can exclude more than one password in the e-residue attack as described above. In the following, we show that the protocol PEKEP is secure against e-residue attack.

Theorem 1. *Let n, $n > 1$, be an odd integer with prime-power factorization $n = p_1^{a_1} p_2^{a_2} \dots p_r^{a_r}$. Let m be a non-negative integer and e an odd prime such that for any prime-power $p_i^{a_i}$ of the factorization of n, $e^{m+1} \nmid \phi(p_i^{a_i}), 1 \le i \le r$. If z is an e^m-th power residue of n, then for any $\lambda \in \mathbb{Z}_n^*$, the congruence $(\lambda x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$.*

Proof. To prove that $(\lambda x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$, we only need to prove that, for each prime power $p_i^{a_i}$ of the factorization of n, the following congruence

$$(\lambda x^e)^{e^m} \equiv z \pmod{p_i^{a_i}} \tag{1}$$

has a solution in $\mathbb{Z}_{p_i^{a_i}}^*$.

Let $n_i = p_i^{a_i}, 1 \le i \le r$. Then $\phi(n_i) = p_i^{a_i - 1}(p_i - 1)$. Since n is odd, p_i is an odd prime. Thus, the integer n_i possesses a primitive root. Let g be a primitive root of n_i, that is, $g^{\phi(n_i)} = 1 \bmod n_i$, and for any $0 \le i, j \le \phi(n_i) - 1, i \not= j$, $g^i \not\equiv g^j \bmod n_i$. Let $\gcd(e^m, \phi(n_i)) = e^c, 0 \le c \le m$. We consider the following two cases:

(1) If $c = 0$, then e and $\phi(n_i)$ are relatively prime. In this case, it is clear that the congruence $(\lambda x^e)^{e^m} \equiv z \pmod{n_i}$ has a unique solution in $\mathbb{Z}_{n_i}^*$.

(2) Next, we consider the case that $1 \le c \le m$. Since z is an e^m-th power residue of n, the congruence $y^{e^m} \equiv z \pmod{n}$ has solutions in $\mathbb{Z}_n^*$. By the Chinese Remainder Theorem, the following congruence

$$y^{e^m} \equiv z \pmod{n_i} \tag{2}$$

has solutions in $\mathbb{Z}_{n_i}^*$. Let $\mathrm{ind}_g z$ denote the index of z to the base g modulo n_i and let $y \in \mathbb{Z}_{n_i}^*$ be a solution of (2), then $g^{e^m \mathrm{ind}_g y - \mathrm{ind}_g z} \equiv 1 \pmod{n_i}$. Since the order of g modulo n_i is $\phi(n_i)$, we have

$$e^m \mathrm{ind}_g y \equiv \mathrm{ind}_g z \pmod{\phi(n_i)} \tag{3}$$

Also since $\gcd(e^m, \phi(n_i)) = e^c$, equation (3) has exactly e^c incongruent solutions modulo $\phi(n_i)$ when taking $\mathrm{ind}_g y$ as variable. This indicates that equation (2) has e^c solutions in $\mathbb{Z}_{n_i}^*$. Let y_0 denote one of the solutions of (2), then the e^c incongruent solutions of (3) are given by

$$\mathrm{ind}_g y = \mathrm{ind}_g y_0 + t\phi(n_i)/e^c \pmod{\phi(n_i)}, \quad 0 \le t \le e^c - 1. \tag{4}$$

For any $\lambda \in \mathbb{Z}_n^*$, we have

$$\mathrm{ind}_g y - \mathrm{ind}_g \lambda \equiv \mathrm{ind}_g y_0 - \mathrm{ind}_g \lambda + t\phi(n_i)/e^c \pmod{\phi(n_i)}, \quad 0 \le t \le e^c - 1.$$

Under the condition that $e^{m+1} \nmid \phi(n_i)$, it is clear that $e \nmid \phi(n_i)/e^c$. Hence, $\phi(n_i)/e^c \equiv 1 \pmod{e}$. So, there exists an integer t, $0 \le t \le e^c - 1$, such that

$$\mathrm{ind}_g y_0 - \mathrm{ind}_g \lambda + t\phi(n_i)/e^c \equiv 0 \pmod{e},$$

which implies that there exists an integer $y \in \mathbb{Z}_{n_i}^*$, such that $y^{e^m} \equiv z \pmod{n_i}$ and $y\lambda^{-1}$ is an e-th power residue of n_i. Therefore, equation (1) has a solution in $\mathbb{Z}_{n_i}^*$, which proves the theorem. □

In PEKEP, Bob sets m equal to $\lfloor \log_e n \rfloor$. Thus, for every prime-power $p_i^{a_i}$ of the factorization of n, we have $e^{m+1} > n \geq p_i^{a_i}$. By Theorem 1, for any $\lambda \in \mathbb{Z}_n^*$, the congruence $(\lambda x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$, where z is the e-th power residue computed by Bob. Hence, by repeating (off-line) the three steps as described previously, Eva could not exclude any password from the space $\mathcal{D}$. So, the protocol PEKEP is secure against e-residue attacks. In Section 5, we will provide a formal analysis of PEKEP within the security model described in Section 2.

At the beginning of PEKEP, Bob needs to test the primality of the public exponent e selected by Alice. When e is small, e.g., $e = 17$, the primality test only induces moderate overhead on Bob. When e is large (e.g., $e > 2^{512}$), however, the computational load for the primality test would be tremendous. In the following, we show that primality test of large public exponents by Bob could be avoided with slight modification of PEKEP. In the protocol PEKEP, Bob can actually select a small prime number e' (e.g., $e' = 3$) and replaces Alice's public key (n, e) by (n, e'), that is, Bob computes m, α, z, η, sk using (n, e') instead of Alice's public key (n, e). Theorem 1 demonstrates that the replacement does not lead to e-residue attacks, even if e' is not relatively prime to $\phi(n)$. So, when the public exponent e received from Alice exceeds a threshold, Bob replaces e by a smaller prime number e' $(2 < e' < e)$ of his own choosing. Bob sends r_B, z, and e' to Alice in the second flow. After receiving e' from Bob, Alice tests if e' is relatively prime to $\phi(n)$. If $\gcd(e', \phi(n)) \; / = 1$, Alice sends a random number $r \in \{0,1\}^k$ to Bob; Alice may select a smaller prime number for e in the next communication session. If $\gcd(e', \phi(n)) = 1$, Alice replaces her decryption key by d' and then proceeds as specified in Fig. 1, where $e'd' \equiv 1 \pmod{\phi(n)}$.

In each run of PEKEP, Bob computes $m+1$ encryptions using Alice's public key (n, e), where $m = \lfloor \log_e n \rfloor$. The computation time for the $m+1$ encryptions is $\mathcal{O}((\log_2 n)^3)$, which means that the computational load on Bob is about the same as that in SNAPI. As discussed above, however, Bob does not have to perform primality test of large public exponents. Hence, the protocol PEKEP still improves on SNAPI by reducing the cost of primality test of RSA public exponent. Since Alice has knowledge of $\phi(n)$, she only needs to perform two decryptions in each run of PEKEP; one using the decryption key $d_1 = d$ and another using the decryption key $d_2 = d^m \bmod \phi(n)$. Note that the computational load on Bob is high even when e is small. In Section 4, we present a computationally-efficient key exchange protocol which greatly reduces the computational load on Bob.

4 Computationally-Efficient Key Exchange Protocol

In this section, we present a *Computationally-Efficient Key Exchange Protocol* (CEKEP), which is described in Fig. 2. The protocol CEKEP is based on PEKEP, but the number of encryptions performed by Bob is less than $\lfloor \log_e n \rfloor$, where (n, e) is the public key of Alice. In the protocol CEKEP, Bob selects a small number ε, $0 < \varepsilon \leq 2^{-80}$, which determines the probability of a successful e-residue attack against the protocol CEKEP. Alice starts the protocol CEKEP

by sending her public key n, e and two random numbers $\rho, r_A \in_R \{0,1\}^k$ to Bob. Bob verifies if e is an odd prime and n is an odd integer. If not, Bob rejects. Else, Bob computers an integer m based on e and ε as $m = \lceil \log_e \varepsilon^{-1} \rceil$. Then Bob selects a random number $\varrho \in_R \{0,1\}^k$ such that $\gamma = H(n, e, \rho, \varrho, A, B, m)$ is relatively prime to n. Bob sends ϱ and m to Alice. After receiving ϱ and m, Alice computes $u = D^m(\gamma)$ and sends it back to Bob. Subsequently, Bob verifies if Alice has made the right decryption, i.e., $E^m(u) = \gamma$. If $\gamma \neq E^m(u)$, Bob rejects. Else, Alice and Bob executes the rest of the protocol as in PEKEP.

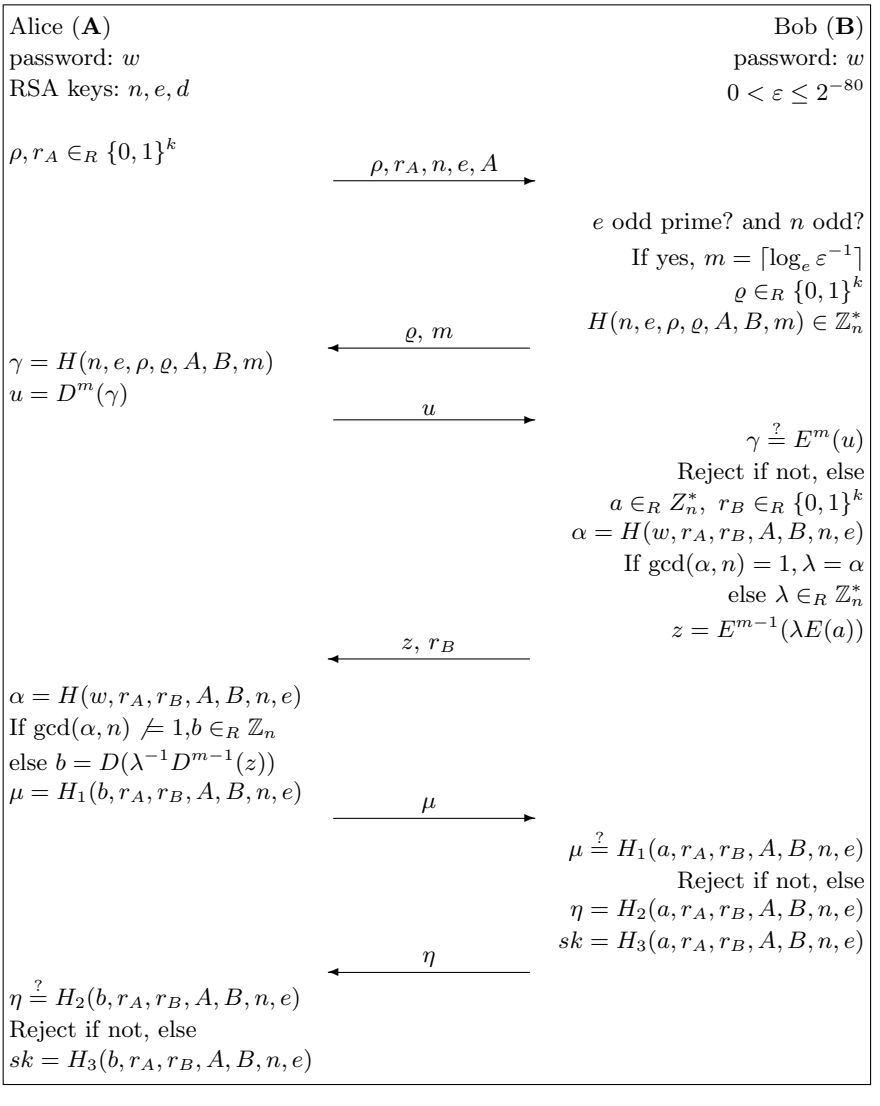


Fig. 2. Computationally-Efficient Key Exchange Protocol (CEKEP)

A major difference between CEKEP and PEKEP is that the protocol CEKEP adds two additional flows between Alice and Bob. Through the two flows, Alice and Bob establish a random number $\gamma \in \mathbb{Z}_n^*$. Then Alice decrypts the random number γ repeatedly m times. If the m repeated decryption is correct, i.e., $\gamma = E^m(u)$, then it can be concluded that, except with probability as small as e^{-m}, the integer e^m does not divide $\phi(p_i^{a_i})$ for every prime-power $p_i^{a_i}$ of the factorization of n.

Theorem 2. *Let n, $n > 1$, be an odd integer with prime-power factorization $n = p_1^{a_1} p_2^{a_2} \dots p_r^{a_r}$. Let m be a positive integer and e an odd prime. If there exists a prime power, say $p_i^{a_i}$, of the factorization of n such that $e^m \mid \phi(p_i^{a_i})$, then for an integer γ randomly selected from $\mathbb{Z}_n^*$, the probability that γ is an e^m-th power residue of n is less than or equal to e^{-m}.*

Proof. Let $n_i = p_i^{a_i}$ be a prime power of the factorization of n such that $e^m \mid \phi(n_i)$. Since n is odd, n_i possesses a primitive root. Let g be a primitive root of n_i. For an integer γ randomly selected from $\mathbb{Z}_n^*$, let $\mathrm{ind}_g\gamma$ denote the index of γ to the base g modulo n_i. Then γ is an e^m-th power residue of n_i if and only if the congruence $x^{e^m} \equiv \gamma \pmod{n_i}$ has a solution, or equivalently, if and only if

$$g^{e^m \,\mathrm{ind}_g x - \mathrm{ind}_g \gamma} \equiv 1 \pmod{n_i},$$

which is equivalent to

$$e^m \mathrm{ind}_g x \equiv \mathrm{ind}_g \gamma \pmod{\phi(n_i)}.$$

Since $e^m \mid \phi(n_i)$, γ is an e^m-th power residue of n_i if and only if $e^m \mid \mathrm{ind}_g\gamma$.

Let $n_i' = n/n_i$, then n_i and n_i' are relatively prime. For any integer $\beta \in \mathbb{Z}_n^*$, it is clear that $\beta \mod n_i$ and $\beta \mod n_i'$ are integers of $\mathbb{Z}_{n_i}^*$ and $\mathbb{Z}_{n_i'}^*$, respectively. On the other hand, for two integers $\alpha_1 \in \mathbb{Z}_{n_i}^*$ and $\alpha_2 \in \mathbb{Z}_{n_i'}^*$, by the Chinese Remainder Theorem, there is an unique integer $\alpha \in \mathbb{Z}_n^*$, such that $\alpha \equiv \alpha_1 \pmod{n_i}$, and $\alpha \equiv \alpha_2 \pmod{n_i'}$. So, the number of integers $\alpha \in \mathbb{Z}_n^*$ which satisfy the congruence $\alpha \equiv \alpha_1 \pmod{n_i}$ is $\phi(n_i')$. If γ is randomly selected from $\mathbb{Z}_n^*$, then for any integer s, $0 \le s \le \phi(n_i) - 1$, we have

$$Pr(g^s = \gamma \mod n_i) = \phi(n_i')/\phi(n) = 1/\phi(n_i),$$

which implies that $Pr(\mathrm{ind}_g\gamma = s) = 1/\phi(n_i)$. Hence,

$$\begin{aligned} Pr(e^m \mid \mathrm{ind}_g\gamma) &= \sum_{e^m \mid s,\, 0 \le s < \phi(n_i)} Pr(\mathrm{ind}_g\gamma = s) \\ &= \phi(n_i) e^{-m}/\phi(n_i) \\ &= e^{-m} \end{aligned}$$

which indicates that, for an integer γ randomly selected from $\mathbb{Z}_n^*$, the probability that γ is an e^m-th power residue of n_i is equal to e^{-m}. So, the probability that γ is an e^m-th power residue of n does not exceed e^{-m}. □

Theorem 2 demonstrates that, if there exits a prime-power $p_i^{a_i}$ of the factorization of n such that $e^m \mid \phi(p_i^{a_i})$, then for a random number $\gamma \in \mathbb{Z}_n^*$, the probability that Alice can decrypt γ repeatedly m times is less than or equal to e^{-m}. If the number u received from Alice satisfies the equation $E^m(u) = u^{e^m} = \gamma \mod n$, i.e., γ is an e^m-power residue of n, then Bob is ensured with probability greater than or equal to $1 - e^{-m}$ that, for every prime-power $p_i^{a_i}$ of the factorization of n, $e^m \nmid \phi(p_i^{a_i})$. Since $m = \lceil \log_e \varepsilon^{-1} \rceil$, $e^{-m} \leq \varepsilon$. By Theorem 1, it is clear that the probability for an adversary to launch a successful e-residue attack against CEKEP is upper-bounded by ε.

In the protocol CEKEP, Alice proves to Bob in an interactive manner (via flow 2 and flow 3) that for every prime-power $p_i^{a_i}$ of the factorization of n, $e^m \nmid \phi(p_i^{a_i})$. In the interactive procedure, however, only one decrypted message is sent from Alice to Bob. The communication overhead on Alice and Bob is greatly reduced in comparison with that in [27, 24, 9]. In CEKEP, the computational burden on Bob includes two modulo exponentiations, i.e., $u^{e^m} \mod n$ and $(\lambda a^e)^{e^{m-1}} \mod n$, where $m = \lceil \log_e \varepsilon^{-1} \rceil$. When $e < \varepsilon^{-1}$, each modulo exponentiation has an exponent consisting of $\mathcal{O}(\lceil \log_2 \varepsilon^{-1} \rceil)$ bits. The computation time for the two modulo exponentiations is $\mathcal{O}(2(\log_2 \varepsilon^{-1})(\log_2 n)^2)$. If $\varepsilon^{-1} \ll n$, then the computational load on Bob is greatly reduced in CEKEP in comparison with that in PEKEP (or in SNAPI). The parameter ε determines the computational load on Bob. It also determines the level of security against e-residue attacks. In practice, Bob can make a trade-off between the computational load and the security level offered by the protocol. When $\varepsilon = 2^{-80}$, for example, Bob needs to compute two modular exponentiation, each having an exponent of about 80 bits. In this case, the computational load on Bob is lighter than that in a Diffie-Hellman based password-authenticated key exchange protocol. In the Diffie-Hellman based EKE variant, for example, Bob needs to compute two modular exponentiation, each having an exponent of at least 160 bits.

5 Formal Security Analysis

In this section, we analyze the security of PEKEP and CEKEP within the formal model of security given in Section 2. Our analysis is based on the random-oracle model of Bellare and Rogaway [4]. In this model, a hash function is modeled as an oracle which returns a random number for each new query. If the same query is asked twice, identical answers are returned by the oracle. In our analysis, we also assume the intractability of the RSA problem.

RSA Assumption: Let ℓ be the security parameter of RSA. Let key generator GE define a family of RSA functions, i.e., $(e, d, n) \leftarrow GE(1^\ell)$, where n is the product of two primes of the same size, $\gcd(e, \phi(n)) = 1$, and $ed \equiv 1 \pmod{\phi(n)}$. For any probabilistic polynomial-time algorithm $\mathcal{C}$ of running time t, the following probability

$$\mathsf{Adv}_{\mathcal{C}}^{rsa}(t) = Pr(x^e = c \mod n : (e, d, n) \leftarrow GE(1^\ell), c \in_R \{0,1\}^\ell, x \leftarrow \mathcal{C}(1^\ell, c, e, n))$$

is negligible. In the following, we use $\mathsf{Adv}^{rsa}(t)$ to denote $\max_{\mathcal{C}}\{\mathsf{Adv}^{rsa}_{\mathcal{C}}(t)\}$, where the maximum is taken over all polynomial-time algorithms of running time t.

Theorem 3. *Let $\mathcal{A}$ be an adversary which runs in time t and makes Q_{send}, $Q_{send} \leq |\mathcal{D}|$, queries of type* Send *to different instances. Then the adversary's advantage in attacking the protocol PEKEP is bounded by*

$$\mathsf{Adv}^{ake}_{\mathcal{A}} \leq \frac{Q_{send}}{|\mathcal{D}|} + (Q_{execute} + 3Q_{send})\mathsf{Adv}^{rsa}(\mathcal{O}(t)) + \mathcal{O}(\frac{(Q_{execute} + 2Q_{send})Q_{oh}}{2^k}),$$

where $Q_{execute}$ denotes the number of queries of type Execute *and Q_{oh} denotes the number of random oracle calls.*

We prove Theorem 3 through a series of hybrid experiments. In each experiment, we modify the way session keys are chosen for instances involved in protocol execution. We start by choosing random session keys (not output by random oracles) for instances for which the Execute oracle is called. We then proceed to choose random session keys for instances for which the Send oracle is called. These instances are gradually changed over five hybrid experiments and in the last hybrid experiment, all the session keys are chosen uniformly at random. Thus, the adversary $\mathcal{A}$ can not distinguish them from random. We denote these hybrid experiments by $P_0, P_1, \ldots, P_4$ and by $\mathsf{Adv}(\mathcal{A}, P_i)$ the advantage of $\mathcal{A}$ when participating in experiment P_i. The hybrid experiment P_0 describes the real adversary attack. For $0 \leq i \leq 3$, we show that the difference between $\mathsf{Adv}(\mathcal{A}, P_i)$ and $\mathsf{Adv}(\mathcal{A}, P_{i+1})$ is negligible. We bound the advantage of $\mathcal{A}$ in P_4 by $Q_{send}/|D| + \epsilon$. Hence, the advantage of $\mathcal{A}$ in P_0 (i.e., in the real attack) is bounded by $Q_{send}/|D| + \epsilon$. Due to lack of space, the proof of Theorem 3 is omitted and can be found in the full version of this paper [28].

It is easy to check that the protocol PEKEP satisfies the first condition of Definition 1. Theorem 3 indicates that the protocol PEKEP also satisfies the second condition of Definition 1 and hence is a secure password-authenticated key exchange protocol. Similarly, we can also show that the protocol CEKEP satisfies the two conditions of Definition 1. In summary, we have the following theorem 4.

Theorem 4. *Both protocols, PEKEP and CEKEP, are secure password authenticated key exchange protocols under the RSA assumption and the random oracle model.*

We notice that the random oracle model in Theorem 4 is less desirable than a standard cryptographic assumption. To avoid the random oracle model, we could use the proof technique of [12], which require a public-key encryption scheme secure against chosen-ciphertext attacks. Unfortunately, the most commonly used RSA schemes (e.g. [3, 7]) which are secure against chosen-ciphertext attacks are also based on the random oracle model. Nevertheless, it is encouraging to see that efficient password-authenticated key exchange protocols with security proof in the random oracle model can be constructed without severe restriction on the public key of RSA.

6 Conclusion

In this paper, we investigate the design of RSA-based password-authenticated key exchange protocols that do not restrict the size of RSA public exponent. Based on number-theoretic techniques, we develop a Password Enabled Key Exchange Protocol (PEKEP) which can use both large and small primes as RSA public exponent. We show that the protocol PEKEP is secure against e-residue attacks. We also provide a formal security analysis of PEKEP under the RSA assumption and the random oracle model. Based on PEKEP, we develop a computationally-efficient key exchange protocol to mitigate the burden on communication entities. Both protocols, PEKEP and CEKEP, do not require public parameters; Alice and Bob only need to establish a shared password *in advance* and do not need to establish other common parameters such as a prime number p and a generator g of the cyclic group modulo p. This is appealing in environments where entities have insufficient resources to generate or validate public parameters with certain properties, e.g., primality.

References

1. E. Bach and J. Shallit, *Algorithmic Number Theory*, vol. 1: *Efficient Algorithms*, MIT Press, 1997.
2. M. Bellare, D. Pointcheval, and P. Rogaway, Authenticated key exchange secure against dictionary attack, *Advances in Cryptology - EUROCRYPT 2000 Proceedings*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 139-155.
3. M. Bellare and P. Rogaway, Optimal asymmetric encryption, *Advances in Cryptology - EUROCRYPT '94 proceedings*, Lecture Notes in Computer Science, vol. 950, Springer-Verlag, 1995, pp. 92–111.
4. M. Bellare and P. Rogaway, Entity Authentication and key distribution, *Advances in Cryptology - CRYPTO'93 Proceedings*, Lecture Notes in Computer Science, vol. 773, Springer-Verlag, 1994, pp. 22-26.
5. S. M. Bellovin and M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, *Proc. of the IEEE Symposium on Research in Security and Privacy*, Oakland, May 1992, pp. 72-84.
6. S. M. Bellovin and M. Merritt, Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise, *Proc. of the 1st ACM Conference on Computer and Communications Security*, ACM, November 1993, pp. 244-250.
7. D. Boneh, Simplified OAEP for the RSA and Rabin functions, *Advances in Cryptology - CRYPTO 2001 Proceedings*, Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, 2001, pp. 275-291.
8. V. Boyko, P. MacKenzie, and S. Patel, Provably secure password authenticated key exchange using Diffie-Hellman, *Advances in Cryptology - EUROCRYPT 2000 Proceedings*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 156-171.
9. D. Catalano, D. Pointcheval, and T. Pornin, IPAKE: Isomorphisms for Password-based Authenticated Key Exchange, to appear in *CRYPTO 2004 Proceedings*.

10. R. Gennaro and Y. Lindell, A framework for password-based authenticated key exchange, *Advances in Cryptology - EUROCRYPT 2003 Proceedings*, Lecture Notes in Computer Science, vol. 2656, Springer-Verlag, 2003, pp.524-542.
11. O. Goldreich and Y. Lindell, Session-key generation using human passwords only, *Advances in Cryptology - CRYPTO 2001 Proceedings*, Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, 2001, pp.408-432.
12. S. Halevi and H. Krawczyk, Public-key cryptography and password protocols, *Proc. of the Fifth ACM Conference on Computer and Communications Security*, 1998, pp. 122-131.
13. D. Jablon, Strong password-only authenticated key exchange, *Computer Communication Review, ACM SIGCOMM*, vol. 26, no. 5, 1996, pp. 5-26.
14. D. Jablon, http://www.integritysciences.com.
15. J. Katz, R. Ostrovsky, and M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, *Advances in Cryptology – EUROCRYPT 2001 Proceedings*, Lecture Notes in Computer Science, vol. 2045, Springer-Verlag, 2001.
16. K. Kobara and H. Imai, Pretty-simple password-authenticated key-exchange under standard assumptions, *IEICE Trans.*, vol. E85-A, no. 10, 2002, pp. 2229-2237.
17. T. Kwon, Authentication and key agreement via memorable passwords, *Proc. Network and Distributed System Security Symposium*, February 7-9, 2001.
18. S. Lucks, Open key exchange: How to defeat dictionary attacks without encrypting public keys, *Proc. Security Protocol Workshop*, Lecture Notes in Computer Science, vol. 1361, Springer-Verlag, 1997, pp. 79-90.
19. P. MacKenzie, S. Patel, and R. Swaminathan, Password-authenticated key exchange based on RSA, *Advances in Cryptology—ASIACRYPT 2000 Proceedings*, Lecture Notes in Computer Science, vol. 1976, Springer-Verlag, 2000, pp. 599–613.
20. A. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
21. M. Nguyen and S. P. Vadhan, Simpler session-key generation from short random passwords, *Proc. TCC 2004*, Lecture Notes in Computer Science, vol. 2951, Springer-Verlag, 2004, pp. 428-445.
22. S. Patel, Number theoretic attacks on secure password schemes, *Proc. IEEE Symposium on Security and Privacy*, Oakland, California, May 5-7, 1997.
23. K. H. Rosen, *Elementary Number Theory and Its Applications*, 4th ed., Addison Wesley Longman, 2000.
24. D. Wong, A. Chan, and F. Zhu, More efficient password authenticated key exchange based on RSA, *INDOCRYPT 2003 Proceedings*, Lecture Notes in Computer Science, vol. 2904, Springer-Verlag, 2003, pp. 375-387.
25. T. Wu, The secure remote password protocol , *Proc. Network and Distributed System Security Symposium*, San Diego, March 1998, pp. 97-111.
26. T. Wu, A real-world analysis of Kerberos password security, *Proc. Network and Distributed System Security Symposium*, February 3-5, 1999.
27. F. Zhu, D. Wong, A. Chan, and R. Ye, RSA-based password authenticated key exchange for imbalanced wireless networks, *Proc. Information Security Conference 2003 (ISC'02)*, Lecture Notes in Computer Science, vol. 2433, Springer-Verlag, 2002, pp.150-161.
28. M. Zhang, New approaches to password authenticated key exchange based on RSA, Cryptology ePrint Archive, Report 2004/033.

Constant-Round Authenticated Group Key Exchange for Dynamic Groups*

Hyun-Jeong Kim**, Su-Mi Lee, and Dong Hoon Lee***

Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea
{khj, smlee}@cist.korea.ac.kr, donghlee@korea.ac.kr

Abstract. An authenticated group key exchange (AGKE) scheme allows a group of users in a public network to share a session key which may later be used to achieve desirable cryptographic goals. In the paper, we study AGKE schemes for dynamically changing groups in *ad hoc* networks, i.e., for environments such that a member of a group may join and/or leave at any given time and a group key is exchanged without the help of any central sever. Difficulties in group key managements under such environments are caused by dynamically changing group and existence of no trustee. In most AGKE schemes proposed so far in the literature, the number of rounds is linear with respect to the number of group members. Such schemes are neither scalable nor practical since the number of group members may be quite large and the efficiency of the schemes is severely degraded with only one member's delay. We propose an efficient provably secure AGKE scheme with constant-round. The propose scheme is still contributory and efficient, where each user executes three modular exponentiations and at most $O(n)$ XOR operations.

Keywords: dynamic authenticated group key exchange, ad hoc networks.

1 Introduction

Recently, secure and efficient AGKE protocols have received much attention with increasing applicability in various collaborative and distributive group settings such as multicast communication, audio-video conference, multiplayer game, etc. In addition to provable security, the recent researches in group key exchange have concentrated on the efficiency which is related to the costs of communication and computation. Especially the number of rounds may be of critical concern in practical environment where the number of group members are quite large and

* Supported by the Ministry of Information & Communications, Korea, under the Information Technology Research Center (ITRC) Support Program.

** A guest researcher at ESAT/SCD-COSIC,K.U.Leuven from 2003 to 2004.

*** Supported by grant No. R01-2001-000-00537-0 from the Korea Science & Engineering Foundation.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 245–259, 2004.

a group is dynamic. As noted in [10], even in the case of a group where only few members have a slow network connection, the efficiency of the protocol with n rounds for a group of n members can be severely degraded. Furthermore, it is clear that a scheme with n rounds is not scalable.

In the paper, we design a secure and efficient dynamic AGKE protocol for *ad hoc* networks [26]. Most group communication environments are dynamic, where users can join and leave a group frequently. In particular, many group key exchange protocols [1, 7, 20, 23, 25] have considered *ad hoc* networks, *i.e.*, absent fixed infrastructure. IEEE 802.11 standards [18] includes as a component an ad hoc network environment such as IBSS (Independent Basic Service Set). Difficulties in designing a secure and efficient dynamic group key exchange scheme arise from the facts that a group key should be updated whenever a membership changes and exchanged without any trustee.

1.1 Overview

Related Work: The security models and provably secure protocols for authenticated static group key exchange have been first proposed by Bresson *et al.* [11], in which the security models have been based on the secure 2-party key exchange models constructed by Bellare *et al.* [2, 5, 6]. Their scheme requires $O(n)$ rounds.

Authenticated static group key exchange protocols with constant round have been proposed by Tzeng and Tzeng [24] and Boyd and Nieto [9]. In the protocol [24] with a fixed constant-round, however, the cost of communication is very high. Each member should compute n modular exponentiations for a group key exchange and additionally perform $3n$ modular exponentiations for authentications, since non-interactive proof systems are used in the authentication process. Boyd and Nieto have proven the security of the protocol [9] in the random oracle model [4]. In [9], group members consist of one member called *initiator* and other members called *responders*. While the responders only perform one signature verification, one decryption in a public cryptosystem and one operation of one-way hash function, the initiator has a heavy burden caused by $(n-1)$ encryptions in a public cryptosystem and one signature generation. Furthermore, both of these protocols [9, 24] cannot provide forward secrecy.

Katz and Yung have proposed a scalable authenticated static group key exchange protocol [19] which is based on [15] introduced by Burmester and Desmedt. Burmester and Desmedt's protocol provides 2-round and more efficient computation rate of group members than previous protocols [9, 24]; each member performs 3 modular exponentiations and $(\frac{n^2}{2} + \frac{3n}{2} - 3)$ modular multiplications. However, Burmester and Desmedt's protocol has not proposed any authentication method and any clear security proof. In [19], Katz and Yung propose a scalable compiler which transforms a secure group key exchange protocol into a secure authenticated group key exchange protocol. The compiler preserves forward secrecy of an original protocol. As Katz and Yung adapt this compiler to Burmester and Desmedt's protocol, they construct a 3-round authenticated static group key exchange protocol. Each member performs the same modular

computations as the protocol [15] and additionally performs 2 signature generations and $(2n-2)$ signature verifications. The rate of modular exponentiations in [15, 19] is constant, but still the rate of modular multiplications is dependent on the number of group members.

More recently, Bresson and Catalano have proposed a provably authenticated static group key exchange protocol with 2-round in the standard model [10]. The protocol is based on standard secret sharing techniques. The protocol is inefficient from a point of view of the computation rate. Each member should perform more than $3n$ modular exponentiations, $3n$ modular multiplications, n signature generations and n signature verifications.

For dynamic groups, Bresson *et al.* improved the protocol [11] into dynamic group key exchange protocols in [12, 13]. However, Bresson *et al.*'s protocols do not have constant-round; in their schemes, each group member embeds its secret in the intermediate keying materials and forwards the results generated with the secret to the next group member. This makes the number of rounds in setup/join algorithms linear with respect to the number of group members. Though the number of rounds in leave algorithm is constant-round, for the constant-round leave algorithm all members should store data of which length is linear with respect to the number of group members.

Bresson *et al.* [14] have introduced a provably secure authenticated group key distribution protocol with 2-round in the random oracle model [4], which is suitable for restricted power devices and wireless environments. They have concentrated on an efficient computation rate of a group member with a mobile device. In the protocol, however, there exists a base station as a trustee. The computation rate of the base station is similar to the maximum rate of group members in other protocols without any central server; n modular exponentiations, n signature verifications, n one-way hash function operations, and n XOR operations.

Our Contribution: We propose a 2-round dynamic AGKE protocol without using any trustee. All legitimate members can also detect errors and stop execution the protocol instantly, if invalid messages are broadcasted by corrupted members. For dynamic group communications, we propose *setup*, *join*, and *leave* algorithms with 2-round. In the setup algorithm, each group member performs at most 3 modular exponentiations, 4 one-way hash function operations, and n XOR operations. Since the operation dependent on the number of group members is the XOR operation, the total cost of computations can be highly reduced, compared to the previous protocols. For authentication, each group member generates 2 signatures and performs $2n$ signature verifications: this computation rate is similar to other AGKE protocols using secure signature schemes [19]. Our join/leave algorithms are executed for generations of new session keys, whenever some members join or leave. Simply, setup algorithms of static constant-round AGKE protocols can be performed, whenever group membership changes. In our join/leave algorithms, however, the communication rate and the total computation rate of group members are dependent on the number of joining/leaving members. Therefore our joining/leaving algorithms are more

efficient than the setup algorithms with the rates dependent on the number of total members, when the number of joining/leaving members is smaller than the number of remaining group members. With reduced round complexity, our protocol is still *contributory*; in our protocol, each member can participate in the generation of a group key with a one-time random value without any trust party.

In Table 1, we show efficiency analysis between our protocol and Bresson *et al.*'s dynamic AGKE protocol (**Bresson(Dyn)**) [13]. While the number of round in **Bresson(Dyn)** is depending on the number of group members, the proposed scheme is of constant round without degrading efficiency. However, our protocol cannot avoid the number of verification operations per each member increasing as like other authenticated group key exchanges [19, 24]. Our further research is to decrease or fix the number of verification operations. The following efficiency measures, *Round*, *Communication*, *Message* and *Computation* are similar to the measures defined by Katz and Yung in [19].

- *Storage*: the storage rate of a member.
- *Round*: the number of rounds during the execution of protocol.
- *Comm.*: the maximum number of bits that a member sends during the execution of protocol.

Table 1. The analysis of efficiencies

Protocol		Bresson(Dyn)	Our Protocol
Storage		Secret-$\|p\|$, Non Secret-$N\|p\|$	Secret-$3\|h\|$
Setup	Round	N	2
	Comm.	$N\|p\| + \|\sigma\|$	$\|p\| + 3\|h\| + 2\|\sigma\|$
	Mess.	$O(N^2)\|p\| + N\|\sigma\|$	$O(N)(\|p\| + \|h\| + \|\sigma\|)$
	Comp.	$Ne + s + v$	$3e + 4h + (N+1)x + 2s + O(N)v$
Join	Round	$O(N_j)$	2
	Comm.	$(N + N_j)\|p\| + \|\sigma\|$	$\|p\| + 3\|h\| + 2\|\sigma\|$
	Mess.	$O(NN_j)\|p\| + O(N_j)\|\sigma\|$	$O(N_j)(\|h\| + \|p\| + \|\sigma\|)$
	Comp.	$(N + N_j)e + 2s + N_j v$	$3e + 4h + (N_j + 1)x + 2s + O(N_j)v$
Leave	Round	1	2
	Comm.	$(N - N_\ell)\|p\| + \|\sigma\|$	$\|p\| + 3\|h\| + 2\|\sigma\|$
	Mess.	$(N - N_\ell)\|p\| + \|\sigma\|$	$N_\ell\|p\| + (N + N_\ell)(\|h\| + \|\sigma\|)$
	Comp.	$(N - N_\ell)e + s$	$3e + 4h + (N+1)x + 2s + (N_\ell + N)v$

Notations of Table1: $|\sigma|$-the length of a signature, $|h|$-the output size of a hash function, $|p|$-the length of a prime number p where p is an order of a cycle group $\mathbb{G}$; N-the number of members, N_j-the number of joining members, N_ℓ-the number of leaving members; s-the cost of a signing operation, v-the cost of a verifying operation, e-the cost of a modular exponentiation, h-the cost of a hash function operation, x-the cost of a XOR operation. Note that $|h| \leq |p|$ may be satisfied in general. We do not consider the post computation rates in our protocol.

– *Mess.*: the total length of all bits transmitted during the execution of protocol.
– *Comp.*: the maximum computation rate of a member during the execution of protocol.

2 The Model

In this section we present a security model for a dynamic AGKE protocol based on [11, 12] by Bresson *et al.* and [19] by Katz and Yung.

Participants. A nonempty set $\mathcal{U}$ is a set of users who are able to participate in an AGKE protocol $\mathcal{P}$. Each user generates secret/public key pairs (sk, pk) and the list of all public keys are known by all users. These key pairs are long-lived and used for signature generation/verification. An adversary is not a participant, but can control all communication on a network and corrupt group members.

Partnering. Whenever group membership changes, a new group $G_v = \{u_1, \cdots, u_n\}$ is formed and each group member of G_v can obtain a new session key sk_v through an instance performing $\mathcal{P}$: the index v increases whenever group membership changes and G_0 denotes the initial group. $\Pi^j_{u_i}$ denotes an instance j of a group member u_i. An instance $\Pi^j_{u_i}$ has unique session identifier $\mathtt{sid}^j_{u_i}$ and partner identifier $\mathtt{pid}^j_{u_i}$. After the group key exchange protocol $\mathcal{P}$ has been terminated successfully, $\Pi^j_{u_i}$ has a unique session key identifier $\mathtt{sk}^j_{u_i}$ corresponding to the session key sk_v. $\mathtt{pid}^j_{u_i}$ corresponds to a set of group members $G^j_{u_i} = G_v \backslash \{u_i\}$. When the group key exchange protocol $\mathcal{P}$ has been successfully terminated in the instance $\Pi^j_{u_i}$, each member u_k of $G^j_{u_i}$ has an instance $\Pi^{j_k}_{u_k}$ $(1 \leq k \neq i \leq n)$ containing $\{\mathtt{sid}^{j_k}_{u_k}, \mathtt{pid}^{j_k}_{u_k}, \mathtt{sk}^{j_k}_{u_k}\}$ such that $\mathtt{sid}^{j_k}_{u_k} = \mathtt{sid}^j_{u_i}$, $\mathtt{pid}^{j_k}_{u_k} = G_v \setminus \{u_k\}$ and $\mathtt{sk}^{j_k}_{u_k} = \mathtt{sk}^j_{u_i}$: we state that the instances $\Pi^j_{u_i}$ and $\Pi^{j_k}_{u_k}$ are *partnered* [19].

Protocol Model. A dynamic AGKE protocol $\mathcal{P}$ consists of the following algorithms:

– *Key Generation*: With an input value 1^k where k is a security parameter, this probabilistic polynomial time algorithm outputs a long-lived key for each user of $\mathcal{U}$.
– *Setup*(G_0): This algorithm starts the protocol $\mathcal{P}$ and the initial group G_0 is generated.
– *Join*$(\mathcal{J}, G_{v-1})$: Inputs to this algorithm are a set of joining members' identities denoted by $\mathcal{J}$ and the current group G_{v-1}. The output of this algorithm is a new group $G_v = G_{v-1} \cup \mathcal{J}$ and all members of G_v share a new session key sk_v secretly.
– *Leave*$(\mathcal{R}, G_{v-1})$: Inputs of this algorithm are a set of leaving members' identities denoted by $\mathcal{R}$ and the current group G_{v-1}. The output of this algorithm is a new group $G_v = G_{v-1} \setminus \mathcal{R}$ and all members of G_v share a new session key sk_v secretly.

Security Model. We define the capabilities of an adversary. We allow the adversary to potentially control all communication in the network via access to a set of oracles as defined below. We consider an *experiment* in which the adversary asks queries to oracles, and the oracles answer back to the adversary. Oracle queries model attacks which an adversary may use in the real system. We consider the following types of queries in this paper.

- $\mathsf{Send}(\Pi^j_{u_i}, m)$: $\mathcal{A}$ sends a message m to an instance $\Pi^j_{u_i}$. When $\Pi^j_{u_i}$ receives m, it responds according to the group key exchange protocol. An adversary may use this query to perform *active* attacks by modifying and inserting the messages of the key-exchange protocol. Impersonation attacks and man-in-the-middle attacks are also possible using this query.
- $\mathsf{Setup}(G_0)$, $\mathsf{Join}(\mathcal{J}, G_{v-1})$, $\mathsf{Leave}(\mathcal{R}, G_{v-1})$: Using these queries, $\mathcal{A}$ can start the *Setup*, *Join* or *Leave* algorithm.
- $\mathsf{Reveal}(\Pi^j_{u_i})$: $\mathcal{A}$ can obtain a session key $\mathtt{sk}$ which has been exchanged between the instance $\Pi^j_{u_i}$ and partnered instances, while u_i's long-lived key are concealed. This query models *known key* attacks (or Denning-sacco attacks).
- $\mathsf{Corrupt}(u_i)$: $\mathcal{A}$ can obtain u_i's long-lived key. In our protocol, we consider *adaptive corruptions* [22]; in general, adaptive corruptions mean *weak corruptions* in which an adversary can obtain an honest member's long-lived key, but cannot obtain the member's "ephemeral" keys.
- $\mathsf{Test}(\Pi^j_{u_i})$: This query is used to define the advantage of an adversary. $\mathcal{A}$ executes this query on a *fresh* instance $\Pi^j_{u_i}$ at any time, but only once (other queries have no restriction). When $\mathcal{A}$ asks this query, it receives a session key $\mathtt{sk}$ of the instance $\Pi^j_{u_i}$ if $b = 1$ or a random string if $b = 0$ where b is the result of a coin flip. Finally, $\mathcal{A}$ outputs a bit b'.

To define a meaningful notion of security, we must first define *freshness*.

Definition 1. An instance $\Pi^j_{u_i}$ is *fresh* if both the following conditions are true at the end of the experiment described above:

(a) None of the instance $\Pi^j_{u_i}$ and its partnered instances has received an adversary's Reveal query.
(b) No one of u_i and other members in $G^j_{u_i}$ has received an adversary's $\mathsf{Corrupt}$ query before adversary's Send queries.

Let $\mathcal{P}$ be a group key exchange protocol and let $\mathcal{A}$ be an active adversary against $\mathcal{P}$. When $\mathcal{A}$ asks a Test query to a fresh instance $\Pi^j_{u_i}$ in $\mathcal{P}$, $\mathcal{A}$ receives the result of the coin flip b which is either a session key or a random value and then outputs a bit b'. If the probability that $\mathcal{A}$ correctly guesses the bit b is negligible, $\mathcal{P}$ is secure in the sense that $\mathcal{A}$ cannot obtain any information about a session key through re-keying broadcast messages. Let $Adv^{agke}_{\mathcal{A},\mathcal{P}}$ denote the advantage for $\mathcal{A}$'s guess over the result of a coin-flip in a Test query with $\mathcal{P}$. Then, $Adv^{agke}_{\mathcal{A},\mathcal{P}}$ is defined as follows.

$$Adv^{agke}_{\mathcal{P},\mathcal{A}} = \Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0] = 2\Pr[b' = b] - 1.$$

We say that $\mathcal{P}$ is a secure AGKE if $Adv^{agke}_{\mathcal{P}} = \max_{\mathcal{A}}\{Adv^{agke}_{\mathcal{P},\mathcal{A}}\}$ is negligible.

For the security of authentication, we consider the ability of $\mathcal{A}$ for impersonation attacks against a group member u_i in an instance $\Pi^j_{u_i}$ [11]. For impersonation attacks, $\mathcal{A}$ should be able to forge a signature of the group member u_i in the instance $\Pi^j_{u_i}$. If it is computationally infeasible that $\mathcal{A}$ generates a valid signature with any message under a chosen message attack, we say that the signature scheme is CMA-secure. Let $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme where $\mathcal{K}, \mathcal{S}$ and $\mathcal{V}$ are *key generation*, *signing* and *verification* algorithms. Formally, let $Succ^{cma}_{\Sigma,\mathcal{A}}$ be a success probability of $\mathcal{A}$'s existential forgery under a chosen message attack against Σ. Then, we state that Σ is CMA-secure [21] if $Succ^{cma}_{\Sigma} = \max_{\mathcal{A}}\{Succ^{cma}_{\Sigma,\mathcal{A}}\}$ is negligible.

Let $\mathsf{G} = \langle g \rangle$ be a group. Given g^x and g^y, CDH problem is to compute a value g^{xy} [17]. For the CDH problem, we consider a probability $Succ^{cdh}_{\mathbb{G}}$ such that

$$Succ^{cdh}_{\mathbb{G},\mathcal{A}} = \Pr\left[C = g^{xy} | g^x, g^y \leftarrow \mathbb{G}; C \leftarrow \mathcal{A}(g^x, g^y)\right],$$
$$Succ^{cdh}_{\mathbb{G}} = \max_{\mathcal{A}}\{Succ^{cdh}_{\mathbb{G},\mathcal{A}}\}$$

where $\mathcal{A}$ is a CDH attacker against a group $\mathbb{G}$.

3 A Constant-Round AGKE Protocol

Our protocol is based on the Computational Diffie-Hellman (CDH) assumption and a secure signature scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$. A group key space belongs to $\{0,1\}^\ell$ where ℓ is a security parameter. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. g and p are public parameters and $\ell \leq |p|$ is satisfied. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ be a one-way hash function.

Key Generation. Each user u_i of $\mathcal{U}$ has a private/public key pair (sk_{u_i}, pk_{u_i}) for signing/verifying. The list of public keys is published to all users.

Setup. Let $G_0 = \{u_1, \cdots, u_n\}$ be an initial group. We consider a ring structure among the members of G_0, *i.e.*, members' indices could be considered on the circulation of $\{1, \cdots, n\}$. $L(i)$ ($R(i)$, resp.) means the left (right, resp.) index of i on the ring for $i \in \{1, \cdots, n\}$. Let $I_0 = ID_{u_1}||\cdots||ID_{u_n}$. Figure 1 shows the example of this algorithm with four members.

- **Round 1.** Each member u_i randomly chooses $k_i \in \{0,1\}^\ell$ and $x_i \in \mathbb{Z}^*_p$, computes $y_i = g^{x_i}$ and keeps k_i secretly. The last member u_n computes $\mathcal{H}(k_n||0)$. Each member u_i generates a signature $\sigma^1_i = \mathcal{S}_{sk_{u_i}}(M^1_i||I_0||0)$ where $M^1_i = y_i$ for $1 \leq i \leq n-1$ and $M^1_n = \mathcal{H}(k_n||0)||y_n$, and broadcasts $M^1_i||\sigma^1_i$.
- **Round 2.** All members receive $(M^1_i||\sigma^1_i)$'s and verify σ^1_i's. If some signatures are not valid, this process fails and halts. Otherwise, u_i computes $t^L_i = \mathcal{H}(y^{x_i}_{L(i)}||I_0||0)$, $t^R_i = \mathcal{H}(y^{x_i}_{R(i)}||I_0||0)$ and generates $T_i = t^L_i \oplus t^R_i$. The last member u_n additionally computes $\widehat{T} = k_n \oplus t^R_n$. Each member u_i generates $\sigma^2_i = \mathcal{S}_{sk_{u_i}}(M^2_i||I_0||0)$ and broadcasts $M^2_i||\sigma^2_i$ where $M^2_i = k_i||T_i$ for $1 \leq i \leq n-1$ and $M^2_n = \widehat{T}||T_n$.

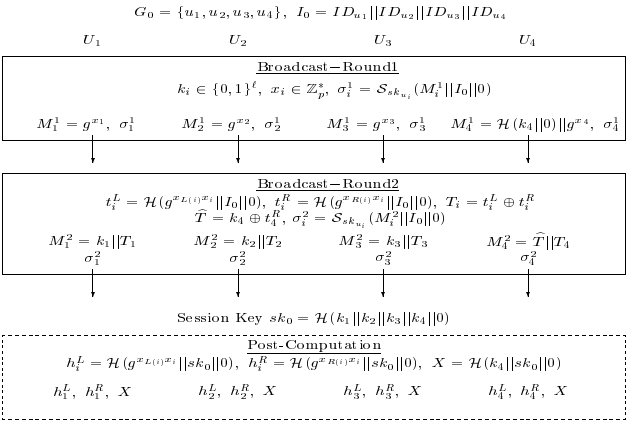


Fig. 1. *Setup* algorithm with $G_0 = \{u_1, u_2, u_3, u_4\}$

– **Key Computation.** SESSION KEY COMPUTATION: All members verify signatures σ_i^2's. If all signatures are valid, u_i computes $\widetilde{t}_{i+1}^R, \widetilde{t}_{i+2}^R, \cdots, \widetilde{t}_{i+(n-1)}^R$ $(= \widetilde{t}_i^L)$ by using t_i^R:

$$\widetilde{t}_{i+1}^R = T_{i+1} \oplus t_i^R,\ \ \widetilde{t}_{i+2}^R = T_{i+2} \oplus \widetilde{t}_{i+1}^R,\ \ \cdots,\ \ \widetilde{t}_{i+(n-1)}^R = T_{i+(n-1)} \oplus \widetilde{t}_{i+(n-2)}^R.$$

Finally u_i can check if $t_i^L = \widetilde{t}_i^L$ holds. Even though wrong messages (or no message) are broadcasted by illegal members or system faults, honest members can notice the errors through the above check process and then halt the protocol. However, it is not easy to find who transmitted illegal messages. When members want to find illegal members, all members participating in this protocol should reveal their secret values x_i's. If the above check process has been valid, all members have $\widetilde{t}_n^R$ $(= t_n^R)$. Then they can obtain $\widetilde{k}_n$ from $\widehat{T}$ and check if $\mathcal{H}(\widetilde{k}_n||0) = \mathcal{H}(k_n||0)$ holds. Note that *Key Control* can be guaranteed by this check value and the one-way hash function $\mathcal{H}$. All members compute a session key like as

$$sk_0 = \mathcal{H}(k_1||k_2||\cdots||k_{n-1}||k_n||0).$$

POST-COMPUTATION: Each member u_i generates $h_i^L = \mathcal{H}(y_{L(i)}^{x_i}||sk_0||0)$, $h_i^R = \mathcal{H}(y_{R(i)}^{x_i}||sk_0||0)$ and $X = \mathcal{H}(k_n||sk_0||0)$ and saves (h_i^L, h_i^R, X, sk_0) secretly. All members should erase other ephemeral data.

Join. Let $G_{v-1} = \{u_1, \cdots, u_n\}$ $(v \geq 1)$ be the current group and $\mathcal{J} = \{u_{n+1}, \cdots, u_{n+n'}\}$ $(n' \geq 1)$ be a set of new members. We divide G_{v-1} into three parts $\{u_1\}$, $\{u_2, \cdots, u_{n-1}\}$ and $\{u_n\}$, and consider u_2 as an agent of $\{u_2, \cdots, u_{n-1}\}$. For convenience of explanation, we allow that $u_{n+n'+1}$, $u_{n+n'+2}$ and $u_{n+n'+3}$ denote u_1, u_2 and u_n. In this algorithm, we consider a ring structure among the members $u_{n+1}, \cdots, u_{n+n'+3}$. Let $\mathcal{G}$ be the set $\{u_{n+1}, \cdots, u_{n+n'+3}\}$ and $I_v = ID_{u_1}||\cdots||ID_{u_{n+n'}}$. Figure 2 shows the example of this algorithm.

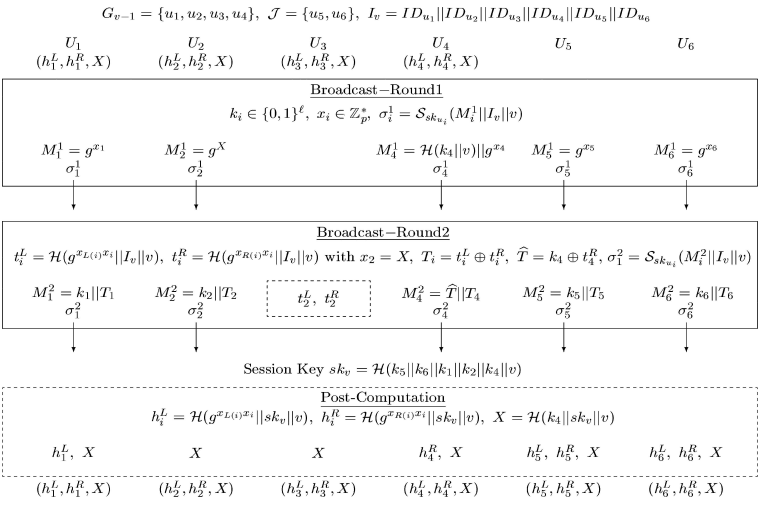


Fig. 2. *Join* algorithm with $G_{v-1} = \{u_1, u_2, u_3, u_4\}$ and $\mathcal{J} = \{u_5, u_6\}$

- **Round 1.** Each member u_{n+i} of $\mathcal{G}$ randomly chooses $k_{n+i} \in \{0,1\}^\ell$ and $x_{n+i} \in \mathbb{Z}_p^*$, computes $y_{n+i} = g^{x_{n+i}}$ and keeps k_{n+i} secretly. The member $u_{n+n'+2}$ $(= u_2)$ computes $y_{n+n'+2} = g^X$ by using the secret value X instead of $x_{n+n'+2}$ and the member $u_{n+n'+3}$ $(= u_n)$ computes $\mathcal{H}(k_{n+n'+3}||v)$. Each member u_{n+i} generates $\sigma_{n+i}^1 = \mathcal{S}_{sk_{u_{n+i}}}(M_{n+i}^1||I_v||v)$ where $M_{n+i}^1 = y_{n+i}$ for $1 \le i \le n'+2$ and $M_{n+n'+3}^1 = \mathcal{H}(k_{n+n'+3}||v)||y_{n+n'+3}$, and broadcasts $M_{n+i}^1||\sigma_{n+i}^1$.
- **Round 2.** All members receive $(M_{n+i}^1||\sigma_{n+i}^1)$'s and verify σ_{n+i}^1's. Each member u_{n+i} computes $t_{n+i}^L = \mathcal{H}(y_{L(n+i)}^{x_{n+i}}||I_v||v)$, $t_{n+i}^R = \mathcal{H}(y_{R(n+i)}^{x_{n+i}}||I_v||v)$ and generates $T_{n+i} = t_{n+i}^L \oplus t_{n+i}^R$. The member $u_{n+n'+3}$ additionally computes $\widehat{T} = k_{n+n'+3} \oplus t_{n+n'+3}^R$. Each member u_{n+i} generates $\sigma_{n+i}^2 = \mathcal{S}_{sk_{u_{n+i}}}(M_{n+i}^2||I_v||v)$ and broadcasts $M_{n+i}^2||\sigma_{n+i}^2$ where $M_{n+i}^2 = k_{n+i}||T_{n+i}$ for $1 \le i \le n'+2$ and $M_{n+n'+3}^2 = \widehat{T}||T_{n+n'+3}$. All members of $\{u_3, \cdots, u_{n-1}\}$ compute $t_{n+n'+2}^L$ and $t_{n+n'+2}^R$ by using X.
- **Key Computation.** Session Key Computation: All members verify σ_{n+i}^2's. If all signatures are valid, each member u_{n+i} computes $\widetilde{t}_{n+i+1}^R, \cdots, \widetilde{t}_{n+i+n'-1}^R (= \widetilde{t}_{n+i}^L)$ by using t_{n+i}^R and checks if $t_{n+i}^L = \widetilde{t}_{n+i}^L$ holds. Also, the members $u_3, \cdots, u_{n-1}$ can check it by using $t_{n+n'+2}^L$ and $t_{n+n'+2}^R$. Finally all members can obtain $k_{n+n'+3}$ from $\widehat{T}$ and compute a new session key sk_v as follows:
$$sk_{v+1} = \mathcal{H}(k_{n+1}|| \cdots ||k_{n+n'+3}||v).$$
Post-Computation: Each new member u_{n+i} $(1 \le i \le n')$ generates $h_{n+i}^L = \mathcal{H}(y_{L(n+i)}^{x_{n+i}}||sk_v||v)$ and $h_{n+i}^R = \mathcal{H}(y_{R(n+i)}^{x_{n+i}}||sk_v||v)$. u_1 and u_n respectively compute $h_1^L = \mathcal{H}(y_{n+n'}^{x_1}||sk_v||v)$ and $h_n^R = \mathcal{H}(y_{n+1}^{x_n}||sk_v||v)$ instead of the previous value h_1^L $(= h_n^R)$. All members compute a new value $X = \mathcal{H}(k_n||sk_v||v)$. Each member u_i saves h_i^L, h_i^R, X and sk_v secretly.

Leave. Let $G_{v-1} = \{u_1, u_2, \cdots, u_n\}$ be the current group and $\mathcal{R} = \{u_{l_1}, u_{l_2}, \cdots, u_{l_{n''}}\}$ with $\{l_1, \cdots, l_{n''}\} \subset \{1, 2, \cdots, n\}$ be a set of revoked members. Let $\mathcal{N}(\mathcal{R})$ be a set of all left/right members of revoked members, *i.e.*, $\mathcal{N}(\mathcal{R}) = \{u_{l_1-1}, u_{l_1+1}, \cdots, u_{l_{n''}-1}, u_{l_{n''}+1}\}$. For generating a new group $G_v = G_{v-1} \setminus \mathcal{R}$ with a new session key s_v, a new Diffie-Hellman value should be shared between two members u_{l_j-1} and u_{l_j+1} $(1 \leq j \leq n'')$. In this algorithm, we consider a ring structure among members of G_v and we newly index the members as $G_v = \{u_1, u_2, \cdots, u_{n-n''}\}$. Let $I_v = ID_{n_1}||\cdots||ID_{n-n''}$. Figure 3 shows the example of this algorithm.

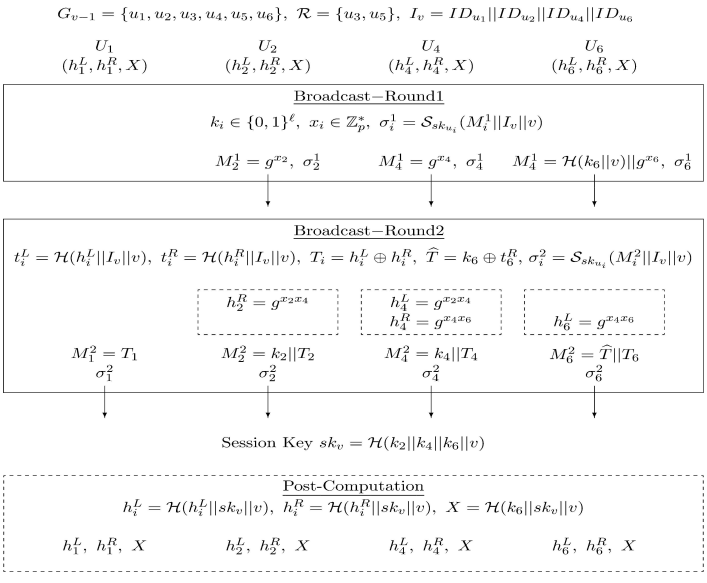


Fig. 3. *Leave* algorithm with $G_{v-1} = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ and $\mathcal{R} = \{u_3, u_5\}$

- **Round 1.** Each member u_w of $\mathcal{N}(\mathcal{R})$ randomly chooses $k_w \in \{0,1\}^\ell$ and $x_w \in \mathbb{Z}_p^*$, computes $y_w = g^{x_w}$ and keeps k_w secretly. The member $u_{l_{n''}+1}$ computes $\mathcal{H}(k_{l_{n''}+1}||v)$. u_w generates $\sigma_w^1 = \mathcal{S}_{sk_{u_w}}(M_w^1||I_v||v)$ where $M_w^1 = y_w$ with $w \in \{l_1 - 1, l_1 + 1, \cdots, l_{n''} - 1\}$ and $M_{l_{n''}+1}^1 = \mathcal{H}(k_{l_{n''}+1}||v)||y_{l_{n''}+1}$, and broadcasts $M_w^1||\sigma_w^1$.
- **Round 2.** All members of G_v verify signatures σ_w^1's. If all signatures are valid, each member u_{l_j-1} (resp. u_{l_j+1}) of $\mathcal{N}(\mathcal{R})$ regenerates $h_{l_j-1}^R = y_{l_j+1}^{x_{l_j-1}}$ (resp. $h_{l_j+1}^L = y_{l_j-1}^{x_{l_j+1}}$). Then each member u_i of G_v computes $t_i^L = \mathcal{H}(h_i^L||I_v||v)$, $t_i^R = \mathcal{H}(h_i^R||I_v||v)$ and $T_i = t_i^L \oplus t_i^R$. The member $u_{l_{n''}+1}$ additionally computes $\widehat{T} = k_{l_{n''}+1} \oplus t_{l_{n''}+1}^R$. Each member u_i generates a signature $\sigma_i^2 = \mathcal{S}_{sk_{u_i}}(M_i^2||I_v||v)$ and broadcasts $M_i^2||\sigma_i^2$ where $M_{l_{n''}+1}^2 = \widehat{T}||T_{l_{n''}+1}$, $M_i^2 = k_i||T_i$ for other members except $u_{l_{n''}+1}$ of $\mathcal{N}(\mathcal{R})$ and $M_i^2 = T_i$ for members of $G_v \setminus \mathcal{N}(\mathcal{R})$.

– **Key Computation.** Session Key Computation: All members verify signatures σ_i^2's. If all signatures are valid, each member u_i computes $\widetilde{t}_{i+1}^R, \widetilde{t}_{i+2}^R, \cdots, \widetilde{t}_{i+(n-n''-1)}^R (= \widetilde{t}_i^L)$ by using t_i^R. Finally, each member u_i checks if $t_i^L = \widetilde{t}_i^L$ holds. Then all members computes a session key as follows:

$$sk_v = \mathcal{H}(k_{l_1-1}||k_{l_1+1}||\cdots||k_{l_{n''}-1}||k_{l_{n''}+1}||v).$$

Post-Computation: Each member u_i regenerates $h_i^L = \mathcal{H}(h_i^L||sk_v||v)$, $h_i^R = \mathcal{H}(h_i^R||sk_v||v)$ and $X = \mathcal{H}(k_{l_{n''}+1}||sk_v||v)$ and saves h_i^L, h_i^R, X and the session key sk_v secretly.

4 The Security

In this section, we prove the security of our protocol in the random oracle model.

4.1 Security Proof

The security of our protocol $\mathcal{P}$ is dependent on the probabilities $Succ_{\Sigma}^{cma}$ and $Succ_{\mathbb{G}}^{cdh}$, since an adversary $\mathcal{A}$ against $\mathcal{P}$ can obtain information about a session key only by two methods: $\mathcal{A}$ successfully performs either signature forgery attacks or CDH attacks. Even if random values k_i's were selected identically in different instances, $\mathcal{A}$ could not get any information about a session key because of the index v and the random hash oracle $\mathcal{H}$. Our proof method is similar to that in [14].

Theorem 1. *Let $\mathcal{A}$ be an active adversary against our protocol $\mathcal{P}$ in the random oracle model. Let q_s be the number of* Send *queries and q_H be the number of queries to the hash oracle $\mathcal{H}$. Then,*

$$Adv_{\mathcal{P}}^{agke} \leq 2n \cdot Succ_{\Sigma}^{cma}(t, q_s) + 2q_H q_s^2 \cdot Succ_{\mathbb{G}}^{cdh}(t)$$

where n is the maximum number of group members and t is the adversary's running time.

Proof. We consider $\mathcal{A}$'s attacks as a sequence of simulated protocols, which is denoted by a sequence of games $\{\mathsf{Game}_0, \cdots, \mathsf{Game}_3\}$. In each game, $\mathcal{A}$ executes Test query and get a result of a coin flip b. Each $\mathbf{Succ}_i$ denotes an event in which $\mathcal{A}$'s a guessing bit b' is equal to b in each Game_i. Each Game_i is simulated as follows:

Game_0: This game is equal to the real protocol $\mathcal{P}$. All group members obtain a pair of valid signing/verifying key and randomly choose k_i's and x_i's. In this game, $\mathcal{A}$'s advantage is equal to the advantage in the real protocol $\mathcal{P}$. Thus,

$$\Pr[\mathbf{Succ}_0] = \frac{Adv_{\mathcal{P},\mathcal{A}}^{agke} + 1}{2} \tag{1}$$

Game_1: In this game, we consider a special event SigForge in which $\mathcal{A}$ executes a Send query with a message $\overline{m}$ instead of a group member u_i in an instance $\Pi_{u_i}^j$

and the message is verified and accepted by all group members. In particular, the message $\overline{m}$ previously has not been used in any instances and a $\mathsf{Corrupt}(u_i)$ query has not been executed to the member u_i. When the event $\mathsf{SigForge}$ occurs, this game halts and $\mathcal{A}$'s output b' is determined randomly. The difference between $\mathcal{A}$'s outputs in games Game_0 and Game_1 is dependent on the event $\mathsf{SigForge}$. That is,

$$|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_0]| \leq \Pr[\text{SigForge}].$$

If one correctly guesses a member impersonated by $\mathcal{A}$ and the event $\mathsf{SigForge}$ occurs to the member, one can be suceessful in the existential forgery against a pair of signing/verifying key under CMA. Therefore we know that

$$Succ^{cma}_{\Sigma,\mathcal{A}}(t, q_s) \geq \tfrac{1}{n} \Pr[\text{SigForge}].$$

Finally, we get

$$|\Pr[\mathbf{Succ}_1] - \Pr[\mathbf{Succ}_0]| \leq \Pr[\text{SigForge}] \leq n \cdot Succ^{cma}_{\Sigma,\mathcal{A}}(t, q_s) \tag{2}$$

Game_2: In this game, a Diffie-Hellman triple $(A = g^a, B = g^b, C = g^{ab})$ is given. Whenever two successive members u_i and u_{i+1} should choose random values x_i and x_{i+1} and compute $y_i = g^{x_i}$ and $y_{i+1} = g^{x_{i+1}}$, we simulate this game with $y_i = A^{c_i}$ and $y_{i+1} = B^{c_{i+1}}$ where c_i and c_{i+1} are random values in $\mathbb{Z}^*_p$. Then a hash value t^R_i $(= t^L_{i+1})$ is computed by using $C^{c_i c_{i+1}}$. We know that this game is equal to Game_1 as long as c_i and c_{i+1} are selected randomly. Therefore,

$$\Pr[\mathbf{Succ}_2] = \Pr[\mathbf{Succ}_1] \tag{3}$$

Game_3: In this game, a pair $(A = g^a, B = g^b)$ is given and there is no information about the Diffie-Hellman value $C = g^{ab}$. Whenever two successive members u_i and u_{i+1} should choose random values x_i and x_{i+1} and compute y_i and y_{i+1}, we simulate this game like Game_2. However, when u_i or u_{i+1} should broadcast a message with a hash value t^R_i $(= t^L_{i+1})$, a random value r in $\{0,1\}^\ell$ is used as the hash value. Now, we consider an event Hash in which $\mathcal{A}$ detects the fact that the broadcasted hash value t^R_i (or t^L_{i+1}) is incorrect by using $\mathcal{A}$'s hash oracle queries. This event is possible when $\mathcal{A}$ sends a correctly guessing value $C^{c_i c_{i+1}}$ to the hash oracle H and receives a hash value. At that time, $\mathcal{A}$ recognizes that the value is different from the previous random value r. When the event Hash occurs, this game is halted and $\mathcal{A}$'s output b' is randomly chosen. Therefore,

$$|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_2]| \leq \Pr[\text{Hash}].$$

Given (A, B) one can obtain a valid Diffie-Hellman value C if both of the following situations occur; (1) two successive members compute $y_i = A^{c_i}$ and $y_{i+1} = B^{c_{i+1}}$ and use a random value r as a hash value t^R_i, (2) $\mathcal{A}$ executes a hash oracle query with a correctly guessing value $C^{c_i c_{i+1}}$ after (1), *i.e.*, the event Hash occurs. Therefore

$$Succ^{cdh}_{\mathbb{G},\mathcal{A}}(t) \geq \tfrac{1}{q_H q_s^2} \Pr[\text{Hash}].$$

Finally, we get

$$|\Pr[\mathbf{Succ}_3] - \Pr[\mathbf{Succ}_2]| \leq \Pr[\mathrm{Hash}] \leq q_H q_s^2 \cdot Succ^{cdh}_{\mathbb{G},\mathcal{A}}(t) \quad (4)$$

Furthermore, $\mathcal{A}$ has no advantage for guessing a coin-flip bit b in this game since the hash oracle $\mathcal{H}$ has been supposed the random oracle and each input of the hash oracle is used only once owing to the index v. Therefore $\Pr[\mathbf{Succ}_3] = \frac{1}{2}$.

From the above results, the theorem is proved. □

4.2 Forward Secrecy of AGKE Protocol

For a secure group key exchange protocol, forward secrecy is one of essential security requirements. *Forward secrecy* means that the compromise of one or more members' long-lived keys should give no information for the compromise of any earlier session key.

In an AGKE protocol, a member's long-lived key is the member's signing key for authentication. Most dynamic AGKE protocols have considered an adversary with weak corruption ability and have guaranteed forward secrecy for a member's signing key. Bresson *et al.*'s protocols [12, 13] have offered forward secrecy for a member's signing key. However, another secret value of a member (really, it is an exponent) is as important as a signing key. If a member's exponent is revealed, earlier session keys can be revealed as well as later session keys. Furthermore, unless a member is a leader of a group, the member's secret exponent key never changes from joining to leaving. Therefore the secret exponent should be definitely considered as a long-lived key, even though the value is saved in a smart card. Also, in Bresson *et al.* [14], forward secrecy for a member's signing key can be guaranteed, but forward secrecy for a member's Diffie-Hellman value cannot be guaranteed: a member's Diffie-Hellman value is never changed until the member leaves. Therefore this value should be also considered as a long-lived key.

In our AGKE protocol, a member secretly keeps a signing key as a long-lived key and three hashed values as short-lived keys: every time a session key is changed, member's short-lived keys are also changed. In the paper we consider and prove forward secrecy for member's long-lived key, but our protocol also guarantees forward secrecy for member's short-lived keys. When an adversary obtain some members' short-lived keys, it can obtain later session keys, but cannot obtain previous session keys easily. Therefore our protocol can guarantee forward secrecy against an adversary with strong corruption capability [22] in which an advasary can obtain a member's short-lived key as well as the member's long-lived key.

5 Conclusion

We have proposed an efficient and secure constant-round AGKE protocol for dynamic groups in the random oracle model. We note that each membership

change in dynamic group could be handled by running other constant round static AGKE protocols from scratch. But our *Join* and *Leave* algorithms are more efficient than *Setup* algorithms of other constant round AGKE protocols for static groups when the number of joining/leaving members is smaller than the number of remaining group members. Hereafter, research in a provably secure constant-round AGKE protocol for dynamic groups under standard assumptions should be studied.

References

1. N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. In *Computer Communication*, pp.1627-1237, 2000.
2. M. Bellare, R. Canetti and H. Krawczyk. A Modular Approach to The Design and Analysis of Authentication and Key-Exchange Protocols. In *Proc. of the 30th Annual Symposium on the Theory of Computing (STOC)*, ACM Press, 1998.
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology - Eurocrypt'00*, LNCS 1807, Springer-Verlag, pp.139-155, 2000.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS'93*, 1993.
5. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology - Crypto'93*, LNCS 773, Springer-Verlag, pp.232-249, 1994.
6. M. Bellare and P. Rogaway. Provably Secure Session Key Distribtuion: The Three Party Case. In *Proc. of the 27th Annual Symposium on the Theory of Computing*, ACM Press, 1995.
7. J. Binkley. Authenticated Ad Hoc Routing at The Link Layer for Mobile Systems. In *Wireless Network*, 1999.
8. V. Boyko, P.D. MacKenzie and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In *Proc. of Eurocrypt 2000*, LNCS 1807, Springer-Verlag, pp.156-171, 2000.
9. C. Boyd and J.M.G. Nietio. Round-Optimal Contributory Conference Key Agreement. In *Proc. of Public-Key Cryptography*, LNCS 2567, Springer-Verlag, pp.161-174, 2003.
10. E. Bresson and D. Catalano. Constant Round Authenticated Group Key Agreement via Distributed Computation. In *Proc. of PKC 2004*, LNCS 2947, Springer-Verlag, pp.115-129, 2004.
11. E. Bresson, O. Chevassut and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proc. of the 8th ACM Conference on Computer and Communications Security*, pp.255-264, 2001.
12. E. Bresson, O. Chevassut and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange-The Dynamic Case. In *Advances in Cryptology - Asiacrypt'01*, LNCS 2248, Springer-Verlag, pp.290-309, 2001.
13. E. Bresson, O. Chevassut and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *Advances in Cryptology - Eurocrpt'02*, LNCS 2332, Springer-Verlag, pp.321-336, 2002.
14. E. Bresson, O. Chevassut, A. Essiari and D. Pointcheval. Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices. In *The Fifth IEEE International Conference on Mobile and Wireless Communications Networks*, 2003.

15. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *Pre-proceedings of Eurocrypt'94*, pp.279-290, 1994.
16. L. Chen and C. Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003)*, pp.219-233, 2003.
17. W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transanctions on Information Theory*, vol.IT-22(6), pp.644-654, 1976.
18. IEEE Std 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification, 1999 edition.
19. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *Advances in Cryptology - Crypto'03* , LNCS 2729, Springer-Verlag, pp.110-125, 2003.
20. H. Luo and S. Lu. Ubiquitous and Roubust Authentication Services for Ad Hoc Wireless Networks. In *Technical Report*, 2000.
21. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. In *Journal of Cryptology*, 13(3):361-396, 2000.
22. V. Shoup. On Formal Models for Secure Key Exchange. In *Technical Report RZ 3120*, IBM Zurich Research Lab., 1999.
23. F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Adhoc Wireless Networks. In *AT&T Software Symposium*, 1999.
24. W.-G. Tzeng and Z.-J. Tzeng. Round Efficient Conference Key Agreement Protocols with Provable Security. In *Advances in Cryptology - Asiacrypt'00*, LNCS 1976, Springer-Verlag, pp.614-628, 2000.
25. L. Venkatraman and D. P. Agrawal. A Novel Authentication Scheme for Ad Hoc Networks. In *WCND'00*, 2000.
26. L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. In *IEEE Network*, 1999.

A Public-Key Black-Box Traitor Tracing Scheme with Sublinear Ciphertext Size Against Self-Defensive Pirates

Tatsuyuki Matsushita[1,2] and Hideki Imai[2]

[1] TOSHIBA Corporate Research & Development Center,
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
tatsuyuki.matsushita@toshiba.co.jp
[2] Institute of Industrial Science, The University of Tokyo,
4-6-1, Komaba, Meguro-ku, Tokyo 153-8505, Japan
imai@iis.u-tokyo.ac.jp

Abstract. We propose a public-key traitor tracing scheme in which (1) the size of a ciphertext is sublinear in the number of receivers and (2) black-box tracing is efficiently achieved against self-defensive pirate decoders. When assuming that a pirate decoder can take some self-defensive reaction (e.g., erasing all of the internal keys and shutting down) to escape from tracing if it detects tracing, it has been an open question to construct a sublinear black-box traitor tracing scheme that can detect efficiently at least one traitor (who builds the pirate decoder) with overwhelming probability, although a tracing algorithm that works successfully against self-defensive pirate decoders itself is known. In this paper, we answer affirmatively the above question by presenting a concrete construction of a public-key black-box tracing scheme in which the known tracing algorithm can be used while keeping the size of a ciphertext sublinear.

Keywords: Public-key traitor tracing, Black-box tracing, Self-defensive pirates.

1 Introduction

Consider content distribution (e.g., pay-TV) in which digital contents should be available only to subscribers. A data supplier broadcasts an encrypted version of the digital contents (e.g., a movie) to subscribers, and only subscribers can decrypt them with their decryption keys given in advance. In this application, malicious subscribers may redistribute their decryption keys to non-subscribers. This piracy is serious since it allows the non-subscribers to have illegal access to the contents.

To prevent the piracy, *traitor tracing* [3] has been studied extensively. In traitor tracing, each subscriber is given a distinct decryption key (*personal key*) which is contained in a decryption device (*decoder*), and the data supplier broadcasts both the contents encrypted with a *session key* and the encrypted session

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 260–275, 2004.

key (*header*). The subscribers can obtain the session key (and consequently the contents) by inputting the received header to their decoders. In this scenario, malicious subscribers (*traitors*) may give away their personal keys to a pirated version of a decoder (*pirate decoder*). Once the pirate decoder is found, at least one of the traitors who join the piracy can be identified from it. A traitor tracing scheme discourages traitors from committing the piracy since the confiscated pirate decoder can be traced back to its producers.

Among traitor tracing schemes, our interest is in a black-box tracing scheme in the public-key setting. In black-box tracing, a tracer does not break open the pirate decoder but uses it as a black box. Briefly, the tracer chooses a set of suspects and tests whether traitors are included in it only by observing the behavior of the pirate decoder on chosen inputs. Since traitors can be identified no matter how the pirate decoder is implemented, it is desirable to support black-box tracing. In the public-key setting, there are one or more public keys and subscribers can decrypt the header by using their personal keys. Since no secret information is needed to build the header and to execute the tracing algorithm, anyone can work as a data supplier and/or a tracer. This property is desirable as well because of the following two reasons: (1) it enhances the sender-scalability in the sense that plural data suppliers can use the same system and (2) it provides public verifiability of the tracing result, which is a stronger deterrent to the piracy.

As a public-key black-box tracing scheme, the schemes of [7, 2] are known.[1] While these are efficient in the sense that the size of a personal key is constant and that of a header is linear only in the maximum number of traitors in a coalition, the running time of the tracing algorithm is exponential in the maximum coalition size, hence impractical. The convergence time for the tracing algorithm is improved to be practical in the schemes of [5, 9] by integrating the mechanism of revocation of any number of subscribers into black-box tracing. However, if it is assumed that a pirate decoder can take measures (e.g., it erases all of the internal keys and shuts down once it detects tracing) that might escape from tracing, tracing is impossible since the identities of suspects are revealed in the inputs for black-box tracing. In this paper, we consider this type of pirate decoders.

1.1 Our Result

We explain our contribution by comparing previous schemes against self-defensive pirate decoders with ours. (See Table 1.) As mentioned above, in the scheme of [2, 8] the tracer can only do *black-box confirmation* in which the number of suspects examined in one test has to be limited to k, where k is the maximum coalition size. Therefore, the black-box confirmation algorithm needs to be executed on all of the possible $\binom{n}{k}$ sets of suspects in the worst case, where n is the total number of subscribers. This results in an impractical tracing algorithm. Note that there is a trade-off between the running time of the tracing algorithm

[1] The scheme of [7] is improved in [10, 8].

Table 1. Summary of our result (n: the total number of subscribers, k: the maximum coalition size, c: a constant $(0 < c < 1)$)

	Personal-key size	Header size	# of sets of suspects required for tracing	Type of tracing
[2, 8]	$O(1)$	$O(k)$	$\binom{n}{k}$	Black-box confirmation
[6] $(c = 1/2)$	$O(1)$	$O(\sqrt{n})$	$\sqrt{n}$	Black-box list-tracing
Ours $(k = \sqrt{n/8})$	$O(1)$	$O(\sqrt{n})$	n	Black-box tracing

and the transmission overhead. For instance, if we set $k = n-1$, then the number of sets of suspects required for tracing is reduced to n, but the size of a header is linear in n, hence inefficient. It has been an open question to obtain a traitor tracing scheme with both practical black-box tracing and sublinear header size, as pointed out in [2].

In [6], a partial solution to this question is presented by introducing a relaxation idea called as *list-tracing* in which the output of the tracing algorithm is a set of suspects, i.e., a suspect list, and it is guaranteed that at least one traitor is included in it. The scheme of [6] is based on that of [4] and achieves both practical black-box list-tracing and sublinear header size. Unfortunately, this approach incurs another trade-off between the size of the suspect list and that of a header. In order to reduce the header size the suspect list needs to be larger, but the probability that the tracer detects a traitor correctly is in inverse proportion to the size of the suspect list, if the tracer attempts to identify the traitor only from the suspect list.

In this paper, we solve the open question without the list-tracing approach. By applying the key-generation method of [9] to the scheme of [8], a sublinear public-key black-box tracing scheme against self-defensive pirate decoders can be obtained. Note that the improvement we achieve is not in the tracing algorithm itself but in a concrete construction of a public-key black-box tracing scheme in which the known tracing algorithm that can identify at least one traitor with overwhelming probability from the self-defensive pirate decoder can be used while keeping the size of a header sublinear.

The rest of the paper is organized as follows. In Sect. 2, the assumptions on the pirate decoder are described. We propose a sublinear public-key black-box tracing scheme in Sect. 3. The proposed scheme is analyzed in terms of security and efficiency in Sect. 4 and Sect. 5, respectively. We present our conclusions in Sect. 6.

2 Assumptions on Pirate Decoders

Let a *valid input* denote a header for the normal broadcast and an *invalid input* denote a header for black-box tracing. In this paper, we make two assumptions on the pirate decoder.

Assumptions 1. The pirate decoder can take measures that might escape from tracing if it detects tracing.

In Assumption 1 the pirate decoder outputs the correct plaintext only when it gets a valid input or an invalid input which is indistinguishable from a valid one. If the pirate decoder detects that it is examined in the tracing process, it will evade tracing by, e.g., erasing all of the internal keys and shutting down. As well as such self-defensive reaction, the pirate decoder can take aggressive countermeasures (e.g., crashing the host system or releasing a virus) as described in [6]. Note that (1) for simplicity we assume that the reaction is triggered deterministically, i.e., it is activated once the pirate decoder detects tracing and (2) our scheme can be easily extended to the general probabilistic case. In order to identify efficiently traitors from the pirate decoder with the reaction mechanism, it is necessary that a tracing algorithm can decide at least one traitor immediately when the reaction is triggered.

Assumptions 2. The tracer can reset the pirate decoder to its initial state each time the tracer gives the input to it.

Assumption 2 means that each test during black-box tracing can be done independently. We do not consider the pirate decoder that records the previous inputs submitted by the tracer and reacts based on its record.

The pirate decoder assumed in the paper can be viewed as a *type-2* pirate decoder categorized in [6].

3 Proposed Scheme

First, we describe an outline of the proposed scheme. Secondly, an explicit construction of our scheme is shown.

3.1 Outline

Our scheme consists of the four phases.

Key Generation: A trusted party generates and secretly gives every subscriber a distinct personal key. The personal key is stored in the decoder.

Encryption: The data supplier encrypts (1) the contents with the session key and (2) the session key itself as a header. Then, the data supplier broadcasts the encrypted contents and the header. To avoid complication, we assume that (1) a symmetric encryption algorithm used for encryption of the contents is secure and publicly known and (2) a broadcast channel is reliable in the sense that the received information is not altered.

Decryption: When receiving the header, subscribers compute the session key (and consequently the contents) by inputting it to their decoders.

Black-Box Ttracing: Suppose that the pirate decoder is confiscated. In the jth test, the tracer chooses a subscriber, u_j, and builds the header in which the subscribers, $u_1, \ldots, u_j$, are revoked and the others are not, where $u_1, \ldots, u_{j-1}$ has been selected in the $(j-1)$th test. The tracer inputs this header to the pirate decoder and observes whether it decrypts correctly or not. If its output is (1)

correct on the input where a set of revoked subscribers is $\mathcal{X}$ and (2) incorrect on the input where a set of revoked ones is $\mathcal{X} \cup \{u\}$, then the tracer decides that the subscriber, u, is a traitor.

3.2 Protocol

Let n be the total number of subscribers and k be the maximum number of traitors in a coalition. Let p, q be primes s.t. $q|p-1$ and $q \geq n + 2k - 1$. Let g be a qth root of unity over $\mathbb{Z}_p^*$ and G_q be a subgroup of $\mathbb{Z}_p^*$ of order q. Let $\mathcal{U}$ be a set of subscribers ($\mathcal{U} \subseteq \mathbb{Z}_q \backslash \{0\}$). All of the participants agree on p, q, and g. The calculations are done over $\mathbb{Z}_p^*$ unless otherwise specified.

Key Generation: The key-generation method is similar to that of [9]. Split $\mathcal{U}$ into ℓ disjoint subsets $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$. These subsets are publicly known. Choose $a_0, \ldots, a_{2k-1}, b_0, \ldots, b_{\ell-1} \in_{\mathrm{R}} \mathbb{Z}_q$. Then, compute the public key e as follows:

$$\begin{aligned} e &= (g, y_{0,0}, \ldots, y_{0,2k-1}, y_{1,0}, \ldots, y_{1,\ell-1}) \\ &= (g, g^{a_0}, \ldots, g^{a_{2k-1}}, g^{b_0}, \ldots, g^{b_{\ell-1}}). \end{aligned}$$

Suppose that $u \in \mathcal{U}_i$. The subscriber u's personal key is $(u, i, f_i(u))$ where

$$f_i(u) = \sum_{j=0}^{2k-1} a_{i,j} u^j \bmod q,$$

$$a_{i,j} = \begin{cases} a_j & (j \not\equiv i \bmod 2k), \\ b_i & (j = i \bmod 2k). \end{cases}$$

Encryption: Select the session key $s \in_{\mathrm{R}} G_q$ and random numbers $R_0, R_1 \in_{\mathrm{R}} \mathbb{Z}_q$. Build the header $H = (H_0, \ldots, H_{\ell-1})$ by repeating the following procedure for $0 \leq i \leq \ell - 1$.

– Set $r_i = R_0$ or R_1, and compute H_i as follows.

$$\begin{aligned} H_i &= (\hat{h}_i, h_{i,0}, \ldots, h_{i,2k-1}), \\ \hat{h}_i &= g^{r_i}, \\ h_{i,j} &= \begin{cases} y_{0,j}^{r_i} & (j \not\equiv i \bmod 2k), \\ s y_{1,i}^{r_i} & (j = i \bmod 2k). \end{cases} \end{aligned}$$

Note that all of the subscribers in $\mathcal{U}_i$ can be revoked by replacing $s y_{1,i}^{r_i}$ with g^{z_i} where $z_i \in_{\mathrm{R}} \mathbb{Z}_q$ is a random number.

Decryption: Suppose that $u \in \mathcal{U}_i$. The subscriber u can correctly compute the session key s from H_i as follows.

$$\begin{aligned} &\left\{ \left(h_{i,0} \times h_{i,1}^u \times \cdots \times h_{i,2k-1}^{u^{2k-1}} \right) \Big/ \hat{h}_i^{f_i(u)} \right\}^{1/u^{i \bmod 2k}} \\ &\quad = \left\{ \left(y_{0,0}^{r_i} \times y_{0,1}^{r_i u} \times \cdots \times y_{1,i}^{r_i u^{i \bmod 2k}} \times \cdots \times y_{0,2k-1}^{r_i u^{2k-1}} \right) \Big/ g^{r_i f_i(u)} \right\}^{1/u^{i \bmod 2k}} \\ &\quad = \left\{ s^{u^{i \bmod 2k}} g^{r_i \sum_{j=0}^{2k-1} a_{i,j} u^j} \Big/ g^{r_i f_i(u)} \right\}^{1/u^{i \bmod 2k}} \\ &\quad = s. \end{aligned}$$

Black-Box Tracing: The black-box tracing algorithm is based on that of [8]. The difference is that while in [8] suspects must be narrowed down to k subscribers before the execution of black-box confirmation, in ours no such preprocessing, which runs in exponential time, is needed. The inputs of the tracing algorithm are $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$ and the pirate decoder, and the output is a traitor's ID.

For simplicity, we assume that $|\mathcal{U}_0| = \cdots = |\mathcal{U}_{\ell-1}| = 2k,\ \ell = n/2k,\ 2k|n$. Label all of the elements in $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$ as follows.

$$\begin{aligned}\mathcal{U}_0 &= \{u_1, \ldots, u_{2k}\},\\ \mathcal{U}_1 &= \{u_{2k+1}, \ldots, u_{4k}\},\\ &\vdots\\ \mathcal{U}_{\ell-1} &= \{u_{n-2k+1}, \ldots, u_n\}.\end{aligned}$$

For $1 \le j \le n$, repeat the following procedure.

- Set $ctr_j = 0$ and then repeat the following test m times. In each test, the session key s and random numbers R_0, R_1 are chosen randomly.
  1. Set $\mathcal{X} = \{u_1, \ldots, u_j\}$ and build the header $H = (H_0, \ldots, H_{\ell-1})$ by repeating the following procedure for $0 \le i \le \ell - 1$. The same notations are used as in the encryption phase and a random number $z_i \in_{\mathrm{R}} \mathbb{Z}_q$ is selected randomly in each time.
     - If there exists a subset $\mathcal{U}_t$ $(0 \le t \le \ell - 1)$ s.t. $\mathcal{X} \cap \mathcal{U}_t \ / = \emptyset$ and $\mathcal{X} \cap \mathcal{U}_t \ / = \mathcal{U}_t$, then first, suppose that $\mathcal{U}_t \setminus \mathcal{X} = \{x_1, \ldots, x_w\}$ and choose $2k - w - 1$ distinct elements $x_{w+1}, \ldots, x_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q \setminus (\mathcal{U} \cup \{0\})$ when $2k - w - 1 > 0$. Secondly, find $c_0, \ldots, c_{2k-1} \in \mathbb{Z}_q$ s.t. $\sum_{j=0}^{2k-1} c_j x_\alpha^j = 0 \bmod q$ for $1 \le \alpha \le 2k - 1$. Finally, compute H_t as follows.

$$\begin{aligned}\hat{h}_t &= g^{R_1},\\ h_{t,j} &= \begin{cases} g^{c_j} y_{0,j}^{R_1} & (j \ / = t \bmod 2k),\\ s g^{c_j} y_{1,t}^{R_1} & (j = t \bmod 2k).\end{cases}\end{aligned}$$

     For $i \ / = t$, set $r_i = R_0$ if $\mathcal{X} \cap \mathcal{U}_i = \emptyset$. Otherwise $(\mathcal{X} \cap \mathcal{U}_i = \mathcal{U}_i)$, set $r_i = R_0$ or R_1. Then, compute H_i as follows.

$$\begin{aligned}\hat{h}_i &= g^{r_i},\\ h_{i,j} &= \begin{cases} y_{0,j}^{R_0} & (j \ / = i \bmod 2k, r_i = R_0),\\ g^{c_j} y_{0,j}^{R_1} & (j \ / = i \bmod 2k, r_i = R_1),\\ s y_{1,i}^{R_0} & (j = i \bmod 2k, \mathcal{X} \cap \mathcal{U}_i = \emptyset),\\ g^{z_i} & (j = i \bmod 2k, \mathcal{X} \cap \mathcal{U}_i = \mathcal{U}_i).\end{cases}\end{aligned}$$

     - Otherwise ($\mathcal{X} \cap \mathcal{U}_i = \emptyset$ or $\mathcal{X} \cap \mathcal{U}_i = \mathcal{U}_i$ for any i), H_i is the same as in the encryption phase.

2. Give H to the pirate decoder and observe its output.
3. If it decrypts correctly, then increment ctr_j by one. (If a self-defensive reaction is triggered, then decide that the subscriber u_j is a traitor.)

Finally, find an integer $j \in \{1, \ldots, n\}$ s.t. $ctr_{j-1} - ctr_j$ is the maximum and then decide that the subscriber u_j is a traitor, where $ctr_0 = m$.

4 Security

The security of our scheme is based on the difficulty of the Decision Diffie-Hellman problem (DDH) [1]. Informally, the assumption that DDH in G_q is intractable means that no probabilistic polynomial-time (p.p.t. for short) algorithm can distinguish with non-negligible advantage between the two distributions $\langle g_1, g_2, g_1^a, g_2^a \rangle$ and $\langle g_1, g_2, g_1^a, g_2^b \rangle$ where $g_1,\ g_2 \in_{\mathrm{R}} G_q$ and $a, b \in_{\mathrm{R}} \mathbb{Z}_q$. We call a 4-tuple coming from the former distribution as a Diffie-Hellman tuple. Let $\mathcal{M}^{\mathrm{DDH}}$ be a p.p.t. algorithm which solves DDH in G_q. For two p.p.t. algorithms $\mathcal{M}_0, \mathcal{M}_1$, we mean by $\mathcal{M}_0 \Rightarrow \mathcal{M}_1$ that the existence of $\mathcal{M}_0$ implies that of $\mathcal{M}_1$ and by $\mathcal{M}_0 \Leftrightarrow \mathcal{M}_1$ that $\mathcal{M}_0 \Rightarrow \mathcal{M}_1$ and $\mathcal{M}_1 \Rightarrow \mathcal{M}_0$.

4.1 Indistinguishability of a Session Key

Theorem 1 (Indistinguishability of a Session Key). *When given a header, the computational complexity for the non-subscribers to distinguish the session key corresponding to the header from a random element in G_q is as difficult as DDH in G_q.*

Proof. Let $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}}$ be a p.p.t. algorithm the non-subscribers use to distinguish between the session key corresponding to the header and a random element in G_q. We prove that $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}} \Leftrightarrow \mathcal{M}^{\mathrm{DDH}}$. First, it is clear that $\mathcal{M}^{\mathrm{DDH}} \Rightarrow \mathcal{M}_{\mathcal{U}}^{\mathrm{dist}}$. Secondly, we show that $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}} \Rightarrow \mathcal{M}^{\mathrm{DDH}}$ by constructing $\mathcal{M}^{\mathrm{DDH}}$ using $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}}$ as a subroutine. The construction of $\mathcal{M}^{\mathrm{DDH}}$ is as follows.

Algorithm 1 (P.p.t. Algorithm $\mathcal{M}^{\mathrm{DDH}}$).

Input: a challenge 4-tuple, (g_1, g_2, g_3, g_4).
Output: "Diffie-Hellman tuple" or "Random tuple."

Step 1. Choose a set of subscribers $\mathcal{U}$ ($\subseteq \mathbb{Z}_q \backslash \{0\}$) and split $\mathcal{U}$ into ℓ disjoint subsets $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$. For $0 \le i \le \ell - 1,\ 0 \le j \le 2k-1$, choose random numbers $\mu, \lambda_i, a_j \in_{\mathrm{R}} \mathbb{Z}_q$ and compute the public key $e = (g_1, g_1^{a_0}, \ldots, g_1^{a_{2k-1}}, g_1^{b_0}, \ldots, g_1^{b_{\ell-1}})$ where $g_1^{b_i} = g_1^{\lambda_i} g_2^{\mu}$.

Step 2. Select the session key $s \in_{\mathrm{R}} G_q$ and a random number $r \in_{\mathrm{R}} \mathbb{Z}_q$. Compute the header $H = (H_0, \ldots, H_{\ell-1})$ by repeating the following procedure for $0 \le i \le \ell - 1$.

– Set $B_i = 0$ or 1 and compute H_i as follows.

$$H_i = (\hat{h}_i, h_{i,0}, \ldots, h_{i,2k-1}),$$
$$\hat{h}_i = g_1^{B_i r} g_3,$$
$$h_{i,j} = \begin{cases} \left(g_1^{B_i r} g_3\right)^{a_j} & (j \not\equiv i \bmod 2k), \\ s \left(g_1^{B_i r} g_3\right)^{\lambda_i} \left(g_2^{B_i r} g_4\right)^{\mu} & (j = i \bmod 2k). \end{cases}$$

Observe that if the challenge 4-tuple is a Diffie-Hellman tuple, the session key corresponding to the header is s. Otherwise, it is a random element in G_q. Step 3. Give s, H, e to $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}}$. If $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}}$ decides that s is the session key corresponding to H, then output "Diffie-Hellman tuple." Otherwise, output "Random tuple." Since $\mathcal{M}_{\mathcal{U}}^{\mathrm{dist}}$ behaves differently for session keys and random elements in G_q, $\mathcal{M}^{\mathrm{DDH}}$ can solve the given DDH challenge. This completes the proof. □

4.2 Black-Box Traceability

Recall that valid and invalid inputs denote headers for the normal broadcast and those for black-box tracing respectively. In our tracing algorithm subscribers in $\mathcal{X}$ are revoked in invalid inputs. The following three lemmas are used to prove black-box traceability of our scheme.

Lemma 1 (Indistinguishability of an Input). *The computational complexity for any coalition of k non-revoked subscribers to distinguish a valid input from an invalid one is as difficult as DDH in G_q.*

Proof. Let $\mathcal{C}$ be a set of k non-revoked subscribers in a coalition and $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}}$ be a p.p.t. algorithm the coalition $\mathcal{C}$ uses to distinguish a valid input from an invalid one. We prove that $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}} \Leftrightarrow \mathcal{M}^{\mathrm{DDH}}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$. First, it is clear that $\mathcal{M}^{\mathrm{DDH}} \Rightarrow \mathcal{D}_{\mathcal{C}}^{\mathrm{dist}}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$. Secondly, we show that $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}} \Rightarrow \mathcal{M}^{\mathrm{DDH}}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$ by constructing $\mathcal{M}^{\mathrm{DDH}}$ using $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}}$ as a subroutine. The construction of $\mathcal{M}^{\mathrm{DDH}}$ is as follows.

Algorithm 2 (P.p.t. Algorithm $\mathcal{M}^{\mathrm{DDH}}$).

Input: a challenge 4-tuple, (g_1, g_2, g_3, g_4).
Output: "Diffie-Hellman tuple" or "Random tuple."

Step 1. Choose a set of subscribers $\mathcal{U}$ ($\subseteq \mathbb{Z}_q \backslash \{0\}$) and split $\mathcal{U}$ into ℓ disjoint subsets $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$. Select a set of revoked subscribers $\mathcal{X}$ ($\subseteq \mathcal{U}$) with a condition that there is at most one subset $\mathcal{U}_i$ ($0 \le i \le \ell - 1$) s.t. $\mathcal{U}_i \cap \mathcal{X} \not= \emptyset$, $\mathcal{U}_i \cap \mathcal{X} \not= \mathcal{U}_i$. Then, choose a set of k colluders $\mathcal{C}$ s.t. $\mathcal{X} \cap \mathcal{C} = \emptyset$.

Step 2. Suppose that $\mathcal{C} = \{x_1, \ldots, x_k\}$. Choose $k-1$ distinct elements $x_{k+1}, \ldots, x_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q \setminus \mathcal{C}$ and random numbers $\beta_1, \ldots, \beta_k, \lambda, \mu, \psi_t, \omega_t \in_{\mathrm{R}} \mathbb{Z}_q$ for $k+1 \le t \le 2k-1$. Then, there exists a unique polynomial $\alpha(x) = \sum_{m=0}^{2k-1} \alpha_m x^m \bmod q$ s.t. $g_1^{\alpha_0} = g_1^{\lambda} g_2^{\mu}$ and

$$\begin{aligned}(\alpha(x_1),\ldots,\alpha(x_{2k-1}))^{\mathrm{T}} &= (\beta_1,\ldots,\beta_{2k-1})^{\mathrm{T}}\\ &= (\alpha_0,\ldots,\alpha_0)^{\mathrm{T}} + V(\alpha_1,\ldots,\alpha_{2k-1})^{\mathrm{T}} \bmod q,\\ g_1^{\beta_t} &= g_1^{\psi_t} g_2^{\omega_t} \qquad (k+1 \le t \le 2k-1),\end{aligned}$$

where

$$V = \begin{pmatrix} x_1 & \ldots & x_1^{2k-1} \\ \vdots & \ddots & \vdots \\ x_{2k-1} & \ldots & x_{2k-1}^{2k-1} \end{pmatrix} \bmod q.$$

Since V is the Vandermonde matrix, we obtain

$$(\alpha_1,\ldots,\alpha_{2k-1})^{\mathrm{T}} = V^{-1}(\beta_1-\alpha_0,\ldots,\beta_{2k-1}-\alpha_0)^{\mathrm{T}} \bmod q.$$

Let $(v_{m,1},\ldots,v_{m,2k-1})$ be the mth row of V^{-1}. For $1 \le m \le 2k-1$, α_m is represented as follows.

$$\begin{aligned}\alpha_m &= v_{m,1}(\beta_1-\alpha_0) + \cdots + v_{m,2k-1}(\beta_{2k-1}-\alpha_0)\\ &= v_{m,1}\beta_1 + \cdots + v_{m,2k-1}\beta_{2k-1} - \alpha_0(v_{m,1}+\cdots+v_{m,2k-1}) \bmod q.\end{aligned}$$

Therefore, $g_1^{\alpha_m}$ is calculated as follows.

$$g_1^{\alpha_m} = g_1^{v_{m,1}\beta_1+\cdots+v_{m,2k-1}\beta_{2k-1}} \Big/ \left(g_1^{\lambda} g_2^{\mu}\right)^{v_{m,1}+\cdots+v_{m,2k-1}}.$$

Suppose that $x_j \in \mathcal{U}_{i_j}$ $(1 \le j \le k,\ i_j \in \{0,\ldots,\ell-1\})$ and define $\mathcal{J} = \{i_j | 1 \le j \le k,\ x_j \in \mathcal{U}_{i_j}\}$. Choose random numbers $\lambda_i, \mu_i \in_{\mathrm{R}} \mathbb{Z}_q$ for $0 \le i \le \ell-1$ and $\delta_{i_j} \in_{\mathrm{R}} \mathbb{Z}_q$ for all i_j's in $\mathcal{J}$. Then, there exists a unique element $\gamma_{i_j} \in \mathbb{Z}_q$ for each $i_j \in \mathcal{J}$ s.t.

$$\begin{aligned}\delta_{i_j} &= b_{i_j} + \gamma_{i_j} - \alpha_{i_j \bmod 2k} \qquad (i_j \in \mathcal{J}),\\ g_1^{b_i} &= g_1^{\lambda_i} g_2^{\mu_i} \qquad (0 \le i \le \ell-1).\end{aligned}$$

We plan to compute the subscriber x_j's personal key (x_j, i_j, d_j) as follows.

$$\begin{aligned}d_j &= \alpha(x_j) + \delta_{i_j} x_j^{i_j \bmod 2k}\\ &= \alpha_0 + \alpha_1 x_j + \cdots + b_{i_j} x_j^{i_j \bmod 2k} + \cdots + \alpha_{2k-1} x_j^{2k-1} + \gamma_{i_j} x_j^{i_j \bmod 2k}.\end{aligned}$$

To satisfy $d_j = f_{i_j}(x_j)$ where f is the key-generation function defined in 3.2, the coefficients $a_0,\ldots,a_{2k-1}$ are represented as follows. There are at least k elements in $\{0,\ldots,2k-1\} \setminus \{i_j \bmod 2k | i_j \in \mathcal{J}\}$ and we can select k such elements $\theta_1,\ldots,\theta_k$. Then, compute $g_1^{\alpha'_{\theta_1}},\ldots,g_1^{\alpha'_{\theta_k}}$ s.t.

$$\begin{aligned}g_1^{\sum_{\tau\in\{\theta_1,\ldots,\theta_k\}} \alpha'_\tau x_j^\tau} &= g_1^{\gamma_{i_j} x_j^{i_j \bmod 2k}}\\ &= \left(g_1^{\delta_{i_j}} g_1^{\alpha_{i_j \bmod 2k}} \Big/ g_1^{b_{i_j}}\right)^{x_j^{i_j \bmod 2k}} \qquad (1 \le j \le k).\end{aligned}$$

Finally, compute $g_1^{a_m}$ $(0 \le m \le 2k-1)$ and build the public key e.

$$g_1^{a_m} = \begin{cases} g_1^{\alpha_m} & (m \not\in \{\theta_1, \ldots, \theta_k\}), \\ g_1^{\alpha_m} g_1^{\alpha'_m} & (m \in \{\theta_1, \ldots, \theta_k\}), \end{cases}$$

$$e = (g_1, g_1^{a_0}, \ldots, g_1^{a_{2k-1}}, g_1^{b_0}, \ldots, g_1^{b_{\ell-1}}).$$

Step 3. Select the session key $s \in_{\mathrm{R}} G_q$ and a random number $r \in_{\mathrm{R}} \mathbb{Z}_q$. Build the header $H = (H_0, \ldots, H_{\ell-1})$ by repeating the following procedure for $0 \le i \le \ell - 1$.

– If $\mathcal{U}_i \cap \mathcal{X} = \emptyset$, set $B_i = 0$. If $\mathcal{U}_i \cap \mathcal{X} = \mathcal{U}_i$, set $B_i = 0$ or 1. Otherwise $(\mathcal{U}_i \cap \mathcal{X} \neq \emptyset, \mathcal{U}_i \cap \mathcal{X} \neq \mathcal{U}_i)$, set $B_i = 1$. Then, compute H_i as follows.

$$H_i = (\hat{h}_i, h_{i,0}, \ldots, h_{i,2k-1}),$$

$$\hat{h}_i = \begin{cases} g_1^r & (B_i = 0), \\ g_3 & (B_i = 1), \end{cases}$$

$$h_{i,j} = \begin{cases} g_1^{a_j r} & (j \neq i \bmod 2k, B_i = 0), \\ g_3^{a_j} & (j \neq i \bmod 2k, B_i = 1), \\ s g_1^{b_i r} & (j = i \bmod 2k, B_i = 0), \\ s g_3^{\lambda_i} g_4^{\mu_i} & (j = i \bmod 2k, B_i = 1), \end{cases}$$

$$g_3^{a_j} = \begin{cases} g_3^{\alpha_j} & (j \notin \{\theta_1, \ldots, \theta_k\}), \\ g_3^{\alpha_j} g_3^{\alpha'_j} & (j \in \{\theta_1, \ldots, \theta_k\}), \end{cases}$$

$$g_3^{\alpha_j} = g_3^{v_{j,1}\beta_1 + \cdots + v_{j,k}\beta_k} \prod_{t=k+1}^{2k-1} \left(g_3^{\psi_t} g_4^{\omega_t}\right)^{v_{j,t}} \Big/ \left(g_3^{\lambda} g_4^{\mu}\right)^{v_{j,1} + \cdots + v_{j,2k-1}},$$

where $g_3^{\alpha'_{\theta_1}}, \ldots, g_3^{\alpha'_{\theta_k}}$ are computed from the following system of equations.

$$g_3^{\sum_{\tau \in \{\theta_1, \ldots, \theta_k\}} \alpha'_\tau x_z^\tau} = \left(g_3^{\delta_{i_z}} g_3^{\alpha_{i_z \bmod 2k}} \Big/ g_3^{\lambda_{i_z}} g_4^{\mu_{i_z}}\right)^{x_z^{i_z \bmod 2k}} \qquad (1 \le z \le k).$$

Observe that if the challenge 4-tuple is a Diffie-Hellman tuple, H is a valid input. Otherwise, it is an invalid one in which the k colluders in $\mathcal{C}$ are not revoked.

Step 4. Give $H, e, (x_1, i_1, d_1), \ldots, (x_k, i_k, d_k)$ to $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}}$. If $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}}$ decides that H is a valid input, then output "Diffie-Hellman tuple." Otherwise output "Random tuple." Since $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}}$ behaves differently for valid inputs and invalid ones, $\mathcal{M}^{\mathrm{DDH}}$ can solve the given DDH challenge.

Since $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$ can be chosen arbitrarily in Step 4.2, it holds that $\mathcal{D}_{\mathcal{C}}^{\mathrm{dist}} \Rightarrow \mathcal{M}^{\mathrm{DDH}}$ for any $\mathcal{C}$ with $\mathcal{X} \cap \mathcal{C} = \emptyset$, $|\mathcal{C}| = k$. This completes the proof. □

Lemma 2 (Secrecy of a Session Key in an Invalid Input). *When given an invalid input, the computational complexity for any coalition of k subscribers revoked in the invalid input to compute the session key corresponding to the input is at least as difficult as DDH in G_q.*

Proof. Let $\mathcal{C}$ be a set of k colluders revoked in the invalid input and $\mathcal{M}_{\mathcal{C}}^{\mathrm{comp}}$ be a p.p.t. algorithm the coalition $\mathcal{C}$ uses to compute the session key corresponding to the input. Let $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$ be a p.p.t. algorithm the coalition $\mathcal{C}$ uses to distinguish the session key corresponding to the input from a random element in G_q. We prove that $\mathcal{M}_{\mathcal{C}}^{\mathrm{comp}} \Rightarrow \mathcal{M}^{\mathrm{DDH}}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$.

Since it is clear that $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$ can be constructed by using $\mathcal{M}_{\mathcal{C}}^{\mathrm{comp}}$ as a subroutine, it holds that $\mathcal{M}_{\mathcal{C}}^{\mathrm{comp}} \Rightarrow \mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$. Therefore, we show that $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}} \Rightarrow \mathcal{M}^{\mathrm{DDH}}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X},\ |\mathcal{C}| = k$ by constructing $\mathcal{M}^{\mathrm{DDH}}$ using $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$ as a subroutine. The construction of $\mathcal{M}^{\mathrm{DDH}}$ is as follows.

Algorithm 3 (P.p.t. Algorithm $\mathcal{M}^{\mathrm{DDH}}$).

Input: a challenge 4-tuple, (g_1, g_2, g_3, g_4).
Output: "Diffie-Hellman tuple" or "Random tuple."

Step 1. Choose a set of subscribers $\mathcal{U}$ ($\subseteq \mathbb{Z}_q \backslash \{0\}$) and split $\mathcal{U}$ into ℓ disjoint subsets $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$. Select a set of revoked subscribers $\mathcal{X}$ ($\subseteq \mathcal{U}$) with a condition that there is at most one subset $\mathcal{U}_i$ ($0 \le i \le \ell - 1$) s.t. $\mathcal{U}_i \cap \mathcal{X}$ / $\not\emptyset$, $\mathcal{U}_i \cap \mathcal{X}$ / $\not= \mathcal{U}_i$. Then, choose a set of k colluders $\mathcal{C}$ s.t. $\mathcal{C} \subseteq \mathcal{X}$.

Step 2. Suppose that $\mathcal{C} = \{x_1, \ldots, x_k\}$. Construct the personal key (x_j, i_j, d_j) given to the subscriber, $x_j \in \mathcal{U}_{i_j}$, and the public key $e = (g_1, g_1^{a_0}, \ldots, g_1^{a_{2k-1}}, g_1^{b_0}, \ldots, g_1^{b_{\ell-1}})$ by executing the same procedure as in Step 4.2 of Algorithm 2.

Step 3. Select the session key $s \in_{\mathrm{R}} G_q$ and random numbers $r, x, y \in_{\mathrm{R}} \mathbb{Z}_q$. Build the header $H = (H_0, \ldots, H_{\ell-1})$ by repeating the following procedure to compute $H_i = (\hat{h}_i, h_{i,0}, \ldots, h_{i,2k-1})$ for $0 \le i \le \ell - 1$.

– If $\mathcal{X} \cap \mathcal{U}_i = \emptyset$, then compute H_i as follows.

$$\hat{h}_i = g_3^r,$$
$$h_{i,j} = \begin{cases} g_3^{a_j r} & (j \ / \ \not= \bmod 2k), \\ s\left(g_3^{\lambda_i} g_4^{\mu_i}\right)^r & (j = i \bmod 2k). \end{cases}$$

– If $\mathcal{X} \cap \mathcal{U}_i = \mathcal{U}_i$, then set $B_i = 0$ or 1 and compute H_i as follows. In each time, a random number $z_i \in_{\mathrm{R}} \mathbb{Z}_q$ is selected randomly.

$$\hat{h}_i = \begin{cases} g_3^r & (B_i = 0), \\ g_1^x g_3^y & (B_i = 1), \end{cases}$$
$$h_{i,j} = \begin{cases} h'_{i,j} & (j \ / \ \not= \bmod 2k), \\ g_1^{z_i} & (j = i \bmod 2k), \end{cases}$$

where if there exists a subset $\mathcal{U}_t$ ($0 \le t \le \ell - 1$) s.t. $\mathcal{X} \cap \mathcal{U}_t$ / $\not= \emptyset$ and $\mathcal{X} \cap \mathcal{U}_t$ / $\not= \mathcal{U}_t$, then

$$h'_{i,j} = \begin{cases} g_3^{a_j r} & (B_i = 0), \\ g_1^{c_j} (g_1^x g_3^y)^{a_j} & (B_i = 1). \end{cases}$$

Otherwise ($\mathcal{X} \cap \mathcal{U}_i = \emptyset$ or $\mathcal{X} \cap \mathcal{U}_i = \mathcal{U}_i$ for any i),

$$h'_{i,j} = \begin{cases} g_3^{a_j r} & (B_i = 0), \\ (g_1^x g_3^y)^{a_j} & (B_i = 1), \end{cases}$$

– If $\mathcal{X} \cap \mathcal{U}_i \neq \emptyset$ and $\mathcal{X} \cap \mathcal{U}_i \neq \mathcal{U}_i$, then first, suppose that $\mathcal{U}_i \setminus \mathcal{X} = \{u_1, \ldots, u_w\}$ and choose $2k - w - 1$ distinct elements $u_{w+1}, \ldots, u_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q \setminus (\mathcal{U} \cup \{0\})$ when $2k - w - 1 > 0$. Secondly, find $c_0, \ldots, c_{2k-1} \in_{\mathrm{R}} \mathbb{Z}_q$ s.t. $\sum_{j=0}^{2k-1} c_j u_\alpha^j = 0 \bmod q$ for $1 \leq \alpha \leq 2k - 1$. Finally, compute H_i as follows.

$$\hat{h}_i = g_1^x g_3^y,$$
$$h_{i,j} = \begin{cases} g_1^{c_j} (g_1^x g_3^y)^{a_j} & (j \neq i \bmod 2k), \\ s g_1^{c_j} (g_1^x g_3^y)^{\lambda_i} (g_2^x g_4^y)^{\mu_i} & (j = i \bmod 2k). \end{cases}$$

In this procedure, $g_3^{a_j}$ is computed as in Step 4.2 of Algorithm 2. Observe that if the challenge 4-tuple is a Diffie-Hellman tuple, s is the session key corresponding to H. Otherwise, it is not.

Step 4. Give $s, H, e, (x_1, i_1, d_1), \ldots, (x_k, i_k, d_k)$ to $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$. If $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$ decides that s is the session key corresponding to H, then output "Diffie-Hellman tuple." Otherwise output "Random tuple." Since $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}}$ behaves differently for session keys and random elements in G_q, $\mathcal{M}^{\mathrm{DDH}}$ can solve the given DDH challenge.

Since $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X}$, $|\mathcal{C}| = k$ can be chosen arbitrarily in Step 4.2, it holds that $\mathcal{M}_{\mathcal{C}}^{\mathrm{dist}} \Rightarrow \mathcal{M}^{\mathrm{DDH}}$ for any $\mathcal{C}$ with $\mathcal{C} \subseteq \mathcal{X}$, $|\mathcal{C}| = k$. This completes the proof. □

Lemma 3 (Indistinguishability of a Suspect). *The computational complexity for any coalition of k subscribers to distinguish (1) an invalid input in which a given subscriber other than the k ones is not revoked from (2) an invalid one in which the subscriber is revoked is as difficult as DDH in G_q.*

Sketch of Proof. Due to space limitation, we describe a sketch of the proof. Let $\mathcal{C}$ be a set of k colluders. Let $\mathcal{A}_{\mathcal{C}}^{\mathrm{dist}}$ be a p.p.t. algorithm the coalition $\mathcal{C}$ uses to distinguish an invalid input in which the given subscriber is not revoked from an invalid one in which the subscriber is revoked. Similarly in the proofs of the other lemmas, we construct $\mathcal{M}^{\mathrm{DDH}}$ using $\mathcal{A}_{\mathcal{C}}^{\mathrm{dist}}$ as a subroutine.

Algorithm 4 (P.p.t. Algorithm $\mathcal{M}^{\mathrm{DDH}}$).

Input: a challenge 4-tuple, (g_1, g_2, g_3, g_4).
Output: "Diffie-Hellman tuple" or "Random tuple."

Step 1. Choose a set of subscribers $\mathcal{U}$ ($\subseteq \mathbb{Z}_q \setminus \{0\}$) and split $\mathcal{U}$ into ℓ disjoint subsets $\mathcal{U}_0, \ldots, \mathcal{U}_{\ell-1}$. Select a set of k colluders $\mathcal{C}$ and one subscriber $u \in_{\mathrm{R}} \mathcal{U} \setminus \mathcal{C}$. Suppose that $u \in \mathcal{U}_t$, $\mathcal{U}_i \cap \mathcal{X} = \mathcal{U}_i$ for $0 \leq i \leq t - 1$, and $\mathcal{U}_i \cap \mathcal{X} = \emptyset$ for $t + 1 \leq i \leq \ell - 1$. There are three possible relations between $\mathcal{U}_t$ and $\mathcal{X}$: (1) $\mathcal{U}_t \cap \mathcal{X} \neq \mathcal{U}_t$, $\mathcal{U}_t \cap \mathcal{X} \neq \emptyset$ both when $u \notin \mathcal{X}$ and $u \in \mathcal{X}$, (2) $\mathcal{U}_t \cap \mathcal{X} = \emptyset$ when

$u \notin \mathcal{X}$, and $\mathcal{U}_t \cap \mathcal{X} = \{u\}$ when $u \in \mathcal{X}$, (3) $\mathcal{U}_t \cap \mathcal{X} = \mathcal{U}_t \setminus \{u\}$ when $u \notin \mathcal{X}$, and $\mathcal{U}_t \cap \mathcal{X} = \mathcal{U}_t$ when $u \in \mathcal{X}$.

Step 2. Suppose that $\mathcal{C} = \{x_1, \ldots, x_k\}$. Construct the personal key (x_j, i_j, d_j) given to the subscriber, $x_j \in \mathcal{U}_{i_j}$, and the public key $e = (g_1, g_1^{a_0}, \ldots, g_1^{a_{2k-1}}, g_1^{b_0}, \ldots, g_1^{b_{\ell-1}})$ by executing the same procedure as in Step 4.2 of Algorithm 2.

Step 3. Build the header H in which (1) if the challenge 4-tuple is a Diffie-Hellman tuple, the subscriber u is not revoked and (2) otherwise, the subscriber u is revoked, in each case. The construction of H is similar to that in Step 4.2 of Algorithm 3.

Step 4. Give $u, H, e, (x_1, i_1, d_1), \ldots, (x_k, i_k, d_k)$ to $\mathcal{A}_{\mathcal{C}}^{\mathrm{dist}}$. Since $\mathcal{A}_{\mathcal{C}}^{\mathrm{dist}}$ behaves differently for invalid inputs in which the subscriber u is not revoked and invalid ones in which the subscriber u is revoked, $\mathcal{M}^{\mathrm{DDH}}$ can solve the given DDH challenge. □

From Lemma 1, Lemma 2, and Lemma 3, it follows that the next theorem holds.

Theorem 2 (Black-Box Traceability). *In the proposed scheme, from the pirate decoder constructed by a coalition of at most k traitors, at least one of them can be identified with probability $1 - \varepsilon$ where ε is negligible.*

Proof. Recall that ctr_j $(0 \leq ctr_j \leq m)$ denotes the number of times of observing that the pirate decoder decrypts correctly the input in which $\mathcal{X} = \{u_1, \ldots, u_j\}$, i.e., the subscribers $u_1, \ldots, u_j$ are revoked. Define $j = 0$ if $\mathcal{X} = \emptyset$, i.e., the input is valid. It is clear that $ctr_0 = m$. From Lemma 2, it holds that $ctr_n = 0$ with overwhelming probability. From the triangular inequality, it follows that there exists an integer $j \in \{1, \ldots, n\}$ s.t. $ctr_{j-1} - ctr_j \geq m/n$. If the subscriber u_j is not a traitor, $ctr_{j-1} - ctr_j \ll m/n$ since it follows from Lemma 3 that the pirate decoder cannot distinguish an invalid input in which $\mathcal{X} = \{u_1, \ldots, u_{j-1}\}$ from an invalid one in which $\mathcal{X} = \{u_1, \ldots, u_j\}$ with non-negligible advantage. Therefore, the subscriber u_j is a traitor with overwhelming probability if $ctr_{j-1} - ctr_j$ is the maximum.

Next, consider the case where the reaction mechanism is activated. From Lemma 1, no such reaction is triggered as long as $\mathcal{X} \cap \mathcal{C} = \emptyset$ where $\mathcal{C}$ denotes a set of the colluders. Therefore, if the reaction is triggered in the input in which $\mathcal{X} = \{u_1, \ldots, u_j\}$, it holds that $\{u_1, \ldots, u_j\} \cap \mathcal{C} \neq \emptyset$. In this case, if the subscriber u_j is not a traitor, the pirate decoder must have taken the reaction in the previous input in which $\mathcal{X} = \{u_1, \ldots, u_{j-1}\}$ since it follows from Lemma 3 that the pirate decoder cannot distinguish an invalid input in which $\mathcal{X} = \{u_1, \ldots, u_j\}$ from an invalid one in which $\mathcal{X} = \{u_1, \ldots, u_{j-1}\}$ with non-negligible advantage. Hence, if the reaction is triggered in the input in which $\mathcal{X} = \{u_1, \ldots, u_j\}$, it holds that the subscriber u_j is a traitor with overwhelming probability. □

Note that our scheme can be easily applied to the case where the pirate decoder takes the reaction in a probabilistic way.

5 Efficiency

In Table 2, the previous schemes and ours are compared from the viewpoints of each subscriber's storage, the transmission overhead, the number of sets of suspects required for tracing, the detection probability, and the computational cost for decryption. The scheme of [2] is omitted since its efficiency is almost the same as that of [8] in the above criteria. We suppose that the standard ElGamal encryption scheme is straightforwardly used in the scheme of [6].

Table 2. Efficiency comparison ($\mathcal{P}$, $\mathcal{S}$, $\mathcal{H}$: sets of possible personal keys, session keys, and headers respectively, n: the total number of subscribers, k: the maximum coalition size, c: a constant $(0 < c < 1)$, ε: negligible probability)

	Each subscriber's storage ($\log \lvert\mathcal{P}\rvert / \log \lvert\mathcal{S}\rvert$)	Transmission overhead ($\log \lvert\mathcal{H}\rvert / \log \lvert\mathcal{S}\rvert$)	# of sets of suspects for tracing	Detection probability	# of exp.'s for decryption
[8]	1	$2k+1$	$\binom{n}{k}$	$1-\varepsilon$	$O(k)$
[6]	$(1-c)^{-1}$	$(1-c)^{-1}n^{1-c}$	n^{1-c}	n^{-c}	$O((1-c)^{-1})$
[6] $(c = 1/2)$	2	$2\sqrt{n}$	$\sqrt{n}$	$1/\sqrt{n}$	$O(1)$
Ours	1	$4k + n/2k + 2$	n	$1-\varepsilon$	$O(k)$
Ours $(k = \sqrt{n/8})$	1	$2\sqrt{2n}+2$	n	$1-\varepsilon$	$O(\sqrt{n})$

In the scheme of [6], the size of a personal key is determined by a constant c $(0 < c < 1)$ selected when initializing the system. In the other schemes, the size of a personal key is constant. In the scheme of [8], the efficient transmission overhead which is linear only in k is achieved where k is the maximum coalition size. However, the scheme of [8] can only support black-box confirmation in which only k suspects can be tested in one confirmation. Therefore, the tracer needs to execute the confirmation algorithm on all of the possible $\binom{n}{k}$ sets of suspects at the worst case, where n is the total number of subscribers. Since the number of sets of suspects required for tracing is directly affected to the running time of the tracing algorithm, the scheme of [8] is impractical from this viewpoint. On the other hand, in the scheme of [6] and ours the number of sets of suspects required for tracing is drastically reduced and hence the practical convergence time for tracing is achieved.

In the scheme of [6], the output of the tracing algorithm is the list of suspects in which at least one traitor is included with overwhelming probability. If the tracer attempts to identify the traitor only from the suspect list, the probability that the tracer correctly detect the traitor is n^{-c}, since the list size is n^c. Due to its combinatorial construction, there is a trade-off between the transmission overhead and the detection probability in the scheme of [6]. The value of c which gives the smallest header size and detection probability at the same time is $c = 1/2$ and in this case the header size is $O(\sqrt{n})$ and the detection probability is $1/\sqrt{n}$. Although the sublinear header size is achieved in the scheme, its detection probability becomes smaller as n gets larger.

In our scheme, efficient black-box tracing is achieved without the above list-tracing approach, i.e., there is no such trade-off. The header size is linear in k and the number of subsets of subscribers. Especially, if we set $k = \sqrt{n/8}$, the header size is $O(\sqrt{n})$, where we assume that the size of each subset is $2k$. The tracer can identify at least one traitor with overwhelming probability, regardless of n. By applying the key-generation method of [9] to the scheme of [8], our scheme enables the tracer to make it impossible for the revoked subscribers to compute the session key by substituting a random value for the element used only by the subscribers in one of the ℓ disjoint subsets if all of them in the subset are revoked. This helps to remove the restriction of the number of suspects in the previous schemes with black-box confirmation and hence efficient black-box tracing without sacrificing the detection probability is achieved. On the value of m, which is the number of repetition times of the test in the tracing algorithm, it is shown in [6] that at least one traitor can be identified with overwhelming probability if $m = O(n^2 \log^2 n)$. By using this result, it can be said that the running time of the tracing algorithm is $O(n^3 \log^2 n)$.

The main differences between the scheme of [6] and ours are the detection probability and the computational cost for decryption. While in the scheme of [6] the detection probability gets smaller as the value of n increases, in our scheme it is independent of n and always overwhelming. On the other hand, the scheme of [6] is efficient from the viewpoint of the computational cost for decryption. In the previous scheme, only a few exponentiations are needed, while the number of exponentiations required for decryption is $O(k)$ in ours. This can be alleviated by using a technique of vector-addition chain exponentiation [11, p.622].

6 Conclusions

In this paper, we have proposed a sublinear public-key black-box tracing scheme against self-defensive pirate decoders. This can be viewed as a solution to the open question to build a sublinear traitor tracing scheme that supports efficient black-box tracing against self-defensive pirate decoders with negligible probability of error.

References

1. D. Boneh: "The Decision Diffie-Hellman Problem", In *Proc. of the Third Algorithmic Number Theory Symposium*, LNCS 1423, Springer-Verlag, pp. 48–63, 1998.
2. D. Boneh and M. Franklin: "An Efficient Public Key Traitor Tracing Scheme", In *Proc. of CRYPTO '99*, LNCS 1666, Springer-Verlag, pp. 338–353, 1999.
3. B. Chor, A. Fiat, and M. Naor: "Tracing Traitors", In *Proc. of CRYPTO '94*, LNCS 839, Springer-Verlag, pp. 257–270, 1994.
4. B. Chor, A. Fiat, M. Naor, and B. Pinkas: "Tracing Traitors", *IEEE Transactions on Information Theory*, Vol. 46, No. 3, pp. 893–910, 2000.
5. Y. Dodis and N. Fazio: "Public Key Broadcast Encryption for Stateless Receivers", *ACM Workshop on Digital Rights Management (DRM '02)*, 2002.

6. A. Kiayias and M. Yung: "On Crafty Pirates and Foxy Tracers", In *Proc of Revised Papers from the ACM CCS-8 Workshop on Security and Privacy in Digital Rights Management (SPDRM '01)*, LNCS 2320, Springer-Verlag, pp. 22–39, 2002.
7. K. Kurosawa and Y. Desmedt: "Optimum Traitor Tracing and Asymmetric Schemes", In *Proc. of EUROCRYPT '98*, LNCS 1403, Springer-Verlag, pp. 145–157, 1998.
8. K. Kurosawa and T. Yoshida: "Linear Code Implies Public-Key Traitor Tracing", In *Proc. of PKC '02*, LNCS 2274, Springer-Verlag, pp. 172–187, 2002.
9. T. Matsushita: "A Flexibly Revocable Key-Distribution Scheme for Efficient Black-Box Tracing", In *Proc. of ICICS '02*, LNCS 2513, Springer-Verlag, pp. 197–208, 2002.
10. T. Matsushita and H. Imai: "Black-box Traitor Tracing against Arbitrary Pirate Decoders", In *Proc. of the 1st Workshop on Information Security Applications (WISA '00)*, pp. 265–274, 2000.
11. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices

R. Gennaro*, D. Leigh**, R. Sundaram***,+, and W. Yerazunis†

*IBM T.J.Watson Research Center, Yorktown Heights, NY, USA
rosario@watson.ibm.com
**Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
leigh@merl.com
***Northeastern University, Boston, MA, USA
koods@ccsneu.edu
†Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
yerazunis@merl.com

Abstract. We present a *batch* version of Schnorr's identification scheme. Our scheme uses higher degree polynomials that enable the execution of several Schnorr's protocol at a cost very close to that of a single execution. We present a full proof of security that our scheme is secure against impersonation attacks.

The main application of this result is a very efficient way for a party to prove that it holds several secret keys (i.e. identities), where each identity is linked to a specific authorization. This approach protects the privacy of the prover allowing her to prove only the required set of authorizations required to perform a given task, without disclosing whether she is in possession of other privileges or not.

We also show that our scheme is suitable to be implemented on low-bandwidth communication devices. We present an implementation of a smart card employing recent technology for the use of LEDs (Light Emitting Diodes) for bidirectional communication. Another contribution of our paper is to show that this new technology allows the implementation of strong cryptography.

1 Introduction

Identification, also known as entity authentication, is a process by which a *verifier* gains assurance that the identity of a *prover* is as claimed, i.e. there is no impersonation [MOV97, Sch96]. An identification scheme enables a prover holding a secret key to identify itself to a verifier holding the corresponding public key.

+ This research funded by MERL.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 276–292, 2004.

The primary objectives of an identification protocol are *completeness* - in the case of honest parties the prover is successfully able to authenticate itself to the verifier, and *soundness* - a dishonest prover has a negligible probability of convincing a verifier.

There are various grades of dishonesty and corresponding levels of security. The goal of the adversary is to impersonate the prover. As per the standard security framework [FFS88], the adversary is allowed various attacks on the honest prover, which complete before the impersonation attempt. A typical requirement of identification protocols is that they be secure against impersonation under *passive* attack, where the adversarial prover has access to transcripts of prover-verifier interactions. A stronger requirement is that protocols be secure against *active* attacks where the adversarial prover can actively play the role of a cheating verifier with the prover numerous times before the impersonation attempt. Security against impersonation under active attack has been the traditional goal of identification schemes.

However, in recent times interest has been growing in still stronger attacks, e.g. *concurrent attacks*. In these attacks, just like with active attacks the adversarial prover gets to play the role of cheating verifier prior to impersonation with the key distinction that the adversary is allowed to interact with multiple honest prover clones concurrently [FFS88].

It is very important to keep in mind that, in the real world, identification protocols provide assurances only at the instant of time when the protocol is successfully completed. It is therefore important to ensure that the identification process is tied to some form of ongoing *real-world* integrity service. At some level all identification schemes are vulnerable to the adversary who cuts in immediately after the successful identification of the legitimate party.

Zero-knowledge Protocols. A paradigm introduced in [FFS88] to construct identification protocols, is to construct *zero-knowledge proofs of knowledge*. These are protocols which allow a prover to demonstrate knowledge of a secret while revealing no information whatsoever other than the one bit of information regarding the possession of the secret [GMR85, FFS88].

A protocol is said to be *honest-verifier zero-knowledge* if it is zero-knowledge when interacting with honest verifiers. An honest-verifier zero-knowledge protocol has a weaker security guarantee than a general zero-knowledge protocol since it is possible that a dishonest verifier can extract information from the prover in the former protocol.

However, when used as identification schemes, the ultimate measure of the worth of a protocol lies in its security against impersonation attempts and a protocol that is secure against impersonation against concurrent attacks is considered to be "secure" even if it is "only" honest-verifier zero-knowledge. This happens if one is able to show that whatever information is leaked to the dishonest verifier, it does not help him in any impersonation attack.

Schnorr in [Sch91] presents such a protocol, based on the hardness of computing discrete logarithms. The details are described in the body of the paper, but here we just remark that Schnorr's protocol is an honest-verifier zero-knowledge

proof of knowledge of a discrete logarithm. Recently Bellare and Palacio in [BPa02] showed that under a slightly stronger assumption on the security of discrete logarithms, Schnorr's protocol is a secure identification scheme against concurrent attacks.

1.1 Authorization

Authorization is the conveyance to the verifier that the prover, has the sanction to gain access to a particular resource set, or belongs to a certain privilege class. Authorization may be effected by the proving of one or of multiple identities.

Consider now the following access control scenario. Users of a given system belong to various privilege classes. Access control classes for the data are defined using these privileges, i.e. as the users who own a given subset of privileges. For example the access control class for a given piece of data D, is defined as the users who own privileges P_1, P_2, P_3.

A way to implement such an access control system is to give each user a certified public key. The certificate would indicate the subset of privileges associated with this public key. Then in order to gain access, Alice performs an identification protocol based on her public key, and if her privileges are a superset of the ones required for the access she is attempting, access is granted.

There are several drawbacks with this approach. But the main one is a blatant violation of Alice's privacy. Whenever Alice proves her identity she reveals *all* her privileges, when, theoretically, in order to gain access she should have had to reveal only a subset of them.

It is clear that there are situation which warrant a privacy-preserving authorization mechanism, in which Alice can gain access by proving she owns the minimal set of required privileges.

This can be done by associating a different public key to each privilege. Then Alice would prove that she knows the secret keys required for the authorization. Using typical proofs of knowledge, like Schnorr's, to prove knowledge of k keys the user has to perform k proofs. Although these proofs can be performed in parallel, keeping the round complexity the same, the computational complexity goes up by a factor of k.

Thus an interesting question is if it is possible to perform a proof of knowledge of d secrets at the cost of less than d proofs. We answer this question in the affirmative (see below).

Another advantage of associating different keys to different privileges, is that the latter can be easily transferred simply by transferring the corresponding secret key.

1.2 Our Contributions

We present a *batch* version of Schnorr's protocol. In our scheme the prover can prove knowledge of d secret keys (discrete logarithms), at a cost slightly superior to the cost of a single Schnorr's protocol, thus saving a factor of d in computation and bandwidth over the best previously known solutions. We use degree d polynomials to represent an ordered list of d identities. We show that the result-

ing scheme is not only honest-verifier zero-knowledge, but that it is also a secure identification scheme against impersonation under concurrent attacks [BPa02].

This immediately yields a very efficient privacy-preserving authorization mechanism along the lines described in the previous section.

Finally, in order to showcase the efficiency of our proposal we present an implementation in a very low-bandwidth environment. We use recently proposed technology to use Light Emitting Diodes (LEDs) as bi-directional communication devices. We believe that another interesting contribution of our paper is to show that this new technology is robust enough to implement strong cryptographic solutions.

1.3 Related Work

Besides the works cited above [GMR85, FFS88, Sch91], another widely used protocol for identification is the one proposed by Guillou and Quisquateur in [GQu88]. This scheme is more efficient than Schnorr's and very suitable to low-power computation devices. When proving multiple identities simultaneously, our batch technique makes the advantage of using GQ over Schnorr's disappear very quickly. Indeed only for very small values of d (the number of identities being proven), d parallel executions of GQ beat our batch Schnorr protocol in efficiency[1]. It would be interesting to devise a batch version of the Guillou-Quisquateur protocol, but we were not able to do so.

In any case, when comparing our scheme with running d Guillou-Quisquateur schemes, one should remember that the GQ scheme is based on a different assumption (RSA inversion) than the Schnorr's protocol.

Our new identification scheme is related to the concept of batch verification of signatures [BGR98]. As far as we know there has not been any work on batch verification for identification protocols. A straightforward application of the techniques in [BGR98] to our problem would yield a much less efficient protocol. Moreover, the mathematical techniques we use are fundamentally different than the ones in [BGR98].

Recently the area of privacy-preserving protocol has received a lot of attention. We refer the reader especially to the works by Camenisch and Lysyanskaya [CL01, CL02], where the concept of *group signature* is used to show how a user can prove membership in a certain privilege class, without revealing her true identity. These solutions offer a very strong privacy guarantee, as a user can safely prove his privileges to various verifiers, who would not be able to *link* her various transactions. On the other hand our solution does not protect the

[1] Jumping ahead, assume we perform Schnorr's scheme, with parameters p, q such that $|p| = 1024$ and $|q| = 160$. Then one execution of the protocol costs about 240 multiplications for the prover (i.e. one exponentiation mod$p$, with a 160-bit exponent). On the other hand, if we perform the GQ scheme over a 1024-bit RSA modulus, using a small public exponent (like 3), and security parameter 80, then the prover's cost is about 80 multiplications. Thus GQ is approximately 3 times as fast as Schnorr's. Which means that for $d > 3$, i.e. when proving more than 3 identities simultaneously, our batch Schnorr protocol becomes more attractive than GQ.

identity of the user, but simply allows her to prove the minimal set of privileges required for a given transaction. But if verifiers collude they can link the user's transactions and reconstruct the set of privileges she holds. For example if Alice proves to Bob that she belongs to privilege class P1, and to Charles that she belongs to the P2 class, it is possible for Bob and Charles together to understand that Alice holds both P1 and P2 privileges. On the other hand our solution is much simpler and more efficient that solutions based on group signatures, thus it could be preferable in a scenario in which collusion is not really a problem. Moreover some of our batching techniques can be used to speed-up solutions based on group signatures (since there, the proof of possession of several keys is a subprotocol).

2 Preliminaries

In this section we recall the basic definition of proof of knowledge, the computational assumptions that we are going to need, and Schnorr's protocol. In the following the acronym PPT stands for "probabilistic polynomial-time".

2.1 Proofs of Knowledge

Polynomial Time Relationships. Let $\mathcal{R}$ be a polynomial time computable relationship, i.e. a language of pairs (y, w) such that it can be decided in polynomial time in $|y|$ if $(y, w) \in \mathcal{R}$ or not. With $\mathcal{L}_{\mathcal{R}}$ we denote the language induced by $\mathcal{R}$ i.e. $\mathcal{L}_{\mathcal{R}} = \{y \ : \ \exists w \ : \ (y, w) \in \mathcal{R}\}$.

More formally an ensemble of polynomial time relationships $\mathcal{PTR}$ consists of a collection of families $\mathcal{PTR} = \cup_n \mathcal{PTR}_n$ where each $\mathcal{PTR}_n$ is a family of polynomial time relationships $\mathcal{R}_n$. To an ensemble $\mathcal{PTR}$ we associate a randomized *instance generator* algorithm IG that on input 1^n outputs the description of a relationship $\mathcal{R}_n$. In the following we will drop the suffix n when obvious from the context.

Example: The instance generator algorithm on input 1^n outputs an n-bit prime q, a $poly(n)$-prime p, such that $q|p-1$ and an element g of order q in Z_p^*. The corresponding relationship is that of pairs $(y, w) \subset Z_p^* \times Z_q$ such that $y = g^w \bmod p$.

Proofs of Knowledge. In a proof of knowledge for a relationship $\mathcal{R}$, two parties, Prover P and Verifier V, interact on a common input y. The Prover also holds a secret input w, such that $(y, w) \in \mathcal{R}$. The goal of the protocol is to convince V that P indeed knows such w. Ideally this proof should not reveal any information about w to the verifier, i.e. be zero-knowledge.

The protocol should thus satisfy certain constraints. In particular it must be *complete*: if P knows w then V should accept. It should be *sound*: for any (possibly dishonest) prover who does not know w, the verifier should almost always reject. Finally it should be *zero-knowledge*: no (poly-time) verifier (no matter what possibly dishonest strategy she follows during the proof) can learn any information about w.

A formal definition of proofs of knowledge can be found in [BGo93] (improving on the original definition in [FFS88]). Informally, the concept of "knowing" w is formalized by showing that w can be computed in polynomial-time if we have black-box access to P. This is done by constructing a *witness extractor* which runs in probabilistic polynomial time, and computes w with a probability related to the probability that the Prover makes the Verifier accept.

The concept of zero-knowledge is formalized via the existence of a probabilistic polynomial time simulator S that on input y and interacting with a possibly cheating Verifier outputs transcripts with the same probability distribution as the real prover (who knows w).

A formal definition follows. With $[\mathsf{P}(y,w),\mathsf{V}(y)]$ we denote the output of the protocol, i.e. 1 iff V accepts. With $\pi_{\mathsf{P}}(n)$ we denote the probability that a prover P makes the verifier accept, i.e.

$$\pi_{\mathsf{P}}(n) = Prob[\ \mathcal{R}_n \leftarrow \mathsf{IG}(1^n)\ ;\ [\mathsf{P}(y,\cdot),\mathsf{V}(y)] = 1\ ]$$

where the statement y can be chosen by P.

Definition 1. *We say that* (P,V) *is a proof of knowledge for a relationship* $(\mathcal{PTR},\mathsf{IG})$ *if the following properties are satisfied:*

Completeness. *For all* $(y,w) \in \mathcal{R}_n$ *(for all* $\mathcal{R}_n$*) we have that* $[\mathsf{P}(y,w),\mathsf{V}(y)]$=1.

Witness Extraction. *There exist a probabilistic polynomial time* knowledge extractor KE*, a function* $\kappa : \{0,1\}^* \rightarrow [0,1]$ *and a negligible function* ϵ*, such that for all PPT* P'*, if* $\pi_{\mathsf{P}'}(n) > \kappa(n)$ *then* KE*, given rewind access to* P'*, computes* w *such that* $(y,w) \in \mathcal{R}_n$ *with probability at least* $\pi_{\mathsf{P}'}(n) - \kappa(n) - \epsilon(n)$.

Zero-Knowledge. *For every PPT Verifier* V' *there exist a probabilistic polynomial time* simulator $\mathsf{SIM}_{\mathsf{V}'}$*, such that for all* $(y,w) \in \mathcal{R}_n$ *the two random variables*

$$\mathsf{View}[\mathsf{P}(y,w),\mathsf{V}'(y)]$$
$$\mathsf{View}[\mathsf{SIM}_{\mathsf{V}'}(y),\mathsf{V}'(y)]$$

are indistinguishable.

The function κ is called the *knowledge error* and measures the probability, inherent to the protocol, that a cheating prover can convince the verifier without knowing w. What we require is that if a prover convinces the verifier with a probability higher than κ, then we can extract the witness with a success probability related to the difference.

2.2 Identification Schemes

In an identification scheme a prover P and a verifier V interact on input a public key (generated together with its matching secret key by a key generation algorithm KG). The prover holds the matching secret key, and his goal is to convince the Verifier of this fact, and thus of his identity.

An *impersonation attack* is when an adversary $\mathcal{A}$ tries to convince the verifier that he is the honest prover. This kind of attack is called a *passive attack*, if the adversary attempts impersonation only after having witnessed several correct executions of the identification protocol between the prover and honest verifiers. We said that an attack, is *active*, if the adversary before trying to impersonate the prover has engaged with him in the identification protocol, playing the role of a (possibly dishonest) verifier.

Finally, and this is the notion we consider in this paper, we say that an active impersonation attack is a *concurrent attack* if the interactions between the adversary and the honest prover before the impersonation attack, can be carried out in a concurrent fashion (i.e. with an arbitrary scheduling of messages).

So we can consider the following game. A pair of keys sk,pk is chosen according to the distribution induced by KG on input 1^n the security parameter. The prover is given sk and pk is made public. Then the adversary $\mathcal{A}$ engages as a verifier in several concurrent executions of the identification protocol with the prover. He then finally runs one execution of the protocol, as the prover, with an honest verifier. We denote with $\mathsf{adv}_{\mathcal{A}}(n)$ the probability that $\mathcal{A}$ makes the verifier accept at the end of this game.

Definition 2. *We say that an identification protocol is secure against concurrent impersonation attack if* $\mathsf{adv}_{\mathcal{A}}(n)$ *is negligible in* n.

2.3 Discrete Logarithm Assumptions

Since its introduction in the seminal paper by Diffie and Hellman [DH78], the discrete logarithm assumption has been widely used to construct cryptographic algorithms. Here we are going to use a well established variant of the assumption that considers the hardness of computing discrete logs in subgroups of prime order.

Consider the example we described above. On input a security parameter 1^n, we generate an n-bit prime q, a $poly(n)$-prime p, such that $q|p-1$ and an element g of order q in Z_p^* (the multiplicative group of integers mod$p$). In the group generated by g we can consider the exponentiation function that maps $w \in Z_q$ to $y = g^w \bmod p$. The discrete log assumption says that if we choose w at random then it is infeasible to compute w, when given only y.

Assumption 1. *We assume that computing discrete logarithm is hard, i.e. for every PPT Turing Machine* $\mathcal{I}$ *(for inverter) the following probability*

$$\mathsf{adv}_{\mathcal{I}}(n) = Prob[\mathcal{I}(p, q, g, y = g^w \bmod p) = w]$$

is negligible in n. *The probability is taken over the internal coin tosses of* $\mathcal{I}$, *the random choices of* q *as* n*-bit prime,* p *as a* $poly(n)$*-bit prime such that* $q|p-1$, $w \in_R Z_q$, *while* g *is an arbitrary element of order* q *in* Z_p^*.

In the following we are going to use a stronger variant of the discrete log assumption, introduced in [BNPS01, BPa02]. In this variant once we have selected p, q at random and chose g, we give the inverter $\mathcal{I}$ access to two oracles. The

challenge oracle Ch, when invoked outputs a random element in the group generated by g. The *discrete log* oracle DL when queried on a value $y \in (g)$ will output w such that $y = g^w \bmod p$. The goal of $\mathcal{I}$ is to invert *all* the values issued to him by the challenge oracle (and $\mathcal{I}$ must invoke Ch at least once), but is restricted to invoke DL a number of times, which is strictly smaller than the number of times he invoked Ch. We denote with $\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(n)$ the probability (taken over the choices of p, q and the internal coin tosses of $\mathcal{I}$ and Ch) that $\mathcal{I}$ succeeds in this game.

Assumption 2. *We assume that the problem of* one more inversion *of discrete logarithms is hard, i.e. we assume that* $\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(n)$ *is negligible in* n.

Note that when the number of queries to Ch is equal to 1, this assumption is equivalent to Assumption 1. Though Assumption 2 is new, it looks reasonable and has the advantage enunciated in [BPa02] of reducing the security of our identification scheme to the hardness of a well-specified number theoretic problem.

2.4 Schnorr's Identification Scheme

Let p and q be two primes such that $q|p-1$ and $|q| = n$. Let $g \not= 1$ be an element of order q in Z_p^*. Let G_q be the subgroup generated by g. The integers p, q, g are known and can be common to a group of users.

An identity consists of a private/public key pair. The private key w is a random non-negative integer less than q. The public key is computed as $y = g^{-w} \bmod p$.

The protocol is described in Figure 1.

It is well known that $\mathsf{Schnorr}$ is an honest-verifier zero-knowledge proof of knowledge of w, the discrete logarithm of y. The reader is referred to [Sch91] for details. The protocol is only honest-verifier ZK, because if a dishonest verifier chooses the challenge e in a non-random way (particularly dependent on the first message x) we are not able to simulate the interaction[2].

However in [BPa02] it is shown that the Schnorr scheme is secure against impersonation, under concurrent attacks, under the assumption that discrete logarithm is secure under one more inversion in the underlying group.

3 The New Identification Scheme

In this section we present our generalization of Schnorr's scheme to the case in which the prover wants to prove multiple identities.

A naive generalization of Schnorr's scheme would be to do the simultaneous authentication of d identities by composing d rounds in parallel. In other words

[2] Note that it is also necessary to check that y is in the proper group, by checking that $y^q = 1 \bmod p$, see [Bur90]. This can be added as a verification step, or the verifier can trust the certification authority that certified y to have performed the test. A similar requirement holds for our protocol.

Schnorr

Common Input: p, q, g, y. A security parameter t.
Secret Input for the Prover: $w \in Z_q$ such that $y = g^{-w} \bmod p$.

1. *Commitment by Prover.* Prover picks $r \in_R Z_q$ and sends $x = g^r \bmod p$ to the Verifier.

   Prover $\xrightarrow{x = g^r}$ Verifier

2. *Challenge from Verifier.* Verifier picks a number $e \in_R [1..2^t]$ and sends it to the Prover.

   Prover $\xleftarrow{e}$ Verifier

3. *Response from Prover.* Prover computes $s = r + w \cdot e \bmod q$ and sends it to the Verifier.

   Prover $\xrightarrow{s = r + w \cdot e}$ Verifier

   The Verifier checks that $x = g^s \cdot y^e \bmod p$ and accepts if and only if equality holds.

Fig. 1. Schnorr's protocol

the prover would send over d commitments and the verifier would reply with d challenges - one per identity. Note that this scheme has a communication and computation cost that is d times the cost of Schnorr's original scheme. A possible improvement would be to use the same challenge for all rounds, and apply batch verification techniques (such as the ones in [BGR98]) to the last verification step. Even with these improvements, the communication and computation cost of the whole scheme would still be higher by a factor of d (the prover would still have to send and compute d commitments).

We propose a more efficient scheme where the prover sends *one* commitment and the verifier sends one challenge across all identities. The prover's response is generalized from a degree one polynomial to a degree d polynomial formed from the d secret keys. We are able to show that the resulting scheme is sound and further that it is secure against impersonation under concurrent attacks by extending the corresponding arguments in [Sch91] and [BPa02] respectively. We present two theorems that demonstrate that the new scheme is an honest-verifier zero knowledge proof of knowledge and also a secure identification against impersonation under concurrent attacks.

The parameters are very similar to Schnorr. Let p and q be two primes such that $q|p-1$. Let $g \not= 1$ be an element of order q in Z_p^*. The integers p, q, g are public, and can be common to a group of users.

We have d identities, each consisting of a private/public key pair indexed by i. The private keys w_i are non-negative integers less than q, chosen uniformly at random. The public keys are computed as $y_i = g^{-w_i} \bmod p$.

The Prover initiates the protocol by sending over the list of public keys y_i for which it claims to possess the corresponding private keys w_i. The protocol is described in Figure 2.

Batch-Schnorr

Common Input: $p, q, g, y_1, \ldots, y_d$. A security parameter t.
Secret Input for the Prover: $w_i \in Z_q$ such that $y_i = g^{-w_i} \bmod p$.

1. *Commitment by Prover.* Prover picks $r \in_R Z_q$ and sends $x = g^r \bmod p$ to the Verifier.

   Prover $\xrightarrow{\quad x = g^r \quad}$ Verifier

2. *Challenge from Verifier.* Verifier picks a number $e \in_R [1..2^{(t+\log d)}]$ and sends it to the Prover.

   Prover $\xleftarrow{\quad e \quad}$ Verifier

3. *Response from Prover.* Prover computes $s = r + \Sigma_i w_i \cdot e^i \bmod q$ and sends it to the Verifier.

   Prover $\xrightarrow{\quad s = r + \Sigma_i w_i \cdot e^i \quad}$ Verifier

   The Verifier checks that $x = g^y \cdot \Pi_i y_i^{e^i} \bmod p$, and accepts if and only if equality holds.

Fig. 2. Batch version of Schnorr's protocol

Theorem 1. Batch-Schnorr *is an honest-verifier zero-knowledge proof of knowledge for d discrete logarithms.*

The complete proof is provided in [GLSY].

Notice that the protocol is not zero-knowledge in the general case since a dishonest verifier could choose a challenge that is dependent on the commitment making it difficult to generate transcripts with the same distribution, without knowing the secret keys. Informally, however the reason no information is revealed is that the numbers x and y, the commitment and the response, are essentially random. This is the intuition behind the proof of security as an identification scheme.

The following theorem (Theorem 2) shows that Batch-Schnorr is an identification scheme secure against impersonation under concurrent attacks. As mentioned before, this is our ultimate end goal. We extend the proof in [BPa02]

(which shows the security of Schnorr's scheme under this kind of attack) to our scheme. We remind the readers that in a concurrent attack the adversarial prover is allowed to play the role of the cheating verifier and interact concurrently with multiple honest prover clones prior to the impersonation attempt. Similar to [BPa02] our proof is based on the assumption that discrete exponentiation is secure under d more inversions in the underlying group (Assumption 2).

Let $\mathcal{A}$ denote the adversary that first takes on the role of fraudulent verifier and interacts concurrently with several honest prover clones before subsequently taking on the role of fraudulent prover. Let $\mathsf{adv}_{\mathcal{A}}(k)$ denote the probability that $\mathcal{A}$ is successful at impersonation.

Theorem 2. *If $\mathcal{A}$ succeeds in an impersonation attack on* Batch-Schnorr *with probability $\mathsf{adv}_{\mathcal{A}}(k)$ then there exists an inverter $\mathcal{I}$ such that for every k*

$$\mathsf{adv}_{\mathcal{A}}(k) \leq 2^{-t} + (\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(k))^{1/(d+1)}.$$

Proof. We show how to construct an inverter $\mathcal{I}$ that interacts with $\mathcal{A}$ the impersonation adversary. Via this interaction $\mathcal{I}$ will compute the discrete logarithm of all the n points it gets from the challenge oracle, by querying the discrete log oracle, at most $n - d$ times.

First the inverter $\mathcal{I}$ queries Ch, the challenge oracle, d times and obtains d random group elements $y_i = g^{-w_i}$. It then runs $\mathcal{A}$ in cheating verifier mode using the y_i's as the public key. For the j^{th} clone prover the commitment x_j is obtained by querying the challenge oracle Ch. The third round response s_j to challenge e_j is computed by querying the discrete log oracle DL on the value $x_j \Pi_{k=1}^{d} y_k^{-e_j^k}$. Notice that this is a perfect simulation of a real concurrent attack. With n we denote the total number of queries to the challenge oracle. Notice that $\mathcal{I}$ queried the discrete log oracle only $n - d$ times.

Now $\mathcal{I}$ runs $\mathcal{A}$ in cheating prover mode $d + 1$ times, rewinding it each time to the beginning (of the phase in which $\mathcal{A}$ acts as a prover). This in particular means that the commitment issued by $\mathcal{A}$ stays the same, since its internal state is the same. If any two challenges are the same then the inverter $\mathcal{I}$ fails.

Let $x = g^r$ be the commitment and let s_i be the response corresponding to the distinct challenges e_i. If the cheating prover $\mathcal{A}$ fails even once then the inverter $\mathcal{I}$ fails. If the cheating prover $\mathcal{A}$ succeeds each of the $d + 1$ times, then the inverter $\mathcal{I}$ has $d + 1$ equations of the form $s_i = r + \Sigma_j w_j \cdot e_i^j$, with $d + 1$ unknowns, r and the d secret keys w_j. By inverting the Van der Monde matrix formed from these equations, they can be solved to obtain the w_j's. These are the answers to the first d queries $\mathcal{I}$ made to Ch.

Recall that $\mathcal{I}$ must answer *all* the challenges he received from Ch. But the answer to each query x_j can be easily computed as $s_j - \Sigma_k w_k \cdot e_j^k$.

Thus with $n - d$ queries, $\mathcal{I}$ succeeds in inverting all the n points asked to the challenge oracle. The probability of success is the probability that $\mathcal{A}$ succeeds $d + 1$ times. We will now estimate this probability. We first prove an auxiliary result that is a generalization of an equivalent result in [BPa02].

Lemma 1 (Generalized Reset Lemma). *Consider any prover (potentially cheating). Let A and B be random variables over the space of the random coins, RP, of the prover. Let A denote the probability, taken over e, that the verifier accepts. Let B denote the probability, taken over e's, that when the verifier is reset and run $d+1$ times, a different e is generated each time and the verifier accepts each time. Let $acc = E(A)$ and $res = E(B)$. Then $acc \leq 2^{-t} + res^{1/(d+1)}$.*

Proof. Let $1/c = 2^{t+\log d}$ be the size of the challenge set i.e. $\#e$. It is easy to see that $B \geq A(A-c)(A-2c)\dots(A-dc)$. This implies that $B \geq (A-dc)^{(d+1)}$ which yields that $E(A) \leq dc + E(B)^{1/(d+1)}$ or $E(A) \leq 2^{-t} + E(B)^{1/d+1}$.

Now observe that $\mathsf{adv}_{\mathcal{A}}(k) = E(acc)$, where the expectation is taken over the choice of y_i and the knowledge gained as the cheating verifier. Similarly $\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(k) = E(res)$. Applying the reset lemma we see that

$$adv_{\mathcal{A}}(k) = E(acc) \leq E(2^{-t} + (res)^{1/(d+1)}) = 2^{-t} + E((res)^{1/(d+1)})$$

then by applying Jensen's inequality

$$adv_{\mathcal{A}}(k) \leq 2^{-t} + (E(res))^{1/(d+1)} = 2^{-t} + (\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}})^{1/(d+1)}$$

This completes the proof of Theorem 2.

The following corollary is a straightforward consequence of Theorem 2.

Corollary 1. *Under Assumption 2, and if $t = \omega(\log k)$, then* Batch-Schnorr *is a secure identification scheme against i mpersonation under concurrent attack.*

Note that the assumption that t is super-logarithmic in k is necessary, otherwise the scheme can be broken by guessing the verifier's challenge.

3.1 Efficiency Analysis

For a list of d identities Batch-Schnorr uses only $O(logd)$ more bits of communication than Schnorr's scheme for a single identity (assuming the same security level).

In terms of computation Batch-Schnorr requires $2d$ extra modular multiplications for the prover. The verifier has to perform $d+1$ modular exponentiations, while in Schnorr's scheme it has to perform 2.

Notice that this is much faster than the known way of proving d identities simultaneously, which consists of d copies of Schnorr's protocol (in the particular the verifier would have to perform $2d$ exponentiations instead of $d+1$).

3.2 Authorization Using Multiple Identities

As we discussed in the Introduction, our identification scheme is suitable to implement Authorization using multiple identities without incurring a huge efficiency cost.

When a user joins a particular privilege class he is given a new public key, its matching secret key and a certificate that associates the key to that particular

privilege class. Another possibility would be to have a unique key for each class, but that would make revocation very difficult to handle, as revoking one user in the class (say because her key was compromised or because she does not belong to the class anymore), would involve replacing the key of all users in the class.

When a user needs to access certain data or services he uses our identification protocol to prove possession of the minimal set of privileges required for that access to take place.

Another advantage of our approach is that it is easy to transfer privileges among parties. A motivating scenario for our application is having a smart card be able to talk to another smart card and transfer a subset of privileges to it (equivalent to a high ranking employee in a corporation enabling selective access to a lower ranked employee). In our scheme using multiple identities it is a simple matter to transfer the private keys corresponding to the selected list of privileges.

4 Implementation

In order to test the efficiency of our scheme we have performed an implementation of our scheme. To carry out the implementation we used a recently proposed technology based on Light Emitting Diodes. We believe that another contribution of our paper is to show that this new technology allows the implementation of strong cryptography.

Light Emitting Diodes, or LEDs, are one of the most ubiquitous interface components. Their diverse applications include numeric displays, flashlights, vehicle brake lights (and possibly even headlights [Hel03]), traffic signals and the omni-present power-on indicator. LEDs are so commonly used as light emitters that people often forget that they are fundamentally photodiodes and hence light *detectors*. Although LEDs are not optimized for light detection they are very effective at it. The interchangeability between solid-state light emission and detection was widely publicized in the 1970s by Forrest W. Mims [Mim86, Mim93], but has since largely been forgotten.

Recently, a novel microprocessor interface circuit was invented which can alternately emit and detect light using an LED [DYL02]. In addition to the LED and two digital I/O pins of the microprocessor, the circuit requires only a single current limiting resistor. When forward-biased the LED emits light and when back-biased it detects/measures the ambient light. The implications of LED-based data communication are significant, since it is essentially a software interface technique that uses existing hardware with minimal modification. "Every LED connected to a microprocessor can be thought of as a generic two-way data port" [DYL02]. One can conceive of numerous applications e.g. using the power light on consumer appliances as a maintenance port for reading service information and uploading new firmware, or capturing a car stereo's fault log through the front panel display.

We show how to build smartcards that communicate via LEDs and implement our Batch-Schnorr protocol.

4.1 Hardware

Batch-Schnorr was implemented using the Microchip PIC16LF628 microcontroller. The hardware was composed of a small printed circuit board 2cm by 4cm, a single push-button switch, an LED, a 3-volt lithium coin-cell battery, a capacitor and two resistors. The PIC uses 8-bit instruction words and runs at 5 MIPS (million instructions per second). It has 16KB of write-able storage. The prototypes were also equipped with an in-circuit programming connector, which allowed us to download code into the microcontroller. We also devised a small adapter board to convert this connector to Microchip's standard RJ-11 in-circuit debugging module. A mass produced version should cost less than a dollar more than a similar LED keychain flashlight. The range of communication is a few centimeters at best and the data rate is 250 bits/second in each direction. We implemented Batch-Schnorr representing the prover in its full functionality. The verifier was implemented as a LED directly controlled by a PC.

See Appendix B for a picture of our implementation.

4.2 Security

We chose a security parameter setting of $t = 95$. This is generally considered adequate security (see [Sch96]) for most practical purposes. We used $d = 32$. This made $t + \log d = 100$. This forced the prime q to be 200 bits long because of the existence of the $O(q^{1/2})$ baby-step-giant-step algorithm for finding discrete logs (see [Sch91]). In conjunction with the existence of the general number field sieve (see [LOd91]) this, in turn, forced the prime p to be about 1500 bits long.

4.3 Prover

The bulk of the implementation effort lay in the code for the prover. An important aspect of our implementation of Batch-Schnorr was that storage was at a premium. This is common with most smart cards where the storage is needed both for code as well as data.

The main operation performed by the prover is modular multiplication. We initially attempted an implementation of the Fast Fourier Transform (see [Str88]) of Cooley and Tukey, which takes $O(n \log n)$ bit operations. However it turned out that our practical implementations of this scheme had high code complexity, even though it is more efficient asymptotically.

Hence, we adopted a scheme that utilizes a pre-computed table to substantially save on both code complexity as well as computation time. For each of the private keys we stored a pre-computed table of the residues modulo q of the product of the private key with the powers of 2 up to $2^{1+\log q}$. Then to multiply the private key with any given number we added the residues corresponding to the powers of 2 present in the binary representation of that number. The residue modulo q of $2^{1+\log q}$ enabled us to reduce the overflow when doing addition, so that we always had a number with $\log q$ bits. Upon receiving the challenge e we first computed a similar table consisting of the residues modulo q of the product of e with the powers of 2 up to $2^{1+\log q}$. We then used this table to compute

the powers of e, and then used the pre-computed tables of the secret keys to compute $y = r + \Sigma_i si \cdot e^i \bmod q$. This enhancement enabled the implementation to run in less than 2 seconds for our choice of the security parameters.

5 Conclusion and Extensions

We have presented a batch version of Schnorr's protocol. In our scheme a prover can prove knowledge of d keys at essentially the same cost as proving knowledge of a single key. We believe this protocol can find several applications in the cryptography literature.

We discussed the application of privacy-preserving authorization mechanisms. Also we presented an implementation of our protocol employing a new technology to use Light Emitting Diodes as two-way communication devices. We believe this to be another interesting contribution of our paper.

In terms of future research, it would be interesting to devise a batch version of the Guillou-Quisquateur identification protocol.

References

[BGR98] M. Bellare, J. Garay and T. Rabin. "Fast batch verification for modular exponentiation and digital signatures". Advances in Cryptology- Eurocrypt '98 Proceedings, Lecture Notes in Computer Science Vol. 1403, K. Nyberg ed, Springer-Verlag, 1998.

[BGo93] M. Bellare and O. Goldreich. "On defining proofs of knowledge". Advances in Cryptology - CRYPTO '92 Proceedings, Lecture Notes in Computer Science Vol. 740, E. Brickell ed, Springer-Verlag, 1993.

[BNPS01] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. *The one-more RSA Inversion Problem.* Financial Cryptography'01. Final version available at `http://eprint.iacr.org/2002/002`

[BPa02] Bellare M., and Palacio A., "GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks," Advances in Cryptology-CRYPTO '02, (2002).

[Bur90] Burmester M., "A remark on the efficiency of identification schemes," EUROCRYPT '90, pp. 493-495 (1990).

[CL01] J. Camenisch and A. Lysyanskaya. "An efficient system for non-transferable anonymous credentials with optional anonymity revocation." EUROCRYPT'01 pp.93-118.

[CL02] J. Camenisch and A. Lysyanskaya. "Signature schemes with efficient protocols." Security in Communication Networks Workshop. 2002

[Cha90] Chaum D., "Zero knowledge undeniable signatures," Advances in Cryptology - CRYPTO '90, pp 458-464 (1990).

[CDM00] Cramer R., Damgard I., and MacKenzie P.D., "Efficient zero-knowledge proofs of knowledge without intractability assumptions," Public Key Cryptography '00, pp. 354-372 (2002).

[CSh98] Cramer R., and Shoup V., "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," Advances in Cryptology-CRYPTO '98, pp. 13-25 (1998).

[CSh99] R.Cramer and V.Shoup. "Signature schemes based on the Strong RSA assumption". 6th ACM Conference on Computer and Communication Security 1999.

[DPr89] Davies D. W., and Price W. L., "Security for computer networks," Wiley (1989).

[deR91] de Rooij P., "On the security of the Schnorr scheme using preprocessing," EUROCRYPT '91, pp. 71-80 (1991).

[DYL02] Dietz, P., Yerazunis W., and Leigh, D., "Very low-cost sensing and communication using bidirectional LEDs," to appear in Ubicomp 2003. http://www.merl.com/papers/TR2003-35/ (2002) patent pending.

[DH78] W. Diffie and M. Hellman. *New Directions in Cryptography.* IEEE Transactions on Information Theory, IT-22(6):644–654. 1976.

[DDN00] D.Dolev, C.Dwork and M.Naor. "Non-malleable Cryptography". SIAM J. Comp. 30(2):391–437, 2000.

[FFS88] Feige U., Fiat A., and Shamir A., "Zero-knowledge proofs of identity," Journal of Cryptology, vol. 1, pp. 77-94 (1988).

[For94] Ford W., "Computer communications security: principles, standard protocols and techniques," Prentice Hall (1994).

[GHR99] Gennaro R., Halevi S. and Rabin T., "Secure Hash-and-Sign Signatures Without the Random Oracle". Eurocrypt '99 LNCS no. 1592, pages 123-139 (1999).

[GKR97] Gennaro R., Krawczyk H. and Rabin T., "RSA-based undeniable signatures," Advances in Cryptography - CRYPTO '97, pp 132-149 (1997). Also in Journal of Cryptology, vol 13., pp 397-416 (2000).

[GLSY] Gennaro R., Leigh D., Sundaram R. and Yerazunis W., "Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices," Tech Report. http://www.ccs.neu.edu/ koods/papers.html (2004).

[GMR85] Goldwasser S., Micali S., and Rackoff C., "The knowledge complexity of interactive proof-systems," Proceedings of the 17th Annual ACM Symposium on Theory of Computing, pp. 291-304 (1985).

[GMR88] S.Goldwasser, S.Micali, and R.L.Rivest. "A digital signature scheme secure against adaptive chosen-message attacks". *SIAM J. Computing*, 17(2):281–308, April 1988.

[GQu88] Guillou L. C., and Quisquater J. -J., "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," EUROCRYPT '88, pp. 123-128 (1988).

[GUg86] Guillou L. C., and Ugon M., "Smart card: a highly reliable and portable security device," Advances in Cryptology-CRYPTO '86, pp. 464-479 (1986).

[GUQ92] Guillou L. C., Ugon M., and Quisquater J. -J., "The smart card: a standardized security device dedicated to public cryptography," Contemporary Cryptology: The Science of Information Integrity, IEEE, pp. 561-613 (1992).

[Hel03] Helms N., "Bright LEDs power headlights," Electronics News, http://www.dialelectronics.com.au/articles/1e/0c01631e.asp (2003).

[Kah67] Kahn D., "The codebreakers: the story of secret writing," Macmillan (1967).

[LOd91] LaMacchia B. A., and Odlyzko A. M., "Computation of discrete logarithms in prime fields," Designs, Codes and Cryptography, vol. 1, pp. 46-62 (1991).

[Mim86] Mims F. M., "Siliconnections: Coming of age in the electronic era," McGraw Hill (1986).
[Mim93] Mims F. M., "LED circuits and projects," H. W. Sams and Co. (1993).
[MOV97] Menezes A. J., van Oorschot P. C., and Vanstone S. A., "Handbook of applied cryptography," CRC Press (1997). http://www.cacr.math.uwaterloo.ca/hac/
[MTh79] Morris R., and Thompson K., "Password security: a case history," Communications of the ACM, vol. 22, pp. 594-597 (1979).
[NSc78] Needham R. M., and Schroeder M. D., "Using encryption for authentication in large networks of computers," Communications of the ACM, vol. 21, pp. 993-999 (1978).
[QGB89] Quisquater J. -J., Guillou L., and Berson T., "How to explain zero-knowledge protocols to your children," Advances in Cryptology-CRYPTO '89, LNCS 435, pp. 628-631 (1989).
[Sch96] Schneier B., "Applied cryptography," Wiley (1996).
[Sch91] Schnorr C. P., "Efficient signature generation for smart cards," Journal of Cryptology, vol. 4, no. 3, pp. 161-174 (1991).
[Str88] Strang G., "Linear algebra and its applications," Harcourt Brace (1988).
[Ted02] Tedeschi, W., "Trying to shift shape of PC screens," http://www.nytimes.com/2002/11/04/technology/04ECOM.html

A LightKey Image

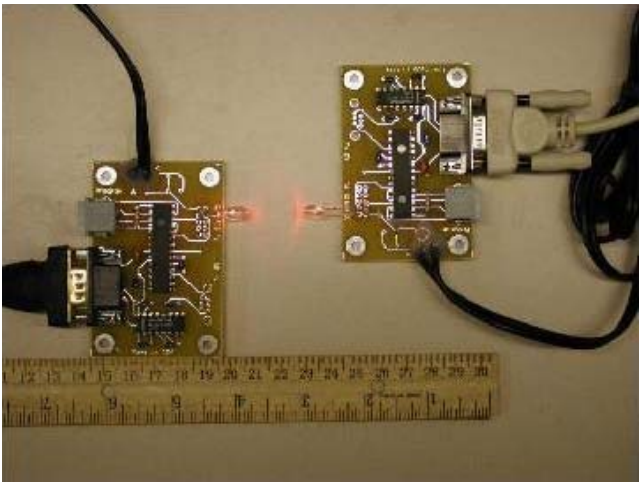


Fig. 3. LightKey Image

Secret Handshakes from CA-Oblivious Encryption

Claude Castelluccia[1,2], Stanisław Jarecki[1], and Gene Tsudik[1]

[1] Information and Computer Science, University of California, Irvine, CA 92697, USA
{ccastell, stasio, gts}@ics.uci.edu
[2] INRIA Rhône-Alpes, 655 Avenue de l'Europe, 38334 Saint Ismier CEDEX, France
claude.castelluccia@inrialpes.fr

Abstract. Secret handshakes were recently introduced [BDS+03] to allow members of the same group to authenticate each other *secretly*, in the sense that someone who is *not* a group member cannot tell, by engaging some party in the handshake protocol, whether that party is a member of this group. On the other hand, any two parties who *are* members of the same group will recognize each other as members. Thus, a secret handshake protocol can be used in any scenario where group members need to identify each other without revealing their group affiliations to outsiders.

The work of [BDS+03] constructed secret handshakes secure under the *Bilinear Diffie-Hellman* (BDH) assumption in the Random Oracle Model (ROM). We show how to build secret handshake protocols secure under a more standard cryptographic assumption of *Computational Diffie Hellman* (CDH), using a novel tool of *CA-oblivious* public key encryption, which is an encryption scheme s.t. neither the public key nor the ciphertext reveal any information about the Certification Authority (CA) which certified the public key. We construct such CA-oblivious encryption, and hence a handshake scheme, based on CDH (in ROM). The new scheme takes 3 communication rounds like the [BDS+03] scheme, but it is about twice cheaper computationally.

Keywords: authentication, privacy, anonymity, encryption.

1 Introduction

A secret handshake scheme, introduced by Balfanz et al. [BDS+03], allows two members of the same group to identify each other *secretly*, in the sense that each party reveals his/her affiliation to the other only if the other party is also a group member. For example, a CIA agent Alice might want to authenticate herself to Bob, but only if Bob is also a CIA agent. Moreover, if Bob is *not* a CIA agent, the protocol should not help Bob in determining whether Alice is a CIA agent or not. This secrecy property can be extended to ensure that group members' affiliations are revealed only to members who hold specific *roles* in the group. For example, Alice might want to authenticate herself as a CIA agent with security level one if *and only if* Bob is a CIA agent with security clearance two, and vice versa.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 293–307, 2004.

In other words, if A is a member of group G_a with role r_a and B is a member of G_b with role r_b, a secret handshake scheme guarantees the following [BDS+03]:

- A and B authenticate each other if and only if $G_a = G_b$.[1]
- If $G_a \neq G_b$ then both parties learn only *the sole fact* that $G_a \neq G_b$.
- A can choose not to reveal anything about herself unless B is a member with particular role r_b (and vice versa).[2]
- An eavesdropper or a man in the middle learn nothing from the protocol.

As observed in [BDS+03], secret handshakes seem to require new cryptographic protocols since they can not be easily obtained from existing tools in the "cryptographic toolbox". For example, group signatures [CVH91, ACJT00] might appear to be an attractive building block for secret handshakes. However, they offer anonymity and unlinkability of group members' signatures, not secrecy of membership itself. In the interactive variant of group signatures, called *identity escrow* [KP98], one party can prove to another its membership in a group in an anonymous fashion. However, what turns out to be quite difficult is the seemingly simple issue of two parties proving group membership to each other simultaneously, in such a way that one party never reveals its group membership to another unless the former is also a member of the same group.

Secret Handshake Scheme as a "CA-oblivious PKI". To be usable in practice, a secret handshake scheme must provide efficient revocation of any group member by the Group Authority (GA) which administers the group. To support this functionality we will consider secret handshake schemes which, like the scheme of [BDS+03], are similar to PKI's (Public Key Infrastructures), where the role of a group authority corresponds to that of a Certification Authority (CA) in a PKI. Namely, to become a member of a group a party needs the GA to issue a certificate on an ID bitstring which the CA agrees to assign to this party. The certificate must include a CA-specific *trapdoor* which corresponds to this ID.[3] To revoke some party, the CA puts that party's ID on a revocation list. To perform a handshake, two parties first exchange their ID's, and then proceed only if the ID of the other party is not on the revocation list of their CA. Since the secret handshake protocol must hide one's group affiliation from outsiders, the ID's will be random strings picked from the same domain by all the CA's.[4]

[1] However, as noted by [BDS+03], a handshake protocol cannot be *fair* in the sense that if $G_a = G_b$ then one party is going to learn about it first and could abort the protocol and thus withhold their group affiliation from the counterparty.

[2] To simplify the presentation, we will ignore roles for most of the paper. However, as we show in appendix A.1, they can be added easily.

[3] For example, in an identity based encryption scheme, the trapdoor is a secret key corresponding to the public key which can be recovered from ID and the public parameters associated with the CA. In a standard PKI system, this correspondence has an added level of indirection: The trapdoor t is a secret key corresponding to the public key PK which is in turn bound to the ID string by a signature of CA on the $(ID|PK)$ pair.

[4] To make protocol runs executed by the same party *unlinkable*, [BDS+03] propose that a single user gets multiple (ID,certificate) pairs, each to be used only once.

In this setting, constructing a secret handshake scheme amounts to solving the following protocol problem: For a given CA, Alice wants to prove to Bob that she possesses a trapdoor t_A issued by this CA on her ID_A, but only if Bob possesses a trapdoor t_B issued by *the same* CA on his ID_B (and vice versa). Moreover, the protocol must be "CA-oblivious" in the sense that if a cheating Bob is not in the group administered by a given CA, and hence does not hold a CA-specific trapdoor t_B associated with ID_B, then his interaction with Alice must not help him in guessing if Alice belongs to this group or not. (And vice versa for an honest Bob and a cheating Alice.) While this protocol problem can be solved in principle with general 2-party secure computation techniques, the issue remains whether it can be solved with a *practical* protocol, at a cost comparable to standard authentication protocols.

Existing Solutions Based on Bilinear Maps. The secret handshake protocol of [BDS$^+$03] is based on bilinear maps, which can be constructed using Weil pairings on elliptic curves [Jou02, Gag02]. The protocol of [BDS$^+$03] builds on the non-interactive key-agreement scheme of [SOK00], and works as follows. As in the identity based encryption scheme of [BF01], A and B can compute each other's public keys from each other's ID's and from the public parameters associated with the CA. If Alice is a group member, she can use her trapdoor t_A corresponding to PK_A to non-interactively compute a session key from (t_A, PK_B). Similarly, if Bob is a group member he can compute *the same* session key from (t_B, PK_A). The two parties can then verify if they computed the same key via a standard MAC-based challenge-response protocol. Under the *Bilinear Diffie-Hellman* (BDH) assumption, it is easy to show (in the Random Oracle Model) that an attacker who does not hold the correct trapdoor cannot compute the session key. Moreover, the MAC-based challenge response confirmation protocol has the needed property that without the knowledge of the key, one learns nothing from the counterparty's responses.

Thus, the "CA-obliviousness" property of the protocol of [BDS$^+$03] follows from two properties of cryptosystems built on bilinear maps: (1) that the receiver's public key can be recovered by the sender from the receiver's ID, and thus the receiver does not need to send any information revealing his CA affiliation to the sender, and (2) knowing their public keys, the two parties can establish a session key non-interactively, and thus they again do not reveal any CA-specific information. Given that the first property relies on identity based encryption, and that the only practical IBE known so far is based on bilinear maps [BF01], it seems that BDH is indeed needed for secret handshakes.

Our Contributions. In this paper we show that efficient secret handshake (SH) schemes can be built using weaker and more standard assumption than the BDH, namely the *Computational Diffie Hellman* (CDH) assumptions. However, our security arguments, just like those for the BDH-based scheme of [BDS$^+$03] remain in the so-called Random Oracle Model (ROM). Moreover, the proposed scheme is computationally at least twice cheaper than the scheme of [BDS$^+$03].

We show this in several steps: First, we generalize the IBE-based secret handshake solution sketched above by showing that an efficient *four-rounds* secret handshake protocol can be built using any *PKI-enabled* encryption with the additional property of *CA-obliviousness*. We define the notion of (chosen-*plaintext* secure) PKI-enabled encryption, which generalizes both the Identity Based Encryption schemes, and the standard encryption schemes used in the context of a PKI system like X.509. We define the CA-obliviousness property for this notion of PKI-enabled encryption, which requires that both the public-key-related information which the receiver provides to the sender, and the ciphertext sent from the sender to the receiver, do not reveal which CA issued the receiver's certificate. We then show that every CA-oblivious PKI-enabled encryption leads to a four-round secret handshake protocol whose cost is one decryption and one encryption for each party. We also show an alternative construction, which creates a three-round secret handshake protocol using any CA-oblivious PKI-enabled encryption equipped with the so-called zero-knowledge "signature of knowledge" [CS97] of the private decryption key.

Next, we combine ElGamal encryption and Schnorr signatures to construct a practical CA-oblivious PKI-enabled encryption secure under the CDH assumption (in ROM), which thus leads to a four-round secret handshake protocol secure under CDH. However, since this encryption admits a very practical (in ROM) ZK signature of knowledge of the private key, which is simply the Schnorr signature scheme itself, this results in a secret handshake scheme which takes three rounds, like the scheme of [BDS$^+$03], and which involves one multiexponentiation and one or two exponentiations per player. Compared to the cost of the scheme of [BDS$^+$03], where each player computes a pairing of two elements one of which is known in advance, this is about twice less expensive, according to the results of Barreto et al. [BKLS02].

We also improve the functionality of a secret handshake system by showing that our CDH-based SH schemes support "blinded" issuance of the member certificates in the sense that the CA does not learn the trapdoors included in the certificate, and thus, in contrast to the BDH-based SH scheme of [BDS$^+$03], the CA cannot impersonate that member.

Finally, we note that the CA-oblivious encryption we devise can be also applied to provide a CDH-based solution to the *Hidden Credentials* problem [HBSO03], which generalizes the notion of secret handshakes to general PKI trust evaluations where two communicating partners are not necessarily certified by the same group/certification authority. This problem was also given only a BDH-based solution so far, in [HBSO03].

Related Work. As described in [BDS$^+$03], existing anonymity tools such as anonymous credentials, group signatures, matchmaking protocols, or accumulators, have different goals than secret handshakes, and it is indeed unclear how to achieve a secret handshake scheme from any of them. Thus we will briefly discuss here only the new work of [LDB03], which proposes a new notion "oblivious signature-based envelopes", which is closely related to the secret handshake problem. The oblivious envelope notion they define is very similar to our notion of

PKI-enabled encryption, but with a weaker obliviousness property. Namely, they only require that the encrypting party does not know if the receiver possesses a CA-certified public/private key or not, but the protocol does not hide the identity of the CA itself from the receiver. In contrast, our CA-oblivious encryption notion requires the protocol to hide this identify. Thus, while our CA-oblivious encryption gives an oblivious signature-based envelope for Schnorr signatures, the other direction is not clear. In particular, it remains an open problem if CA-oblivious encryption and/or secret handshakes can be constructed based on the RSA assumption.[5]

Organization. In section 2 we revise the definitions of an SH scheme [BDS+03], restricting them to "PKI-like" SH schemes we consider here. In section 3 we define the notion of a *PKI-enabled encryption*, and the *CA-obliviousness* property for such encryption. In section 4 we construct a CA-oblivious encryption secure under CDH in ROM. In section 5 we give two general constructions of SH schemes from any CA-oblivious encryption. In appendix A we show how to support roles and blinded issuing of CA certificates.

2 Definition of Secret Handshakes

We adapt the definition of a secure Secret Handshake [SH] scheme from [BDS+03] to what we call "PKI-like" SH schemes. Our definitions might potentially restrict the notion of a secret handshake scheme, but both the SH scheme of [BDS+03] and our SH schemes fall into this category. We define an SH scheme as a tuple of probabilistic algorithms Setup, CreateGroup, AddMember, and Handshake s.t.

- Setup is an algorithm executed publicly on the high-enough security parameter k, to generate the public parameters params common to all subsequently generated groups.
- CreateGroup is a key generation algorithm executed by a GA, which, on input of params, outputs the group public key G, and the GA's private key t_G.
- AddMember is a protocol executed between a group member and the GA on GA's input t_G and shared inputs: params, G, and the bitstring ID (called a *pseudonym* in [BDS+03]) of size regulated by params. The group member's private output is the trapdoor t produced by GA for the above ID.
- Handshake is the authentication protocol, i.e. the SH protocol itself, executed between players A, B on public input ID_A, ID_B, and params. The private input of A is (t_A, G_A) and the private input of B is (t_B, G_B). The output of the protocol for either party is either a *reject* or *accept*.

We note that AddMember can be executed multiple times for the same group member, resulting in multiple (ID, t) authentication tokens for that member. We

[5] In the poster advertising the preliminary version of these results in PODC'04, we erroneously claimed that we know how to get RSA-based CA-oblivious encryption scheme, but this claim was incorrect, and this issue is still an open problem.

also note that in all the SH schemes discussed here the output of the Handshake protocol can be extended to include an authenticated session key along with the "accept" decision.

2.1 Basic Security Properties

An SH scheme must be complete, impersonator resistant, and detector resistant:[6]

Completeness. If honest members A, B of the same group run Handshake with valid trapdoors t_A, t_B generated for their ID strings ID_A, ID_B and for the same group $G_A = G_B$, then both parties output "accept".

Impersonator Resistance. Intuitively, the impersonator resistance property is violated if an honest party V who is a member of group G authenticates an adversary $\mathcal{A}$ as a group member, even though $\mathcal{A}$ **is not** a member of G. Formally, we say that an SH scheme is *impersonator resistant* if every polynomially bounded adversary $\mathcal{A}$ has negligible probability of winning in the following game, for *any* string ID_V which models the ID string of the victim in the impersonation attack:

1. We execute $\mathsf{params} \leftarrow \mathsf{Setup}(1^k)$, and $(G, t_G) \leftarrow \mathsf{CreateGroup}(\mathsf{params})$.
2. $\mathcal{A}$, on input (G, ID_V), invokes the AddMember algorithm on any number of group members ID_i of his choice. (The GA's inputs are ID_i's, G, and t_G.)
3. $\mathcal{A}$ announces a new $ID_{\mathcal{A}}$ string, different from all the ID_i's above. (This models a situation where the ID_i's belong to group members who are malicious but who might be revoked.)
4. $\mathcal{A}$ interacts with the honest player V in the Handshake protocol, on common inputs $(ID_{\mathcal{A}}, ID_V)$, and on V's private inputs G and t_V, where $t_V \leftarrow \mathsf{AddMember}((G, ID_V), t_G)$.

We say that $\mathcal{A}$ *wins* if V outputs "accept" in the above Handshake instance.

We note that the above impersonator resistance property is rather weak, and that stronger versions of this property are possible, and indeed advisable. Namely, the attacker *should* be allowed to run the protocol several times against V, and be able to ask for additional trapdoors after each attempt, before he announces that he is ready for the true challenge. Also, the attacker *could* be allowed to ask for trapdoors on additional $ID_i \neq ID_A$ strings during the challenge protocol with V. We adopt the simplest and weakest definition here to reduce the level of formalism in the paper. Nevertheless, we believe that our schemes remain secure under these stronger notions as well.

Remark: We note that even such strengthened notion of impostor resistance is not strong enough to be used in practice. For example, the resulting notion

[6] Once we restrict the notion of SH schemes to the PKI-like SH schemes, the security properties defined originally in [BDS+03] can be stated in a simpler way. Specifically, their properties of *impersonator resistance* and *impersonator tracing* are subsumed by our *impersonator resistance*, and their *detector resistance and tracing* is subsumed by what we call *detector resistance*.

makes no claims of security against the man in the middle attacks, and no claims if the adversary triggers a handshake protocol with an honest owner of the $ID_{\mathcal{A}}$ identity at any time *before* the adversary tries to authenticate himself to V under this identity. Therefore we do not claim that the above impostor resistance property is sufficient in practice. Instead, the above *authentication*-like notion of impostor resistance has to be first extended to *Authenticated Key Agreement* [AKE]. We discuss this further in the Section 2.2 below.

Detector Resistance. Intuitively, an adversary $\mathcal{A}$ violates the detector resistance property if it can decide whether some honest party V is a member of some group G, even though $\mathcal{A}$ **is not** a member of G. Formally, we say that an SH scheme is *detector resistant* if there exists a probabilistic polynomial-time algorithm SIM, s.t. any polynomially bounded adversary $\mathcal{A}$ cannot distinguish between the following two games with the probability which is non-negligibly higher than 1/2, for *any* target ID string ID_V:

Steps 1-3 proceed as in the definition of *Impersonator Resistance*, i.e. on input ID_V and a randomly generated G, $\mathcal{A}$ queries GA on adaptively chosen ID_i's and announces some challenge string $ID_{\mathcal{A}}$, $ID_{\mathcal{A}} \neq ID_i$ for all i.

4-1. In game 1, $\mathcal{A}$ interacts with an algorithm for the honest player V in the $\mathsf{Handshake}$ protocol, on common inputs $(ID_{\mathcal{A}}, ID_V)$, and on V's private inputs G and $t_V = \mathsf{AddMember}((G, ID_V), t_G)$.

4-2. In game 2, $\mathcal{A}$ interacts with SIM on common inputs $(ID_{\mathcal{A}}, ID_V)$.

5. $\mathcal{A}$ can query GA on additional strings $ID_i \neq ID_{\mathcal{A}}$.

6. $\mathcal{A}$ outputs "1" or "2", making a judgment about which game he saw.

Similarly to impersonator resistance, stronger notions of detector resistance are possible and indeed advisable. In particular, the adversary should be able to trigger several executions of the handshake protocol with player V, and he should be able to interleave these instances with instances executed with the rightful owner of the $ID_{\mathcal{A}}$ identity. We adopt the above weak notion for simplicity, but our schemes satisfy these stronger notion as well.

2.2 Extensions and Other Security Properties

Authenticated Key Exchange. As mentioned in the previous section, the impostor resistance property defined above is only a weak authentication-like property which does not give sufficient guarantees in practice. Moreover, in practice one would like to extend the notion of a secret handshake from one where participants' outputs are binary decisions "accept" / "reject", to authenticated key exchange, where parties output instead either "reject" or a secure session *key*. We believe that the SH schemes we propose, just like the original SH protocol of [BDS+03], can be easily extended to AKE protocols using the standard AKE protocol techniques. However, the formal security analysis of the resulting protocols requires adoption of AKE formalism [BR93, CK02, Sho99], which is beyond the scope of this paper.

Group-Affiliation Secrecy Against Eavesdroppers. Our schemes also protect secrecy of participants' group affiliations against eavesdroppers, even if the eavesdropper is a malicious member of the same group. An observer of our SH protocols does not even learn if the participants belong to the same group or not. We do not formally define security against eavesdroppers, because it is very similar to the security against active attackers which we do define, the impersonator and detector resistance. Moreover, if the protocol participants first establish a secure anonymous session, e.g. using SSL or IKE, and then run the SH protocol over it, the resulting protocol is trivially secure against eavesdroppers.

Unlinkability. A potentially desirable property identified in [BDS+03], is *unlinkability*, which extends privacy protection for group members by requiring that instances of the handshake protocol performed by the same party cannot be efficiently linked. This can be achieved trivially (but inefficiently) by issuing to each group member a list of one-time certificates, each issued on a randomly chosen ID, to be discarded after a single use. Unfortunately, an honest member's supply of one-time certificates can be depleted by an active attacker who initiates the handshake protocol enough times. Indeed, while one can run our SH schemes using multiple certificates to offer some heuristic protections against linking, constructing an efficient and perfectly unlinkable SH scheme remains an open problem.

3 Definition of PKI-Enabled CA-Oblivious Encryption

We define the notion of *PKI-enabled* encryption, which models the use of standard encryption in the context of a PKI system, and also generalizes Identity Based Encryption. We define *one-way security* for PKI-enabled encryption, adapting a standard (although weak) notion of one-way security of encryption to our context, and we define a novel *CA-obliviousness* property for such schemes.

A PKI-enabled encryption is defined by the following algorithms:

- Initialize is run on a high-enough security parameter, k, to generate the public parameters params common to all subsequently generated Certification Authorities (CAs).
- CAInit is a key generation algorithm executed by a CA. It takes as inputs the system parameters params and returns the public key G and the private key t_G of the CA.
- Certify is a protocol executed between a CA and a user who needs to be certified by this CA. It takes CA's private input t_G, and public inputs G (assume that G encodes params) and string ID which identifies the user, and returns *trapdoor* t and *certificate* ω as the user's outputs.
- Recover is an algorithm used by a *sender*, a party who wants to send an encrypted message to a user identified by some string ID, to recover that user's public key. It takes inputs (G, ID, ω) and outputs a public key PK.
- Enc is the actual encryption algorithm which takes inputs message m and the public key PK (assume that PK encodes params and G), and outputs a ciphertext c.

– Dec is the decryption algorithm which takes as inputs the ciphertext c and the trapdoor t (as well as possibly params, G, ID, and ω, all of which can be encoded in t), and returns m.

The above algorithms must satisfy the obvious *correctness* property that the decryption procedure always inverts encryption correctly.

It is easy to see (see footnote 3) that this notion of encryption indeed models both regular encryption schemes in the PKI context as well as the Identity Based encryption schemes.

One-Way Security. We define the security of PKI-enabled encryption only in the relatively weak sense of so-called *one-way* security, namely that the attacker who does not own a trapdoor for some public key cannot decrypt an encryption of a random message. This is a weaker notion than the standard *semantic* security for an encryption, but we adopt it here because (1) it simplifies the definition of security, (2) one-way security is all we need in our construction of a secure SH scheme, and (3) in the Random Oracle Model, it is always possible to convert a one-way secure encryption into a semantically secure encryption, or even a CCA-secure encryption using the method of Fujisaki and Okamoto [FO99].

The definition of security for PKI-enabled encryption is very similar to the definition of security of an IBE scheme: We say that a PKI-enabled encryption scheme is *One-Way* (OW) secure on message space $\mathcal{M}$ under *Chosen-Plaintext Attack* (CPA), if every polynomially-bounded adversary $\mathcal{A}$ has only negligible probability of winning the following game:

1. The Initialize and CAInit algorithms are run, and the resulting public key G is given to $\mathcal{A}$.
2. $\mathcal{A}$ repeatedly triggers the Certify protocol under the public key G, on ID strings ID_i of $\mathcal{A}$'s choice. In each instance $\mathcal{A}$ receives (t_i, ω_i) from the CA.
3. $\mathcal{A}$ announces a pair $(ID_{\mathcal{A}}, \omega)$, where $ID_{\mathcal{A}} \neq ID_i$ for all ID_i's queried above.
4. $\mathcal{A}$ receives $c = \mathsf{Enc}_{PK}(m)$ for a random message $m \in \mathcal{M}$ and $PK = \mathsf{Recover}(G, ID_{\mathcal{A}}, \omega)$.
5. $\mathcal{A}$ is allowed to trigger the Certify algorithm on new $ID_i \neq ID_{\mathcal{A}}$ strings of his choice, getting additional (t_i, ω_i) pairs from the CA.
6. $\mathcal{A}$ outputs a message m'. If $m' = m$ then we say that $\mathcal{A}$ *wins*.

CA-Obliviousness. Informally, PKI-enabled encryption is CA-oblivious if (1) the receiver's message to the sender, i.e., the pair (ID, ω), hides the identity of the CA which certified this ID; and (2) the sender's messages to the receiver, i.e., ciphertexts, do not leak any information about the CA which the *sender* assumed in computing the receiver's public key. Consequently, in a standard exchange of messages between the receiver and the sender, neither party can guess which CA is assumed by the other one. Formally, we call a PKI-enabled encryption scheme *CA-oblivious* under two conditions:

(I) It is *"Receiver CA-oblivious"*, i.e., if there exists a probabilistic polynomial-time algorithm $SIM_{(R)}$, s.t. no polynomially-bounded adversary $\mathcal{A}$ can distinguish between the following two games with probability non-negligibly higher than 1/2, for *any* target ID string ID_R:

1. The Initialize and CAInit algorithms are executed, and the resulting parameters params and the public key G is given to $\mathcal{A}$.
2. $\mathcal{A}$ can trigger the Certify protocol on any number of ID_i's.

3-1. In game 1, $\mathcal{A}$ gets (ID_R, ω_R), where ω_R is output by the Certify protocol on G and ID_R.

3-2. In game 2, $\mathcal{A}$ gets (ID_R, r) where $r = SIM_{(R)}(\mathsf{params})$.

4. $\mathcal{A}$ can trigger the Certify protocol some more on any $ID_i \neq ID_R$.
5. $\mathcal{A}$ outputs "1" or "2", making a judgment about which game he saw.

(II) It is *"Sender CA-oblivious"*, i.e., if there exists a probabilistic polynomial-time algorithm $SIM_{(S)}$ s.t. no polynomially-bounded adversary $\mathcal{A}$ can distinguish between the following two games, with probability non-negligibly higher than 1/2:

1. The Initialize and CAInit algorithms are executed, and the resulting parameters params and the public key G is given to $\mathcal{A}$.
2. $\mathcal{A}$ can trigger the Certify protocol any number of times, for public key G and group members ID_i's of $\mathcal{A}$'s choice.
3. $\mathcal{A}$ announces pair (ID_R, ω_R) on which he wants to be tested, where $ID_R \neq ID_i$ for all i.

4-1. In game 1, $\mathcal{A}$ gets $c = \mathsf{Enc}_{PK_R}(m)$ for random $m \in \mathcal{M}$ and $PK_R = \mathsf{Recover}(G, ID_R, \omega_R)$.

4-2. In game 2, $\mathcal{A}$ gets $c = SIM_{(S)}(\mathsf{params})$.

5. $\mathcal{A}$ can query GA on some more ID_i's s.t. $\forall_i, ID_i \neq ID_R$.
6. $\mathcal{A}$ outputs "1" or "2", making a judgment about which game he saw.

4 Construction of CA-Oblivious Encryption

We construct a CA-oblivious PKI-enabled encryption scheme secure based on the CDH assumption in the Random Oracle Model.[7]

- Initialize picks the standard discrete logarithm parameters (p, q, g) of security k, i.e., primes p, q of size polynomial in k, s.t. g is a generator of a subgroup in $\mathbb{Z}_p^*$ of order q. Initialize also defines hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^k$. (Both hash functions are modeled as random oracles, but we note that H' is not essential in this construction and can be easily removed.)
- CAInit picks random private key $x \in \mathbb{Z}_q$ and public key $y = g^x \bmod p$.
- In Certify on public inputs (y, ID), the CA computes the Schnorr signature on string ID under the key y [Sch89], i.e., a pair $(\omega, t) \in (\mathbb{Z}_p^*, \mathbb{Z}_q)$ s.t. $g^t = \omega y^{H(\omega, ID)} \bmod p$. The user's outputs are the trapdoor t and the certificate ω. The signature is computed as $\omega = g^r \bmod p$, and $t = r + xH(\omega, ID) \bmod q$, for random $r \leftarrow \mathbb{Z}_q$.

[7] We remark that since the Identity Based Encryption scheme of [BF01] is also a CA-oblivious PKI-based encryption scheme, the SH construction of Section 5 applied to that encryption scheme implies efficient BDH-based SH schemes.

- $\mathsf{Recover}(y, ID, \omega)$ outputs $PK = \omega y^{H(\omega, ID)} \bmod p$.
- $\mathsf{Enc}_{PK}(m)$ is an ElGamal encryption of message $m \in \{0,1\}^k$ under the public key PK: It outputs a ciphertext $[c_1, c_2] = [g^r \bmod p, m \oplus H'(PK^r \bmod p)]$, for random $r \in \mathbb{Z}_q$.
- Dec is an ElGamal decryption, outputing $m = c_2 \oplus H'(c_1^t \bmod p)$.

Theorem 1. *The above encryption scheme is CA-oblivious and One-Way secure under the CDH assumption in the Random Oracle Model.*

Proof (of One-Way Security). Assume that an adversary $\mathcal{A}$ breaks one-wayness of this encryption scheme. This means that after receiving n Schnorr signatures (t_i, ω_i) on ID_i's of his choice, $\mathcal{A}$ sends a tuple (ID, ω) s.t. $ID \neq ID_i$ for all the above ID_i's, and (in ROM), to break one-wayness $\mathcal{A}$ must query the H' oracle on $c_1^t \bmod p$ where $g^t = \omega y^{H(\omega, ID)} \bmod p$. Therefore, $\mathcal{A}$ must exponentiate a random element c_1 it received to the exponent t. Hence, what we need to argue that, even though $\mathcal{A}$ receives n signatures (t_i, ω_i) on her ID_i's, she cannot produce a new pair (ID, ω) s.t. she can exponentiate a random elements c_1 to exponent t where $g^t = \omega * y^{h(\omega, ID)}$. Now, this is very similar to proving the chosen message attack security of the underlying Schnorr signature scheme, where one argues that, after receiving n signatures, $\mathcal{A}$ cannot produce a new triple (ID, ω, t) s.t. $g^t = \omega * y^{h(\omega, ID)}$. Hence, our proof is very similar to the forking-lemma proof for Schnorr signature security in [PS96]. However, here we reduce the successful attack not to computing discrete logarithm, but to breaking the CDH assumption by computing m^x on input $y = g^x$ and a random value m.

To reduce $\mathcal{A}$'s ability to succeed in this protocol to computing m^x on the Diffie-Hellman challenge (g, g^x, m), we first simulate, as in the proof of Schnorr signature security, the signatures (t_i, ω_i) that $\mathcal{A}$ gets on her ID_i's, by taking random t_i, c_i, computing $\omega_i = g^{t_i} * y^{-c_i} \bmod p$, and assigning $H(\omega_i, ID_i)$ to c_i. Since the verification equation is satisfied and t_i, c_i are picked at random, this is indistinguishable from receiving real signatures. Then, as in the forking lemma argument of [PS96], we can argue that if $\mathcal{A}$'s probability of success is ϵ, the probability that $\mathcal{A}$ executed twice in a row succeeds in *both* executions *and* sends the *same* (ID, ω) challenge in both of them, is at least ϵ^2/q_h where q_h is the number of queries $\mathcal{A}$ makes to the hash function H (see [PS96]). The forking lemma used in the security proof of the Schnorr signature scheme shows that if two conversations with an adversary produce triples (t, ω, ID) and (t', ω, ID), where in first conversation $H(\omega, ID) = c$ and in the second $H(\omega, ID) = c'$ for some random c, c', then $x = DL_g(y)$ can be computed as $x = (s - s')/(c - c') \bmod q$, because $g^t = \omega * y^c$ and $g^{t'} = \omega * y^{c'}$. By applying the same forking lemma to our case, adversary $\mathcal{A}$ produces two *exponentiations* m^t and $m^{t'}$, instead of forgeries t, t', but still we have that $x = DL_g(y) = (t - t')/(c - c')$. Therefore, with probability ϵ^2/q_h we can break the CDH challenge and compute $m^x = m^{(t-t')/(c-c')} = (m^t/m^{t'})^{1/(c-c')} \bmod p$.

Note that if the success probability ϵ is higher than negligible, and if $\mathcal{A}^*$ is an efficient algorithm and hence the number of queries q_h is polynomial, then the probability of CDH break ϵ^2/q_h is non-negligible as well.

Proof (of CA-Obliviousness). It is easy to see that neither ω nor the ciphertext $C = [c_1, c_2]$ reveal any information about the CA: Since $\omega = g^r$ for random r, ω is independent from CA's public key y, and hence the scheme is receiver CA-oblivious. Ciphertext $C = [c_1, c_2]$ on a random message m is also independent from the group key y, because $c_1 = g^r$ for random r and c_2 is computed by xoring $H'(PK^r)$ with the random m.

5 Secret Handshakes from CA-Oblivious Encryption

We first show how to built a secure four-rounds SH scheme using CA-oblivious PKI-enabled encryption. Given a CA-oblivious one-way secure PKI-enabled encryption scheme (Initialize, CAInit, Certify, Recover, Enc, Dec), and a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$ modeled as a random oracle, we specify a secret handshake scheme as follows: Algorithms Setup, CreateGroup, and AddMember, are simply set to Initialize, CAInit, and Certify, respectively, while algorithm Handshake proceeds as follows. A's inputs are (ID_a, ω_a, t_a) and B's inputs are (ID_b, ω_b, t_b).[8]

1. $(B \longrightarrow A)$: $ID_b,\ \omega_b$
   - A obtains $PK_b = \mathsf{Recover}(G, ID_b, \omega_b)$
   - A picks $r_a \leftarrow \mathcal{M}$ and $ch_a \leftarrow \{0,1\}^k$
   - A computes $C_a = \mathsf{Enc}_{PK_b}(r_a)$
2. $(A \longrightarrow B)$: $ID_a,\ \omega_a,\ C_a,\ ch_a$
   - B obtains $PK_a = \mathsf{Recover}(G, ID_a, \omega_a)$
   - B obtains $r_a = \mathsf{Dec}_{t_b}(C_a)$
   - B picks $r_b \leftarrow \mathcal{M}$ and $ch_b \leftarrow \{0,1\}^k$
   - B computes $C_b = \mathsf{Enc}_{PK_a}(r_b)$
   - B computes $resp_b = H(r_a, r_b, ch_a)$
3. $(B \longrightarrow A)$: $C_b,\ resp_b,\ ch_b$
   - A obtains $r_b = \mathsf{Dec}_{t_a}(C_b)$
   - if $resp_b \neq H(r_a, r_b, ch_a)$, A outputs FAIL; otherwise A outputs ACCEPT.
   - A computes $resp_a = H(r_a, r_b, ch_b)$
4. $(A \longrightarrow B)$: $resp_a$
   - if $resp_a \neq H(r_a, r_b, ch_b)$, B outputs FAIL; otherwise B outputs ACCEPT.

We note that the above protocol can be easily turned into an Authenticated Key Exchange (AKE) protocol (secure in the ROM model) if the two parties compute their authenticated session key as $K = H(r_a, r_b)$.

Theorem 2. *If the PKI-enabled encryption is CA-oblivious and One-Way secure, the above construction yields a Secret Handshake scheme secure in the Random Oracle Model (ROM).*

[8] Group member's trapdoor on string ID in this SH scheme is a *pair* (ω, t) produced by the Certify protocol. We can also assume that (ID_a, ID_b) are public inputs.

Proof (of Impersonator Resistance). Assume that $\mathcal{A}$ violates with non-negligible probability ϵ the impersonator resistance property against some honest member V identified by ID_V. Assume that $\mathcal{A}$ plays the role of A and V plays the role of B (the other case is easier because B has to speak first). Therefore with prob. ϵ, $\mathcal{A}$ sends a valid $resp_a = H(r_a, r_b, ch_b)$ response to B. In the ROM model, that can happen with non-negligible probability only if $\mathcal{A}$ querries the oracle for $H(\cdot)$ on the input (r_a, r_b, ch_b) s.t., in particular, r_b was the value picked by V and sent to $\mathcal{A}$ in the form of a ciphertext $C_b = \mathsf{Enc}_{PK_a}(r_b)$ for $PK_a = \mathsf{Recover}(G, ID_a, \omega_a)$, where (ID_a, ω_a) are sent by $\mathcal{A}$ in its first message to V. Therefore, in ROM, we can use $\mathcal{A}$ to create a break $\mathcal{A}'$ against the one-way security of the encryption scheme:

On input G, $\mathcal{A}'$ passes the public key G to $\mathcal{A}$. When $\mathcal{A}$ can makes a querry ID_i, so does $\mathcal{A}'$, passing back (ω_i, t_i) to $\mathcal{A}$. When $\mathcal{A}$ announces that he is ready for the impersonation challenge against V, $\mathcal{A}'$ passes as *his* encryption challenge the pair (ID_a, ω_a) sent by $\mathcal{A}$ in his first message to V. On encryption challenge $c = \mathsf{Enc}_{PK_a}(m)$ where m is chosen at random in $\mathcal{M}$, $\mathcal{A}'$ passes the same challenge as its response $C_b = c$ to $\mathcal{A}$, together with a random challenge value ch_b and $resp_b$ picked at random. The only way $\mathcal{A}$ can tell between this communication and a conversation with an honest V is by querying H on (r_a, r_b, ch_a) for $r_b = \mathsf{Dec}_{t_a}(C_b) = m$. Otherwise, as we argued above, he queries H on (r_a, r_b, ch_b) with probability almost ϵ. In either case, since $\mathcal{A}$ can make only polynomially-many queries to H, $\mathcal{A}'$ can pick one such query at random, and $\mathcal{A}'$ will have a non-negligible chance of outputing $r_b = m$. Thus $\mathcal{A}'$ breaks the one-wayness of the encryption scheme.

Proof (of Detector Resistance). We will show a simulator SIM s.t. if $\mathcal{A}$ distinguishes between interactions with SIM and interactions with a group member, we can break the one-way security of the encryption scheme. Assume again that the adversary $\mathcal{A}$ plays the role of A and V plays the role of B. Assume that the underlying encryption scheme is CA-oblivious, and therefore there exist simulators $SIM_{(S)}$ and $SIM_{(R)}$ which satisfy the two CA-obliviousness criteria. We define a simulator SIM, running on input $(ID_A, ID_V, \mathsf{params})$, as follows: (1) To simulate V's first message SH-1, SIM sends $ID_b = ID_V$ together with $\omega_b = SIM_{(R)}(\mathsf{params})$, (2) To simulate B's second message SH-3, SIM sends $resp_b$ and ch_b picked at random, and $C_b = SIM_{(S)}(\mathsf{params})$.

If $\mathcal{A}$ can distinguish a conversation with such SIM from a conversation with a true group member V, then by a standard hybrid argument, since the $SIM_{(S)}$ and $SIM_{(R)}$ simulators produce messages which are indistinguishable from the messages of an honest B, it must be that $\mathcal{A}$ distinguishes random values $resp_b$ chosen by SIM from values $resp_b = H(r_a, r_b, ch_a)$ computed by a real player. But this can happen only if $\mathcal{A}$ makes an oracle query on the triple (r_a, r_b, ch_a), in which case we can use $\mathcal{A}$, exactly in the same manner as we did in the proof of impersonator resistance, to attack the one-way security of the underlying encryption scheme.

5.1 Three-Round Secret Handshake Scheme

We can eliminate one communication round in the above protocol using the zero-knowledge signature of knowledge [CS97] of the trapdoor t that corresponds to the public key $PK = \mathsf{Recover}(G, ID, \omega)$, which we will denote $\mathsf{sig}_t(m)$. One can easily construct such signatures in ROM if this relation admits a 3-round honest-verifier special-soundness proof system [CS97]. The protocol proceeds as follows, using the same notation as above:

1. $(B \longrightarrow A)$: (ID_b, ω_b, ch_b)
   – A computes $PK_b = \mathsf{Recover}(G, ID_b, \omega_b)$ and $c = \mathsf{Enc}_{PK_b}(r_a, \mathsf{sig}_{t_a}(ch_b))$
2. $(A \longrightarrow B)$: $(ID_a, \omega_a, cha_a, c)$
   – B accepts if c decrypts to (r_a, sig) where sig verifies as a signature on ch_b under the public key $PK_a = \mathsf{Recover}(G, ID_a, \omega_a)$
3. $(B \longrightarrow A)$: $resp_b = H(r_a, ch_a)$
   – A accepts if $resp_b = H(r_a, ch_a)$

In the case of the CDH-based encryption of Section 4, the above signature of knowledge is simply a Schnorr signature, and the resulting computational cost is one or two exponentiation and one multiexponentiation per player.

References

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO'2000*, 2000.

[BDS+03] D. Balfanz, G. Durfee, N. Shankar, D.K. Smetters, J. Staddon, and H.C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, 2003.

[BF01] D. Boneh and M. Franklin. Identity based encryption from weil pairing. In *Advances in Cryptography - CRYPT0 2001*, Santa Barbara, CA, August 2001.

[BKLS02] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptography - CRYPT0 2002*, pages 354–368, Santa Barbara, CA, August 2002.

[BR93] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO'93*, 1993.

[CK02] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology - EUROCRYPT 2002*, 2002.

[CS97] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, ETH Zurich, 1997.

[CVH91] D. Chaum and E. Van Heyst. Group signatures. In Springer-Verlag, editor, *Advances in Cryptology - EUROCRYPT'91*, volume 547, pages 257–265, 1991.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology-CRYPTO'99*, pages 537–554, August 1999.

[Gag02] Martin Gagne. Applications of bilinear maps in cryptography. Master's thesis, University of Waterloo, 2002.

[HBSO03] J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *2nd ACM Workshop on Privacy in the Electronic Society*, October 2003.

[Jou02] A. Joux. The weil and tate pairings as building blocks for public key cryptosystems. In *Proceedings of the 5th International Symposium on Algorithmic Number Theory*, 2002.

[KP98] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptography - CRYPT0 1998*, Santa Barbara, CA, August 1998.

[LDB03] N. Li, W. Du, and D. Boneh. Oblivious signature-based enevelope. In *Proceedings of 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, Boston, Massachusetts, July 13-16 2003.

[PS96] D. Pointcheval and J. Stern. Security proofs for signatures. In *Eurocrypt'96*, pages 387 – 398, 1996.

[Sch89] C. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptography - CRYPT0 1989*, Santa Barbara, CA, August 1989.

[Sho99] V. Shoup. On formal models for secure key exchange. Technical Report RZ3120, IBM, April 1999.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium in Cryptography and Information Security*, Okinawa, Japan, January 2000.

A Achieving Additional Properties

A.1 Roles

Our schemes can easily be extended to handle group member roles (as in the SH scheme of [BDS$^+$03]), in a way that a member can choose not to reveal anything about herself unless the other party is a member with a particular role r (and vice versa). This functionality can be provided by modifiying the AddMember and Recover procedures as follows:

- AddMember: takes as inputs $params$, G, t_G and an arbritary string $ID \in \{0,1\}^*$ and returns (ω, t) where t is a trapdoor and ω is a public parameter. (ω, t) are constructed using the string $ID|r$ (instead of ID as in the original procedure), where r is the role that the CA is assigning to the user.
- Recover: takes as input $params$, G, ID and ω (provided by another user B). It outputs a public key PK using as input $ID|r$ (instead of ID as in the original Recover procedure), where r is the role that $\mathcal{A}$ chooses to have a secret hanshake with.

A.2 Trapdoor Secrecy

Since CA computes the user's trapdoor t, it can impersonate that user. Would that be problematic, AddMember can easily be modified to blind the trapdoor if in the AddMember protocol the user supplies the CA with $b = g^{\delta} \bmod p$, where δ is the user's temporary secret. The CA can then reply with $\omega = g^k * b \bmod p$, where k is a random value in $\mathbb{Z}_q$, and $t' = k + H(\omega, ID) * t_G \bmod q$, and the user computes his trapdoor as $t = t' + \delta \bmod q$.

k-Times Anonymous Authentication (Extended Abstract)

Isamu Teranishi, Jun Furukawa, and Kazue Sako

Internet Systems Research Laboratories, NEC Corporation,
1753 Shimonumabe, Nakahara-Ku, Kawasaki 211-8666, Japan
teranisi@ah.jp.nec.com, j-furukawa@ay.jp.nec.com, k-sako@ab.jp.nec.com

Abstract. We propose an authentication scheme in which users can be authenticated anonymously so long as times that they are authenticated is within an allowable number. The proposed scheme has two features that allow 1) no one, not even an authority, identify users who have been authenticated within the allowable number, and that allow 2) anyone to trace, without help from the authority, dishonest users who have been authenticated beyond the allowable number by using the records of these authentications. Although identity escrow/group signature schemes allow users to be anonymously authenticated, the authorities in these schemes have the unnecessary ability to trace any user. Moreover, since it is only the authority who is able to trace users, one needs to make cumbersome inquiries to the authority to see how many times a user has been authenticated. Our scheme can be applied to e-voting, e-cash, electronic coupons, and trial browsing of content. In these applications, our scheme, unlike the previous one, conceals users' participation from protocols and guarantees that they will remain anonymous to everyone.

1 Introduction

1.1 Background

Many applications, such as e-voting [19, 21, 27, 29, 32], e-cash [1, 9, 12, 16, 30], electronic coupons [25, 26, 28], and trial browsing of content, often need to allow users to anonymously use these to protect privacy. At the same time, these applications need to restrict the number of times users can use them. These applications have three common requirements. The first is that they should provide honest users as much privacy as possible. The second is that they should be able to trace dishonest users easily. The third is that they should be able to restrict the number of times users can use applications.

However, if an application provider authenticates each user by receiving the user's signature when the user accesses it, a problem arises in that the provider is able to know who is using the application.

By following the authentication procedure of an identity escrow/group signature scheme [2, 4, 6, 7, 15, 24, 22], instead of an ordinary authentication scheme, users can be authenticated by the application provider without revealing their

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 308–322, 2004.

ID to it. However, this method also does not fully satisfy all the requirements. First, an authority called the group manager can identify honest users. Second, providers needs to make cumbersome inquiries of the group manager to trace dishonest users. Third, there is no easy way for the provider to restrict the number of times users can use applications.

1.2 Properties of Proposed Scheme

We propose an authentication scheme called *k-times anonymous authentication* (k-TAA) that satisfies the three requirements mentioned in the previous section. An authority called the *group manager* first registers users in the proposed scheme. Each application provider(AP) then publishes the number of times a user is allowed to use their application. The registered users can be authenticated by various APs.

The proposed scheme satisfies the following properties:

1. No one, *not even the group manager*, is able to identify the authenticated user, if authenticated user is honest.
2. No one, *not even the group manager*, is able to decide whether two authentication procedures are performed by the same user or not, if the user(s) is/are honest.
3. Any user who was accurately detected as having accessed more than the allowed number of times can be correctly traced *using only the authentication log of the AP and public information.*
4. No colluders, *not even the group manager*, are able to be authenticated by an AP provider on behalf of an honest user.
5. Once a user has been registered by the group manager, the user does not need to access the group manager.
6. Each AP can independently determine the maximum number of times a registered user can anonymously access the AP.

We stress that the group manager of our scheme has less authority than one of an identity escrow/group signature scheme. He cannot trace honest users. His sole role is registering users.

The proposed scheme also has directly uses as a *k-times anonymous signature.*

We formalize security requirements of k-TAA, then prove that the proposed scheme is secure under strong RSA assumption and DDH assumption.

1.3 Comparison with Related Work

Using known schemes, one can construct a scheme that has similar properties to ours. However, these schemes have some problems.

Blind Signature Scheme. Using the blind signature scheme [13], one can construct a scheme that has similar properties to ours. In each authentication, a user receives the group manager's blind signature and sends this signature to an AP. The AP accepts the authentication if the signature sent is valid. However, the scheme does not work well when there are multiple APs and their allowed number of access times is more than one.

Electronic Cash That Can be Spent k-times. Using multi-show cash [9] (i.e., electronic cash that one can spend multiple times), we can construct another scheme that has similar properties to ours. The group manager plays the role of the bank. Before accessing an AP for the first time, a user asks the bank to give him digital cash that can be spent k times, where k is the number of the access times allowed by the AP. This cash plays the role of a ticket that allows users to access the AP, i.e., users send the digital cash to the AP when they are authenticated by the AP.

This scheme, however, has three drawbacks. First, the scheme is not efficient in the sense that users must access the group manager every time they access a new AP. Second, the group manager can learn which APs each user wants to be authenticated by. Third, one can determine whether two payment protocols have been performed by the same user or not by comparing the multi-show cash that was used in the protocols.

Electronic Coupon. By using electronic coupons [25] as tickets, instead of electronic cash, one can construct another scheme, which also has similar properties to ours. This scheme, however, has the same problems that identity escrow/group signature schemes have. That is, the group manager can trace honest users, and an AP needs to make cumbersome inquiries of the group manager to trace dishonest users. The scheme also has a problem in that one can sometimes determine whether two authentication procedures have been performed by the same user or not[1].

List Signature and Direct Anonymous Attestation. Independently proposed schemes [5] and [11] are similar to ours. However, these schemes are unmatched to our purpose. 1) These scheme cannot use two or more times signature. 2) A verifier of [5] cannot trace dishonest user without help of an authority. The scheme [11] has no way to trace dishonest user. 3) An authority of [5] can identify the authenticated user.

1.4 Applications

An example of an application of the k-TAA is trial browsing of content. Each provider wants to provide users with a service that allows them to browse content such as movies or music freely on trial. To protect user privacy, the providers allow users to use them anonymously. To prevent users from using the service too many times, the providers want to restrict the number of times that a user can access the service.

This privileged service is only provided to certain group members, say a member of the XXX community. The head of this community plays the role of the group manager, and registers users on behalf of providers in advance.

[1] Although the authors of [25] claim that no one can determine this, it does not. The reason is nearly same as that k-TAA scheme which an AP is able to know how many times users accesses to him. See 1) of 3.4.

The properties of the proposed scheme enables all honest users to browse content anonymously for an permitted number, but users who access beyond the allowed number of times are identified.

It can also be applied to voting, transferable cash, and coupons. In the one- or multiple-voting scheme constructed with the proposed scheme, a voter computes one- or k-times anonymous signatures on his ballot, and sends these anonymously to an election administrator. In this scheme, even authorities are unable to know whether a user has voted or not.

We can add transferability to the electronic cash scheme [9] with our scheme. To transfer cash to another entity, the owner of the cash computes a one-time anonymous signature on the electronic cash, and sends it with the signature to the receiver. Although a transferable electronic cash scheme has already been proposed in [16], our scheme has an advantage in that users does not need to access the bank each time they transfers cash to another entity.

One can construct an electronic coupon scheme by applying the k-times anonymous signature scheme directly. Our method has an advantage in that even an authority can not trace an honest user while anyone can trace a dishonest user.

2 Model

2.1 Entities

Three types of entities take part in the model, namely, the *group manager* (GM), *users*, and *application providers* (AP). The k-TAA scheme is comprised of the following five procedures: *setup*, *joining*, *bound announcement*, *authentication*, and *public tracing*.

In the setup, the GM generates a *group public key* / *group secret key* pair, and publishes the group public key. Joining is done between the GM and user who wants to join the group. After the procedure, the user obtains a *member public key* / *member secret key* pair. A user who has completed the joining procedure is called a *group member*.

In the bound announcement procedure, an AP announces the number of times each group member is allowed to access him. The AP v publishes his $\mathtt{ID}_v$, and the upper bound k_v.

An authentication procedure is performed between a user and an AP. The AP accepts the user if the user is a group member and has not accessed him more than the allowable times. The AP detects and rejects the user if he is not a group member, or if he is a group member but has accessed him more times than the announced bound allows. The AP records the data sent by the accepted or detected user in the *authentication log*.

Using only the public information and the authentication log, anyone can do public tracing. The procedure outputs some user ID i, "`GM`", or "`NO-ONE`", which respectively mean "the user i is authenticated by the AP more times than the announced bound", "the GM published the public information maliciously ", and "the public tracing procedure cannot find malicious entities". Note that

we allow AP to delete some data from the log. Even if a member has been authenticated over the number of times, the tracing outputs `NO-ONE` if the AP deletes data about the member's authentication.

2.2 Requirements

A secure k-TAA must satisfy the following requirements:

- (**Correctness):** An honest group member will be accepted in authentication with an honest AP.
- (**Total Anonymity):** No one is able to identify the authenticated member, or to decide whether two accepted authentication procedures are performed by the same group member, if the authenticated user(s) has followed the authentication procedure within the permitted number of times per AP. These are satisfied even if other group members, the GM, and all APs collude with each other.
- (**Detectability):** Public tracing using an honest AP's authentication log does not output "`NO-ONE`", if a colluding subset of group members has been authenticated beyond the total number of times each colluding group member is able to be authenticated by the AP.
- (**Exculpability for Users):** Public tracing does not output the ID of an honest user, even if other group members, the GM, and all APs collude with each other.
- (**Exculpability for the GM):** Public tracing does not output `GM` if the GM is honest. This is satisfied even if every group members and every APs collude with one another.

Note that these requirements implies the followings:

- (**Unforgeability):** Without the help of the GM or group members, no colluding group non-members can be authenticated as group members.
- (**Coalition Resistance):** A colluding subset of group members cannot generate a member public key/private key pair, which is not generated in the joining procedures.
- (**Traceability):** Any member who is detected of having accessed an AP predetermined bound can be traced from public information and the AP's authentication log.

As reasons the unforgeability and the coalition resistance properties are satisfied are almost the same as for the group signature case [7], we have not included an explanation. Traceability property is clearly satisfied.

3 Proposed Scheme

3.1 Notations and Terminologies

Let $\mathbb{N}$ and $\mathbb{Z}_n$ denote the ring of natural numbers and natural numbers from 0 to $n-1$, and $\mathrm{QR}(n)$ be the multiple group of quadratic residues of $\mathbb{Z}_n$. Let $\mathcal{H}_X$

denote a full domain hash function onto set X. Let $\texttt{PROOF}(x \text{ s.t. } R(x))$ denote the proof of knowledge of x that satisfies the relation $R(x)$. We call prime p a *safe prime* if $(p-1)/2$ is also a prime number. We call n a *rigid integer* if natural number n can be factorized into two safe primes of equal length. Let G be a group with known order q, on which DDH problem is hard to solve. For simplification, we assume the bit length of q is equal to a security parameter κ.

3.2 Key Ideas

The proposed scheme is a modification of a group signature scheme. The GM is disabled from tracing an honest member, and anyone can identify who accessed over a number of times. Say an AP wishes to set the bound at k. Every time a member wants to be authenticated by the AP, he computes k intrinsic basis B_1, ..., B_k of AP which is called a *tag base*, then picks a tag base B_i which he has not used before. As long as the member uses different tag bases, he will not be identified. However, if he used the same tag base, anyone can identify who used the tag base twice.

3.3 Summary of Proposed Scheme

Let G be a group on which DDH problems are hard to solve. In the setup, the GM publishes a rigid integer n, elements $a, a_0 \in_U \mathrm{QR}(n)$, and an element b of G.

In joining, the a user and GM compute a member public key/secret key pair $((A,e),x)$ such that an equation $a^x a_0 = A^e$ is satisfied, x and e are elements of some previously determined intervals, and e is prime, and add b^x and his ID to public list, which is called *identification list*.

A tag base is a pair $(t, \check{t})$ of elements of the group G. They must be a hash values of some data, to prevent to be known the discrete logarithm of each others. In each authentication, an AP sends random number ℓ to a member, then the member sends back a *tag* $(\tau, \check{\tau}) = (t^x, (b^\ell \check{t})^x)$ with a validity proof. If the member does not have computed two tags using the same tag base, no one is able to trace that user, since DDH problem on G is hard to solve. However, if the member computes another tag $(\tau', \check{\tau}') = (t^x, (b^{\ell'} \check{t})^x)$ using the same tag base, AP can search these from his authentication log since these satisfy $\tau = \tau'$, and one can compute $(\check{\tau}/\check{\tau}')^{1/(\ell-\ell')} = ((b^\ell \check{t})^x/(b^{\ell'} \check{t})^x)^{1/(\ell-\ell')} = b^x$. Since identification list preserves user ID which corresponds b^x, one can identify the member.

3.4 Concerns

To construct the scheme we propose, we need to consider the followings:

1) *If an AP is able to know, w, the member accesses to him, the total anonymity property is not satisfied.* Suppose the number of times, w_1, that member M_1 has accessed to an AP does not equals the number of times, w_2, that member M_2 has accessed to the same AP. If $w \neq w_1$ is satisfied, the AP can affirm that the member is not M_1.

2) *If one can know the discrete logarithm of two tag bases, one can identify members* using the equation $\beta = (\check{\tau}_1{}^z/\check{\tau}_2)^{1/(l_1 z - l_2)}$. Here, β is a part of the public key

of a member, $\check{\tau}_1$ and $\check{\tau}_2$ are second coordinate of tags computed by the member using tag bases t_1 and t_2 which satisfy $t_2 = {t_1}^z$. Similerly, *If* ${\tau_1}^{x_1} = {\tau_1}^{x_2}$ *are satisfied, a user who know* x_1 *can perform as a user who know* x_2

3) *An AP can add false data to the log.*

4) In joining, *a secret key* x *of a member must be selected randomly*, since "one more unforgeability" of member key pair is assured only if the condition is satisfied. (See Lemma 2).

5) In joining, *a user must add* b^x *to the identification list before he know* (A, e). If a user can know (A, e) before he adds b^x, he can stop the joining procedure, and get a member key pair $(x, (A, e))$ such that b^x is not written in the identification list. Therefore, he can be anonymously authenticated any number of times since b^x is needed to tracing procedure.

6) As a similar reason plain signature schemes needs CA, the proposed scheme needs some mechanizm to assure the correctness of the correspondence between each entity and his public key.

7)*If* G *is unknown order group, the number of exponentiations of public tracing is linear to the size of members.* In this case, since one cannot compute $(\check{\tau}/\check{\tau}')^{1/(\ell-\ell')}$, one must cumbersomely compute $\beta^{(\ell-\ell')}$ for each element β of the identification list and then check whether $\beta^{(\ell-\ell')} = \check{\tau}/\check{\tau}'$ is satisfied.

To avoid attacks of 1), ..., 5), we construct the proposed scheme which satisifes the following: 1) the validity proof conceals w, 2) tag bases are hash values of some data, 3) an authentication log contains validity proofs which members have computed, 4) x is randomized by the GM, and 5) b^x is added to the identification list before the GM computes $A = (a^x a_0)^{1/e} \bmod n$.

To avoid attacks of 6), we assume the GM's public key is distributed by some trust entity. Additionally, we assume some assumption about the identification list, to assuer the correspondence between each member and his public key. See 4.2 for more detailed discussion.

To avoid inefficient tracing descibed in 7), we set G as a known order group, especially $G \not\models \mathrm{QR}(n)$.

3.5 Description of Proposed Scheme

PARAMETERS

The security parameters of our scheme are ν, ε, μ, and κ. Let λ and γ be parameters which are determined by the security parameters. (See Section 5 for a detailed description). We set Λ, Γ as sets of integers that were in $(0, 2^\lambda)$ and $(2^\gamma, 2^\gamma + 2^\lambda)$ respectively. Let $\{G_\kappa\}_{\kappa\in\mathbb{N}}$ be a set of cyclic groups with a known order. Let G be G_κ.

The parameters ν, ε, μ, and κ respectively control the difficulty of solving flexible RSA problem (the problem is also called strong RSA problem) on $\mathbb{Z}_n$, the tightness of the statistical zero-knowledge property, the soundness of the scheme, and the difficulty of solving DDH problem on G. We set, for example, $\nu = 1024$, $\varepsilon = \mu = \kappa = 160$, and set G as an elliptic curve group.

SETUP

1. The GM randomly chooses 2ν-bit rigid integer n. Then, it randomly chooses μ-bit string $R_{\mathtt{GM}}$, and computes $((a', a_0'), b) = \mathcal{H}_{\mathbb{Z}_{n^2} \times G}(R_{\mathtt{GM}})$ and $(a, a_0) = (a'^2, a_0'^2) \bmod n \in \mathrm{QR}(n)^2$. The group secret key is (p_1, p_2) and the group public key is $(n, R_{\mathtt{GM}}, a, a_0, b)$.

JOINING

1. User U_i selects $x' \in_U \Lambda$, and sends its commitment C to the GM with a validity proof.
2. The GM verifies the proof, and sends $x'' \in_U \Lambda$ to U_i.
3. User U_i confirms that $x'' \in \Lambda$ is satisfied, computes $x = ((x' + x'') \bmod 2^\lambda)$ and $(\alpha, \beta) = (a^x \bmod n, b^x)$, and then adds new data (i, β) to the *identification list* $\mathtt{LIST}$. Then, U_i sends (α, β) to the GM with a validity proof.
4. The GM verifies (i, β) is an element of the identification list, and the proof is valid. Then, the GM generates a prime $e \in_U \Gamma$, computes $A = (\alpha a_0)^{1/e} \bmod n$, and sends (A, e) to user U_i.
5. User U_i confirms that equation $a^x a_0 = A^e \bmod n$ is satisfied, e is a prime, and e is an element of Γ. The new member U_i's secret key is x, and his public key is (α, A, e, β).

BOUND ANNOUNCEMENT

1. AP V publishes $(\mathtt{ID}_\mathrm{V}, k_\mathrm{V})$. Here, $\mathtt{ID}_\mathrm{V}$ is his ID.

Let $(t_1, \check{t}_1) = \mathcal{H}_{G^2}(\mathtt{ID}_\mathrm{V}, k_\mathrm{V}, 1), \ldots, (t_{k_\mathrm{V}}, \check{t}_{k_\mathrm{V}}) = \mathcal{H}_{G^2}(\mathtt{ID}_\mathrm{V}, k_\mathrm{V}, k_\mathrm{V})$. We call $(t_w, \check{t}_w)$ the w-th *tag base* of the AP.

AUTHENTICATION

1. Member M increases counter $C_{\mathtt{ID}_\mathrm{V}, k_\mathrm{V}}$. If value w of counter $C_{\mathtt{ID}_\mathrm{V}, k_\mathrm{V}}$ is greater than k_V, then M sends $\perp$ to V and stops.
2. AP V sends random integer $\ell \in_U [0, 2^{\mu+\varepsilon}] \cap \mathbb{N}$ to M.
3. Member M computes tag $(\tau, \check{\tau}) = (t_w^x, (b^\ell \check{t}_w)^x)$, using M's secret key x and the w-th tag base $(t_w, \check{t}_w)$, computes proof $(\tau, \check{\tau})$ is correctly computed, and sends $(\tau, \check{\tau})$ and the validity proof to V.
4. If the proof is valid and if τ is different from all search tags in his authentication log, V adds tuple $(\tau, \check{\tau}, \ell)$ and the proof to the authentication log $\mathtt{LOG}$ of V, and outputs $\mathtt{accept}$.

PUBLIC TRACING

1. From $\mathtt{LOG}$, one finds two data $(\tau, \check{\tau}, \ell, \mathtt{PROOF})$ and $(\tau', \check{\tau}', \ell', \mathtt{PROOF}')$ that satisfy $\tau = \tau'$ and $\ell \neq \ell'$, and that $\mathtt{PROOF}$ and $\mathtt{PROOF}'$ are valid. If one cannot find such data, then one outputs $\mathtt{NO\text{-}ONE}$.

2. One computes $\beta' = (\check{\tau}/\check{\tau}')^{1/(\ell-\ell')} = ((b^{\ell}\check{t})^x/(b^{\ell'}\check{t})^x)^{1/(\ell-\ell')} = b^x$, and searches pair (i, β) that satisfies $\beta = \beta'$ from the identification list. Then, one outputs a member's ID i. If there is no such (i, β), then one affirms that the GM has deleted some data from the identification list, and outputs `GM`.

3.6 Details

- setup.
  - The GM must additionally publish 1) $(g, h) \in_U \mathrm{QR}(n)^2$, which shall be used by users to compute commitment C in joining, 2) a zero-knowledge proof that n is a rigid integer, and 3) a zero-knowledge proof that (g, h) is an element of $\mathrm{QR}(n)$. The GM provides the proof 2) using the technique of [10], and provide the proof 3) by proving knowledge of $(\tilde{g}', \tilde{h}') \in {\mathbb{Z}_n}^2$, which satisfies $(\tilde{g}'^2, \tilde{h}'^2) = (g, h) \bmod n$.
- joining.
  - At step 1, the user must compute $((a', a_0'), b) = \mathcal{H}_{{\mathbb{Z}_n}^2 \times G}(R_{\mathtt{GM}})$, and verify equation $(a, a_0) = (a'^2, {a_0'}^2) \bmod n$, and the proofs.
  - Commitment C is $g^{x'} h^{s'} \bmod n$. Here s' is a $(2\nu + \varepsilon)$-bit random natural number.
  - The formal description of validity proofs of step 1 and 3 are, respectively, $\mathtt{PROOF}_1 = \mathtt{PROOF}((\tilde{x}', \tilde{s}')$ s.t. $\tilde{x}' \in \Lambda \wedge C = g^{\tilde{x}'} h^{\tilde{s}'} \bmod n)$ and $\mathtt{PROOF}_2 = \mathtt{PROOF}((\tilde{x}, \tilde{\theta}, \tilde{s}')$, which satisfies the (a), ..., (d) below.), where (a) $\tilde{x} \in \Lambda$, (b) $a^{\tilde{x}} = \alpha \bmod n$, (c) $Cg^{x''} = g^{\tilde{x}}(g^{2^\lambda})^{\tilde{\theta}} h^{\tilde{s}'} \bmod n$, and (d) $b^{\tilde{x}} = \beta$. These proofs must be statistically zero knowledge on security parameter ε. We have omitted a detailed description of proofs. See [8] for the proof that committed number lies in the interval.
- authentication.
  - At step 4, if the proof is invalid, V outputs `reject` and stops. If τ is already written in the identification list, V adds tuple $(\tau, \check{\tau}, \ell)$ and the proof to the `LOG` of the AP, outputs $(\mathtt{detect}, \mathtt{LOG})$ and stops.
  - The proof of step 3 is rather more complex. Its details are described in the full version of this paper.

3.7 Efficiency

The proposed scheme satisfies the followings:

- **(Compactness).** The GM is able to add new members to the group without modifying any keys which was previously generated. In particular, the size of the member's key pair does not depend on the group size.
- Once a user has been registered by the GM, the user does not need to access the GM.
- Each AP is able to solely determine the bound of himself.
- The computational cost of authentication is $\mathcal{O}(k_{\mathrm{V}})$. However, if G is taken as an elliptic curve group, the factor which depend on k_{V} is small, since the exponentiation on G is faster than that on $\mathbb{Z}_n$.
- The number of exponentiations of public tracing is independent of the size of an authentication log and the identification list.

3.8 Variants of Proposed Scheme

1) Although the proposed scheme merely restricts the number of authentications, one can construct, using the "and/or"-proof technique, a scheme such as "a trace procedure identifies a user if and only if the user is authenticated either 1) k_1 times from AP V_1 or 2) k_2 times from AP V_2 and k_3 times from AP V_3".
2) By changing a data in LIST from (i, β) to $(\mathcal{H}(\beta), \mathcal{E}_\beta(i))$, one can construct a k-TAA scheme in which no one, except a member himself and the GM, can detect who is a member of the group. Here, $\mathcal{H}$ is a hash function and $\mathcal{E}$ is a symmetric encryption scheme. To trace dishonest user, one computes β as in the proposed scheme, and then computes $\mathcal{H}(\beta)$, searches (h, e) from LIST that satisfies $h = \mathcal{H}(\beta)$, and decrypts e.

4 Formal Security Requirements

4.1 Notations

We describe the five procedures for a k-TAA scheme as SETUP, JOIN = ($\mathcal{U}_{\texttt{JOIN-GM}}$, $\mathcal{U}_{\texttt{JOIN-U}}$), BOUND-ANNOUNCEMENT, (abbrev. BD-ANN), AUTH = ($\mathcal{U}_{\texttt{AUTH-U}}$,$\mathcal{U}_{\texttt{AUTH-AP}}$), and TRACE. The procedures $\mathcal{U}_{\texttt{JOIN-GM}}$ and $\mathcal{U}_{\texttt{JOIN-U}}$ (resp. $\mathcal{U}_{\texttt{AUTH-AP}}$ and $\mathcal{U}_{\texttt{AUTH-U}}$) are what the GM and user (resp. AP and user) follow in joining (resp. authentication). Let $(\texttt{gpk}, \texttt{gsk})$ and $(\texttt{mpk}, \texttt{msk})$ denote the public key/secret key pair of group manager and member respectively.

4.2 List Oracle Model

We must assume the existence of an infrastructure which enables to assure the correct correspondence between each member and his public key to formalize the security requirements. If we do not assume such thing, no scheme satisfies the exculpability properties for users as in a group signature case[7]. One of a such infrastructure is a PKI, but a formalization on the PKI model is rather complicated, since it must include description of the signing oracle, what an adversary can do in a PKI key setup, etc. To simplify, we introduce new model *list oracle model*. In the model, it is assumed the existence of a *list oracle* $\mathcal{O}_{\texttt{LIST}}$, which manages the identification list[2] LIST. The oracle $\mathcal{O}_{\texttt{LIST}}$ allows anyone to view any data of LIST. However, it allows entities to write data $(i, \texttt{mpk}_i)$ to LIST only if the entity is user i or i's colluder and to delete data of LIST only if the entity is the GM or GM's colluder. We need to stress that even the GM cannot write data $(i, \texttt{mpk}_i)$ without colluding with the user i, and even user i cannot delete data $(i, \texttt{mpk}_i)$ without colluding with the GM. A more formal definition is described in Figure 1, where X is a set of entities which collude with an entity who accesses to $\mathcal{O}_{\texttt{LIST}}$

Note that *a scheme on the list oracle model can be easily transformed into a scheme on the PKI model*, by changing $(i, \texttt{mpk}_i)$ to $(\texttt{mpk}_i, \sigma_i(\texttt{mpk}_i))$ and LIST

[2] Although the LIST of the proposed scheme stores a parts of public keys, β, we deal with the case LIST stores the whole public key, to simplify.

to $(\mathtt{LIST}, \sigma_{\mathtt{GM}}(\mathtt{LIST}))$. Here, $\sigma_i(\cdot)$ is a signature of an entity i. The authority of the GM in the list oracle model to delete data from LIST corresponds to the authority of the GM in the PKI model to publish $(\mathtt{LIST}', \sigma_{\mathtt{GM}}(\mathtt{LIST}'))$ in spite of $(\mathtt{LIST}, \sigma_{\mathtt{GM}}(\mathtt{LIST}))$. Here $\mathtt{LIST}' = \mathtt{LIST} \setminus \{(\mathtt{mpk}_i, \sigma_i(\mathtt{mpk}_i))\}$.

4.3 Experiments

An adversary is allowed the following in experiments on security properties:

- If an adversary colludes with the GM, the adversary can maliciously execute SETUP and $\mathcal{U}_{\mathtt{JOIN\text{-}GM}}(\mathtt{gpk}, \mathtt{gsk})$.
- If an adversary colludes with a user i, the adversary can maliciously execute $\mathcal{U}_{\mathtt{JOIN\text{-}U}}(\mathtt{gpk}, i)$ and $\mathcal{U}_{\mathtt{AUTH\text{-}U}}(\mathtt{gpk}, \mathtt{msk})$ where msk is a secret key of i.
- If an adversary colludes with an AP, the adversary can choose the public information $(\mathtt{ID}, k)$ of the AP and maliciously execute $\mathcal{U}_{\mathtt{AUTH\text{-}AP}}(\mathtt{gpk}, (\mathtt{ID}, k))$. Moreover, the adversary can use different AP information $(\mathtt{ID}, k)$ for each authentication.
- An adversary is only allowed to execute many joining and authentication procedures sequentially.

Total Anonymity. An adversary is allowed to collude with the GM, all APs, and all users except target users i_1 and i_2. It is also allowed to authenticate the oracle $\mathcal{O}_{\mathtt{QUERY}}(b, \mathtt{gpk}, (i_1, i_2), (\mathtt{ID}, k), (d, \cdot))$ once only for $d = 0, 1$. If it sends (d, M) to $\mathcal{O}_{\mathtt{QUERY}}$, oracle $\mathcal{O}_{\mathtt{QUERY}}$ regards M as data sent by a member and executes $\mathcal{U}_{\mathtt{AUTH\text{-}U}}$ using the key pair of user $i_{b \oplus d+1}$ and the APs public information $(\mathtt{ID}, k)$. Recall that k-TAA schemes provide anonymity only if a member has been authenticated less than the allowed number of times. Therefore, the adversary must authenticate user i_1 or i_2 using $(\mathtt{ID}, k)$ within k times. If the adversary keeps to the rule and outputs b, the adversary wins. See Figure 1 for the formal definition of $\mathcal{O}_{\mathtt{QUERY}}$. Here, $S_{\mathtt{QUERY}}$ is a set, using which $\mathcal{O}_{\mathtt{QUERY}}$ memorize the session IDs.

Contrary to [7, 22], the secret key of the target users is not input to an adversary. If the secret keys is input to an adversary, the adversary is able to determine b as follows: it colludes with AP publishing $(\mathtt{ID}, k)$, authenticated k times from the AP using i_1's secret key, and obtains the log LOG for the authentications. Then, it communicates with $\mathcal{O}_{\mathtt{QUERY}}(b, \mathtt{gpk}, (i_1, i_2), (\mathtt{ID}, k), (0, \cdot))$ and obtains the log L of the authentications. Secret b equals to 0 if and only if $\mathtt{TRACE}^{\mathcal{O}_{\mathtt{LIST}}(\emptyset, \cdot)}(\mathtt{gpk}, \mathtt{LOG} \cup \{L\}) = i_1$ is satisfied.

Detectability. An adversary is allowed to collude with all of group members. If the adversary succeeds in being accepted by some AP in more than kn authentications, the adversary wins. Here, k is the number of times the AP allows access for each user, and n is the number of users who collude with the adversary.

Exculpability (for users, and for GM). An adversary is allowed to collude with all entities except the target entity. If the adversary succeeds in computing the log with which the public tracing procedure outputs the ID of the target entity, the adversary wins.

$***\mathbf{Exp}^{\text{anon-}((i_1,i_2),(\mathtt{ID},k),b)}_{\mathcal{A},\mathcal{H}}(\omega)***$
$(\mathtt{gpk}, St) \leftarrow \mathcal{A}(1^\omega)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{LIST}}(\{i_1,i_2\}^c,\cdot),\mathcal{O}_{\mathtt{JOIN-U}}(\mathtt{gpk},\cdot),\mathcal{O}_{\mathtt{AUTH-U}}(\mathtt{gpk},\cdot),\mathcal{O}_{\mathtt{QUERY}}(b,\mathtt{gpk},(i_1,i_2),(\mathtt{ID},k),(\cdot,\cdot))}(St)$
If ($\mathcal{O}_{\mathtt{QUERY}}$ has output `OVER`) Return $\perp$.
Return b'.

$***\mathcal{O}_{\mathtt{LIST}}(X, M)$ ***
Parse M as $(\mathtt{command}, i, \mathtt{mpk})$.
If($\mathtt{command} = \mathtt{view}$ and $\mathtt{mpk} =$"-")
  If($^\exists\mathtt{mpk}'$ s.t. $(i, \mathtt{mpk}') \in \mathtt{LIST}$)
    Return $\mathtt{mpk}'$.
If($\mathtt{command} = \mathtt{add}$)
  If($i \in X$ and $^\nexists\mathtt{mpk}'$ s.t. $(i, \mathtt{mpk}) \in \mathtt{LIST}$)
    $\mathtt{LIST} \leftarrow \mathtt{LIST} \cup \{(i, \mathtt{mpk})\}$.
Else if($\mathtt{command} = \mathtt{delete}$)
  If ($\mathtt{GM} \in X$) $\mathtt{LIST} \leftarrow \mathtt{LIST} \setminus \{(i, \mathtt{mpk})\}$.
Return $\perp$.

$***\mathcal{O}_{\mathtt{QUERY}}(b, \mathtt{gpk}, (i_1, i_2), (\mathtt{ID}, k)(d, M))***$
If($d \notin \{0, 1\}$) Return $\perp$.
If($^\nexists\mathtt{sid}$ s.t. $(d, \mathtt{sid}) \in S_{\mathtt{QUERY}}$)
  Choose new session ID $\mathtt{sid}$
    which has been ever used.
  $S_{\mathtt{QUERY}} \leftarrow S_{\mathtt{QUERY}} \cup \{(d, \mathtt{sid})\}$.
Return
  $\mathcal{O}_{\mathtt{AUTH-U}}(\mathtt{gpk}, (\mathtt{sid}, i_{1+(b\oplus d)}, (\mathtt{ID}, k)||M))$.

$***\mathbf{Exp}^{\text{decis}}_{\mathcal{A},\mathcal{H}}(\omega)***$
$(\mathtt{gpk}, \mathtt{gsk}) \leftarrow \mathtt{SETUP}(1^\omega)$
$\mathcal{A}^{\mathcal{O}_{\mathtt{LIST}}(\{\mathtt{GM}\}^c,\cdot),\mathcal{O}_{\mathtt{JOIN-GM}}(\mathtt{gpk},\mathtt{gsk},\cdot),\mathcal{O}_{\mathtt{AUTH-AP}}(\mathtt{gpk},\cdot,\cdot)}(1^\omega)$.
If ($^\exists(\mathtt{ID}, k) \in S_{\mathtt{AUTH-AP}}$ s.t. $\#\mathtt{LOG}_{\mathtt{ID},k} > k \cdot \#\mathtt{LIST}$)
  Return $\mathtt{TRACE}^{\mathcal{O}_{\mathtt{LIST}}(\emptyset,\cdot)}(\mathtt{gpk}, \mathtt{LOG}_{\mathtt{ID},k})$.
Return $\perp$.

$***\mathbf{Exp}^{\text{excul-}i_1}_{\mathcal{A},\mathcal{H}}(\omega)***$
$(\mathtt{gpk}, St) \leftarrow \mathcal{A}(1^\omega)$.
$\mathtt{LOG} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{LIST}}(\{i_1\}^c,\cdot),\mathcal{O}_{\mathtt{JOIN-U}}(\mathtt{gpk},\cdot),\mathcal{O}_{\mathtt{AUTH-U}}(\mathtt{gpk},\cdot)}(St)$.
Return $\mathtt{TRACE}^{\mathcal{O}_{\mathtt{LIST}}(\emptyset,\cdot)}(\mathtt{gpk}, \mathtt{LOG})$.

$***\mathbf{Exp}^{\text{excul-GM}}_{\mathcal{A},\mathcal{H}}(\omega)***$
$(\mathtt{gpk}, \mathtt{gsk}) \leftarrow \mathtt{SETUP}(1^\omega)$
$\mathtt{LOG} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathtt{LIST}}(\{\mathtt{GM}\}^c,\cdot),\mathcal{O}_{\mathtt{JOIN-GM}}(\mathtt{gpk},\mathtt{gsk},\cdot)}(\omega)$.
Return $\mathtt{TRACE}^{\mathcal{O}_{\mathtt{LIST}}(\emptyset,\cdot)}(\mathtt{gpk}, \mathtt{LOG})$.

Comments:
1. To simplify, we abbreviate
the hash oracle $\mathcal{O}_{\mathcal{H}}$.
2. $\mathcal{O}_{\mathtt{AUTH-U}}(\mathtt{gpk}, (\cdot, i, \cdot))$ outputs `OVER`
if $\mathcal{A}$ authenticate user i
more than allowed number of times.

Fig. 1. The oracles and the experiments

Figure 1 denotes the experiments, formally. $\mathcal{O}_{\mathtt{JOIN-GM}}$, $\mathcal{O}_{\mathtt{JOIN-U}}$, $\mathcal{O}_{\mathtt{AUTH-U}}$, and $\mathcal{O}_{\mathtt{AUTH-AP}}$ are the oracles that manage and execute multiple sessions of $\mathcal{U}_{\mathtt{JOIN-GM}}$, $\mathcal{U}_{\mathtt{JOIN-U}}$, $\mathcal{U}_{\mathtt{AUTH-U}}$, and $\mathcal{U}_{\mathtt{AUTH-AP}}$ respectively, and ω is the security parameter. The set $S_{\mathtt{AUTH-AP}}$ contains all AP's information that was used by $\mathcal{O}_{\mathtt{AUTH-AP}}$, and $\mathtt{LOG}_{\mathtt{ID},k}$ is the log of authentications engaged by $\mathcal{O}_{\mathtt{AUTH-AP}}$ using AP's information $(\mathtt{ID}, k)$. See the full version of this paper for a formal definition of the oracles.

4.4 Definition

Definition 1. *Let ω be a security parameter, $\mathcal{A}$ be an adversary, b be an element of $\{0, 1\}$, i_1 and i_2 be natural numbers, and $(\mathtt{ID}, k)$ is a some AP's public information.*

If $\mathbf{Adv}^{\text{anon-}((i_1,i_2),(\mathtt{ID},k))}_{\mathcal{A}}(\omega) = |\Pr(\mathbf{Exp}^{\text{anon-}(0,(i_1,i_2),(\mathtt{ID},k))}_{\mathcal{A},\mathcal{H}}(\omega) = 1) - \Pr(\mathbf{Exp}^{\text{anon-}(1,(i_1,i_2),(\mathtt{ID},k))}_{\mathcal{A},\mathcal{H}}(\omega) = 1)|$ *is negligible for security parameter ω for all $(\mathcal{A}, i_1, i_2, (\mathtt{ID}, k))$, we say a k-TAA scheme satisfies* total anonymity.

If $\mathbf{Adv}^{\text{decis}}_{\mathcal{A}}(\omega) = \Pr(\mathbf{Exp}^{\text{decis}}_{\mathcal{A},\mathcal{H}}(\omega) = \mathtt{NO\text{-}ONE})$ *is negligible for security parameter ω for all $\mathcal{A}$, we say a k-TAA scheme satisfies* detectability.

If $\mathbf{Adv}_{\mathcal{A}}^{excul\text{-}i_1}(\omega) = \Pr(\mathbf{Exp}_{\mathcal{A},\mathcal{H}}^{excul-i_1}(\omega) = i_1)$ *is negligible for security parameter* ω *for all* $(\mathcal{A}, i_1)$*, we say a k-TAA scheme satisfies* exculpability for users.

If $\mathbf{Adv}_{\mathcal{A}}^{excul\text{-}\mathtt{GM}}(\omega) = \Pr(\mathbf{Exp}_{\mathcal{A},\mathcal{H}}^{excul-\mathtt{GM}}(\omega) = \mathtt{GM}))$ *is negligible for security parameter* ω *for all* $\mathcal{A}$*, we say a k-TAA scheme satisfies* exculpability for users.

5 Security of Proposed Scheme

To prove the security of the proposed scheme, we use two key lemmata. First, since each member generates element a^x of $\mathrm{QR}(n)$ and element b^x of G using the same x, we must be particularly concerned about secrecy. We will prove the difficulty of a variant in the DDH problem, where two components of a DH-tuple are elements of $\mathrm{QR}(n)$ and the other two components are elements of G:

Lemma 1. (Separation Lemma) *Let* a *be an element of* $\mathrm{QR}(n)$*, and* b *be an element of* G*. Then, the following two distributions are statistically indistinguishable: 1) the distribution of* $(a^x \bmod n, b^x) \in \mathrm{QR}(n) \times G$*, where* x *is randomly chosen from* Λ*, and 2) the distribution of* $(\alpha, \beta) \in \mathrm{QR}(n) \times G$*, where* α *and* β *are randomly chosen from* $\mathrm{QR}(n)$ *and* G *respectively.*

Since $|\Lambda|$ is ε times greater than $|\mathrm{QR}(n) \times G|$, the variation distance between the two distributions is less than $1/2^{\varepsilon}$, and therefore, Lemma 1 holds. Note that, if we injudiciously choose a narrow Λ, the security of the proposed scheme will rely on a non-standard assumption that those two distributions will still be computationally indistinguishable.

Detectability and GM's exculpability of the proposed scheme depends on "one more unforgeability" of a $((A, e), x)$:

Lemma 2. *If a) a member's secret key is randomly generated in each joining procedure, and if b) for all* $x \in \Lambda$ *and* $e \in \Gamma$*,* $x < e$ *is satisfied, then no adversary can generate a* (x, A, e) *which satisfies* $a^x a_0 = A^e$*,* $x \in \Lambda$*, and* $e \in \Gamma$*, and which has not been made in the joinings.*

The proof for Lemma 2 is almost same as the proof for Theorem 1 of [2].

The proposed scheme satisfies the conditions for Lemma 2. Conditions a) and b), respectively, follow the method of choosing x in the joining procedure, and the choice of (λ, γ).

Using these lemmata, we can prove the security of the proposed scheme. See the full version of this paper for the detailed proof.

Theorem 1. *Let* λ *be* $2\nu+\kappa+\varepsilon$*, and* γ *be* $\lambda+\mu+\varepsilon+8$*. Then, the proposed scheme on list oracle model satisfies the security requirements of Definition 1 under the strong RSA assumption, the DDH assumption on* $\{G_\kappa\}$*, and the random oracle assumption.*

References

1. How to Date Blind Signatures, M. Abe, and E. Fujisaki, In *ASIACRYPT 1996*, *LNCS* 1163, pp. 244-251, Springer-Verlag, 1996.
2. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO 2000*, *LNCS* 1880, pp. 255-270, Springer-Verlag, 2000.
3. Giuseppe Ateniese and Breno de Medeiros. Efficient Group Signatures without Trapdoors. In *ASIACRYPT 2003*, *LNCS* 2094, pp. 246-268, Springer-Verlag, 2003.
4. G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. In *Financial Cryptography '99*, *LNCS* 1648,pp. 196-211, Springer-Verlag, 1999.
5. Direct Anonymous Attestation. Ernie Brickell, Jan Camenisch, and Liqun Chen. ZISC Information Security Colloquium SS 2004, June 2004 http://www.hpl.hp.com/techreports/2004/HPL-2004-93.pdf
6. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi, Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions In *EUROCRYPT 2003*, *LNCS* 2656, pp. 614-629, Springer-Verlag, 2003.
7. Mihir Bellare, Haixia Shi, and Chong Zhang, Foundations of Group Signatures: The Case of Dynamic Groups. http://eprint.iacr.org/2004/077.ps
8. Fabrice Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In *EUROCRYPT 2000*, *LNCS* 1807, pp. 255-270, Springer-Verlag, 2000.
9. Stefan Brands. An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical Report CS-R9323, Centrum voor Wiskunde en Informatica,
10. Jan Camenisch and Markus Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In *EUROCRYPT'99*, *LNCS* 1592, pp. 107-122, Springer-Verlag, 1999.
11. Sébastien Canard and Jacques Traoré List Signature Schemes and Application to Electronic Voting. In *International Workshop on Coding and Cryptography* 2003. pp.24-28, March 2003.
12. A. Chan, Y. Frankel, and Y.Tsiounis, Easy Come - Easy Go Divisible Cash. *EUROCRYPT '98*, *LNCS* 1403, pp. 614-629, Springer-Verlag, 1998.
13. D. Chaum. Blind signature system, In *CRYPTO'83*, pp. 153-153, Plenum Press, 1984
14. D. Chaum. *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM, vol. 24, No. 2, pp. 84-88, (1981).
15. D. Chaum and E. van Heijst. Group signatures. In *EUROCRYPT '91*, vol. *LNCS* 547, pp. 257-265, Springer-Verlag, 1991.
16. D. Chaum, T. Pedersen, Transferred Cash Grows in Size, In *EUROCRYPT'92*, *LNCS* 658, pp. 390-407, Springer-Verlag, 1993.
17. R.Cramer, I.Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO'94*, *LNCS* 2139, pp. 174-187, Springer-Verlag, 1994.
18. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, Victor Shoup. Anonymous Identification in Ad Hoc Groups. In *EUROCRYPT 2004*, *LNCS* 3027, pp. 609-626, Springer-Verlag, 2004.
19. I. Damgård and M. Jurik. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-key system. In *Proceedings. of Public Key Cryptography 2001. LNCS* 1992, pp. 119-136, Springer-Verlag, 2001.

20. A. Fiat and A. Shamir. How to prove yourself: practical solution to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pp. 186-194, Springer-Verlag, 1987.
21. Jun Furukawa, and Kazue Sako. An Efficient Scheme for Proving a Shuffle. In *CRYPTO 2001*, *LNCS* 2139, pp. 368-387, Springer-Verlag, 2001.
22. Aggelos Kiayias, and Moti Yung. Group Signatures: Provable Secure, Efficient Constructions and Anonymity from Trapdoor Holders. http://eprint.iacr.org/2004/076.ps
23. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable Signatures. In *EUROCRYPT 2004*, *LNCS* 3027, pp. 571-589, Springer-Verlag, 2004.
24. Joe Kilian, and Erez Petrank. Identity Escrow. In *CRYPTO'98*, *LNCS* 1462, pp. 169-185, Springer-Verlag, 1998.
25. Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable Electronic Coupon Protocol with Anonymity Control, In *ISW'99*, *LNCS* 1729, pp. 37-46, Springer-Verlag, 1999.
26. Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Electronic Coupon Ticket Protocol with Unlinkable Transcripts of Payments. In *Proceedings of the 1999 Symposium on Cryptography and Information Security*, pp. 359-363, 1999. (Japanese).
27. C.A. Neff, A Verifiable Secret Shuffle and its Application to E-Voting, ACMCCS 01 pp. 116-125 2001.
28. Tatsuaki Okamoto and Kazuo Ohta. One-Time Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol E81-A, No. 1, pp. 2-10, 1998.
29. M. Ookubo, F. Miura, M. Abe A. Fujioka, and T. Okamoto. An improvement of a practical secret voting scheme. In *ISW'99*, *LNCS* 1729, pp. 37-46, Springer-Verlag, 1999.
30. Chris Pavlovski, Colin Boyd, and Ernest Foo. Detachable Electronic Coins. In *Information and Communication Security, Second International Conference, ICICS'99*, *LNCS* 1726, pp. 54-70, Springer-Verlag, 1999.
31. K. Sako. Restricted Anonymous Participation. In *Proceedings of the 2000 Symposium on Cryptography and Information Security*, B12, January 2000. (Japanese).
32. K. Sako and J. Kilian. Secure Voting using Partially Compatible Homomorphisms. In *CRYPTO '94*, *LNCS* 839, pp. 411–424, Springer-Verlag, 1994.
33. Isamu Teranishi and J. Furukawa. Tag Signature. (Preliminary version of this paper). In *Proceedings of the 2003 Symposium on Cryptography and Information Security*, 6C-2, January 2003.

The XL-Algorithm and a Conjecture from Commutative Algebra

Claus Diem

Institute for Experimental Mathematics, University of Duisburg-Essen, Essen, Germany

Abstract. The "XL-algorithm" is a computational method to solve overdetermined systems of polynomial equations which is based on a generalization of the well-known method of linearization; it was introduced to cryptology at Eurocrypt 2000.

In this paper, we prove upper bounds on the dimensions of the spaces of equations in the XL-algorithm. These upper bounds provide strong evidence that for any fixed finite field K and any fixed $c \in \mathbb{N}$ the median of the running times of the original XL-algorithm applied to systems of $m = n+c$ quadratic equations in n variables over K which have a solution in K is not subexponential in n. In contrast to this, in the introduction of the original paper on XL, the authors claimed to "provide strong theoretical and practical evidence that the expected running time of this technique is [...] subexponential if m exceeds n by a small number".

More precise upper bounds on the dimensions of the spaces of equations in the XL-algorithm can be obtained if one assumes a standard conjecture from commutative algebra. We state the conjecture and discuss implications on the XL-algorithm.

Keywords: Cryptanalysis, algebraic attacks, overdetermined systems of polynomial equations, extended linearization, Fröberg's Conjecture.

1 Motivation and Introduction

The security of many cryptographic systems would be jeopardized if one could solve certain types of systems of polynomial equations over finite fields. For example, it has been pointed out in [8] that one can with a high probability recover an AES-128 key from one AES-128 plaintext-ciphertext pair if one can solve certain systems with 1600 variables and 8000 quadratic equations over $\mathbb{F}_2$, and it has been pointed out in [14] that one can achieve the same goal if one can solve certain systems with 3986 variables and 3840 (sparse) quadratic equations as well as 1408 linear equations over $\mathbb{F}_{2^8}$.

Of particular importance for cryptological applications are so-called *overdetermined* (or *overdefined*) systems of quadratic equations as for example the ones we just mentioned. Let us consider a system of quadratic polynomial equations

$$f_1(X_1, \ldots, X_n) = 0\,,\ \ldots\ ,\ f_m(X_1, \ldots, X_n) = 0\ , \tag{1}$$

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 323–337, 2004.

where the f_j are polynomials in n indeterminates $X_1, \ldots, X_n$ over an "effective" field K (the field being finite in the cryptological applications). We say that the system is *overdetermined* if the dimension of the K-vector space generated by the f_j is greater than n.

In [7], Courtois, Klimov, Patarin and Shamir propose a computational method called *eXtended Linearization (XL)* or *XL-algorithm* to solve such systems of polynomial equations (see the next section for a description of the method). In the same paper certain heuristics on the running time of this method are stated. These heuristics have subsequently been criticized by Moh ([13]) as being too optimistic, and in Sect. 4 of [13], the method is analyzed with heuristic upper bounds on the dimensions on the spaces of equations in the XL-algorithm. As however the assumptions on which the heuristic in [13–Sect. 4] relies are not very precisely stated, the question whether this heuristic or the original heuristic is more credible remained an open problem among cryptologists. In a recent work by Chen and Yang ([4]), the heuristic of [13–Sect. 4] is stated as a special case of a theorem ([4–Theorem 2]).[1] However, the proof of [4–Theorem 2] (and of [4–Theorem 7]) has some serious flaws.

The main purpose of this paper is to show that under the assumption of a widely believed conjecture of commutative algebra, one can indeed derive the non-trivial upper bounds on the dimensions of the spaces of equations in the XL-algorithm conjectured by Moh and stated in [4–Corollary 6, 1] (see Theorem 1 in Sect. 5 for a more general statement). Moreover, we state upper bounds on the dimensions of the spaces of equations in the original XL-algorithm which can be proven without the assumption of this conjecture. These upper bounds provide strong evidence that for any fixed finite field K and any fixed $c \in \mathbb{N}$ the median of the running times of the original XL-algorithm applied to systems of $m = n + c$ quadratic equations in n variables over K which have a solution in K is not subexponential in n (see the next section for details).

2 The XL-Algorithm and Our Analysis

Let us fix the system of quadratic equations (1) which we assume to have a solution in K and some $D \in \mathbb{N}$. The main idea of the XL-algorithm is to try to solve (1) by linearization of the system of all polynomial equations

$$\prod_{\ell=1}^{k} X_{i_\ell} \cdot f_j(X_1, \ldots, X_n) = 0 \ , \tag{2}$$

where $k \leq D - 2$.

Let U_D be K-vector space generated by the polynomials $\prod_{\ell=1}^{k} X_{i_\ell} \cdot f_j$ with $k \leq D - 2$.

According to [7–Definition 1], the XL-algorithm is as follows. (Except for changes in the notation, the description is verbatim.)

[1] Theorem 2 of [4] is equivalent to the heuristics of [13] if $D < q$ as can be seen by expanding the polynomial in item 1 of Corollary 6 in [4].

The XL-Algorithm. *Execute the following steps:*

1. Multiply: *Generate all the products* $\prod_{\ell=1}^{k} X_{i_\ell} \cdot f_j \in U_D$ *with* $k \leq D-2$.
2. Linearize: *Consider each monomial in* X_i *of degree* $\leq D$ *as an independent variable*[2] *and perform Gaussian elimination on the equation obtained in* 1. *The ordering on the monomials must be such that all the terms containing one [specific] variable (say* X_1*) are eliminated last.*
3. Solve: *Assume that step 2 yields at least one univariate equation in the powers of* X_1*. Solve this equation over the finite fields (e.g. with Berlekamp's algorithm).*[3,4]
4. Repeat: *Simplify the equations and repeat the process to find the values of the other variables.*

Remark 1. In the description of the method in [7], it seems that D is fixed beforehand. As the authors do however not say how D should be determined, it seems to be reasonable to assume that the authors of [7] had in mind that D is in fact a variable which is small (e.g. 2) in the beginning, and that the the XL-algorithm goes to Step 1 with an incremented D whenever in Step 3 no univariate equation in X_1 with a solution in K is found.

Remark 2. In an "extended version" ([6]) of [7], the description of the method is the same as the one in [7] (and the one we present here) except that the authors have inserted the sentence "In all the following notations we suppose the powers of variables taken over K, i.e. reduced modulo q to the range $1, \ldots, q-1$ because of the equation $a^q = a$ of the finite field K." after the third paragraph of Sect. 3. (But the field is not assumed to be finite in the second paragraph of Sect. 3 and the number q is not mentioned before.) Apart from this insertion, there is no substantial difference between Sect. 3 to 7 of [7] and of [6]. Of course, if one identifies the monomials $\prod_{\ell=1}^{k} X_{i_\ell}$ and $X^q \cdot \prod_{\ell=1}^{k} X_{i_\ell}$ the method becomes much faster if q, the field size, is small, $q = 2, 3, 4, 5$ say. According to the way the heuristics in Sect. 6 of [7] and [6] were conducted, this identification was however *not* made in the heuristic analysis of [7] and [6].

Definition 1. *We call the above computational method the* original XL-algorithm. *The variant introduced in [6] is called* reduced XL-algorithm.

Remark 3. Whereas the original XL-algorithm should only be applied to overdetermined systems of (quadratic) polynomial equations, if the field is finite and not too large, it makes sense to apply the reduced XL-algorithm to any system of (quadratic) polynomial equations.

---

[2] The authors of [7] obviously mean that each monomial of degree $\leq D$ should be considered as a new variable.

[3] Note however that according to the second paragraph in [7–Sect. 3], the ground field is not necessarily finite.

[4] It should be avoided to repeatedly select univariate polynomial equations which have more than one solution in K. Moreover, if a univariate polynomial equation is found which does not have a solution in K, the method should terminate and output "unsolvable".

Remark 4. Neither of the two computational methods terminates for every input (even if they are only applied to overdetermined systems which have a solution in K); thus in contrast to their names, they are not algorithms (not even randomized algorithms) in the usual sense (cf. [11–Chapter 1, 1.1]).

As we will see in the next section, there is a strong connection between the original and the reduced XL-algorithm (see Proposition 1). Because of this connection, one can use an analysis of (a generalization of) the original XL-algorithm to analyze also the reduced XL-algorithm (see Theorem 1 and Corollary 1). In order to state the main ideas of our analysis and to compare our results with the conjectures of [7], in this section, we concentrate on the original XL-algorithm.

For $D \in \mathbb{N}$, let $K[X_1, \ldots, X_n]_{\leq D}$ be the K-vector space of polynomials in $X_1, \ldots, X_n$ of (total) degree $\leq D$, and let

$$\chi(D) := \dim_K(K[X_1, \ldots, X_n]_{\leq D}) - \dim_K(U_D) \ . \tag{3}$$

One can surely obtain a non-trivial univariate polynomial by (Gaussian) elimination on (2) if $\chi(D) \leq D$. (This is because the K-vector space $K[X_1]_{\leq D}$ has dimension $D+1$, thus if $\chi(D) \leq D$, then $\dim_K(U_D) + \dim_K(K[X_1]_{\leq D}) > \dim_K(K[X_1, \ldots, X_n]_{\leq D})$, and this implies that $U_D \cap K[X_1]_{\leq D} \neq \{0\}$.) In order to analyze the running time of the XL-algorithm, it is of greatest importance to study the question for which D one can expect this condition to hold (if such a D exists at all). We thus define $D_{\min}$ be the minimal D with $\chi(D) \leq D$ (if no such D exists, we set $D_{\min} = \infty$).

The starting point for our analysis of the XL-algorithm is the interpretation of the original XL-algorithm via the theory of *homogeneous* polynomial ideals pointed out by Moh ([13]). This interpretation opens the door for the usage of well-established methods from commutative algebra – the keywords are Hilbert Theory, Hilbert functions, Hilbert series and Hilbert polynomials.

A crucial observation is that in order to derive lower bounds on $\chi(D)$ (i.e. upper bounds on $\dim_K(U_D)$) it suffices to study the dimensions of the homogeneous parts of algebras defined by *generic* systems of homogeneous polynomials. (This notion will be made precise in Sect. 4.) For $m \leq n+1$, these dimensions are known, and this information suffices to prove that for $m = n + c, c \geq 1$,

$$D_{\min} \geq \frac{n}{\sqrt{c-1}+1} \tag{4}$$

(see Proposition 6.) In contrast to this inequality, it was suggested in [7–Sect. 6.4] that "even for small values of c", $c \geq 2$, one has

$$D_{\min} \approx \sqrt{n} \ . \tag{5}$$

Let us fix the field K and $c \geq 2$ and study the asymptotic behavior of the running time of the original XL-algorithm for $n \longrightarrow \infty$ (and $m = n + c$): If (5) was true, the XL-algorithm in [7] would have a running time (in field operations) which is *subexponential* in n. (This hope was expressed at the end of Sect. 6.1 of

[7] as well as at the end of the introduction of [7].) However by (4), the running time of all instances for which $U_D \cap K[X_1]_{\leq D} = \{0\}$ for all $D < D_{\min}$ is *not subexponential* in n.

If K is a finite field and $\#K$ and n are not "too small" it seems very reasonable to expect: Under all systems (1) which have a solution in K, the portion of systems for which $U_D \cap K[X_1]_{\leq D} \neq \{0\}$ for some $D < D_{\min}$ is negligible. This suggests that for any fixed $c \geq 1$ the median of the running times of the original XL-algorithm applied to systems of $m = n + c$ quadratic equations in n variables over K which have a solution in K is not subexponential in n. The hope stated at the end of Sect. 1 of [7] that "the expected running time of this technique is ... subexponential if m exceeds n by a small number" should be abandoned.

For $c \geq 3$, much more precise lower bounds on $D_{\min}$ then the ones in (4) can be obtained if one assumes a certain conjecture which implies what the dimensions of homogeneous parts of algebras defined by generic systems of homogeneous polynomials should be (see Sect. 5). This conjecture – which is now approximately 20 years old – states that certain linear maps are either injective or surjective, that is, they have maximal rank. Because of this, we speak of the *maximal rank conjecture (MR-conjecture)*.

3 A Generalization of the Original and the Reduced XL-Algorithm

The original as well as the reduced XL-algorithm of can easily be generalized to more general than quadratic systems of polynomial equations (see also [5–Sect. 2]).

For these generalizations, we start off with a system of m polynomial equations

$$f_1(X_1, \ldots, X_n) = 0, \ \ldots \ , f_m(X_1, \ldots, X_n) = 0 \ . \tag{6}$$

The generalization of the original XL-algorithm works just as the original XL-algorithm stated in the previous section with the difference that for some $D \in \mathbb{N}$, one applies Gaussian elimination to the linearized system of all polynomial equations

$$\prod_{\ell=1}^{k} X_{i_\ell} \cdot f_j(X_1, \ldots, X_n) = 0 \ , \tag{7}$$

where $k + \deg(f_j) \leq D$.

Clearly, the reduced XL-algorithm can be generalized in a similar manner. From now on, we refer to these generalizations also as the "original" and the "reduced" XL-algorithm.

Let us fix the following *notations*.

– As in the previous section, let

$$U_D := \langle \textstyle\prod_{\ell=1}^{k} X_{i_\ell} \cdot f_j(X_1, \ldots, X_n) \text{ with } k \leq D - \deg(f_j) \rangle_K \tag{8}$$

– and
$$\chi(D) := \dim_K(K[X_1, \ldots, X_n]_{\leq D}) - \dim_K(U_D) \ . \tag{9}$$
Let $K = \mathbb{F}_q$.
– If $f \in K[X_1, \ldots, X_n]_{\leq D}$, we denote by f^{red} the "reduction" of f, i.e. f^{red} is the polynomial obtained by maximally reducing all exponents in the monomials according to the relations $\prod_{\ell=1}^{k} X_{i_\ell} - X_i^q \cdot \prod_{\ell=1}^{k} X_{i_\ell} = 0$. Note that $(\ldots)^{\mathrm{red}}$ is a homomorphism of K-vector spaces and that $(U_D)^{\mathrm{red}} \leq (K[X_1, \ldots, X_n]_{\leq D})^{\mathrm{red}}$ is the space of equations generated in the reduced XL-algorithm.
– In order to analyze the reduced XL-algorithm, we set
$$\chi^{\mathrm{red}}(D) := \dim_K((K[X_1, \ldots, X_n]_{\leq D})^{\mathrm{red}}) - \dim_K((U_D)^{\mathrm{red}}) \ . \tag{10}$$
– Let $\tilde{U}_D$ be defined just as U_D with respect to the system of $m+n$ polynomials $f_1, \ldots, f_m, X_1^q - X_1, \ldots, X_n^q - X_n$, and let $\tilde{\chi}(D)$ be defined as $\chi(D)$ with respect to $\tilde{U}_D$.

The proof of the following proposition can be found in Appendix A.

Proposition 1. *We have* $\chi^{\mathrm{red}}(D) = \tilde{\chi}(D)$.

Because of this proposition, the results on the original XL-algorithm can easily be carried over to the reduced XL-algorithm. What remains is to derive non-trivial lower bounds on $\chi(D)$.

4 The XL-Algorithm and Hilbert Theory

In the following discussion, we assume that the reader is familiar with basic notions of commutative algebra as can for example be found in the first three chapters of [1].

As mentioned in Sect. 2, our analysis of the XL-algorithm relies on an interpretation via homogeneous polynomial ideals. The main idea is to consider (for some field K, some $n \in \mathbb{N}$ and some $D \in \mathbb{N}$) the homogeneous polynomials of degree D in $n+1$ variables instead of the polynomials of degree $\leq D$ in n variables.

Let K be an arbitrary field, $n \in \mathbb{N}$, and let $f_1, \ldots, f_n \in K[X_1, \ldots, X_n]$. We use the notations of the previous sections, and additionally we denote by $K[X_0, \ldots, X_n]_D$ the K-vector space of all homogeneous polynomials of degree D. More generally, for any positively graded $K[X_0, \ldots, X_n]$-module M, we denote the homogeneous part of degree D of M by M_D.

Let $F_j \in K[X_0, \ldots, X_n]$ be the homogenization of f_j, that is, F_j is the unique homogeneous polynomial in $K[X_0, \ldots, X_n]$ of the same degree as f_j with $F_j(1, X_1, \ldots, X_n) = f_j(X_1, \ldots, X_n)$.

Let
$$\Phi : K[X_1, \ldots, X_n]_{\leq D} \longrightarrow K[X_0, \ldots, X_n]_D$$

be the "degree D homogenization map", that is, the K-linear map given by

$$\prod_{\ell=1}^{k} X_{i_\ell} \mapsto X_0^{D-k} \prod_{\ell=1}^{k} X_{i_\ell} \quad .$$

Then under the isomorphism Φ, the K-vector space U_D corresponds to

$$\langle \prod_{\ell=1}^{k} X_{i_\ell} \cdot F_j(X_0, X_1, \ldots, X_n) \text{ with } k = D - \deg(F_j) \rangle_K \ ,$$

where the products are taken over the variables $X_0, \ldots, X_n$. This space is nothing but the D^{th} homogeneous component of the homogeneous ideal

$$I := (F_1, \ldots, F_m) \lhd K[X_0, \ldots, X_n] \ ,$$

denoted I_D. We have

$$\begin{aligned} \chi(D) &\stackrel{\text{Def}}{=} \dim_K(K[X_1, \ldots, X_n]_{\leq D}) - \dim_K(U_D) \\ &= \dim_K(K[X_0, \ldots, X_n]_D) - \dim_K(I_D) \\ &= \dim_K(K[X_0, \ldots, X_n]_D / I_D) \\ &= \dim_K((K[X_0, \ldots, X_n]/I)_D) \ . \end{aligned} \tag{11}$$

Let $R := K[X_0, \ldots, X_n]$. Recall the following definitions (see e.g. [16–Sect. 1]).

Definition 2. *Let $M = \bigoplus_{i \in \mathbb{N}_0} M_i$ be any finitely generated positively graded R-module. Then the function*

$$\chi_M : \mathbb{N}_0 \longrightarrow \mathbb{N}_0, \quad \chi_M(i) := \dim_K(M_i)$$

is called the Hilbert function *of M.*

The power series with integer coefficients

$$H_M := \sum_{i \in \mathbb{N}_0} \chi_M(i) T^i$$

is called the Hilbert series *of M.*

Note that the above equation (11) states that

$$\chi(D) = \chi_{R/I}(D) \quad \text{for all } D \in \mathbb{N} \ . \tag{12}$$

Let us denote by $X^{\underline{i}}$ the monomial corresponding to the multiindex $\underline{i} \in \mathbb{N}_0^{\{0,\ldots,n\}}$. The following definition can be found in [10].

Definition 3. *A form (i.e. a homogeneous polynomial) $G = \sum_{\underline{i}} a_{\underline{i}} X^{\underline{i}} \in R$ of degree d is* generic *if all monomials of degree d in R have coefficients $a_{\underline{i}}$ in G, and these coefficients are algebraically independent over the prime field of K.*

A generic system of forms *is a system of generic forms $G_j = \sum_{\underline{i}} a_{\underline{i}}^{(j)} X^{\underline{i}}$ as above (not necessarily of the same degree) such that all $a_{\underline{i}}^{(j)}$ are algebraically independent over the prime field of K. An ideal I generated by a generic system of forms is called* generic*, and so is the R-algebra R/I.*

Lemma and Definition 4. *The Hilbert series of an ideal generated by a generic system $G_1, \ldots, G_m$ of forms of degrees $d_1, \ldots, d_m$ depends only on the characteristic of the field, the number n and the tuple of numbers $(d_1, \ldots, d_m)$. If the characteristic of the field is 0, we speak of* the generic Hilbert series *of type* $(n+1; m; d_1, \ldots, d_m)$.

Proof. Let K and L be two fields over the same prime field F, and let $G_1, \ldots, G_m \in K[X_0, \ldots, X_n], G'_1, \ldots, G'_m \in L[X_0, \ldots, X_n]$ be two generic systems of forms such that $\deg(G_j) = \deg(G'_j)$ for all j. Let $G_j = \sum_{\underline{i}} a_{\underline{i}}^{(j)} X^{\underline{i}}$, $G'_j = \sum_{\underline{i}} a'^{(j)}_{\underline{i}} X^{\underline{i}}$. Let k and l respectively be the subfields of K and L generated by the coefficients of G_j and G'_j over F. Then there exists a (unique) isomorphism between k and l under which $a_i^{(j)}$ corresponds to $a'^{(j)}_i$. We thus have for all $D \in \mathbb{N}_0$

$$\begin{gathered}\chi_{K[X_0,\ldots X_n]/(G_1,\ldots,G_m)}(D) = \dim_K((K[X_0,\ldots X_n]/(G_1,\ldots,G_m))_D) = \\ \dim_k((k[X_0,\ldots X_n]/(G_1,\ldots,G_m))_D) = \\ \dim_l((l[X_0,\ldots X_n]/(G'_1,\ldots,G'_m))_D) = \\ \dim_L((L[X_0,\ldots X_n]/(G'_1,\ldots,G'_m))_D) = \chi_{L[X_0,\ldots X_n]/(G'_1,\ldots,G'_m)}(D) \ .\end{gathered}$$

□

Together with (12), the following proposition is crucial for our analysis of the XL-algorithm.

Proposition 2. *Let K be any field (of any characteristic), and let $F_1, \ldots, F_m \in R = K[X_0, \ldots, X_n]$ be forms of degree $d_1, \ldots, d_m$ (not necessarily generic). Let H_g be the generic Hilbert series of type $(n+1; m; d_1, \ldots, d_m)$. Let $I := (F_1, \ldots, F_m) \lhd K[X_0, \ldots, X_n]$. Then we have the coefficient-wise inequality*

$$H_{R/I} \geq H_g \ .$$

This proposition seems to be well-known in commutative algebra (see e.g. Sect. 4 of [16]); for the lack of a suitable reference we include the proof in Appendix B.

Because of (12) and this proposition the task is now to study generic Hilbert series. The following proposition is a well-known statement from commutative algebra.

Proposition 3. *Let $m \leq n+1$, and let $G_1, \ldots, G_{m-1}, G_m = G$ be a generic system of forms in $R = K[X_0, \ldots, X_n]$, where G has degree d. Let $J := (G_1, \ldots, G_{m-1}) \lhd R$. Then for all $D \in \mathbb{N}_0$ the multiplication map*

$$G\cdot : (R/J)_D \longrightarrow (R/J)_{D+d},\ \overline{F} \mapsto G \cdot \overline{F}$$

is injective, in particular we have a short exact sequence

$$0 \longrightarrow (R/J)_D \xrightarrow{G\cdot} (R/J)_{D+d} \longrightarrow (R/(J,G))_{D+d} \longrightarrow 0 \ .$$

(Here by (J, G) we denote the ideal of R generated by J and G.)

Indeed, this proposition is nothing but a reformulation of the well-known statement that a generic system of forms in $K[X_0, \ldots, X_n]$ with at most $n+1$ elements forms a regular sequence (cf. [16–Sect. 4], Page 318). (The fact that a generic system of forms is a regular sequence can be seen as follows: By Lemma 3 in Appendix B, it suffices to prove that for all $(d_1, \ldots, d_{n+1}) \in \mathbb{N}^{n+1}$ there exist some forms $F_1, \ldots, F_{n+1} \in R$ of degrees $d_1, \ldots, d_{n+1}$ which form a regular sequence, and by [12–Theorem 16.1], the forms $X_0^{d_1}, \ldots, X_n^{d_{n+1}}$ do form a regular sequence.)

Note that the Hilbert series of $R = K[X_0, \ldots, X_n]$ is

$$H_R = \sum_i \binom{n+i}{i} T^i = \frac{1}{(1-T)^{n+1}} \tag{13}$$

(see e.g. [16–Sect. 1]). Proposition 3 and (13) imply by induction on m:

Proposition 4. *Let $m \leq n+1$, and let $G_1, \ldots, G_m$ be a generic system of forms of degrees $d_1, \ldots, d_m$ in R. Then the Hilbert series of $R/(G_1, \ldots, G_m)$ is*

$$\frac{\prod_{j=1}^{m}(1-T^{d_j})}{(1-T)^{n+1}} \quad .$$

For simplicity we now concentrate until the end of this section on the case of *quadratic* equations.

Proposition 5. *Let $m = n + c$ for some $c \geq 1$. Let $F_1, \ldots, F_m$ be quadratic forms in $R = K[X_0, \ldots, X_n]$. Then the Hilbert series of $R/(F_1, \ldots, F_m)$ is coefficient-wise greater-or-equal*

$$(1-(c-1)T^2)(1+T)^{n+1} \quad .$$

Proof. By Proposition 2 we only have to prove that the generic Hilbert series of type $(n+1; m; 2, \ldots, 2)$ is coefficient-wise greater-or-equal $(1-(c-1)T^2)(1+T)^{n+1}$.

So let K be a field of characteristic 0, let $R = K[X_0, \ldots, X_n]$, and let $G_1, \ldots, G_m$ be a generic system of quadratic forms in R. (The assumption on the characteristic is not necessary for the following argument.) Let $R' := R/(G_1, \ldots, G_{n+1})$, and let I' be the ideal generated by $G_{n+2}, \ldots, G_m$ in R'. Note that by the above proposition, the Hilbert series of R' is $(1+T)^{n+1}$. We have $R/(G_1, \ldots, G_m) \simeq R'/I'$, thus

$$\chi_{R/(G_1,\ldots,G_m)}(D) = \chi_{R'/(G_{n+2},\ldots,G_m)}(D) = \dim_K(R'_D) - \dim_K(I'_D) \quad .$$

Now, for $D \geq 2$, $I'_D = \sum_{j=n+2}^{m} G_j \cdot R'_{D-2}$, where by definition $G_j \cdot R'_{D-2}$ is the image of R'_{D-2} under the multiplication map $G_j \cdot : R'_{D-2} \longrightarrow R'_D$. It follows that

$$\dim_K(I'_D) \leq (m-n-1)\dim_K(R'_{D-2}) \quad .$$

All in all, we have

$$\chi_{R/(G_1,\ldots,G_m)}(D) \geq \chi_{R'}(D) - (c-1)\chi_{R'}(D-2) \ ,$$

thus

$$H_{R/(G_1,\ldots,G_m)} \geq H_{R'} - (c-1)T^2 H_{R'} = (1-(c-1)T^2)(1+T)^{n+1} \ .$$

Proposition 6. *Let K be any field, let $m = n + c$ with $c \geq 1$, let $f_1, \ldots, f_m \in K[X_1, \ldots, X_n]$ be quadratic polynomials, and as in Sect. 2, let $D_{\min}$ be the minimal D with $\chi(D) \leq D$, where $\chi(D)$ is defined as above with respect to these polynomials. Then*

$$D_{\min} \geq \frac{n}{\sqrt{c-1}+1} \ .$$

Sketch of the Proof. Let $c \geq 1$, $m = n + c$, $f_1, \ldots, f_m$ and $\chi(D)$ be as in the proposition. By (12) and the above proposition, we have

$$\chi(D) \geq \left(\frac{(n-D+2)(n-D+3)}{(D-1)D} - (c-1)\right) \cdot \binom{n+1}{D-2}$$

for all $D \geq 2$. The proposition follows from the statement that $\left(\frac{(n-D+2)(n-D+3)}{(D-1)D} - (c-1)\right) \cdot \binom{n+1}{D-2} > D$ for all $D < \frac{n}{\sqrt{c-1}+1}$. We only show the slightly weaker statement that $\frac{(n-D+2)(n-D+3)}{(D-1)D} - (c-1) > 0$ for all $D < \frac{n}{\sqrt{c-1}+1}$.

Let $D < \frac{n}{\sqrt{c-1}+1}$. Then $D\sqrt{c-1} + D < n$, thus $D^2(c-1) < (n-D)^2$. This implies that $(D-1)D(c-1) < (n-D+2)(n-D+3)$, i.e. $\frac{(n-D+2)(n-D+3)}{(D-1)D} - (c-1) > 0$. □

Remark 5. For an application of the XL-algorithm to a system with $m = n$ quadratic equations, one can easily see with Propositions 2 and 4 and (12) that one always has $\chi(D) \geq 2^n$, and for $m = n + 1$ quadratic equations, one has $D_{\min} \geq n + 1$. Both these results are consistent with conjectures in [7].

5 The Maximal Rank Conjecture

The *maximal rank conjecture (MR-conjecture)* which we now state can be thought to be a (potential) generalization of Proposition 3.

Conjecture. Let K be a field of characteristic 0, and let $G_1, \ldots, G_{m-1}, G_m = G$ be a generic system of forms in $R = K[X_0, \ldots, X_n]$, where G has degree d. Let $J := (G_1, \ldots, G_{m-1}) \lhd R$. Then for all $D \in \mathbb{N}_0$ the multiplication map

$$G\cdot : (R/J)_D \longrightarrow (R/J)_{D+d},\ \overline{F} \mapsto G \cdot \overline{F}$$

has *maximal rank*, that is it is injective if $\dim_K((R/J)_D) \leq \dim_K((R/J)_{D+d})$ and it is surjective if $\dim_K((R/J)_D) \geq \dim_K((R/J)_{D+d})$.

This conjecture – which is also known under the name "Fröberg's Conjecture" – can (in an equivalent formulation) be found in [10]. It is also stated in Sect. 4 of the informative overview article [16]. (Note however that the formulations at the beginning of Sect. 4 of [16] are a bit vague.) Interesting facts about this and related conjectures can be found in [15].

The conjecture is known to hold if one of the following five conditions is satisfied: $m \leq n+1$ (see Proposition 3), $n = 1, n = 2$, $m = n+2$, $D = \min_j\{\deg(G_j)\}+1$ (see [10–3.2.] and the citations in [16–Sect. 4]).

The conjecture is equivalent to the statement that

$$\chi_{R/(J,G)}(D) = \max\{\chi_{R/J}(D) - \chi_{R/J}(D-d)\,,\,0\}$$

as one can easily see (cf. [16–Sect. 4]). (Here, we set $\chi_{R/J}(i) = 0$ for $i < 0$.)

Obviously, if $\chi_{R/(J,G)}(D) = 0$, then $\chi_{R/(J,G)}(D') = 0$ for all $D' > D$. Using this fact, the conjecture can be reformulated via Hilbert series as:

$$H_{R/(I,G)} = |(1-T^d)H_{R/I}| \ , \tag{14}$$

where for some power series $p(T)$ with integer coefficients, $|p(T)|$ denotes the power series $q(T) = \sum_i q_i T^i$, where

$$\begin{aligned} q_i = p_i \text{ if } p_j > 0 \text{ for all } j \leq i \\ 0 \text{ if } p_j \leq 0 \text{ for some } j \leq i \ . \end{aligned}$$

Assumption. *From now on, we assume that the maximal rank conjecture is valid.*

Let K be a field of characteristic 0, let $G_1, \ldots, G_m$ be a generic system of forms in R, and let $d_j := \deg(G_j)$. Let $I := (G_1, \ldots, G_m)$. Using (13), (14) and Lemma 5 in Appendix C, we have

$$H_{R/I} = \left| \frac{\prod_{j=1}^{m}(1-T^{d_j})}{(1-T)^{n+1}} \right| \ . \tag{15}$$

Definition 5. *(see [16]) We call the right-hand side of the above equation the* expected Hilbert series *of a generic algebra of type* $(n+1; m; d_1, \ldots, d_m)$.

Proposition 2 implies:

Proposition 7. *Let K be any field (of any characteristic), and let $F_1, \ldots, F_m \in R = K[X_0, \ldots, X_n]$ be forms of degree $d_1, \ldots, d_m$ (not necessarily generic). Let H_e be the corresponding expected Hilbert series. Let $I := (F_1, \ldots, F_m)$. Then we have the coefficient-wise inequality*

$$H_{R/I} \geq H_e \ .$$

Together with (12), this proposition has the following implication for the original XL-algorithm.

Theorem 1. *Let K be any field, let $f_1, \ldots, f_m$ be non-trivial polynomials in $K[X_1, \ldots, X_n]$ with degrees $d_1, \ldots, d_m$. Let $D \in \mathbb{N}$, and let $\chi(D)$ be defined as in (9). Then $\chi(D)$ is greater-or-equal to the D^{th} term of the expected Hilbert series of a generic algebra of type $(n+1; m; d_1, \ldots, d_m)$.*

By Proposition 1, this theorem has the following corollary which can be used to analyze the reduced XL-algorithm.

Corollary 1. *With the notations of the theorem, let $K = \mathbb{F}_q$, and let $\chi^{\mathrm{red}}(D)$ be defined as is (10). Then $\chi^{\mathrm{red}}(D)$ is greater-or-equal to the D^{th} term of the expected Hilbert series of a generic algebra of type $(n+1; m+n; d_1, \ldots, d_m, q, \ldots, q)$.*

Remark 6. Let $D_{n,m}$ be the degree of the expected Hilbert series of a generic algebra of type $(n+1; m; 2, \ldots, 2)$. One can use the methods presented in [2–Sect. 5] to study asymptotic behaviors of $D_{n,m}$. A corresponding study is carried out in [3]. One obtains $D_{n,n+c} \sim \frac{n}{2}$ for any fixed $c \geq 2$ and $n \longrightarrow \infty$. (More precise results for various small c can also be found in [3].) For a fixed $\alpha > 1$, a reformulation of a result in [3] gives $D_{n,\alpha n} \sim (\alpha - \sqrt{\alpha^2 - \alpha} - \frac{1}{2}) \cdot n$ for $n \longrightarrow \infty$. For example, one has $D_{n,2n} \sim C \cdot n$ with $C = \frac{3}{2} - \sqrt{2} \approx 0.0858$ (which is consistent with the "Comparison with $2n$ equations over $\mathbb{Q}$" on page 13 of [2]).

Acknowledgment

I thank G. Böckle, T. Bröcker, C. Cid, A. Conca, G. Frey, S. Galbraith, J. Herzog, J. Scholten, A. Wiebe and B.-Y. Yang for discussions. I am particularly in debt to J. Herzog and A. Wiebe for pointing out the "maximal rank conjecture" to me.

Support by the IST Programme "Ecrypt" of the European Union is gratefully acknowledged.

References

[1] M. Atiyah and I. Macdonald. *Introduction to Commutative Algebra*. Addison-Wesley, Reading, 1969.

[2] M. Bardet, J.-C. Faugère, and B. Salvy. Complexity of Gröbner basis computations for Semi-regular Overdetermined sequences over $\mathbb{F}_2$ with solutions in $\mathbb{F}_2$. INRIA Rapport de recherche No. 5049, 2003.

[3] J.-M. Chen and B.-Y. Yang. All in the XL Familiy: Theory and Practice. manuscript from June 2004.

[4] J.-M. Chen and B.-Y. Yang. Theoretical Analysis of XL over Small Fields. In H. Wang, J. Pieprzyk, V. Varadharajan, editors, *Information Security and Privacy*, volume 3108 of *LNCS*, pages 277-288, Springer-Verlag, Berlin, 2004.

[5] N. Courtois. Higher Order Correlation Attacks, XL algorithm, and Cryptanalysis of Toyocrypt. In P.J. Lee, C.H. Lim, editors, *Advances in Cryptology — ICISC 02*, volume 2587 of *LNCS*, pages 182–199, Springer-Verlag, Berlin, 2002.

[6] N. Courtois, A. Klimov, J. Pararin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. "extended version", available under http://www.minrank.org/xlfull.pdf (as of August 24, 2004).

[7] N. Courtois, A. Klimov, J. Pararin, and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer-Verlag, Berlin, 2000.
[8] N. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 267–287. Springer-Verlag, Berlin, 2002.
[9] D. Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*. Springer-Verlag, New York, 1995.
[10] R. Fröberg. An inequality for Hilbert series of graded algebras. *Math. Scand.*, 56:117–144, 1985.
[11] D. Knuth. *The Art of Computer Programming*. Addison-Wesley, Reading, 1973.
[12] H. Matsumura. *Commutative Ring Theory*. Cambridge University Press, Cambridge, UK, 1986.
[13] T. Moh. On the method of "XL" and its inefficiency to TTM. manuscript from January 28, 2000, available under http://eprint.iacr.org/2001/047.
[14] S. Murphy and M.J.B. Robshaw. Essential algebraic structure within the AES. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 1–16, 2002.
[15] K. Pardue. Generic Sequences of Polynomials. manuscript from March 30, 2000.
[16] G. Valla. Problems and Results on Hilbert Polynomials of Graded Algebras. In J. Elas and J. Giral, editors, *Six Lectures on Commutative Algebra*, volume 166 of *Progress in Mathematics*. Birkhäuser, Basel, 1996.

A On the Connection Between the Original and the Reduced XL-Algorithm

The purpose of this section is to prove Proposition 1.

As in Proposition 1, let $K = \mathbb{F}_q$. Let

$$V_D := \langle \prod_{\ell=1}^{k} X_{i_\ell} \cdot (X_i^q - X_i) \text{ with } k + q \leq D \rangle_K \leq K[X_1, \ldots, X_n]_{\leq D} \ .$$

Lemma 1. *Let U be any K-vector subspace of $K[X_1, \ldots, X_n]_{\leq D}$. Then we have a short exact sequence*

$$0 \longrightarrow U \cap V_D \longrightarrow U \longrightarrow U^{\mathrm{red}} \longrightarrow 0 \ .$$

Proof. It is obvious that $U \cap V_D$ is contained in the kernel of $(\ldots)^{\mathrm{red}}$. The converse follows from the following lemma. □

Lemma 2. *Let $f \in K[X_1, \ldots, X_n]$. Then there exist polynomials $p_1, \ldots, p_n$ of degree $\leq \deg(f) - q$ with*

$$f = p_1 \cdot (X_1^q - X_1) + \cdots p_n \cdot (X_n^q - X_n) + f^{\mathrm{red}} \ .$$

Proof. By the linearity of $(\ldots)^{\mathrm{red}}$, it suffices to prove the statement for monomials, and for monomials it is obvious by the very definition of $(\ldots)^{\mathrm{red}}$. □

Let us use the definitions of Sect. 3.
We have by Lemma 1

$$(K[X_1,\ldots,X_n]_{\leq D})^{\mathrm{red}} \simeq K[X_1,\ldots,X_n]_{\leq D}/V_D \ ,$$

$$(U_D)^{\mathrm{red}} \simeq U_D/(U_D \cap V_D) \simeq (U_D+V_D)/V_D \simeq \tilde{U}_D/V_D \ ,$$

thus

$$\begin{aligned}(K[X_1,\ldots,X_n]_{\leq D})^{\mathrm{red}}/(U_D)^{\mathrm{red}} &\simeq (K[X_1,\ldots,X_n]_{\leq D}/V_D)/(\tilde{U}_D/V_D)\\ &\simeq K[X_1,\ldots,X_n]/\tilde{U}_D \ .\end{aligned}$$

This implies:

$$\begin{aligned}\chi^{\mathrm{red}}(D) &= \dim_K((K[X_1,\ldots,X_n]_{\leq D})^{\mathrm{red}}/(U_D)^{\mathrm{red}})\\ &= \dim_K(K[X_1,\ldots,X_n]_{\leq D}/\tilde{U}_D) \ = \tilde{\chi}(D) \ .\end{aligned}$$

B Hilbert Series of Generic and Arbitrary Algebras

The purpose of this section is to prove Proposition 2. Let us before we come to the proof state two lemmata.

Lemma 3. *Let A be a domain with quotient field Q. Let K be a field and let $\varphi : A \longrightarrow K$ be a homomorphism, and let $\Phi : A[X_0,\ldots,X_n] \longrightarrow K[X_0,\ldots,X_n]$ be the canonical extension of φ. Let $I \lhd A[X_0,\ldots,X_n]$ be a homogeneous ideal (that is, an ideal generated by homogeneous polynomials). Then we have the coefficient-wise inequality*

$$H_{K[X_0,\ldots,X_n]/(\Phi(I))} \geq H_{Q[X_0,\ldots,X_n]/(I)} \ .$$

Proof. Let $D \in \mathbb{N}_0$. The map $\varphi : A \longrightarrow K$ induces a canonical map

$$\begin{aligned}(A[X_0,\ldots,X_n]/I)_D &\longrightarrow (A[X_0,\ldots,X_n]/I)_D \otimes_A K\\ &\simeq (A[X_0,\ldots,X_n]/I \otimes_A K)_D \simeq (K[X_0,\ldots,X_m]/(\Phi(I)))_D.\end{aligned}$$

This implies that

$$\begin{aligned}&\chi_{K[X_0,\ldots,X_n]/(\Phi(I))}(D)\\ &= \dim_K((A[X_0,\ldots,X_n]/I)_D \otimes_A K)\\ &\geq \dim_Q((A[X_0,\ldots,X_n]/I)_D \otimes_A Q) \ \text{ by Lemma 4 below}\\ &= \dim_Q((Q[X_0,\ldots,X_n]/(I))_D)\\ &= \chi_{Q[X_0,\ldots,X_n]/(I)}(D) \ .\end{aligned}$$

Lemma 4. *Let A be a domain with quotient field Q, and let M be a finitely generated A-module. Let K be a field and let $\varphi : A \longrightarrow K$ be a homomorphism. Then*

$$\dim_K(M \otimes_A K) \geq \dim_Q(M \otimes_A Q) \ .$$

Proof. Let $\mathfrak{m}$ be the kernel of φ. Then $M \otimes_A K \simeq M_{\mathfrak{m}} \otimes_{A_{\mathfrak{m}}} K$ and $M \otimes_A Q \simeq M_{\mathfrak{m}} \otimes_{A_{\mathfrak{m}}} Q$. We can thus assume that A is a local ring with maximal ideal $\mathfrak{m} = \ker(\varphi)$. As the dimension of a vector space is stable under base-change, we can further assume that φ is surjective. Now if $m_1, \ldots, m_r$ form modulo $\mathfrak{m}$ a basis of $M \otimes_A K$ over K then by Nakayama's Lemma ([9–Corollary 4.8]) they generate the A-module M, thus they generate $M \otimes_A Q$ over Q. □

Proof of Proposition 2. We keep the notations of the proposition. The proposition follows from Lemma 3 applied to the multivariate polynomial ring $A = \mathbb{Z}[\{a_{\underline{i}}^{(j)}\}]$, the ideal $I = (G_1, \ldots, G_m)$, where $G_1, \ldots, G_m$ with $G_j = \sum_{\underline{i}} a_{\underline{i}}^{(j)} X^{\underline{i}}$ and $\deg(G_j) = d_j$ is a generic system of forms, and the specialization homomorphism $\varphi : A \longrightarrow K$ sending $a_{\underline{i}}^{(j)}$ to the corresponding coefficient of F_j. (Note that the quotient field of A is $\mathbb{Q}(\{a_{\underline{i}}^{(j)}\})$ which has characteristic 0.) □

C A Lemma on Power Series

The following lemma generalizes [10–Lemma 4]. For the convenience of the reader, we include a proof.

Lemma 5. *Let $p(T)$ be a power series with integer coefficients, let $d \in \mathbb{N}$. Then*

$$|(1 - T^d)p(T)| = |(1 - T^d)|p(T)|| \ .$$

Proof. Note that $((1 - T^d)p(T))_i = p_i$ for $i < d$ and $((1 - T^d)p(T))_i = p_i - p_{i-d}$ for $i \geq d$.

Thus the coefficients whose index is $< d$ of both sides agree. Furthermore, if $p_i < 0$ for some $i < d$, then both sides are equal.

Let us assume that for all $i = 0, \ldots, d-1$, we have $p_i > 0$.

If now for all i we have $p_i - p_{i-d} > 0$, then we also have $p_i > 0$ for all i as can easily be seen by induction on i. In this case, both sides agree with $(1 - T^d)p(T)$.

Assume that this is not the case and let a be the least natural number for which $p_a - p_{a-d} \leq 0$.

Then for each $i < a$, we have $p_i > 0$ again by induction on i.

There are two cases: Either $p_a > 0$. Then $|p(T)|_a - |p(T)|_{a-d} = p_a - p_{a-d} < 0$ by definition of a. Or $p_a \leq 0$. Then $|p(T)|_a - |p(T)|_{a-d} = -p_{a-d} < 0$.

We conclude that for $i \leq d-1$, the i-th coefficient of both sides agrees with p_i, for $d < i < a$, the i-th coefficient of both sides agrees with $p_i - p_{i-d}$, and for $i \geq a$, the i-th coefficient of both sides is 0. □

The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

The information in this document reflects only the author's views, is provided as is, and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Comparison Between XL and Gröbner Basis Algorithms

Gwénolé Ars[1], Jean-Charles Faugère[2], Hideki Imai[3], Mitsuru Kawazoe[4], and Makoto Sugita[5]

[1] IRMAR, University of Rennes 1,
Campus de Beaulieu 35042 Rennes, France
gwenole.ars@univ-rennes1.fr
[2] LIP6/CNRS/INRIA, University of Paris VI,
8 rue du Capitaine Scott Paris 75015 Paris, France
Jean-Charles.Faugere@lip6.fr
[3] Institute of Industrial Science, University of Tokyo,
4-6-1 Komaba, Meguro-ku Tokyo, 153-8505, Japan
imai@iis.u-tokyo.ac.jp
[4] Department of Mathematics and Information Sciences, Osaka Prefecture University,
1-1 Gakuen-cho Sakai Osaka 599-8531, Japan
kawazoe@mi.cias.osakafu-u.ac.jp
[5] IT Security Center, Information-technology Promotion Agency, Japan
2-28-8 Honkomagome, Bunkyo-ku Tokyo, 113-6591, Japan
m-sugita@ipa.go.jp

Abstract. This paper compares the XL algorithm with known Gröbner basis algorithms. We show that to solve a system of algebraic equations via the XL algorithm is equivalent to calculate the reduced Gröbner basis of the ideal associated with the system. Moreover we show that the XL algorithm is also a Gröbner basis algorithm which can be represented as a redundant variant of a Gröbner basis algorithm F_4. Then we compare these algorithms on semi-regular sequences, which correspond, in conjecture, to almost all polynomial systems in two cases: over the fields $\mathbb{F}_2$ and $\mathbb{F}_q$ with $q \gg n$. We show that the size of the matrix constructed by XL is large compared to the ones of the F_5 algorithm. Finally, we give an experimental study between XL and the Buchberger algorithm on the cryptosystem HFE and find that the Buchberger algorithm has a better behavior.

Keywords: Multivariate polynomial equations, Algebraic attacks, Solving Systems, Gröbner basis, XL algorithm, Semi-regular Sequences.

1 Introduction

Algebraic attacks are among the most efficient attacks for public key cryptosystems, block ciphers and stream ciphers. They try to recover a secret key by solving a system of algebraic equations. Algebraic attacks were first applied to Matsumoto-Imai Public Key Scheme in [19] by Jacques Patarin and a similar

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 338–353, 2004.

attack was also applied in [15]. Algebraic attacks were also applied to block ciphers in [6], where the complexity for attacking AES and Serpent was evaluated. Moreover, algebraic attacks were applied to stream cipher in [7], [8], [9] and improved in [1].

As a general method to solve a system of algebraic equations, we know Gröbner basis algorithms. The fastest of such algorithms previously known are the F_4 and F_5 algorithms introduced in [11] and [12], respectively.

The XL algorithm was proposed as an efficient algorithm for algebraic attacks. It was first introduced in [20] and applied to an attack for HFE which is an improved version of Matsumoto-Imai Public Key Scheme. It was improved in [5]. As stated in [20], in cryptographic scheme, a system of algebraic equations we are interested in has a unique solution over its defining field. The XL algorithm was proposed as a powerful technique to solve such special systems. In [20], it was stated that the XL algorithm does not try to calculate a whole Gröbner basis and therefore it should be more efficient.

Recently, by using the algorithms F_4 and F_5, 80-bit HFE were first cryptanalyzed in [14], whereas the XL algorithm was not applicable to 80-bit HFE. Time results with an implementation under Magma are presented on A. Steel's web page (http://magma.maths.usyd.edu.au/users/allan/gb/). As we stated above, the F_4 and F_5 algorithms are Gröbner basis algorithms. Why did algebraic cryptanalysis based on these Gröbner basis algorithms exceed XL? We give an answer for this question in this article.

In this paper we clarify a relation between the XL algorithm and Gröbner basis algorithms. Moreover, we study the XL algorithm on semi-regular sequences, which correspond, according to a conjecture in a report [3], to almost all overdefined polynomial systems, and on the cryptosystem HFE.

More precisely, we show the following:

1. The XL algorithm does not introduce explicitly a monomial ordering. But we have proved that if the XL algorithm terminates, it will also terminate with a lexicographic ordering.
2. To solve a system of algebraic equations whose solution in a given finite field is unique amounts to nothing but to calculate the reduced Gröbner basis for the ideal associated with that system.
3. By 2, the XL algorithm is actually a Gröbner basis algorithm. Moreover it is *essentially* the same as the one treated in [17] and can be viewed as a redundant variant of a Gröbner basis algorithm F_4.
4. We study the XL algorithm on semi-regular sequences.
   On $\mathbb{F}_2$, that the degree D of the parameter needed for the XL algorithm is almost the same as the degree of the polynomials in the matrix constructed by the F_5 algorithm. But the complexity of these two algorithms is specified by the size of the matrix: for example, for a quadratic multivariate polynomials with $n = 128$ and $m = 130$, both algorithms reached the same degree 17 and the matrices generated by the XL algorithm will have about 170×10^{20} rows and 6×10^{20} columns compared to squared matrices with only 6×10^{20} rows and columns for the F_5 algorithm.

On the field $\mathbb{F}_q$, with q very large compared to n, we show the XL algorithm terminates for a degree higher than Gröbner basis algorithms with a DRL order. Then it is obvious that XL matrices are huge compared to F_5 matrices.

5. We complete this study on generic systems with a comparison of the XL algorithm and the Buchberger algorithm for a cryptosystem HFE. For this cryptosystem, a Gröbner basis algorithm finds a structure in the multivariate systems and never exceeds a low degree, whereas, for the XL algorithm, the degree seems to still increase with the number of variables n.

The XL algorithm was proposed to be a more efficient algorithm to solve a system of equations under a special condition without trying to calculate a whole Gröbner basis. But our results imply that the XL algorithm is not so efficient as it was expected to be.

In Section 2, we recall the description of the XL algorithm. In Section 3, we give an overview of the theory of Gröbner bases. In Section 4, we clarify a relation between the XL algorithm and the F_4 algorithm. In Section 5, we study the behavior of the XL algorithm on semi-regular sequences. In Section 6, we give experimental results on HFE systems and in Section 7, we conclude this report.

2 The Basic Principle of XL

The XL algorithm is given as an algorithm which solves systems of quadratic equations having a solution in k^n for a finite field $k = \mathbb{F}_q$. Let $\mathcal{A}$ be a system of multivariate equations $f_j = 0$, $(1 \leq j \leq m)$ for $f_j \in k[\mathbf{x}] := k[x_1, \ldots, x_n]$. We denote the ideal generated by all f_j in $\mathcal{A}$ by $\mathcal{I}_\mathcal{A}$. Then, XL is described as follows [20].

Algorithm 1 (The XL Algorithm). For a positive integer D, execute the following steps:

1. **Multiply:** Generate all the products $\prod_{j=1}^{r} x_{\ell_j} * f_i \in \mathcal{I}_\mathcal{A}$ with $r \leq D-2$ and total degree $\leq D$.
2. **Linearize:** Consider each monomial in the x_i of *degree* $\leq D$ as a new variable and perform the Gaussian elimination on the equations obtained in Step 1. The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.
3. **Solve:** Assume that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm).
4. **Repeat:** Simplify the equations and repeat the process to find the values of the other variables.

In the original definition of the XL algorithm in [20], only quadratic equations are treated. If we change the condition "with $r \leq D-2$ and total degree $\leq D$"

in Step 1 to "with $r \leq D - \deg(f_i)$", we can apply XL to a system of equations including a non-quadratic equation. Note that this change does not contradict the original XL setting when a system of equations consists of quadratic equations. So hereafter, we use this generalized version in order to work in general case.

Remark 1. We can replace Step 1 of the XL algorithm by considering f_i^* the *homogenization* of f_i: $f_i^* = Z^d f(\frac{x_1}{Z}, \ldots, \frac{x_n}{Z}) \in k[\mathbf{x}, Z]$ and products mf_i^* with m a monomial with degree $D - \deg(f_i^*)$. All the computation is exactly the same. So the behavior of XL is the same on the homogenization of the system $\mathcal{A}$ as on $\mathcal{A}$. We will use this remark on section 5, and for more properties of homogenization, we refer to [4].

3 Gröbner Basis and Some Algorithms

3.1 Basic Notation and Definitions

Let $k[\mathbf{x}] = k[x_1, \ldots, x_n]$ be a polynomial ring with variables $x_1, \ldots, x_n$ over a field k. For a monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, $|\alpha| := \sum_{i=1}^n \alpha_i$ is called the *total degree* of this monomial. In the following, the set of all monomials in variables $x_1, \ldots, x_n$ is denoted by $M(x_1, \cdots, x_n)$, or simply by M. In the theory of Gröbner bases, we need to consider a *monomial ordering* (cf. [10]). One of such ordering is the *degree reverse lexicographical order* (DRL) defined as follows:

Definition 1 (cf. [14]). *For $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}_{\geq 0}^n$, We say $\mathbf{x}^\alpha >_{DRL} \mathbf{x}^\beta$ if $|\alpha| = \sum_{i=1}^n \alpha_i > |\beta| = \sum_{i=1}^n \beta_i$, or $|\alpha| = |\beta|$ and the right-most nonzero entry of the vector $\alpha - \beta \in \mathbb{Z}^n$ is negative.*

There are many monomial orderings. We choose one of such orderings on T and write it as $<$.

A nonzero polynomial f in $k[\mathbf{x}]$ is written as $f = \sum_\alpha c_\alpha \mathbf{x}^\alpha$, $c_\alpha \neq 0$. We use the following notations:

$T(f) = \{c_{(\alpha_1, \cdots, \alpha_n)} x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid c_{(\alpha_1, \cdots, \alpha_n)} \neq 0\}$: the set of *terms* of f
$M(f) = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} \mid c_{(\alpha_1, \cdots, \alpha_n)} \neq 0\}$: the set of *monomials* of f

We denote the *total degree*, the *leading term*, the *leading coefficient* and the *leading term* with respect to $<$, by $\deg(f)$, $\mathrm{LM}(f)$, $\mathrm{LC}(f)$ and $\mathrm{LT}(f)$ respectively. (For each definition, see [10].)

The ideal in $k[\mathbf{x}]$ generated by a subset F is denoted by $\langle F \rangle$. We also denote by $\langle I_1, \ldots, I_n \rangle$ the minimal ideal containing ideals $I_1, \ldots, I_n$.

Under the above notation, a *Gröbner basis* is defined as follows.

Definition 2. *Let M be the set of all monomial of $k[\mathbf{x}]$ with a fixed ordering. A finite subset $G = \{g_1, \ldots, g_m\}$ of an ideal $\mathcal{I}$ is called a Gröbner basis if*

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_m) \rangle = \langle \mathrm{LT}(\mathcal{I}) \rangle.$$

For a given ideal $\mathcal{I}$, its Gröbner basis is not unique. But the *reduced Gröbner basis*, which is defined as follows, is uniquely determined.

Definition 3. *A Gröbner basis* $G = \{f_1, \ldots f_m\}$ *of an ideal* $\mathcal{I}$ *is called reduced Gröbner basis if for all* i, $\mathrm{LC}(f_i) = 1$ *and any monomial of* f_i *is not divisible by any element of* $\mathrm{LM}(G \backslash \{f_i\})$.

3.2 The Buchberger Algorithm

An algorithm which calculates a Gröbner basis is called a *Gröbner basis algorithm*. The Buchberger algorithm is one of them.

Definition 4. *Let* $f, g \in k[\mathbf{x}]$ *be nonzero polynomials. The S-polynomial of* f *and* g *is the combination*

$$S(f,g) := \mathrm{LC}(g)\frac{\mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LM}(f)} f - \mathrm{LC}(f)\frac{\mathrm{lcm}(\mathrm{LM}(f), \mathrm{LM}(g))}{\mathrm{LM}(g)} g.$$

For a finite set G of polynomials in $k[\mathbf{x}]$ and a polynomial $f \in k[\mathbf{x}]$, we denote by $\bar{f}^G$, a remainder of f on division by G. (For the definition of division by a finite set of polynomials, see [10] for example.)

Theorem 1. *A basis* $G = \{g_1, \ldots g_m\}$ *of an ideal* $\mathcal{I}$ *in* $k[\mathbf{x}]$ *is a Gröbner basis if and only if for all pairs* $i \neq j$, $\overline{S(g_i, g_j)}^G = 0$.

As a result of Theorem 1, we have the *The Buchberger algorithm*:

Algorithm 2 (The Buchberger Algorithm).

Input: an ordered set $F = (f_1, \ldots, f_m)$ in $k[\mathbf{x}]$
Output: a Gröbner basis $G = \{g_1, \ldots, g_s\}$ for $I = \langle f_1, \ldots, f_m \rangle$ with $F \subset G$
$G := F$
Repeat
  $H := G$
  For each pair (p, q), $p \neq q$ in H,
    If $S := \overline{S(p,q)}^H \neq 0$, Then $G := G \cup \{S\}$
Until H=G

We remark that the reduced Gröbner basis is calculated in a finite number of steps from a Gröbner basis.

3.3 Some Other Algorithms

D. Lazard in the articles [17] describes a relationship between the method of the computation of Gröbner bases and the one based on Gaussian Eliminations on matrix for the system $\mathcal{A}$. Moreover there are some other Gröbner basis algorithms based on Gaussian elimination: F_4 [11], $FGLM$ [13] and F_5 [12]. We explain now the relationship between polynomials and matrices.

For a system $\mathcal{A}$ of equations $f_j = 0$ $(j = 1, 2, \ldots, m)$, let us consider a finite list $G = (g_1, \ldots, g_m)$ of elements of the ideal generated by f_j, the ordered set $M_G = [t_1, \ldots, t_l]$ of monomials of all g_i with respect to a fixed order $<$. A matrix A whose (i, j)-entry is given as the coefficient of t_j in g_i is called the *coefficient*

matrix of G. Note that ${}^tG = A\,{}^tM_G$ where ${}^tG$ and ${}^tM_G$ mean the transpose of each. Let $\tilde{A}$ be the row echelon form of A obtained by using elementary row operations in a standard linear algebra[1]. Then we call $\tilde{G}$ given by ${}^t\tilde{G} := \tilde{A}\,{}^tM_G$ the *row echelon basis* of G. When we take the reduced row echelon form of G, we say $\tilde{G}$ *the reduced row echelon basis* of G (In [11], this is called the row echelon basis). Calculation of the reduced row echelon basis is an essential part of F_4.

4 Relation Between XL and Gröbner Basis Algorithms

4.1 The Choice of a Monomial Ordering

To compare the XL algorithm with Gröbner basis algorithms, we need to give an explicit monomial ordering for XL. As the XL algorithm does not give an explicit monomial ordering, we need to introduce the following lemma :

Lemma 1. *Let $\mathcal{A}$ be a system of m multivariate equations with n variables.*

$$\textit{XL terminates for a degree } D \iff \begin{array}{l}\textit{XL terminates for a degree } D\\ \textit{with the Lexicographic ordering}\end{array}$$

Proof. Let be M (respect. M') the coefficient matrix of the list $\{(\prod_{j=1}^{k} x_{i_j}) * f_i\}$ with $k \leq D - \deg(f_i)$ for XL (respect. with the Lexicographic ordering). So we can write $M = \left(A \middle| B\right)$ and $M' = \left(A' \middle| B'\right)$ such that B (respect. B') corresponds to the columns for the univariate monomials. Moreover M', A' and B' are only column permutations of M, A and B.

If XL terminates for a degree D, it means that $\mathrm{rank}(M) > \mathrm{rank}(A)$. Then $\mathrm{rank}(M') > \mathrm{rank}(A')$ and then XL will find an univariate polynomial with the lexicographic ordering. □

4.2 Pre-assumption of the XL Algorithm

Let $k = \mathbb{F}_q$ be a finite field with q elements and let $\mathcal{A}$ be a system of multivariate equations $f_j = 0$ $(1 \leq j \leq m)$ where $f_j \in k[x_1, \ldots, x_n]$. As stated *implicitly* in the introduction of [20], XL was proposed to be an efficient algorithm to solve a system of multivariate equations satisfying the following condition.

Condition 1. The system $\mathcal{A}$ has only one solution $(x_1, \ldots, x_n) = (a_1, \ldots, a_n)$ in k^n. (i.e. $\mathcal{A}$ has a solution $(a_1, \ldots, a_n)$ in k^n and no other solution in k^n.)

Note that the system $\mathcal{A}$ under Condition 1 can have another solution in K^n for some extension field $K(/\!\!\neq k)$ of k. To determine the solution in k^n, we need extra equations $x_i^q - x_i = 0$ $(i = 1, \ldots, n)$. Thus the ideal we have to consider is generated by f_j $(j = 1, \ldots, m)$ and $x_i^q - x_i$ $(i = 1, \ldots, n)$. We denote this ideal by $\widetilde{\mathcal{I}}_{\mathcal{A}}$. Then we have the following important theorem.

[1] This procedure is so-called the Gaussian elimination.

Theorem 2. *Let $\mathcal{A}$ be a system of multivariate equations $f_j = 0$, $j = 1, 2, \ldots, m$ in $k[x_1, \ldots, x_n]$ with $k = \mathbb{F}_q$. Let $\widetilde{\mathcal{I}}_{\mathcal{A}}$ be the ideal $\langle f_1, \ldots, f_m, x_1^q - x_1, \ldots, x_n^q - x_n \rangle$. Then a solution $(x_1, \ldots, x_n) = (a_1, \ldots, a_n) \in k^n$ of $\mathcal{A}$ is unique in k^n if and only if $\widetilde{\mathcal{I}}_{\mathcal{A}} = \langle x_1 - a_1, \ldots, x_n - a_n \rangle$.*

Proof. If $(x_1, \ldots, x_n) = (a_1, \ldots, a_n)$ is a unique solution in k^n of $\mathcal{A}$, $\widetilde{\mathcal{I}}_{\mathcal{A}} \subset \langle x_1 - a_1, \ldots, x_n - a_n \rangle$ and $(a_1, \ldots, a_n)$ is a unique solution in $\bar{k}^n$ of a system which consists of $f_j = 0$ $(j = 1, \ldots, m)$ and $x_i^q - x_i = 0$ $(i = 1, \ldots, n)$ for an algebraic closure $\bar{k}$ of k because $x_i^q - x_i = 0$ has solutions only in k. Then from Hilbert's Nullstellensatz (cf. [10]), for each $i = 1, \ldots, n$, there exists a positive integer ℓ_i such that $(x_i - a_i)^{\ell_i} \in \widetilde{\mathcal{I}}_{\mathcal{A}}$. Since $x_i - a_i = \gcd(x_i^q - x_i, (x_i - a_i)^{\ell_i}) \in \widetilde{\mathcal{I}}_{\mathcal{A}}$, we have $\widetilde{\mathcal{I}}_{\mathcal{A}} = \langle x_1 - a_1, \ldots, x_n - a_n \rangle$. For the converse, it is obvious. □

By this theorem, Condition 1 is equivalent to the following condition.

Condition 2. The reduced Gröbner basis with respect to DRL of the ideal $\widetilde{\mathcal{I}}_{\mathcal{A}} = \langle f_1, \ldots, f_m, x_1^q - x_1, \ldots, x_n^q - x_n \rangle$ is $\{x_1 - a_1, \ldots, x_n - a_n\}$.

Thus the problem to solve $\mathcal{A}$ defined over $k = \mathbb{F}_q$ under the Condition 1 coincides with the problem to calculate the reduced Gröbner basis of the ideal generated by equations in $\mathcal{A}$ and field equations $x_i^q - x_i = 0$ under the Condition 2, which is not a new problem. In particular, if the XL algorithm can solve a system $\mathcal{A}$ of algebraic equations over $\mathbb{F}_q$ under the Condition 1, it actually computes the reduced Gröbner basis of the ideal $\widetilde{\mathcal{I}}_{\mathcal{A}}$.

4.3 Relation Between XL and the F_4 Algorithm

We use the same notation as in (3.1). Here we show the XL algorithm gives a Gröbner basis algorithm which can be viewed as a redundant variant of the F_4 algorithm. (For the description of the original F_4, see [11].) To give such a description, we need the following definition.

Definition 5. *(1) A critical pair of two polynomials (f_i, f_j) is an element of $M^2 \times k[\mathbf{x}] \times M \times k[\mathbf{x}]$, $Pair(f_i, f_j) := (lcm_{ij}, t_i, f_i, t_j, f_j)$ such that*

$$lcm(Pair(f_i, f_j)) = lcm_{ij} = \mathrm{LM}(t_i f_i) = \mathrm{LM}(t_j f_j) = lcm(\mathrm{LM}(f_i), \mathrm{LM}(f_j)).$$

(2) For a critical pair $p_{ij} = Pair(f_i, f_j)$, $\deg(lcm_{ij})$ is called the degree of p_{ij} and denoted by $\deg(p_{ij})$. Let P be a list of critical pairs. For $p = Pair(f, g) \in P$ and $d \in \mathbb{N}$, we define two functions $XLLeft(p, d) = \{(t, f) | t \in M, \deg(t * f) \leq d\}$, and $XLRight(p, d) = \{(t, g) | t \in M, \deg(t * g) \leq d\}$. We write $XLLeft(P, d) = \bigcup_{p \in P} XLLeft(p, d)$ and $XLRight(P, d) = \bigcup_{p \in P} XLRight(p, d)$.

For a list of critical pairs P and a positive integer $d \in \mathbb{N}$, we set

$$Sel(P, d) := \{p \in P \mid \deg(lcm(p)) \leq d\}.$$

Now we give an F_4-like description of the XL algorithm.

Algorithm 3 (The XL Algorithm).

Input: $\begin{cases} F : \text{a finite subset of } k[\mathbf{x}] \\ Sel : \text{fixed as above.} \end{cases}$
Output: a finite subset of $k[\mathbf{x}]$.
$G := F$, $\tilde{F}_0^+ := F$ and $d := 0$
$P := \{Pair(f, g) | f, g \in G \text{ with } f \not= g\}$
While $P \not= \phi$ Do
  $d := d + 1$
  $L_d := \text{XLLeft}(P, d) \cup \text{XLRight}(P, d)$
  $P_d := Sel(P, d)$
  $P := P \setminus P_d$
  $\tilde{F}_d^+ := \text{Reduction}(L_d)$
  For $h \in \tilde{F}_d^+$ Do
    $P := P \cup \{Pair(h, g) | g \in G\}$
    $G := G \cup \{h\}$
Return G

Reduction
  Input: a finite subset L of $M \times k[\mathbf{x}]$
  Output: a finite subset of $k[\mathbf{x}]$ (possibly an empty set).
    $F :=$ Symbolic Preprocessing(L)
    $\tilde{F} :=$ Reduction to Row Echelon Basis of F w.r.t. $<$
    $\tilde{F}^+ := \{f \in \tilde{F} | \text{LM}(f) \not\in \text{LM}(F)\}$
  Return $\tilde{F}^+$

Symbolic Preprocessing
  Input: a finite subset L of $M \times k[\mathbf{x}]$
  Output: a finite subset of $k[\mathbf{x}]$
    $F := \{t * f \mid (t, f) \in L\}$
  Return F

Remark 2. In the original description of XL, it seems that the bound D is taken globally at once. However, to implement XL, there seems to be the following four ways to realize the process determining the optimal value of D. Let $\mathcal{A}$ be a system of equations you want to solve. Then each way is described as follows.

1. Begin with $D = 1$. Do XL described as in Definition 1 for $\mathcal{A}$. If you cannot obtain the solution, set $D := D + 1$ and do XL again for $\mathcal{A}$ with the new D.
2. Begin with $D = 1$. Iterate 'Multiply' and 'Linearize' described as in Definition 1 for $\mathcal{A}$ by adding new equations obtained by 'Linearize' to $\mathcal{A}$. If you cannot solve the resulting system, then return to the original $\mathcal{A}$, set $D := D + 1$ and iterate the same procedure as for $D = 1$. Repeat until you obtain the solution.
3. Begin with $D = 1$. Do XL described as in Definition 1 for $\mathcal{A}$. If you cannot obtain the solution, then set $D := D + 1$, replace $\mathcal{A}$ by the resulting system obtained by 'Linearize' in the previous XL and do XL again for the new $\mathcal{A}$ and D. Repeat until you obtain the solution.

4. Begin with $D = 1$. Iterate 'Multiply' and 'Linearize' described as in Definition 1 for $\mathcal{A}$ by adding new equations obtained by 'Linearize' to $\mathcal{A}$. If you cannot solve the resulting system $\mathcal{A}'$, then replace $\mathcal{A}$ by $\mathcal{A}'$, set $D := D + 1$ and iterate the same procedure as for $D = 1$. Repeat until you obtain the solution.

The first two processes are slightly different from the others. The degree reached for the third and the fourth ones can be lower than the degree of the others. The Gaussian elimination of polynomials with degree D can give polynomials with lower or equal to $D - 1$. For example, let us consider the system $x_2^2 + x_3 = 0$, $x_1x_2 - x_2 = 0$, $x_3^3 + x_1 = 0$. For $D = 3$, the polynomial $x_3x_1 - x_3 = (x_1 - 1)(x_2^2 + x_3) - x_2(x_1x_2 - x_2)$ appear in resulting system obtained by 'Linearize', and then for $D = 4$, the third and fourth methods find the univariate polynomial $x_1^2 - x_1 = (x_1 - 1)(x_3^3 + x_1) - x_3^2(x_3x_1 - x_3)$. Whereas, the two first methods need a degree $D = 5$ to find this polynomial because $x_1^2 - x_1 = (x_1 - 1)(x_3^3 + x_1) - (x_3^2x_1 - x_3^2)(x_2^2 + x_3) + x_3^2x_2(x_1x_2 - x_2)$.

In the above description of XL, we take the third one. You may take one of the other three realizations but the rest of our result holds for all of them. We should remark that XL taking D as in the first one is *essentially* the same as the Gröbner basis algorithm treated in [17].

In the above description of the XL algorithm, we keep some redundancy in the description to show the similarity to the F_4 algorithm. Note that in algebraic attacks using XL, the input F should be a set of polynomials which comes from all equations in a given system of equations $\mathcal{A}$ whose solution in k^n is unique and all field equations $x_i^q - x_i = 0$ for all variables x_i. 'Multiply' in XL corresponds to the calculation of L_d and "Symbolic Preprocessing". And 'linearize' corresponds to "Reduction". Note that, XL in the above description can be viewed as a redundant variant of F_4. This is because XLLeft and XLRight collect more polynomials and therefore the set of polynomials constructed in "Symbolic Preprocessing" is much larger than the one in F_4. In fact, XL collects all the products $\prod_{j=1}^{r} x_{l_j} * f_i$ with $r \leq D - \deg(f_i)$, whereas F_4 collects only polynomials needed in the Gaussian elimination.

The above description enables us to prove the following theorem.

Theorem 3. *Let F be a finite set of polynomials in $k[\mathbf{x}]$. Then Algorithm 3 computes a Gröbner basis G for the ideal $\langle F \rangle$ in $k[\mathbf{x}]$ such that $F \subseteq G$.*

Proof. Let d be a positive integer and G_d the set G obtained for that d in the while-loop. If $\tilde{F}_d^+ \neq \emptyset$, then $\deg h \leq d$ for any $h \in \tilde{F}_d^+$ and hence $h \in L_{d+1}$ in the next loop. Then it is obvious that $h \notin \tilde{F}_{d+1}^+$. Since any $g \in G_{d-1}$ of $\deg g \leq d$ is contained in L_d, $h \notin G_{d-1}$ for any $h \in \tilde{F}_d^+$ and hence we have $G_{d-1} \subsetneq G_d$ when $\tilde{F}_d^+ \neq \emptyset$.

First, we show that Algorithm 3 terminates in a finite number of steps. Suppose that Algorithm 3 does not terminate. Then there is an infinite sequence (d_i) of positive integers such that $d_i < d_{i+1}$ and $\tilde{F}_{d_i}^+ \neq \emptyset$ for all i. From the

above observation, we have an infinite ascending chain $G_{d_i} \subsetneq G_{d_{i+1}} \subsetneq \cdots$. But it contradicts to the fact that the ring $k[\mathbf{x}]$ is noetherian.

Now we show the output G of Algorithm 3 is actually a Gröbner basis of $\langle F \rangle$. Since $G = \bigcup_{d \geq 0} \tilde{F}_d^+$ and $\tilde{F}_d^+ \subset \langle F \rangle$, we have $F \subset G \subset \langle F \rangle$. The remaining task is to show $\overline{S(f,g)}^G = 0$ for all $f \neq g$ in G. Put $\tilde{d} := \deg(Pair(f,g))$. Then the S-polynomial $S(f,g)$ is contained in $L_{\tilde{d}}$ and hence $\overline{S(f,g)}^{G_{\tilde{d}}} = 0$. In particular, we obtain $\overline{S(f,g)}^G = 0$. Thus, by Theorem 1, the output G is actually a Gröbner basis of $\langle F \rangle$. □

5 Semi-regular Sequences

In this section, we try to give a bound on the matrix size of the XL algorithm compared to the matrix size of the F_5 algorithm for most polynomial systems.

5.1 Presentation of Semi-regular Sequences

In the report [3], the notion of semi-regular sequences was presented for overdefined systems over the finite field $\mathbb{F}_2$ and for affine systems. We have to distinguish two important cases for finite fields, $\mathbb{F}_2$ and $\mathbb{F}_q$. In the field $\mathbb{F}_2$, we have a criterion deduced from the Frobenius application. If we are interested in a system $\mathcal{A}$ on a field $\mathbb{F}_q$, with $q \gg n$, i.e. q is very high compared to n, then the trivial relation issued from the Frobenius application will not be reached during computation and all the computation done is similar to computation on $\mathbb{Q}$.

Definition 6.

Homogeneous Semi-regular Sequence: *Let $f_1, \ldots, f_m$ be a sequence of m homogeneous polynomials (i.e. for all monomial t of f_i, $\deg(t) = \deg(f_i)$ in $\mathcal{R}_n^h := \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2, \ldots, x_n^2 \rangle$ or $\mathbb{Q}[x_1, \ldots, x_n]$), and $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ an ideal of $\mathcal{R}_n^h$ or $\mathbb{Q}[x_1, \ldots, x_n]$.*

- *The* degree of regularity *of $\mathcal{I}$ is the minimal degree d such that $\{LT(f) \mid f \in \mathcal{I}, \deg(f) = d\}$ is exactly the set of monomials of degree d in $\mathcal{R}_n^h$, denoted by $D_{reg}(\mathcal{I})$.*
- *$f_1, \ldots, f_m$ is a* homogeneous semi regular sequence on $\mathbb{F}_2$ *if $\mathcal{I} \neq \mathcal{R}_n^h$ and for $i \in \{1, \ldots, m\}$, if $g_i f_i = 0$ in $\mathcal{R}_n^h/\langle f_1, \ldots, f_{i-1} \rangle$ and $\deg(g_i f_i) < D_{reg}(\mathcal{I})$ then $g_i = 0$ in $\mathcal{R}_n^h/\langle f_1, \ldots, f_{i-1}, f_i \rangle$.*
- *$f_1, \ldots, f_m$ is a* homogeneous semi regular sequence on $\mathbb{Q}$ *if $\mathcal{I} \neq \mathbb{Q}[x_1, \ldots, x_n]$ and for $i \in \{1, \ldots, m\}$, if $g_i f_i = 0$ in $\mathbb{Q}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1} \rangle$ and $\deg(g_i f_i) < D_{reg}(\mathcal{I})$ then $g_i = 0$ in $\mathbb{Q}[x_1, \ldots, x_n]/\langle f_1, \ldots, f_{i-1} \rangle$.*

Affine Semi-regular Sequence: *Let $f_1, \ldots, f_m$ be a sequence of m polynomials, and $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ an ideal of $\mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 - x_1, \ldots, x_n^2 - x_n \rangle$ or $\mathbb{Q}[x_1, \ldots, x_n]$. Let f_i^h the homogeneous part of the largest degree of f_i.*

- $f_1, \ldots, f_m$ *is a* semi regular sequence *if* $f_1^h, \ldots, f_m^h$ *is a homogeneous semi-regular sequence.*
- *the* degree of regularity *of* $\mathcal{I}$ *is the degree of regularity of* $\langle f_1^h, \ldots, f_m^h \rangle$*, denoted by* D_{reg}*.*

With this sequence of polynomials, the matrix generated by the F_5 algorithm has a full rank for the degree $d < D_{reg}$. Moreover, all polynomials computed by F_5 have a degree lower or equal to D_{reg}.

This means that, for semi-regular sequences, the number of rows $H_{m,n}(d)$ of the matrix in the homogeneous case, for $d < D_{reg}$, is known, and is given by a recurrence formula $H_{m,n}(d) = H_{m-1,n}(d) + \#\{m_\ell \text{ monomial of degree } d - d_m\} - H_{m,n}(d - d_m)$ with initial conditions $H_{m,n}(d) = 0$ if $m \leq 0$ or $d < min(\deg(f_k) \,|\, k \leq m\}$. Then the number of rows of a matrix for the affine case is $\sum_{d'=1}^{d} H_{m,n}(d')$.

The degree D_{reg} corresponds to the degree d when we will have more rows than columns for the homogeneous part of the largest degree. It is the minimal degree such that $H_{m,n}(d) > \#\{m_\ell \text{ monomial of degree } d\}$. If we consider the series $f(y) = \sum_{d \geq 0}(H_{m,n}(d) - \#\{m_\ell \text{ monomial of degree } d\})y^d$, the degree D_{reg} is given when the coefficient of this series is negative. the expression of f for quadratic equations is:

$\frac{(1+y)^n}{(1+y^2)^m}$ for $\mathbb{F}_2$ $\qquad$ $\frac{(1-y^2)^m}{(1-y)^n}$ for $\mathbb{F}_q$, with $q \gg n$.

Moreover, in the article [3], the authors have made a conjecture verified on many computer experiments:

Conjecture 1. almost all polynomial systems are semi-regular sequences.

As the XL algorithm computes for an homogeneous system, we work on semi-regular sequences such that the homogenization of the sequences is still semi-regular. With these hypotheses, the conjecture is still true.

If we want to find an univariate polynomial for the original description of XL, we need to have a number of rows higher than the number of monomials with degree D minus the number of univariate monomials in X_1 (i.e., X_1 and 1 for $\mathbb{F}_2$ and $1, \ldots, X_1^D$, for $\mathbb{F}_q$).

This means that the degree D of the XL algorithm is given when the coefficient of this series is negative. the expression of f for quadratic equations is :

$\frac{(1+y)^n}{(1-y)(1+y^2)^m} - \frac{1+y}{1-y}$ for $\mathbb{F}_2$ $\qquad$ $\frac{(1-y^2)^m}{(1-y)^{n+1}} - \frac{1}{(1-y)^2}$ for $\mathbb{F}_q$, with $q \gg n$.

5.2 On the Field $\mathbb{F}_2$

Figure 1(a) presents a comparison of the degree reached between the XL algorithm and Gröbner basis computation for a variation of the number of variables n and Figure 1(b) for a variation of the number of equations m.

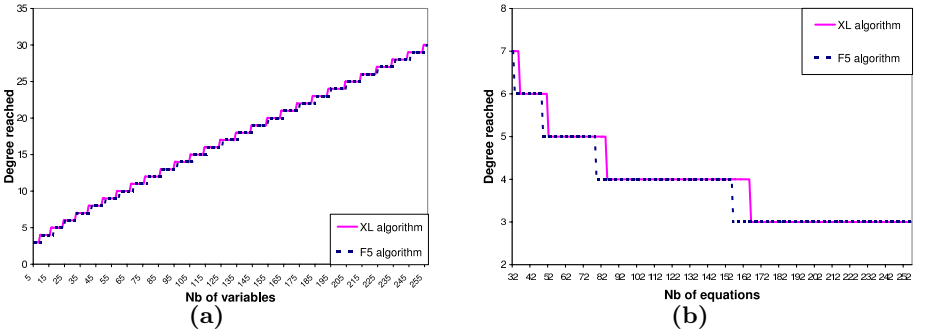


Fig. 1. Behavior of the XL algorithm and the F_5 algorithm on $\mathbb{F}_2$

With these figures, we do not have a noticeable difference between the degree reached by the two algorithms. So we can say that for random systems, the methods of XL and Gröbner basis are almost the same.

For the complexity point of view, if N_D is the size of the matrix constructed, then the whole complexity is the cost of linear algebra on this matrix, which is N_D^w where $w \leq 3$ is the coefficient of linear algebra. The XL algorithm creates matrices with $\sum_{i=1}^{m} \sum_{k=0}^{D-\deg(f_i)} \binom{n}{k}$ rows and $\sum_{k=0}^{D} \binom{n}{k}$ columns, whereas F_5 creates square matrices with $\sum_{k=0}^{D} \binom{n}{k}$ columns.

So the number of columns for F_5 algorithm matrices is lower or equal to the one for XL algorithm matrices whereas the number of rows of the matrices constructed is very different, Figure 2 presents the number of rows of each matrices with a logarithm scale. As we can see, the difference between the two curves gives us a multiplicative constant.

5.3 On the Field $\mathbb{F}_q$, with q Large

Figure 3(a) presents a comparison of the degree reached between the XL algorithm and Gröbner basis computation for a variation of the number of variables n with $m = n + 2$ and Figure 3(b) for a variation of the number of equations m. First we can see that for random polynomials we have always computed a

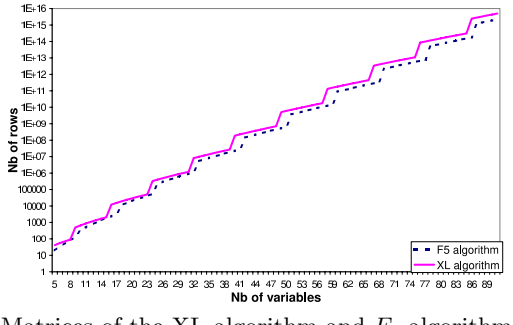


Fig. 2. Matrices of the XL algorithm and F_5 algorithm on $\mathbb{F}_2$

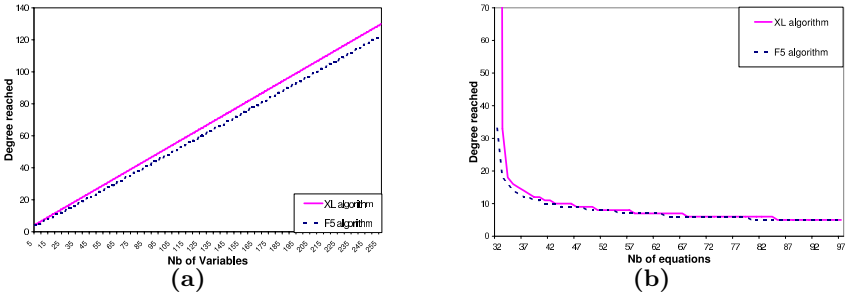


Fig. 3. Behavior of the XL algorithm and the F_5 algorithm on $\mathbb{F}_q$

Gröbner basis before finding the univariate polynomial for a degree D. Moreover, we can see the behavior of the degree of the XL algorithm does not seem to follow the formula $\frac{n}{\sqrt{m}}$ as it was said in [6].

As the complexity is N_D^w, where N_D is the size of the matrix constructed and w the coefficient of linear algebra and the XL algorithm has a higher degree D than the F_5 algorithm, the difference of the size of constructed matrices is very important. For example, for quadratic multivariate polynomials with $n = 128$ and $m = 130$, the XL algorithm reached a degree 66 whereas the F_5 algorithm reached a degree 61. So the matrices generated by the XL algorithm will have about 94317×10^{49} rows and 6332×10^{49} columns compared to squared matrices with only 8.4×10^{49} rows and columns for the F_5 algorithm.

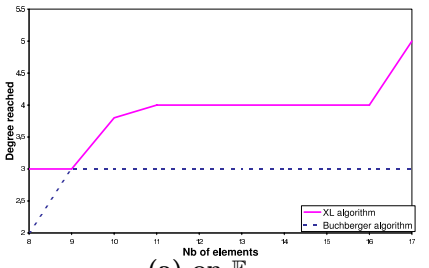
For the case $m = n$, the number of solutions with multiplicity of a random system with quadratic equations is $\prod_{i=1}^{m} \deg(f_i) = 2^n$, which is the Bezout bound. So the univariate polynomial has this degree and XL will terminate for this degree. Whereas, the computation of the Gröbner basis will not exceed $1 + \sum_{i=1}^{n} (\deg(f_i) - 1) = n + 1$ for any ordering. This computation is done with a DRL ordering and then we use the FGLM algorithm [13, 10] to find the wanted ordering.

All this study is still true if $D < q$ and not only for $q \gg n$.

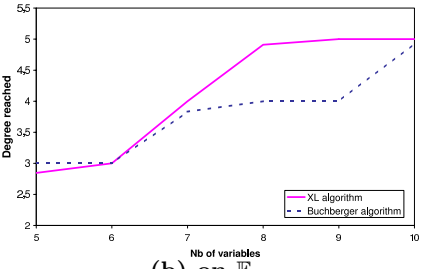
6 Example on HFE Systems

In cryptography, the systems studied seem to be random but have a structure behind them. So we need to make experimental tests on cryptosystems to have an idea of the efficiency of both algorithms.

Hidden Field Equations (HFE) is an asymmetric cryptosystem. It does not use the number theory but it is based on multivariate polynomials over a finite field (cf [18]). The idea of HFE is to take a secret univariate polynomial (the private key) on an extension of the finite field, then to express this polynomial on the finite field. We thus obtain an algebraic system (the public key). This system is composed with polynomials of degree 2.



(a) on $\mathbb{F}_2$



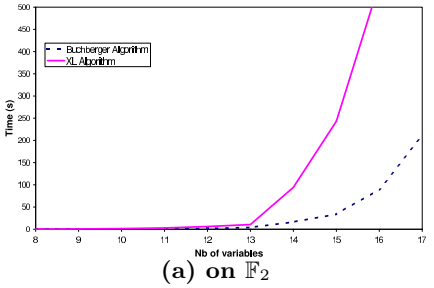
(b) on $\mathbb{F}_{16}$

Fig. 4. Comparison between XL and Gröbner algorithms on HFE

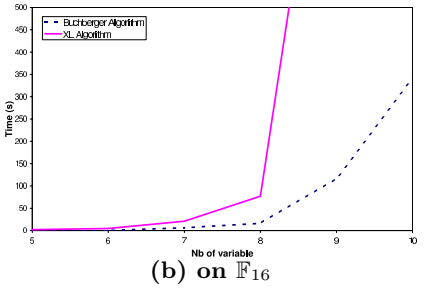
We have implemented the XL algorithm in Magma to test on the examples. Moreover as the XL algorithm has a better behavior for $m > n$, we have fixed some variables to be in the case $m = n + 2$. We studied on both cases presented in section 5, for the field $\mathbb{F}_2$, we use secret polynomials with degree 17 and with degree 24 for the field $\mathbb{F}_{16}$.

With Figure 4(a), we see that the XL algorithm's maximal degree increases whereas for Gröbner basis computation, the degree of resolution does not change and does not exceed 3. In fact, the XL algorithm seems to follow Figure 1(a). So XL does not seem to find a difference between a random system and the HFE cryptosystem contrary to Gröbner basis computation.

Figure 4(b) confirms that the Buchberger algorithm is still better than the XL algorithm on a bigger field for a number of elements higher than 6.



(a) on $\mathbb{F}_2$



(b) on $\mathbb{F}_{16}$

Fig. 5. Time comparison between XL and Gröbner algorithms on HFE

We present then time computation on figure 5. For the XL algorithm, the main part of computation is done in the Gaussian elimination and not in the other part of the algorithm. As we can see, the Buchberger algorithm has a better behavior than the XL algorithm.

7 Conclusion

In this paper, we compared the XL algorithm with Gröbner basis algorithms. First, we showed that to solve a system of algebraic equations treated in XL is equivalent to calculate the reduced Gröbner basis of the ideal associated with the system. Moreover we showed that the XL algorithm is also a Gröbner basis algorithm which can be represented as a redundant variant of a Gröbner basis algorithm F_4. Then we compared these algorithms on semi-regular sequences in two cases: in the fields $\mathbb{F}_2$ and $\mathbb{F}_q$ with $q \gg n$. We showed that the size of the matrix constructed by XL is huge compared to the ones of F_5 algorithm. We gave an experimental study between XL and Buchberger algorithms on the cryptosystem HFE and found that the Buchberger algorithm had a better behavior. Our results imply that the XL algorithm is not so efficient as it was expected.

References

1. F. Armknecht, M. Krause, "Algebraic Attacks on Combiners with Memory", Crypto 2003, LNCS 2729, pp. 162-176, Springer.
2. G. Ars and J.-C. Faugère. "Comparison of XL and Gröbner Basis Algorithms over Finite Fields." , Technical report, INRIA Rocquencourt, 2004.
3. M. Bardet, J.-C. Faugère, and B. Salvy. "Complexity of Gröbner basis computation for semi-regular sequences over $\mathbb{F}_2$ with solutions in $\mathbb{F}_2$." , Technical report, INRIA Rocquencourt, 2003.
4. T. Becker and V. Weispfenning. "Gröbner Basis : A Computational Approach to Commutative Algebra", Springer-Verlag, New York, 1993.
5. N. Courtois, "The security of Hidden Field Equations (HFE)", Cryptographers' Track RSA Conference 2001, San Francisco 8-12 April 2001, LNCS 2020, Springer, pp. 266-281.
6. N. Courtois and J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations", Asiacrypt 2002, LNCS 2501, Springer.
7. N. Courtois, "Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt", ICISC 2002, LNCS 2587, Springer.
8. N. Courtois and W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback", Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345-359, Springer.
9. N. Courtois, "Fast Algebraic Attacks on Stream Ciphers with Linear Feedback", Crypto 2003, LNCS 2729, Springer.
10. D. Cox, J. Little, and D. O'Shea, "Using Algebraic Geometry", Springer-Verlag, New York, 1998.
11. J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases (F_4)", Journal of Pure and Applied Algebra 139 (1999) pp. 61-88.
12. J.-C. Faugère, "A new efficient algorithm for computing Gröbner basis without reduction to zero (F_5)", In T. Mora, editor, Proceeding of ISSAC, pages 75-83, ACM Press, July 2002.
13. J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. "Efficient computation of zero-dimensional Gröbner bases by change of ordering". Journal of Symbolic Computation, 16(4):329–344, 1993.
14. J.-C. Faugère and A. Joux, "Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner bases", Crypto 2003, LNCS 2729, pp. 44-60, Springer.

15. A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced Oil and Vinegar Signature Schemes", Eurocrypt 1999, Springer-Verlag, pp. 216-222.
16. A. Kipnis and A. Shamir, "Cryptanalysis of the HFE Public Key Cryptosystem", Proceedings of Crypto'99, Springer-Verlag.
17. D. Lazard, "Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations", Computer algebra (London, 1983), LNCS 162, pp. 146–156, Springer.
18. J. Patarin, "Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms", Lecture Notes in Computer Science, 1070:33–48, 1996.
19. J. Patarin, "Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88", Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.
20. A. Shamir, J. Patarin, N. Courtois, and A. Klimov, "Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations", Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
21. M. Sugita and H. Imai, "Relations between Algebraic Attacks and Gröbner Base Algorithms", In The 2004 Symposium on Cryptography and Information Security, Japan – SCIS 2004, Jan.27–Feb.30, 2004.
22. M. Sugita, M. Kawazoe and H. Imai, "Relation between XL Algorithm and Gröbner Bases Algorithms", Cryptology ePrint Archive, Report 2004/112, 2004, http://eprint.iacr.org/.

Generic Homomorphic Undeniable Signatures

Jean Monnerat* and Serge Vaudenay**

EPFL, Switzerland
http://lasecwww.epfl.ch

Abstract. We introduce a new computational problem related to the interpolation of group homomorphisms which generalizes many famous cryptographic problems including discrete logarithm, Diffie-Hellman, and RSA. As an application, we propose a generic undeniable signature scheme which generalizes the MOVA schemes. Our scheme is generic in the sense that we transform a private group homomorphism from public groups G to H (the order of H being public) into an undeniable signature scheme. It is provably secure in the random oracle model provided that the interpolation problem is hard and it offers the advantage of making the signature size arbitrarily short (depending on a security level). We (im)prove some security results from MOVA. We also propose a new example with complexity similar to RSA and with 3-byte signatures.

1 Introduction

An undeniable signature scheme is similar to a classical digital signature except that the recipient of a message cannot verify its validity alone: he needs to interact with the signer in order to be convinced of the validity of the signature. This opposes to the so called universal verifiability of classical digital signatures where anybody knowing the signer's public key is able to verify the signature at any time. In some applications such as signing a contract, it is desirable to keep the signer's privacy by limiting the ability to verify this signature. However, an undeniable signature does not abandon the non-repudiation property. Indeed, in case of a dispute, the signer could be compelled by an authority to prove the invalidity of a signature, otherwise this would be considered as an attempt of denying a valid signature. An undeniable signature scheme is composed of a signature generation algorithm, a confirmation protocol to prove the validity of a signature, and a denial protocol to prove the invalidity of an invalid signature.

Since the invention of the first undeniable signature scheme proposed by Chaum and van Antwerpen [9], a certain amount of work has been dedicated to its development and different improvements [5, 7, 8, 11, 12]. Until the proposition

* Supported in part by a grant of the Swiss National Science Foundation, 200021-101453/1.

** Supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 354–371, 2004.

of an undeniable signature scheme based on RSA by Gennaro et al. [15], all previous undeniable signatures were based on the discrete logarithm problem. More recently, three undeniable signatures based on different problems have been proposed. The first one is based on pairings [18], the second one is based on a quadratic field [4], and the third one (MOVA) is based on characters [19].

In traditional digital signature schemes, the security collapses when the signature is too short because of universal verifiability: an attacker can try to guess a signature until it is valid in order to forge it. One advantage of undeniable signatures is that the security smoothly decreases with the signature length. As an example, we can think of 20-bit signatures which cannot be forged but with a probability of success of 2^{-20}. The forger can increase it in an on-line attack, but this can easily be detected and thwarted. So, undeniable signatures could in principle be arbitrarily small e.g. as small as a MAC, although no such signatures were proposed so far except MOVA signatures.

In this paper, we provide a new computational problem called Group Homomorphism Interpolation (GHI) problem whose solution consists in finding the image of a given point under an homomorphism which interpolates some given points. This generalizes and improves the MOVA scheme based on characters. Section 2 provides some theoretical results about the GHI problem. Section 3 contains several interactive proof protocols and some related security results that will be used for our undeniable signature from Section 4. Section 5 is devoted to a new example and further discussions. Finally, Section 6 concludes.

2 The Group Homomorphism Interpolation Problem

2.1 Problem Definitions

Given two Abelian groups G, H, and $S := \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq G \times H$, we say that the set of points S *interpolates in a group homomorphism* if there exists a group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for $i = 1, \ldots, s$. We say that a set of points $B \subseteq G \times H$ *interpolates in a group homomorphism with another set of points* $A \subseteq G \times H$ if $A \cup B$ interpolates in a group homomorphism. We state here the Group Homomorphism Interpolation problem (GHI problem) and its decisional problem (GHID problem).

S-GHI Problem (Group Homomorphism Interpolation Problem)
Parameters: two Abelian groups G and H, a set of s points $S \subseteq G \times H$.
Input: $x \in G$.
Problem: find $y \in H$ such that (x, y) interpolates with S in a group homomorphism.

S-GHID Problem (GHI Decisional Problem)
Parameters: two Abelian groups G and H, a set of s points $S \subseteq G \times H$.
Input: a point $(x, y) \in G \times H$.
Problem: does (x, y) interpolate with S in a group homomorphism?

We also consider the following problems.

d-MGGD Problem (Modular Group Generation Decisional Problem)
Parameters: an Abelian group G, an integer d.
Input: a set of values $S_1 = \{x_1, \ldots, x_s\} \subseteq G$.
Problem: does S_1 modulo dG span G/dG.

(d, S_1)-MSR Problem (Modular System Representation Problem)
Parameters: an Abelian group G, a set $S_1 = \{x_1, \ldots, x_s\} \subseteq G$, an integer d.
Input: $x \in G$.
Problem: find $a_1, \ldots, a_s \in \mathbf{Z}$ such that $x \in a_1x_1 + \cdots + a_sx_s + dG$.

d-Root Problem (dth Root Problem)
Parameters: an Abelian group G, an integer d.
Input: $x \in G$.
Problem: find $r \in G$ such that $x = dr$.

2.2 Preliminaries

Here is a first straightforward condition to solve the GHID problem.

Lemma 1. *Let G, H be two finite Abelian groups. We denote by d the order of H. The set $S = \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq G \times H$ interpolates in a group homomorphism if and only if for any $a_1, \ldots, a_s \in \mathbf{Z}$ such that $a_1x_1 + \cdots + a_sx_s \in dG$ we have $a_1y_1 + \cdots + a_sy_s = 0$.*

Let us now consider uniqueness criteria. We first notice that when the x-coordinates of points in S modulo dG generate G/dG (hence satisfy the MGGD problem), then there is at most one interpolating homomorphism. The following result says that this is a necessary condition as well.

Lemma 2. *Let G, H be two finite Abelian groups. We denote d the order of H. Let $x_1, \ldots, x_s \in G$ which span G'. The following properties are equivalent. In this case, we say that $x_1, \ldots, x_s$ H-generate G.*

1. *For all $y_1, \ldots, y_s \in H$, there exists at most one group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for all $i = 1, \ldots, s$.*
2. *There exists a unique group homomorphism $\varphi : G \longrightarrow H$ such that $\varphi(x_i) = 0$ for $i = 1, \ldots, s$, namely $\varphi = 0$.*
3. *The set $\mathrm{Hom}(G/G', H)$ of all group homomorphisms from G/G' to H is restricted to $\{0\}$.*
4. $\gcd(\#(G/G'), d) = 1$.
5. $G' + dG = G$.

Note that the criterion 4 suggests that H is only involved by the prime factors of its order. In what follows the smallest prime factor p will be important. Note that if $G = H$, these criteria mean that $x_1, \ldots, x_s$ generate G.

We can often meet the GHI and GHID problems in cryptography as the following examples suggest.

Example 1. We take a cyclic group G of order q, $H = \mathbf{Z}_q$, and a generator g of G. The set $S = \{(g, 1)\}$ interpolates in a unique group homomorphism, and the GHI problem is exactly the discrete logarithm problem.

Example 2. We take a cyclic group $G = H$, and a generator g of G. For any $a \in \mathbf{Z}$, $S = \{(g, ag)\}$ interpolates in a unique group homomorphism: the exponentiation to the power a. The GHI and GHID problems are exactly the Diffie-Hellman problem [13] and the Diffie-Hellman Decisional problem.

Example 3. Let $n = pq$ such that p, q are different odd primes and $H = \{-1, +1\}$. We let $x_1, x_2 \in \mathbf{Z}_n^*$ be such that x_1 is a quadratic residue modulo p and not modulo q, and that x_2 is a quadratic residue modulo q, and not modulo p. We notice that $S = \{(x_1, 1), (x_2, -1)\}$ interpolates in a unique group homomorphism which is $(\cdot/p)$. Since it is easy to compute $(\cdot/n)$, the quadratic residuosity problem [16] with the information x_1 and x_2 is equivalent to the GHI and GHID problems.

Example 4. Here, we consider the well known RSA cryptosystem [21]. Let $n = pq$ be an RSA modulus and $G = H = \mathbf{Z}_n^*$. Let $f : \mathbf{Z}_n^* \to \mathbf{Z}_n^*$ be defined by $f(x) = x^e \bmod n$ for an exponent e such that $\gcd(e, \varphi(n)) = 1$ [21]. Given enough many pairs $(x_i^e \bmod n, x_i) \in \mathbf{Z}_n^* \times \mathbf{Z}_n^*$, $i = 1, \ldots, s$, for the first coordinates to generate $\mathbf{Z}_n^*$, the RSA decryption problem is solved by a GHI oracle.This application of GHI problem to the decryption problem can be adapted to every homomorphic encryption scheme, e.g. Paillier [20].

Example 5. Given $d \in \{2, 3, 4\}$ and given an integer n such that d divides $\varphi(n)$, we let $G = \mathbf{Z}_n^*$ and $H = \mathbf{Z}_d$. The GHI problem is the MOVA^d problem [19].

Example 6. We show here how we can apply the GHI problem to the Bilinear Diffie-Hellman Problem (BDHP). Let $\hat{e} : \mathbf{G}_1 \times \mathbf{G}_1 \to \mathbf{G}_2$ be a bilinear, non-degenerate and computable mapping, where $\mathbf{G}_1$ and $\mathbf{G_2}$ are cyclic groups of order a large prime p. Let P be a generator of $\mathbf{G}_1$, we can state the BDHP as follows: given three random elements aP, bP and $cP \in \mathbf{G}_1$, compute $\hat{e}(P, P)^{abc}$. ($\mathbf{G_1}$ resp. $\mathbf{G}_2$ is written additively resp. multiplicatively.) BDHP is equivalent to GHI problem with $S = \{(P, \hat{e}(aP, bP))\}$ and $x_1 = cP$.

Note that Examples 2,3,4,5,6, include trapdoors in order to interpolate the group homomorphism. Except Examples 2,6, they further include trapdoors in order to solve the MSR problem. Also note that the order d of H is publicly known in Examples 1,2,3,5,6. It can further be quite small in Examples 3,5. In what follows we focus on publicly known d and on trapdoor homomorphisms. We will also consider the following example inspired by [1].

Example 7. Let $n = pq$ such that $p = rd + 1$ and q are prime, $\gcd(r, d) = 1$, $\gcd(q - 1, d) = 1$, with d small prime. We take $G = \mathbf{Z}_n^*$ and $H = \mathbf{Z}_d$. We can easily compute a group homomorphism by first raising to the power $r(q - 1)$ then computing a discrete logarithm in a small subgroup.

We finally provide a useful lemma to sample group elements.

Lemma 3. *Let G, H, d be defined as in Lemma 2. Let $x_1, \ldots, x_s \in G$ which H-generate G. The following mapping from $G \times \mathbf{Z}_d^s$ to G is balanced.*

$$g : (r, a_1, \ldots, a_s) \mapsto dr + a_1 x_1 + \ldots + a_s x_s$$

2.3 Problem Reductions

We assume that S interpolates in a group homomorphism. We notice that the S-GHI problem can be solved with a single oracle call to a (d, S_1)-MSR oracle where S_1 denotes the set of all x coordinates for points in S.

Similarly, the S-GHID problem can be probabilistically solved with a (d, S_1)-MSR oracle by using Lemma 1 and Lemma 3: we generate a random $x' = ax + dr + a_1 x_1 + \cdots + a_s x_s$, we send it to the MSR oracle who will answer $a'_1, \ldots, a'_s$, and we check whether $ay + (a_1 - a'_1)y_1 + \cdots + (a_s - a'_s)y_s = 0$.

Note that once we have witnesses to find the group invariants of G and H, it becomes easy to solve all problems. So GHI and GHID are in NP∩co-NP.

2.4 Problem Approximations

In this section we present our most important results. They are inspired from the theory of checkable proofs [2, 3] and linear cryptanalysis.

Lemma 4. *Given two finite Abelian groups G and H, and a set of s points $S = \{(x_i, y_i) \mid i = 1, \ldots, s\}$, we assume that $x_1, \ldots, x_s$ H-generate G. We let d be the order of H and p be its smallest prime factor. We assume that there exists a function $f : G \longrightarrow H$ such that*

$$\rho := \Pr_{(r,a_1,\ldots,a_s) \in_U G \times \mathbf{Z}_d^s} [f(dr + a_1 x_1 + \cdots + a_s x_s) = a_1 y_1 + \cdots + a_s y_s] > \frac{1}{p}.$$

The set of points (x_i, y_i) interpolates in a group homomorphism. Furthermore, given a random $x \in_U G$, the value $y = f(x)$ matches the unique interpolation with probability ρ.

This improves Theorem 13 from [19] where we have $1/2$ instead of $1/p$.

Proof. Let $K \subseteq \mathbf{Z}_d^s$ be the set of all $(a_1, \ldots, a_s)$ such that $a_1 x_1 + \cdots + a_s x_s \in dG$. We notice that the representation of any G element as a combination of $x_1, \ldots, x_s$ is uniquely defined modulo K. Following Lemma 1, we only have to prove that we have $a_1 y_1 + \cdots + a_s y_s = 0$ for any $(a_1, \ldots, a_s) \in K$. This way, the value $g(x) = a_1 y_1 + \cdots + a_s y_s$ is uniquely defined by $x = dr + a_1 x_1 + \cdots + a_s x_s$ and g is a group homomorphism which corresponds to f with probability ρ.

Let us consider a random $(r, a_1, \ldots, a_s) \in_U G \times \mathbf{Z}_d^s$. ρ is the probability that $f(dr + a_1 x_1 + \cdots + a_s x_s)$ equals $a_1 y_1 + \cdots + a_s y_s$. This probability is also the average over all possible cosets of $\mathbf{Z}_d^s / K$ of the same probability when $(a_1, \ldots, a_s)$ is sampled in the coset only. Hence we deduce the existence of a coset $(a_1, \ldots, a_s) + K$ such that for $(r, b_1, \ldots, b_s) \in_U G \times K$ we have

$$\Pr[f(dr + (a_1 + b_1)x_1 + \cdots + (a_s + b_s)x_s) = (a_1 + b_1)y_1 + \cdots + (a_s + b_s)y_s] \geq \rho.$$

Note that $a_1x_1+\cdots+a_sx_s$ is now a constant x and that $dr+b_1x_1+\cdots+b_sx_s$ can be written dr' where r' is uniformly sampled in G and independent from $b_1,\ldots,b_s$. Hence, there exists r' such that

$$\Pr_{(b_1,\ldots,b_s)\in_U K}[f(dr'+x)=(a_1+b_1)y_1+\cdots+(a_s+b_s)y_s]\geq\rho.$$

So we have

$$\Pr_{(b_1,\ldots,b_s)\in_U K}[b_1y_1+\cdots+b_sy_s=\text{constant}]>\frac{1}{p}.$$

Since $(b_1,\ldots,b_s)\mapsto b_1y_1+\cdots+b_sy_s$ is a group homomorphism from K to a subgroup of H it must be a balanced function. Its kernel is either a subgroup of size at least p or the trivial subgroup $\{0\}$. Hence, the probability must actually be 1 and we have $b_1y_1+\cdots+b_sy_s=0$ for all $(b_1,\ldots,b_s)\in K$. □

The next result says that f can be used in order to solve the GHI problem.

Lemma 5. *Given two finite Abelian groups G and H, and a set of s points $S=\{(x_i,y_i)\mid i=1,\ldots,s\}$, we assume that $x_1,\ldots,x_s$ H-generate G. We assume that we are given the order d of H whose smallest prime factor is p and that we can sample elements in G with a uniform distribution. We assume that we have an oracle function $f:G\longrightarrow H$ such that*

$$\Pr_{(r,a_1,\ldots,a_s)\in_U G\times\mathbf{Z}_d^s}[f(dr+a_1x_1+\cdots+a_sx_s)=a_1y_1+\cdots+a_sy_s]=\frac{1}{p}+\theta$$

with $\theta>0$. Let $\varepsilon>0$ be arbitrarily small. There exists a group homomorphism which interpolates S and which is computable within $4\theta^{-2}\log(p/\varepsilon)$ oracle calls with an error probability less or equal to ε.

Note that this substantially improves Theorem 8 from [19] where we basically have $11/12$ instead of $1/p$. It was further conjectured in [19] that we could replace it by $1/2$. We made here a more precise result.

Proof (Sketch). Due to Lemma 4, the homomorphism g exists and we have $\Pr_{x\in_U G}[f(x)=g(x)]=p^{-1}+\theta$. We use the same techniques which are used in linear cryptanalysis and consider the following algorithm.

Input: $x\in G$
1: **repeat**
2: pick $r\in G,a_1,\ldots,a_s\in\mathbf{Z}_d$ at random
3: $y=f(x+dr+a_1x_1+\cdots+a_sx_s)-a_1y_1-\cdots-a_sy_s$
4: $c=0$
5: **for** $i=1$ to n **do**
6: pick $r\in G,a_1,\ldots,a_s,a\in\mathbf{Z}_d$ at random
7: **if** $f(dr+a_1x_1+\cdots+a_sx_s+ax)=a_1y_1+\cdots+a_sy_s+ay$ (**T**) **then**
8: $c=c+1$

```
    end if
  end for
until c > τn
```
Output: y

We choose $n = 4\theta^{-2}(p^{-1}+\theta)\log(p/\varepsilon)$ and $\tau = p^{-1} + \frac{1}{2}\theta$ and we estimate the error probability of the acceptance test. We consider two types of error:

$$\varepsilon_1 = \Pr_{x\in_U G}[c \le \tau n \mid y = g(x)] \qquad \varepsilon_2 = \Pr_{x\in_U G}[c > \tau n \mid y \neq g(x)]$$

We will now estimate these two values and show that they are negligible. If $y \neq g(x)$, then the test (**T**) works with probability $t_2 \le 1/p$ due to Lemma 4. We also notice that if $y = g(x)$, the probability that the test works is $\frac{1}{p} + \theta$. Hence, using the central limit theorem we obtain

$$\varepsilon_1 \approx \Phi\left(\sqrt{n}\frac{\tau - p^{-1} - \theta}{\sqrt{(p^{-1}+\theta)(1-p^{-1}-\theta)}}\right) \qquad \varepsilon_2 \approx \Phi\left(-\sqrt{n}\frac{\tau - t_2}{\sqrt{t_2(1-t_2)}}\right),$$

when n is large enough and where Φ denotes the distribution function of the standard normal distribution. By looking at the logarithmic derivative of the function $f(t) = (\tau - t)/(\sqrt{t(1-t)})$ and noticing that this one is negative on the interval $[0, \tau]$ we deduce that

$$\varepsilon_2 \le \Phi\left(-\sqrt{n}\frac{\tau - p^{-1}}{\sqrt{p^{-1}(1-p^{-1})}}\right).$$

Using $\tau = p^{-1} + \frac{1}{2}\theta$ provides

$$\varepsilon_2 \le \Phi\left(-\sqrt{n}\frac{\theta}{2\sqrt{p^{-1}(1-p^{-1})}}\right) \approx \frac{1}{\sqrt{2\pi}}\left(e^{\frac{-n\theta^2}{4(p^{-1}(1-p^{-1}))}}\right),$$

where the last approximation holds when n is large enough (ε small). Now, we substitute the expression of n in the above inequality and we obtain

$$\varepsilon_2 \le \frac{1}{\sqrt{2\pi}}\left(\frac{\varepsilon}{p}\right)^{\frac{p+p^2\theta}{p-1}}.$$

Since $\frac{p+p^2\theta}{p-1} \ge 1$ and $\frac{\varepsilon}{p} < 1$ when ε is small, we finally get $\varepsilon_2 \le \varepsilon/(p\sqrt{2\pi}) \le \rho\varepsilon/2$ where $\rho = p^{-1} + \theta$. In a similar way, we can show that $\varepsilon_1 \le \varepsilon/2$. It remains to compute the complexity and the error probability of the algorithm. At first, we observe that the probability α that $c \le \tau n$ in the algorithm is equal to $\rho\varepsilon_1 + (1-\rho)(1-\varepsilon_2)$. From the estimate of ε_1, ε_2, we see that $\alpha \approx 1-\rho$. Moreover, the number of iterations is equal to $\sum_{i=1}^{\infty} i\alpha^{i-1}(1-\alpha) = 1/(1-\alpha) \approx 1/\rho$. Hence, the complexity is $n/\rho = 4(\log(1/\varepsilon) + \log(p))/(\rho - \frac{1}{p})^2$. The probability of error is given by $\sum_{i=1}^{\infty} \alpha^{i-1}(1-\rho)\varepsilon_2 \approx (1-\rho)/\rho\varepsilon_2 \le \varepsilon_2/\rho \le \varepsilon/2$. □

3 Interactive Proof Protocol

3.1 Proof for the GHID Problem

Let G, H, and $S = \{(g_1, e_1), \ldots, (g_s, e_s)\}$ be parameters of a GHI problem, and let d be the order of H. We assume that we have a prover who wants to convince a verifier that he knows an interpolating group homomorphism $f : G \longrightarrow H$ for S. Let ℓ be an integer. He performs the following interaction with a verifier.

GHIproof$_\ell(S)$
Parameters: G, H, d
Input: ℓ, $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$

1. The verifier picks $r_i \in G$ and $a_{i,j} \in \mathbf{Z}_d$ at random for $i = 1, \ldots, \ell$ and $j = 1, \ldots, s$. He computes $u_i = dr_i + a_{i,1}g_1 + \cdots + a_{i,s}g_s$ and $w_i = a_{i,1}e_1 + \cdots + a_{i,s}e_s$ for $i = 1, \ldots, \ell$. He sends $u_1, \ldots, u_\ell$ to the prover.
2. The prover computes $v_i = f(u_i)$ for $i = 1, \ldots, \ell$. He sends a commitment to $v_1, \ldots, v_\ell$ to the verifier.
3. The verifier sends all r_i's and $a_{i,j}$'s to the prover.
4. The prover checks that the u_i's computations are correct. He then opens his commitment.
5. The verifier checks that $v_i = w_i$ for $i = 1, \ldots, \ell$.

From a practical point of view, the verifier can generate the r_i's and $a_{i,j}$'s in a pseudorandom way from a seed and simply disclose the seed in the third step of the protocol. Further note that if d^s is large enough, then the verifier can send $h(w_1, \ldots, w_s) \oplus$ seed (where h is a hash function) in his first message so that the complete protocol can run in 2 moves instead of 4. In the second move, the prover simply sends seed.

Note that we need a commitment scheme here, e.g. the trapdoor commitment scheme proposed by Bresson et al. [6]. Note that using trapdoor commitment with the verifier's public key strengthens our protocols by providing the non-transferability property [17].

Theorem 1. *Assuming that $g_1, \ldots, g_s$ H-generate an Abelian group G, let d be an integer and $e_1, \ldots, e_s \in H$, where H is an Abelian group of order d. Let p be the smallest prime factor of d. We consider the GHIproof$_\ell(S)$ protocol with $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$.*

i. Completeness: assuming that the prover and the verifier are honest, the protocol always succeeds.
ii. Zero-knowledge: assuming that the commitment scheme is perfectly hiding, the above protocol is perfectly black-box zero-knowledge against any verifier.
iii. Proof of membership: assuming that the protocol succeeds with probability greater than $p^{-\ell}$ with a honest verifier, then S interpolates in a group homomorphism.

iv. Proof of knowledge: for any $\theta > 0$, assuming that the protocol succeeds with probability greater than $(p^{-1} + \theta)^{\ell}$ with a honest verifier and that the commitment scheme is extractable, for any $\varepsilon > 0$ there exists an extractor with a time complexity factor $\mathcal{O}(\log(1/\varepsilon))$ which can compute an interpolating group homomorphism from the prover with probability at least $1 - \varepsilon$.

Proof (Sketch). Property i is quite clear. Property ii is proven by constructing a simulator for the transcript of the protocol without the secret of the prover. Property iii directly follows from Lemma 4. For Property iv, we use Lemma 4 and Lemma 5. □

3.2 Proof for the co-GHID Problem

Let G, H, and $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$ be parameters of a GHI problem, and let d be the order of H. Let $T = \{(x_1, z_1), \ldots, (x_t, z_t)\} \subseteq G \times H$ be a set of t inputs of the GHID problem. We assume that we have a prover who wants to convince a verifier that for at least one k the answer to the GHID problem with (x_k, z_k) is negative. Let ℓ be an integer. He performs the following interaction with a verifier.

coGHIproof$_\ell(S, T)$
Parameters: G, H, d
Input: ℓ, $S = \{(g_1, e_1), \ldots, (g_s, e_s)\}$, $T = \{(x_1, z_1), \ldots, (x_t, z_t)\}$

1. The verifier picks $r_{i,k} \in G$, $a_{i,j,k} \in \mathbf{Z}_d$, and $\lambda_i \in \mathbf{Z}_p^*$ for $i = 1, \ldots, \ell$, $j = 1, \ldots, s$, $k = 1, \ldots, t$, where p is the smallest prime dividing d. He computes $u_{i,k} := dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$ and $w_{i,k} := \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$. Set $u := (u_{1,1}, \ldots, u_{\ell,t})$ and $w := (w_{1,1}, \ldots, w_{\ell,t})$. He sends u and w to the prover.
2. The prover computes $v_{i,k} := f(u_{i,k})$ for $i = 1, \ldots, \ell$, $k = 1, \ldots, t$. Since $w_{i,k} - v_{i,k} = \lambda_i(z_k - y_k)$, he should be able to find every λ_i if the verifier is honest since $w_{i,k} \neq v_{i,k}$ for all i and at least one k. Otherwise, he sets λ_i to a random value. He then sends a commitment to $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ to the verifier.
3. The verifier sends all $r_{i,k}$'s and $a_{i,j,k}$'s to the prover.
4. The prover checks that u and w were correctly computed. He then opens the commitment to λ.
5. The verifier checks that the prover could find the right λ.

This protocol is inspired from denial protocol of Gennaro et al. [15]. We can also transform it into a 2-move protocol.

We notice that λ_i was chosen such that it can be uniquely retrieved for every nonzero values of $\mathbf{Z}_d$ that can be taken by the elements $z_k - y_k$'s. Namely, this is done by the following result.

Lemma 6. *Let H be an Abelian group of order d, and $a, b \in H$ such that $b \neq 0$. Let λ be in $\{1, \ldots, p-1\}$, where p is the smallest prime dividing d. Then, if the equation $a = \lambda b$ has a solution in λ, then this one is unique.*

3.3 Proof for the MGGD Problem

Inspired by [19], we propose here a proof that $S_1 = \{g_1, \ldots, g_s\}$ H-generate G. However, the signer needs expert knowledge about G since he has to be able to solve the (d, S_1)-MSR and d-Root problems. Let ℓ be an integer. He performs the following protocol.

MGGDproof$_\ell(S_1)$
Parameters: G, H, d
Input: ℓ, $S_1 = \{g_1, \ldots, g_s\} \subseteq G$
1: **for** $i = 1$ to ℓ **do**
2: The prover picks a $\delta_1 \in G$ at random and sends a commitment to δ_1 to the verifier.
3: The verifier picks a $\delta_2 \in G$ at random and sends δ_2 to the prover.
4: The prover solves (d, S_1)-MSR on $\delta_1 + \delta_2$ and d-Root and finds $r \in G$, $a_1, \ldots, a_s \in \mathbf{Z}_d$ such that $\delta_1 + \delta_2 = dr + \sum_{j=1}^{s} a_j g_j$. He sends $r, a_1, \ldots, a_s$ to the verifier and opens the commitment to δ_1.
5: The verifier checks that $\delta_1 + \delta_2 = dr + \sum_{j=1}^{s} a_j g_j$ really holds.
6: **end for**

We can prove as in Lemma 4 that if a honest verifier is convinced with probability greater than $p^{-\ell}$, then S_1 solves the d-MGGD problem.

Note that this can be transformed into a non-interactive proof following standard techniques [14]. An efficient way consists of generating pseudorandom $\delta_1, \ldots, \delta_\ell$ from the same seed then solving the (d, S_1)-MSR and d-Root problems on those elements.

4 Undeniable Signature

4.1 Description

We now describe our undeniable signature scheme.

Domain Parameters. We let integers Lkey, Lsig, Icon, Iden be security parameters as well as "group types" for Xgroup and Ygroup. (The group types should define what groups and which sizes to use in order to achieve security.)

An optional parameter Ival is used in Setup Variants 3 and 4 below.

Primitives. We use two deterministic random generators Gen_1 and Gen_2 which produce elements of Xgroup and a commitment scheme.

Setup Variant 1. (signer without expert group knowledge)
The signer selects Abelian groups Xgroup and Ygroup of given types together with a group homomorphism Hom : Xgroup $\longrightarrow$ Ygroup. He computes the order d of Ygroup. He then picks a random string seedK and computes the Lkey first values $(\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}})$ from $\mathrm{Gen}_1(\mathrm{seedK})$ and $\mathrm{Ykey}_j := \mathrm{Hom}(\mathrm{Xkey}_j)$, $j = 1, \ldots, \mathrm{Lkey}$.

The main problem of Setup is that the choice for $(\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}})$ must Ygroup-generate Xgroup in order to ensure non-repudiation of signatures. In Variant 1, Lkey must be large enough so that it is impossible to maliciously select a key which does not guaranty this condition.

Setup Variant 2. (signer with a Registration Authority (RA))
We use here a RA whose role consists of making sure that a key was randomly selected. (Note that, the RA does not check if the key is valid.)

1. The signer selects Abelian groups Xgroup and Ygroup of given type together with a group homomorphism $\mathrm{Hom} : \mathrm{Xgroup} \longrightarrow \mathrm{Ygroup}$. He computes the order d of Ygroup. He submits his identity Id together with Xgroup, Ygroup and d to RA.
2. RA first checks the identity of the signer and that he did not submit too many registration attempts. He then picks a random string seedK that is sent to the signer together with a signature C for
$$(\mathrm{Id}, \mathrm{Xgroup}, \mathrm{Ygroup}, d, \mathrm{seedK}).$$
3. The signer computes the Lkey first values $(\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}})$ from $\mathrm{Gen}_1(\mathrm{seedK})$ and $\mathrm{Ykey}_j := \mathrm{Hom}(\mathrm{Xkey}_j)$, $j = 1, \ldots, \mathrm{Lkey}$.

   Here the RA basically selects the random key so Lkey can be reduced.

Setup Variant 3. (signer with an expert group knowledge)
In this variant we assume that the signer can solve the MSR and Root problems in Xgroup. It works exactly like in the Setup Variant 1, but the signer can further run a $\mathrm{MGGDproof}_{\mathrm{Ival}}$ in order to validate the public key so that Lkey can be further reduced to the smallest possible one.

Setup Variant 4. (signer with an expert group knowledge, non-interactive)
This variant is the same as Variant 3 except that MGGDproof is transformed into a non-interactive proof.

Public Key. $K_P = (\mathrm{Xgroup}, \mathrm{Ygroup}, d, \mathrm{seedK}, (\mathrm{Ykey}_1, \ldots, \mathrm{Ykey}_{\mathrm{Lkey}}))$ with an optional (Id, C) for Variant 2, an optional Ival for Variants 3,4, and an optional non-interactive proof for Variant 4. We say that K_P is valid if $\{\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}\}$ Ygroup-generate Xgroup.

Secret Key. $K_S = \mathrm{Hom}$.

Signature Generation. The message M is used to generate $\mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_{\mathrm{Lsig}}$ from $\mathrm{Gen}_2(M)$. The signer computes $\mathrm{Ysig}_k = \mathrm{Hom}(\mathrm{Xsig}_k)$ for $k = 1, \ldots, \mathrm{Lsig}$. The signature is $(\mathrm{Ysig}_1, \ldots, \mathrm{Ysig}_{\mathrm{Lsig}})$. It consists of $\mathrm{Lsig}. \log_2 d$ bits.

Confirmation Protocol. Compute $\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}$ from the public key, $\mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_{\mathrm{Lsig}}$ from the message, run $\mathrm{GHIproof}_{\mathrm{Icon}}$ on the set
$$S = \{(\mathrm{Xkey}_j, \mathrm{Ykey}_j) | j = 1, \ldots, \mathrm{Lkey}\} \cup \{(\mathrm{Xsig}_k, \mathrm{Ysig}_k) | k = 1, \ldots, \mathrm{Lsig}\}.$$

Denial Protocol. Compute $\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}$ from the public key as well as $\mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_{\mathrm{Lsig}}$ from the message, run $\mathrm{coGHIproof}_{\mathrm{Iden}}$ on the sets

$$S = \{(\mathrm{Xkey}_j, \mathrm{Ykey}_j) | j = 1, \ldots, \mathrm{Lkey}\},\ T = \{(\mathrm{Xsig}_k, \mathrm{Zsig}_k) | k = 1, \ldots, \mathrm{Lsig}\}$$

where $(\mathrm{Zsig}_1, \ldots, \mathrm{Zsig}_{\mathrm{Lsig}})$ is the alleged non-signature.

The undeniable signature scheme of Gennaro et al. [15] which is based on RSA corresponds to a special case of our scheme, namely with Xgroup = Ygroup = $\mathbf{Z}_n^*$, Lkey = Lsig = 1 and the classical RSA signing function as homomorphism Hom. Another example with Lkey = Lsig = 1 is the undeniable signature of Chaum [7]. He considered Xgroup = Ygroup = $\mathbf{Z}_p^*$ for a prime p and the homomorphism consisting in raising an element to the power of the private key. In both examples the signature is quite large. The MOVA scheme [19] is another example with Xgroup = $\mathbf{Z}_n^*$, Hom is a character of order $d \in \{2, 3, 4\}$, and Ygroup is the subgroup of $\mathbf{C}^*$ spanned by $e^{\frac{2i\pi}{d}}$.

4.2 Security Analysis

Theorem 2 (Setup Variants 1,2). *We consider the above undeniable signature. Given a prime q, we let A_q be the subgroup of* Xgroup *of all terms whose orders are powers of q. Given q there is a unique k_q and $a_{q,1} \leq \ldots \leq a_{q,k_q}$ sequence such that A_q is isomorphic to $\mathbf{Z}_{q^{a_{q,1}}} \oplus \ldots \oplus \mathbf{Z}_{q^{a_{q,k_q}}}$. The probability P_{gen} that $\{\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}\}$ Ygroup-generate Xgroup satisfies*

$$P_{\mathrm{gen}} \geq \prod_{q \in \mathbf{P}_d} \left(1 - \frac{k_q}{q^{\mathrm{Lkey}}}\right),$$

where $\mathbf{P}_d$ is the set of all prime factors of $\gcd(\#\mathrm{Xgroup}, d)$.

As an application, if d is prime and if Xgroup is a product of k cyclic groups, we have $P_{\mathrm{gen}} \geq 1 - k.d^{-\mathrm{Lkey}}$.

Theorem 3. *We consider the above undeniable signature scheme. Assuming that the public key is valid, we have the following security results.*

i. If the signer and the verifier are honest, the two protocols complete: a valid signature will always be accepted by the confirmation protocol, and an invalid signature will always be rejected by the denial protocol.

ii. Let $S = \{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$. The scheme resists against existential forgery attacks provided that Gen_2 is a random oracle and the S-GHI problem is intractable.

iii. The confirmation (resp. denial) protocol is sound: if the signer is able to pass the protocol with probability $q > p^{-\mathrm{Icon}}$ (resp. $q > p^{-\mathrm{Iden}}$), then the alleged signature is valid (resp. invalid).

iv. The confirmation protocol is private when the commitment scheme is extractable: for any $\theta, \varepsilon > 0$, from a prover which is able to convince a honest verifier that a given signature is valid with probability $q > (p^{-1} + \theta)^{\mathrm{Icon}}$, we can extract within a complexity factor of $\Omega(\theta^{-2} \log(p/\varepsilon))$ a group homomorphism which solves the GHI problem with success probability $1 - \varepsilon$.

v. The signatures are invisible: for any $\theta, \varepsilon > 0$, from a distinguisher of a valid signature from a random one with advantage $\theta > 0$, we can extract within a complexity factor of $\Omega(\theta^{-2}\log(1/\varepsilon))$ a GHID problem solver with success probability $1 - \varepsilon$.

vi. The confirmation (resp. denial) protocol is perfectly black-box zero-knowledge when the commitment scheme is perfectly hiding: we can build a simulator for the protocol without the secret key for any verifier.

In short, if we take Xgroup $= \mathbf{Z}_n^*$ where n is a product of two prime numbers, and Lsig = Icon = Iden $= s_{\text{online}}/\log_2 p$, we cannot contradict the confirmation or denial protocols but with a probability at most $2^{-s_{\text{online}}}$, and signatures are invisible provided that generators are random oracles and that the interpolation problem is hard. For Variant 2, we can take Lkey $= s_{\text{online}}/\log_2 p$ and this generates invalid keys with probability less than $2^{1-s_{\text{online}}}$. For Variant 1, we can take Lkey $= s_{\text{offline}}/\log_2 p$ so that the signer cannot create invalid keys within a complexity less than $2^{s_{\text{online}}}$. For Variants 3,4, Lkey can be as low as possible. We can take Ival $= s_{\text{online}}/\log_2 p$ for Variant 3 (so that invalid keys are accepted with probability less than $2^{-s_{\text{online}}}$), and Ival $= s_{\text{offline}}/\log_2 p$ for Variant 4 so that the signer cannot create invalid keys within a complexity less than $2^{s_{\text{online}}}$. We suggest $s_{\text{offline}} = 80$ and $s_{\text{online}} = 20$.

5 Example and Further Discussions

5.1 Setting Proposal

We consider Example 7 with a small prime d e.g. $d = 2^{20} + 7$. We take Xgroup $= \mathbf{Z}_n^*$, Ygroup $= \mathbf{Z}_d$, Lkey = Lsig = Icon = Iden = 1 and we consider Variant 3 and 4 of the Setup protocol. If Xkey $\in$ Xgroup is not a dth power residue then it Ygroup-generates Xgroup. For any Ykey $\in \mathbf{Z}_d$ there is a unique group homomorphism Hom such that Hom(Xkey) = Ykey. With this example we can sign with a single element of $\mathbf{Z}_d$ and a public key $(n, d, \text{seedK}, \text{Ykey})$.

Note that the group homomorphism computation requires raising to the power r in $\mathbf{Z}_p^*$ and computing the discrete logarithm in a cyclic group of about 2^{20} elements. This can be precomputed in a table of 2.5 MB as detailed below.

We first precompute a (large) table of all (Xsig_i, i) with $\text{Xsig}_i = \text{Xkey}^{ir} \pmod p$ for $i = 0, 1, \ldots, d-1$. Note that i can be encoded into 20 bits. Next we insert all (Xsig_i, i) pairs in a hash table of 2^{20} entries keyed by Xsig_i: put i at position $h(\text{Xsig}_i)$ unless there is a collision. Resolving collisions can be done by standard techniques, for instance see [10] Chapter 12, but note that resolving collisions is not necessary: if Xsig_i is not in the table, we can look for the smallest j such that Xsig_{i+j} is in the table.

Time/memory tradeoffs can also be considered. Remark also that such a tradeoff should not require more than the complexity of the Pollard's rho algorithm for the computation of the discrete logarithm in our example, i.e. approximately 3000 multiplications.

Depending on the application, the signature size of 20 bits may be considered as too small. Of course, we can easily enlarge it e.g. to 48 bits. Our point is that signature size versus security is fully scalable here.

The signature generation requires 1 homomorphism i.e. about one exponentiation in $\mathbf{Z}_p^*$. (Note that this is twice as fast as a 1024-bit RSA signature computation with Chinese remainders.) The complexity of the confirmation protocol is about 35 multiplications in $\mathbf{Z}_n^*$ for the verifier (which can be compared to 17 multiplications in $\mathbf{Z}_n^*$ for RSA if we take $e = 2^{16} + 1$) and 1 homomorphism for the prover. The denial protocol requires almost the same complexity.

Complexities of this setting with all setup variants as well as those of the MOVA scheme with $d = 2$ and a 20-bit signature length are detailed in Table 1. The main advantage of using the above setting instead of MOVA is that the former strongly decreases the number of multiplications in $\mathbf{Z}_n^*$ for the confirmation.

Table 1. Implementation Examples

Setup	d	Lsig, Icon, Iden	Lkey	Ival	Signature cost	Confirmation cost
1	2	20	80		20 Leg. symb.	20 Leg. symb., 730 mult.
2	2	20	20		20 Leg. symb.	20 Leg. symb., 280 mult.
3	2	20	2	20	20 Leg. symb.	20 Leg. symb., 145 mult.
4	2	20	2	80	20 Leg. symb.	20 Leg. symb., 145 mult.
1	$2^{20}+7$	1	4		1 Hom	1 Hom, 65 mult.
2	$2^{20}+7$	1	1		1 Hom	1 Hom, 35 mult.
3	$2^{20}+7$	1	1	1	1 Hom	1 Hom, 35 mult.
4	$2^{20}+7$	1	1	4	1 Hom	1 Hom, 35 mult.

5.2 On the MOVA Scheme

We point out here that our scheme generalizes the MOVA scheme [19] and improves the efficiency of the denial protocol of MOVA. An additional contribution to MOVA is also the improvement of some bounds related to the probability of a function approximating Hom from which we can compute Hom in a polynomial time. Our new bound with $1/p$ allows to formally prove the conjectured security level of MOVA.

5.3 Batch Verification and Selective Convertibility

We point out that our scheme allows a batch verification of signatures. Indeed, the confirmation protocol can be easily adapted in order to confirm several signatures at the same time by putting all $(\mathrm{Xsig}_k, \mathrm{Ysig}_k)$ in a single set S.

Note that the signer with expert group knowledge can selectively convert an undeniable signature into a classical one by solving the MSR and Root problems on all Xsig_k. The conversion consists of revealing the solution to those problems.

6 Conclusion

We have exposed an undeniable signature based on a generic group homomorphism interpolation and we have also analyzed the security in the random oracle model. The principal advantage is the size of the signature that can be chosen arbitrarily short depending on the required security level. Confirmation and denial can be run in a 2-move protocol. We can perform batch verification and have selective convertibility. From this general setting we have also proposed a practical example with 3-byte signatures and a complexity cost which is similar to RSA. We hope that this example will be completed by some various additional settings since group homomorphisms are common objects in cryptography.

As future work, we also aim at extending our techniques to other cryptographic algorithms such as the designated confirmer signatures [8].

Acknowledgments. We wish to thank Anna Lysyanskaya and Wenbo Mao for helpful discussions and comments.

References

1. R. Anderson, S. Vaudenay, B. Preneel, K. Nyberg, *The Newton Channel*, Proc. First International Workshop on Information Hiding, Cambridge, UK, LNCS **1174**, pp. 151–156, Springer, 1996.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, *Proof Verification and Hardness of Approximation Problems*, Proc. 33rd IEEE Symp. on Foundations of Computer Science, pp. 14–23, 1992.
3. L. Babai, L. Fortnow, L. Levin and M. Szegedy, *Checking Computations in Polylogarithmic Time*, Proc. 23rd ACM Symp. on Theory of Computing, pp. 21–31, 1991.
4. I. Biehl, S. Paulus and T. Takagi, *Efficient Undeniable Signature Schemes based on Ideal Arithmetic in Quadratic Orders*, Conference on The Mathematics of Public-Key Cryptography, Toronto, 1999.
5. J. Boyar, D. Chaum, I. Damgård and T. Pedersen, *Convertible Undeniable Signatures*, Advances in Cryptology - Crypto '90, LNCS **537**, pp. 189–205, Springer, 1990.
6. E. Bresson, D. Catalano and D. Pointcheval, *A Simple Public-Key Cryptosytem with a Double Trapdoor Decryption Mechanism and Its applications*, Advances in Cryptology - Asiacrypt '03, LNCS **2894**, pp. 37–54, Springer, 2003.
7. D. Chaum, *Zero-Knowledge Undeniable Signatures*, Advances in Cryptology - Eurocrypt '90, LNCS **473**, pp. 458–464, Springer, 1990.
8. D. Chaum, *Designated Confirmer Signatures*, Advances in Cryptology - Eurocrypt '94, LNCS **950**, pp. 86–91, Springer, 1994.
9. D. Chaum and H. van Antwerpen, *Undeniable Signatures*, Advances in Cryptology - Crypto '89, LNCS **435**, pp. 212–217, Springer, 1989.
10. T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, McGraw Hill, 1990.
11. I. Dåmgard and T. Pedersen, *New Convertible Undeniable Signatures Schemes*, Advances in Cryptology - Eurocrypt '96, LNCS **1070**, pp. 372–386, Springer, 1996.

12. Y. Desmedt and M. Yung, *Weaknesses of Undeniable Signature Schemes*, Advances in Cryptology - Crypto '91, LNCS **576**, pp. 205–220, Springer, 1991.
13. W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, vol. IT-22, pp. 644–654, 1976.
14. A. Fiat, A. Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Advances in Cryptology - Crypto '86, LNCS **263**, pp. 186–194, Springer, 1987.
15. R. Gennaro, T. Rabin and H. Krawczyk, *RSA-Based Undeniable Signatures*, Journal of Cryptology, **13**, pp. 397–416, Springer, 2000.
16. S. Goldwasser and S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences, 28, pp. 270–299, 1984.
17. M. Jakobsson, K. Sako and R. Impagliazzo, *Designated Verifier Proofs and Their Applications*, Advances in Cryptology - Eurocrypt '96, LNCS **1070**, pp. 143-154, 1996.
18. B. Libert and J.-J. Quisquater, *Identity Based Undeniable Signatures*, Proc. RSA Crypto Track '04, LNCS **2964**, pp. 112–125, Springer, 2004.
19. J. Monnerat and S. Vaudenay, *Undeniable Signatures Based on Characters: How to Sign with One Bit*, PKC '04, LNCS **2947**, pp. 69–85, Springer, 2004.
20. P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Advances in Cryptology - Eurocrypt '99, LNCS **1592**, pp. 223–238, Springer, 1999.
21. R. L. Rivest, A. Shamir and L. M. Adleman, *A Method for Obtaining Digital Signatures and Public-key Cryptosystem*, Communications of the ACM, vol. 21, pp. 120–126, 1978.

A Technical Proofs

Proof of Lemma 2. **1** $\Leftrightarrow$ **2** $\Leftrightarrow$ **3.** Straightforward.
3 $\Rightarrow$ **4.** Assume that there exists a common prime factor p of $\#(G/G')$ and d. Then, from the structure of Abelian groups G/G' and H we know that each of these two groups possesses one cyclic subgroup U and V respectively of order p. So, we define a non trivial homomorphism that is the composition of the isomorphism between the two cyclic subgroups and with the reduction modulo U. This contradicts 3.
4 $\Rightarrow$ **5.** If $x \in G$, then d must be invertible modulo the order k of $x \bmod G'$ by 4. Let m such that $m \cdot d \equiv 1 \pmod{k}$. We have $m \cdot d \cdot x \equiv x \pmod{G'}$. Hence, $x - d(m \cdot x) \in G'$ and therefore $x \in G' + dG$.
5 $\Rightarrow$ **2.** If $\varphi \in \mathrm{Hom}(G, H)$ is such that $\varphi|_{G'} = 0$ and $x \in G$, we can write $x = a_1 x_1 + \cdots + a_s x_s + dr$. Thus, $\varphi(x) = d\varphi(r) = 0$. This holds for all $x \in G$, i.e., $\varphi = 0$. □

Proof of Lemma 3. Let n be the order of G. Let $h : G \times \mathbf{Z}_{nd}^s \to G$ be a function defined by $h(r, a_1, \ldots, a_s) = dr + a_1 x_1 + \ldots + a_s x_s$. Obviously, h is an homomorphism. It is onto G due to the property 5 of Lemma 2. Hence, it is balanced onto G. Let $\varphi : G \times \mathbf{Z}_{nd}^s \to G \times \mathbf{Z}_d^s$ be a function defined by $\varphi(r, a_1, \ldots, a_s) \to (r + q_1 x_1 + \cdots + q_s x_s, a_1 \bmod d, \ldots, a_s \bmod d)$, where $a_i - (a_i \bmod d) = dq_i$. We have $g \circ \varphi = h$. Obviously, φ is balanced onto $G \times \mathbf{Z}_d^s$ since $\varphi^{-1}(r, a_1, \ldots, a_s) =$

$\{(r-q_1x_1-\ldots-q_sx_s, a_1+dq_1, \ldots, a_s+dq_s) | (q_1, \ldots, q_s) \in \mathbf{Z}_n^s\}$. If $\#g^{-1}(x) = m$, we have $mn^s = \#\varphi^{-1}\left(g^{-1}(x)\right) = \#h^{-1}(x) = (dn)^s$. Hence, $m = d^s$ does not depend on x, so g is balanced. □

Proof of Theorem 2 (Sketch). The decomposition of Xgroup comes from classical results on the structure of Abelian groups. We observe that we can handle each A_q independently because we can see that two elements generating two different A_q's generate the direct sum of these two groups, since the two respective group orders are coprime. We consider $B_q := A_q/dA_q$ and study the probability that elements generate this group. If $\gcd(d, q) = 1$, then B_q is trivial. So, we focus only on the q's that divide d and denote e_q the largest integer such that $q^{e_q}|d$. We can also deduce that the structure of B_q satisfies

$$B_q \simeq \mathbf{Z}_{q^{a_{q,1}}} \oplus \ldots \oplus \mathbf{Z}_{q^{a_{q,r}}} \oplus \mathbf{Z}_{q^{e_q}} \oplus \ldots \mathbf{Z}_{q^{e_q}},$$

where r is the largest integer such that $a_{q,r} < e_q$. The probability that s elements does not generate B_q can be approximated by the probability that these elements stay in one of the largest non trivial subgroups of B_q, i.e. those of order $\#B_q/q$. The number of such subgroups is equal to k_q. Thus, this probability is greater or equal than $1 - \frac{k_q}{q^s}$. Since these events are independent for the different B_q's, the final probability is obtained by the multiplication of these probabilities. □

Proof of Theorem 3 (Sketch). i. The assertion i is straightforward.
ii. First, we show that an attacker $\mathcal{A}$ having access to a signing oracle can be simulated by an attacker without this access. Indeed, when $\mathcal{A}$ calls the signing oracle on a message M, the signing oracle will first produce a sequence of Lsig values $\mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_{\mathrm{Lsig}} \in \mathrm{Xgroup}$ and then computes $\mathrm{Ysig}_i := \mathrm{Hom}(\mathrm{Xsig}_i)$ for $i = 1, \ldots, \mathrm{Lsig}$. From the point of view of $\mathcal{A}$, this is completely equivalent to dispose of a random source generating pairs of the form $(x, \mathrm{Hom}(x))$ since Gen_2 is modelized as a random oracle. Assuming that S_1 Ygroup-generate Xgroup, we see that this source can be simulated by picking some random $r \in \mathrm{Xgroup}$, a_i's $\in \mathbf{Z}_d$, computing $x := dr + a_1\mathrm{Xkey}_1 + \cdots + a_{\mathrm{Lkey}}\mathrm{Xkey}_{\mathrm{Lkey}}$ and $\mathrm{Hom}(x) = a_1\mathrm{Ykey}_1 + \cdots + a_{\mathrm{Lkey}}\mathrm{Ykey}_{\mathrm{Lkey}}$ using Lemma 3. We denote now x, the challenged element of the GHI problem. We use our attacker $\mathcal{A}$ in order to compute the $\mathrm{Hom}(\mathrm{Xsig}_i)$'s as follows. We simulate Gen_2 by computing $u := dr + x + \sum_{j=1}^{\mathrm{Lkey}} a_j\mathrm{Xkey}_j$ for some random $r \in \mathrm{Xgroup}$, $a_j \in \mathbf{Z}_d$. This is indistinguishable from some uniformly picked element in Xgroup. By standard proofs we show that forged signatures are necessarily one of the Gen_2 queries, so we can deduce $\mathrm{Hom}(x)$ from $\mathrm{Hom}(u)$.
iii. For the confirmation, this directly comes from Theorem 1 property iii. For the denial, a cheating prover willing deny a valid signature has to find the value of λ_i at each round of the protocol. Since $\mathrm{Hom}(u_{i,k}) = w_{i,k}$, the prover does not learn additional information with $w_{i,k}$ and has to find λ_i from $u_{i,k}$ uniquely. He cannot find the λ_i since another distribution of the values $u_{i,k}$ with another λ_i is indistinguishable from the first one. Assuming that the commitment scheme is perfectly binding the cheating prover cannot do better than answering a random λ_i.

iv. This directly comes from Theorem 1 property iv.
v. This works like in Lemma 5. We count how many times (x', y') is accepted after having picked $x' = x + dr + a_1 \mathrm{Xkey}_1 + \cdots + a_{\mathrm{Lkey}} \mathrm{Xkey}_{\mathrm{Lkey}}$ and $y' = y + a_1 \mathrm{Ykey}_1 + \cdots + a_{\mathrm{Lkey}} \mathrm{Ykey}_{\mathrm{Lkey}}$. We use $n = \theta^{-2} \log(1/\varepsilon)$ iterations.
vi. For the confirmation, this comes from property ii in Theorem 1. For the denial, this is done as in [15]. □

Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings

Lan Nguyen and Rei Safavi-Naini

School of Information Technology and Computer Science,
University of Wollongong, Wollongong 2522, Australia
{ldn01, rei}@uow.edu.au

Abstract. We propose a group signature scheme with constant-size public key and signature length that does not require trapdoor. So system parameters can be shared by multiple groups belonging to different organizations. The scheme is provably secure in the formal model recently proposed by Bellare, Shi and Zhang (BSZ04), using random oracle model, Decisional Bilinear Diffie-Hellman and Strong Diffie-Hellman assumptions. We give a more efficient variant scheme and prove its security in a formal model which is a modification of BSZ04 model and has a weaker anonymity requirement. Both schemes are very efficient and the sizes of signatures are approximately one half and one third, respectively, of the sizes of the well-known ACJT00 scheme. We also use the schemes to construct a traceable signature scheme.

1 Introduction

Group signature schemes, introduced by Chaum and Van Heyst [14], allow a group member to sign a message on behalf of the group without revealing his identity and without allowing the message to be linkable to other signed messages that are verifiable with the same public key. Participants in a group signature scheme are a set of *group members* and a *group manager*. The role of the group manager is to register new users by issuing membership certificates that contain registration details, and in case of dispute about a signed message, revoking anonymity of the signed message by 'opening' the signature. In some schemes the functions of the group manager can be split between two managers: an *issuer* and an *opener*. This is a desirable property that allows distribution of trust. It is required that no collusion of the issuer and the opener can frame a group member. Group signatures are among the most important cryptographic primitives for providing privacy and have been used for applications such as anonymous credentials [2], identity escrow [21], voting and bidding [1], and electronic cash [23]. Kiayias et al. [18] also introduced the traceable signature primitive, which is basically the group signature system with added properties allowing a variety of levels for protecting user privacy.

In early group signature schemes [9, 14, 15] the size of the public key and the signature grew with the size of the group and so the schemes were impractical for large groups. Schemes with fixed size group public key and signature

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 372–386, 2004.

length have been first proposed in [13] and later extended in [12, 1, 2]. In Crypto 2000, Ateniese et al. (ACJT00) [1] proposed an efficient group signature scheme with very short length and low computation cost. This scheme is also the only scheme that has been proved to satisfy the informal list of security requirements of group signature schemes. Ateniese and de Medeiros (AdM03) proposed an efficient group signature scheme [2] that is 'without trapdoor' in the sense that none of parties in the system including the group manager need to know the trapdoor. That is the system trapdoor is only used during the initialisation and to generate system parameters. The advantage of this property is that the same trapdoor information can be used to initiate different groups. The importance and usefulness of this property in real-world applications, for example when the group signature scheme is used as a building block of an anonymous credential system among a number of organizations that need to communicate and transfer information about users while protecting their privacy, have been outlined in [2]. A drawback of AdM03 scheme is that it has a single group manager who is responsible for registration of users and opening of signatures, and it is not possible to separate the two functionalities. In AdM03 scheme, the group manager stores the certificate (r, s) of each member. The signature of a group member contains elements χ and E_1 satisfying the equation $E_1 = \chi^r$, and so, to revoke a signature, the group manager (or any party with the knowledge of the certificates) can try all certificates to find the one satisfying the equation. This is an computationally expensive process. The security proof (corrected version) is for the informal list of security requirements, and is given in the generic model [3].

Security of a group signature scheme has been traditionally proved by showing that it satisfies a list of informally defined requirements. Bellare et al. [4] gave a formal security model (BSZ04) for (partially) dynamic groups with four security requirements (Correctness, Anonymity, Traceability and Non-frameability). The model uses various oracles including an Open oracle that takes a signed message and reveals the identity of the signer. The ACJT00 scheme although satisfies the conventional list of requirements but cannot be proved secure in the formal model mainly because of the inclusion of the Open oracle in the model. Kiayias et al. [19] proposed an extension (KY04 scheme) of ACJT00 scheme that is proved secure in their formal model. A new direction in constructing group signature schemes is to use bilinear pairings to shorten the lengths of the signature and key. Boneh et al. [7] proposed a short group signature scheme (BBS04) based on the Strong Diffie-Hellman assumption and a new assumption called the Decisional Linear assumption. The scheme is provably secure in a formal model where the Opening oracle is not available and the Non-frameability property is not required, in comparison with the BSZ04 model. They also showed how to construct an extension, which provides Non-frameability (exculpability). Based on the LRSW assumption [22], Camenisch and Lysyanskaya [11] proposed a group signature scheme (CL04) derived from a signature scheme which allows an efficient zero-knowledge proof of the knowledge of a signature on a committed message, and used it to construct an efficient anonymous credential system.

Our Contribution

In this paper, we first propose a new efficient group signature scheme with a number of attractive properties and prove its security in the BSZ04 model under the Decisional Bilinear Diffie-Hellman and Strong Diffie-Hellman assumptions, using random oracle model. We then give an efficient variant of this scheme and prove its security in the reduced version of BSZ04 model. The only difference between the original BSZ04 model and the reduced version is in modelling anonymity property, as in the reduced version, the adversary does not have access to the Open oracle. This is a plausible model for all cases that the opener is a highly trusted entity and cannot be accessed by the adversary. We also extend the variant scheme to a provably secure traceable signature scheme.

All proposed schemes have fixed lengths for group public key and signature, and so can be used for large size groups. Using elliptic curve cryptography in our schemes results in shorter lengths for signatures and keys. For example, for a comparable level of security as the ACJT00 scheme with 1024 bit composite modulus, our group signature schemes require elliptic curve groups of order 170 bit prime, resulting in the sizes of signatures in our two schemes to be one third and one half, respectively, of the size in ACJT00 scheme. For higher security levels this ratio will be smaller.

Our schemes can be converted into identity escrow systems or extended to support efficient membership revocation, as shown in [26]. The schemes are trapdoor-free. The only other trap-door free scheme is the AdM03 scheme, which uses a trapdoor in the initialisation of the system and assumes that the initialising party "safely forgets" the trapdoor. An advantage of our schemes over AdM03 scheme is that they allow separation of issuer and the opener, hence distribution of trust. Finally in our schemes, the interactive protocol underlying the signature scheme achieves honest verifier perfect zero-knowledge without any computational assumption whereas in the ACJT00 and KY04 schemes, the corresponding protocols achieve honest verifier statistical zero-knowledge under the Strong RSA assumption.

The paper is organized as follows. Section 2 gives related background and section 3 describes our group signature scheme and its security proofs. Section 4 gives a modification of BSZ04 formal model and a variant group signature scheme, and proves that the variant scheme and ACJT00 scheme are secure in the modified model. Section 5 describes our traceable signature scheme and section 6 provides efficiency comparison with ACJT00 scheme.

2 Preliminaries

2.1 Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups generated by P_1 and P_2, respectively, both with order p, a prime, and $\mathbb{G}_M$ be a cyclic multiplicative group with the same order. Suppose there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(P_2) = P_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_M$ be a bilinear pairing with the following properties:

1. **Bilinearity:** $e(aP, bQ) = e(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$
2. **Non-degeneracy:** $e(P_1, P_2) \neq 1$
3. **Computability:** There is an efficient algorithm to compute $e(P, Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$ but our group signature schemes can be easily modified for the case when $\mathbb{G}_1 \neq \mathbb{G}_2$. For a group $\mathbb{G}$ of prime order, hereafter, we denote the set $\mathbb{G}^* = \mathbb{G} \backslash \{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element of the group.

We define a Bilinear Pairing Instance Generator as a Probabilistic Polynomial Time (PPT) algorithm $\mathcal{G}$ that takes as input a security parameter 1^l and returns a uniformly random tuple $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P)$ of bilinear pairing parameters, including a prime number p of size l, a cyclic additive group $\mathbb{G}_1$ of order p, a multiplicative group $\mathbb{G}_M$ of order p, a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_M$ and a generator P of $\mathbb{G}_1$.

2.2 Complexity Assumptions

For a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$, if for every positive number α, there exists a positive integer l_0 such that for every integer $l > l_0$, it holds that $f(l) < l^{-\alpha}$, then f is said to be *negligible*. If there exists a positive number α_0 such that for every positive integer l, it holds that $f(l) < l^{\alpha_0}$, then f is said to be *polynomial-bound*.

The q-SDH assumption originates from a weaker assumption introduced by Mitsunari et. al. [24] to construct traitor tracing schemes [28] and later used by Zhang et al. [30] and Boneh et al. [5] to construct short signatures. It intuitively means that there is no PPT algorithm that can compute a pair $(c, \frac{1}{x+c}P)$, where $c \in \mathbb{Z}_p$, from a tuple $(P, xP, \ldots, x^qP)$, where $x \in_R \mathbb{Z}_p^*$.

q-Strong Diffie-Hellman (q-SDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{q\text{-}SDH}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{q\text{-}SDH}(l) = Pr[(\mathcal{A}(\mathbf{t}, P, xP, \ldots, x^qP) = (c, \frac{1}{x+c}P)) \wedge (c \in \mathbb{Z}_p)]$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$ *and* $x \leftarrow \mathbb{Z}_p^*$.

Intuitively, the DBDH assumption [6] states that there is no PPT algorithm that can distinguish between a tuple $(aP, bP, cP, e(P, P)^{abc})$ and a tuple (aP, bP, cP, Γ), where $\Gamma \in_R \mathbb{G}_M^*$ (i.e., chosen uniformly random from $\mathbb{G}_M^*$) and $a, b, c \in_R \mathbb{Z}_p^*$. It is defined as follows.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption. *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{DBDH}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{DBDH}(l) = |Pr[\mathcal{A}(\mathbf{t}, aP, bP, cP, e(P, P)^{abc}) = 1] - Pr[\mathcal{A}(\mathbf{t}, aP, bP, cP, \Gamma) = 1]|$$

where $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$, $\Gamma \leftarrow \mathbb{G}_M^*$ *and* $a, b, c \leftarrow \mathbb{Z}_p^*$.

2.3 Bilinear Pairing Versions of El Gamal Public Key System

Based on the DBDH assumption, we can construct two bilinear pairing versions of El Gamal public key system. El Gamal^{BP1} provides Indistinguishability against adaptive Chosen Plaintext Attack (IND-CPA) and El Gamal^{BP2} provides Indistinguishability against adaptive Chosen Ciphertext Attack (IND-CCA) in the random oracle model. Due to space limitation, we only provide description of El Gamal^{BP2}. This is the bilinear pairing version of the scheme presented and proved by Fouque and Pointcheval [17]. Description of El Gamal^{BP1} can be found in the full version of this paper [25].

Key generation: Let $p, \mathbb{G}_1, \mathbb{G}_M, e$ be bilinear pairing parameters, as defined above, and G be a generator of $\mathbb{G}_1$. Suppose $x_a, x_b \in_R \mathbb{Z}_p^*$ and $\Theta_a = e(G,G)^{x_a}$ and $\Theta_b = e(G,G)^{x_b}$. The public key $pk = (G, \Theta_a, \Theta_b)$ and the secret key is $sk = (x_a, x_b)$. Choose a hash function $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ (a random oracle).

Encryption: Plaintext $\Delta \in \mathbb{G}_M$ can be encrypted by choosing $t_a, t_b \in_R \mathbb{Z}_p^*$ and computing $(E_a, \Lambda_a) = (t_a G, \Delta\Theta_a^{t_a})$, $(E_b, \Lambda_b) = (t_b G, \Delta\Theta_b^{t_b})$ and a non-interactive zero-knowledge proof $\varsigma = (c, \rho_a, \rho_b)$ of equality of plaintexts between (E_a, Λ_a) and (E_b, Λ_b). The proof ς can be computed by choosing $w_a, w_b \in_R \mathbb{Z}_p$ and computing $c = \mathcal{H}_1(G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b||\Lambda_b||w_a G||w_b G||\Theta_a^{w_a}\Theta_b^{w_b})$, $\rho_a = w_a - t_a c$ and $\rho_b = w_b + t_b c$. The ciphertext is $(E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$.

Decryption: Given a ciphertext $(E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$, first check the validity of ς by verifying

$$c \stackrel{?}{=} \mathcal{H}_1(G||\Theta_a||\Theta_b||E_a||\Lambda_a||E_b||\Lambda_b||\rho_a G + cE_a||\rho_b G - cE_b||\Theta_a^{\rho_a}\Theta_b^{\rho_b}(\Lambda_a/\Lambda_b)^c)$$

then compute the plaintext $\Delta = \Lambda_a / e(E_a, G)^{x_a} = \Lambda_b / e(E_b, G)^{x_b}$.

Security: The security of El Gamal^{BP2} system is stated in Theorem 1.

Theorem 1. *El Gamal^{BP2} encryption scheme is IND-CCA if DBDH assumption holds, in the random oracle model.*

3 The Group Signature Scheme

3.1 Overview

Our group signature scheme is built upon two ordinary signature schemes. The first one is used in the Join, Iss protocol for the issuer to generate a signature (a_i, S_i) for each x_i, which is randomly generated by both a member and the issuer, but known only to the member. The second ordinary signature scheme is used in the GSig algorithm as the non-interactive version of a zero-knowledge protocol, that proves the signer's knowledge of (a_i, S_i) and x_i. The security of the two signature schemes underlies the security of the group signature scheme.

Our group signature scheme is constructed in cyclic groups with bilinear mappings. For simplicity, we present the scheme when the groups $\mathbb{G}_1$ and $\mathbb{G}_2$

are the same, however, it can be easily modified for the general case when $\mathbb{G}_1 \neq \mathbb{G}_2$. The users do not perform any pairing operation when signing, but pairing operation play an important role in the verification algorithm GVf. Intuitively, bilinear pairings allow a party, given $A, B, C, D \in \mathbb{G}_1$, to prove that $log_A B = log_C D$ without knowing $log_A B$ or $log_A C$. This is not possible in cyclic groups without bilinear pairings and where the DDH assumption holds.

3.2 Descriptions

We describe our group signature scheme according to the BSZ04 model, which is omitted in this paper due to space limitation. Our group signature scheme consists of two group managers (the issuer and the opener), and users with unique identities $i \in \mathbb{N}$ (the set of positive integers). Each user can join the group and become a group member. The scheme is specified as a tuple $\mathcal{GS}1 =$(GKg, UKg, Join, Iss, GSig, GVf, Open, Judge) of polynomial-time algorithms which are defined as follows. We assume that the group size and the number of queries asked by the adversary are polynomially-bounded by the security parameter l.

GKg: Suppose l is a security parameter and the Bilinear Pairing Instance Generator $\mathcal{G}$ generates a tuple of bilinear pairing parameters $\mathbf{t} = (p, \mathbb{G}_1, \mathbb{G}_M, e, P) \leftarrow \mathcal{G}(1^l)$, that is also the publicly shared parameters. Choose a hash function $\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, which is assumed to be a random oracle in the security proofs. Choose $P_0, G, H \in_R \mathbb{G}_1$, $x, x'_a, x'_b \in_R \mathbb{Z}_p^*$ and compute $P_{pub} = xP$, $\Theta_a = e(G,G)^{x'_a}$ and $\Theta_b = e(G,G)^{x'_b}$. The group public key is $gpk = (P, P_0, P_{pub}, H, G, \Theta_a, \Theta_b)$, the issuing key is $ik = x$, and the opening key is $ok = (x'_a, x'_b)$.

UKg: This algorithm generates keys that provide authenticity for messages sent by the user in the (Join, Iss) protocol. This algorithm is the key generation algorithm K_S of any digital signature scheme (K_S, $Sign$, Ver) that is unforgeable against chosen message attacks (UNF-CMA). A user i runs the UKg algorithm that takes as input a security parameter 1^l and outputs a personal public and private signature key pair $(upk[i], usk[i])$. Public Key Infrastructure (PKI) can be used here. Although any UNF-CMA signature scheme can be used, but using schemes, whose security is based on DBDH or SDH assumptions, will reduce the underlying assumptions of our group signature scheme. One example of such scheme is in [5].

Join, Iss: In this protocol, a user i and the issuer first jointly generate a random value $x_i \in \mathbb{Z}_p^*$ whose value is only known by the user. The issuer then generates (a_i, S_i) for the user so that $e(a_i P + P_{pub}, S_i) = e(P, x_i P + P_0)$. The user uses $usk[i]$ to sign his messages in the protocol. Note that the formal model assumes the communication to be private and authenticated. We also assume that the communication is protected from replay attacks. The protocol is as follows.

1. user $i \longrightarrow$ issuer: $I = yP + rH$, where $y, r \in_R \mathbb{Z}_p^*$.
2. user $i \longleftarrow$ issuer: $u, v \in_R \mathbb{Z}_p^*$.
3. The user computes $x_i = uy + v$, $P_i = x_i P$.

4. user $i \longrightarrow$ issuer: P_i and a proof of knowledge of (x_i, r') such that $P_i = x_i P$ and $vP + uI - P_i = r'H$ (see [12] for this proof).
5. The issuer verifies the proof, then chooses $a_i \in_R \mathbb{Z}_p^*$ different from all corresponding elements previously issued, and computes $S_i = \frac{1}{a_i + x}(P_i + P_0)$.
6. user $i \longleftarrow$ issuer: a_i, S_i.
7. The user computes $\Delta_i = e(P, S_i)$, verifies if $e(a_i P + P_{pub}, S_i) = e(P, x_i P + P_0)$, and stores the *private signing key* $gsk[i] = (x_i, a_i, S_i, \Delta_i)$. Note that only the user knows x_i. The issuer also computes Δ_i and makes an entry in the table reg: $reg[i] = (i, \Delta_i, \langle \mathsf{Join}, \mathsf{Iss} \rangle$ transcript).

GSig: A group signature of a user i shows his knowledge of (a_i, S_i) and a secret x_i such that: $e(a_i P + P_{pub}, S_i) = e(P, x_i P + P_0)$. The signature does not reveal any information about his knowledge to anyone, except for the opener, who can compute Δ_i by decrypting an encryption of that value. The algorithm for a user i to sign a message $m \in \{0,1\}^*$ is as follows.

1. Encrypt Δ_i by El Gamal^{BP2} with public key (G, Θ_a, Θ_b) as $(E_a = tG, \Lambda_a = \Delta_i \Theta_a^t, E_b, \Lambda_b, \varsigma)$.
2. Perform the non-interactive version of a protocol, which we call the Signing protocol, as follows. Generate $r_1, ..., r_3, k_0, ..., k_5 \in_R \mathbb{Z}_p^*$ and compute
   (a) $U = r_1(a_i P + P_{pub})$; $V = r_2 S_i$; $W = r_1 r_2 (x_i P + P_0)$; $X = r_2 U + r_3 H$; $T_1 = k_1 P + k_2 P_{pub} + k_0 H$; $T_2 = k_3 P + k_2 P_0$; $T_3 = k_4 U + k_0 H$; $T_4 = k_5 G - k_4 E_a$; $\Pi = \Theta_a^{k_5} \Lambda_a^{-k_4}$.
   (b) $c = \mathcal{H}_2(gpk||E_a||\Lambda_a||E_b||\Lambda_b||\varsigma||U||V||W||X\ ||T_1||...||T_4||\Pi||m)$.
   (c) Compute in $\mathbb{Z}_p$: $s_0 = k_0 + cr_3$; $s_1 = k_1 + cr_1 r_2 a_i$; $s_2 = k_2 + cr_1 r_2$; $s_3 = k_3 + cr_1 r_2 x_i$; $s_4 = k_4 + cr_2$; $s_5 = k_5 + cr_2 t$.
3. Output the signature $(c, s_0, ..., s_5, U, V, W, X, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ for m.

GVf: The verification algorithm for $m, (c, s_0, ..., s_5, U, V, W, X, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ outputs accept if and only if verifying the proof ς outputs accept and the following two equations hold: $e(U, V) = e(P, W)$ and $c = \mathcal{H}_2(P||P_0||P_{pub}||H||G||\Theta||E_a||\Lambda_a||E_b||\Lambda_b||\varsigma||U||V||W||X||s_1 P + s_2 P_{pub} + s_0 H - cX||s_3 P + s_2 P_0 - cW||s_4 U + s_0 H - cX||s_5 G - s_4 E_a||\Theta_a^{s_5} \Lambda_a^{-s_4} e(P, cV)||m)$.

Open: To open m and its valid signature $(c, s_0, ..., s_5, U, V, W, X, E_a, \Lambda_a, E_b, \Lambda_b, \varsigma)$ to find the signer, the opener performs the following steps.

1. Use GVf algorithm to check the signature's validity. If the algorithm rejects, return $(0, \varepsilon)$, where ε denotes an empty string.
2. Compute $\Delta_i = \Lambda_a e(E_a, G)^{-x'_a}$ and find the corresponding entry i in the table reg. If no entry is found, return $(0, \varepsilon)$.
3. Return $reg[i]$ and a non-interactive zero-knowledge proof ϱ of knowledge of x'_a so that $\Theta_a = e(G, G)^{x'_a}$ and $\Lambda_a / \Delta_i = e(E_a, G)^{x'_a}$ (see [12] for this proof).

Judge: On an output by the Open algorithm for a message m and its signature ω, the Judge algorithm is performed as follows:

1. If Open algorithm outputs $(0, \varepsilon)$, run GVf algorithm on m, ω. If GVf rejects, return accept; otherwise, return reject.
2. If Open algorithm outputs $(reg[i], \varrho)$, return reject if one of the following happens: (i) on m, ω, GVf algorithm rejects; (ii) verification of the proof ϱ rejects; (iii) the $\langle$Join, Iss$\rangle$ transcript is invalid with regard to $upk[i]$; (iv) $\Delta_i \not= e(P, S_i)$ where S_i is extracted from the $\langle$Join, Iss$\rangle$ transcript. Otherwise, return accept.

Remarks:

- Our scheme is trapdoor-free. This improves efficiency and manageability, and various groups can share the same initial set-up $p, \mathbb{G}_1, \mathbb{G}_M, e, P, P_0, G, H$.
- Our Signing protocol achieves honest verifier perfect zero-knowledge and does not rely on any complexity assumption. This indicates a higher level of unconditional security: from a signature, an adversary with unlimited power (but without access to the *reg* table) can compute only a part of the signer's registration information (S_i), whereas, in the ACJT00 and KY04 schemes, the adversary can find all parts of the signer's private signing key.

3.3 Security Proofs

Theorem 2. *The group signature scheme $\mathcal{GS}1$ provides Correctness.*

Theorem 3. *The group signature scheme $\mathcal{GS}1$ provides Anonymity in the random oracle model if the Decisional Bilinear Diffie-Hellman assumption holds.*

Theorem 4. *The group signature scheme $\mathcal{GS}1$ provides Traceability in the random oracle model if the q-Strong Diffie-Hellman assumption holds, where q is the upper bound of the group size.*

Theorem 5. *The group signature scheme $\mathcal{GS}1$ provides Non-frameability in the random oracle model if the Discrete Logarithm assumption holds over the group $\mathbb{G}_1$ and the digital signature scheme $(K_S, Sign, Ver)$ is UNF-CMA.*

Proofs of these theorems can be found in the full version [25]. We provide here the proofs of two important properties that underlie these theorems, i. e. the Zero-knowledge property of the Signing protocol in GSig algorithm and the Coalition-Resistance of $\mathcal{GS}1$ and $\mathcal{GS}2$. In our definition, Coalition-Resistance intuitively means that a colluding group of signers, with the knowledge of the opening key and access to some oracles, should not be able to generate a new valid user private signing key. For a group signature scheme $\mathcal{GS}$, a PPT adversary $\mathcal{A}$, a PPT predicate $\mathcal{U}$ that can determine the validity of a user private signing key, and any security parameter $l \in \mathbb{N}$, the formula of the experiment for Coalition-Resistance is as follows.

Experiment $Exp^{\mathsf{coal.re}}_{\mathcal{GS},\mathcal{A},\mathcal{U}}(l)$
$(gpk, ik, ok) \leftarrow \mathsf{GKg}(1^l)$; $\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$
$gsk' \leftarrow \mathcal{A}(gpk, ok : \mathsf{CrptU}(\cdot,\cdot), \mathsf{SndToI}(\cdot,\cdot), \mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot), \mathsf{USK}(\cdot))$
If $gsk' \in \{gsk[i] |\ i \in \mathsf{CU} \cup \mathsf{HU}\}$ then return 0 else return $\mathcal{U}(gpk, gsk')$

HU is a set of honest users; CU - a set of corrupted users; GSet - a set of message-signature pairs ; $\mathsf{AddU}(\cdot)$ - add user oracle; $\mathsf{CrptU}(\cdot,\cdot)$ - corrupt user oracle; $\mathsf{SndToI}(\cdot,\cdot)$ - send to issuer oracle; $\mathsf{USK}(\cdot)$ - user secret keys oracle; $\mathsf{RReg}(\cdot)$ - read registration table oracle. The group signature scheme $\mathcal{GS}$ provides Coalition-Resistance if the following function $Adv^{\mathsf{coal.re}}_{\mathcal{GS},A,\mathcal{U}}(l)$ is negligible.

$$Adv^{\mathsf{coal.re}}_{\mathcal{GS},A,\mathcal{U}}(l) = \mathsf{Pr}[Exp^{\mathsf{coal.re}}_{\mathcal{GS},A,\mathcal{U}}(l) = 1]$$

Lemma 1. *The interactive Signing protocol underlying the GSig algorithm is a (honest-verifier) perfect zero-knowledge proof of knowledge of* (a_i, S_i), x_i *and* t *such that* $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$, $E_a = tG$ *and* $\Lambda_a = e(P, S_i)\Theta_a^t$.

Proof. The proof for completeness is straightforward. The proofs of Soundness and Zero-knowledge property are as follows.

Soundness: If the protocol accepts with non-negligible probability, we show that the prover must have the knowledge of (a_i, S_i), x_i and t satisfying the relations stated in the theorem. Suppose the protocol accepts for the same commitment $(U, V, W, X, T_1, ..., T_4, \Pi)$, two different pairs of challenges and responses $(c, s_0, ...s_5)$ and $(c', s'_0, ..., s'_5)$. Let $f_i = \frac{s_i - s'_i}{c - c'}, i = 0, ..., 5$, then: $X = f_1P + f_2P_{pub} + f_0H$; $W = f_3P + f_2P_0$; $X = f_4U + f_0H$; $E_a = f_5f_4^{-1}G$; $e(P, V) = \Theta_a^{-f_5}\Lambda_a^{f_4}$; so $U = f_1f_4^{-1}P + f_2f_4^{-1}P_{pub}$.

Let $a_i = f_1f_2^{-1}$, $S_i = f_4^{-1}V$, $x_i = f_3f_2^{-1}$, $t = f_5f_4^{-1}$, then $E_a = tG$, $\Lambda_a = e(P, S_i)\Theta_a^t$ and $e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0)$, as $e(U, V) = e(P, W)$. So the prover have the knowledge of (a_i, S_i), x_i and t satisfying the relations.

Zero-knowledge: The simulator chooses $c, s_0, ...s_5 \in_R \mathbb{Z}_p$, $b \in_R \mathbb{Z}_p^*$, $X, V \in_R \mathbb{G}_1$ and compute $U = bP$, $W = bV$, $T_1 = s_1P + s_2P_{pub} + s_0H - cX$, $T_2 = s_3P + s_2P_0 - cW$, $T_3 = s_4U + s_0H - cX$, $T_4 = s_5G - s_4E_a$ and $\Pi = \Theta_a^{s_5}\Lambda_a^{-s_4}e(P, cV)$. We can see that the distribution of the simulation is the same as the distribution of the real transcript.

Lemma 2. *If the q-SDH assumption holds, then the group signature schemes* $\mathcal{GS}1$ *and* $\mathcal{GS}2$*, whose group sizes are bounded by* q*, provide Coalition-Resistance, where the predicate* $\mathcal{U}$ *is defined as:*

$$\mathcal{U}(\langle P, P_0, P_{pub}, ...\rangle, \langle x_i, a_i, S_i, \Delta_i\rangle) = 1 \Leftrightarrow e(a_iP + P_{pub}, S_i) = e(P, x_iP + P_0).$$

Proof. We prove the lemma for both $\mathcal{GS}1$ and $\mathcal{GS}2$. Suppose there is a PPT adversary $\mathcal{A}$ that can break the Coalition-Resistance property of $\mathcal{GS}1$ or $\mathcal{GS}2$ with respect to the predicate $\mathcal{U}$ defined above. Let the set of private signing keys generated during $\mathcal{A}$'s attack be $\{(x_i, a_i, S_i, \Delta_i)\}_{i=1}^q$ and let his output be a new private signing key $(x^*, a^*, S^*, \Delta^*)$ with non-negligible probability (that means $(a^*, S^*) \notin \{(a_i, S_i)\}_{i=1}^q$). We show a construction of a PPT adversary $\mathcal{B}$ that can break the q-SDH assumption. Suppose a tuple $challenge = (Q, zQ, \ldots, z^qQ)$ is given, where $z \in_R \mathbb{Z}_p^*$; we show that $\mathcal{B}$ can compute $(c, 1/(z+c)Q)$, where $c \in \mathbb{Z}_p$ with non-negligible probability. We consider two cases.

Case 1: This is a trivial case, where $\mathcal{A}$ outputs $S^* \in \{S_1, ..., S_q\}$ with non-negligible probability. In this case, $\mathcal{B}$ chooses $x, x'_a, x'_b \in_R \mathbb{Z}_p^*$ and $G, H \in_R \mathbb{G}_1$, gives $\mathcal{A}$ the group signature public key $(P = Q, P_0 = zQ, P_{pub} = xP, H, G, \Theta_a = e(G,G)^{x'_a}, \Theta_b = e(G,G)^{x'_b})$ and the opening key (x'_a, x'_b) (no x'_b, Θ'_b in case of $\mathcal{GS}2$), and simulates a set of possible users. Then $\mathcal{B}$ can simulate all oracles that $\mathcal{A}$ needs to access. Suppose a set of private signing keys $\{(x_i, a_i, S_i, \Delta_i)\}_{i=1}^q$ is generated and $\mathcal{A}$ outputs a new $(x^*, a^*, S^*, \Delta^*)$ with non-negligible probability such that $S^* \in \{S_1, ..., S_q\}$. Suppose $S^* = S_j$, where $j \in \{1, ..., q\}$, then $\frac{1}{a^*+x}(x^*P + P_0) = \frac{1}{a_j+x}(x_jP + P_0)$, so $(a_j - a^*)P_0 = (a^*x_j - a_jx^* + x_jx - x^*x)P$. Therefore, z is computable by $\mathcal{B}$ from this, and so is $(c, 1/(z+c)Q)$, for any $c \in \mathbb{Z}_p$.

Case 2: This is when the first case does not hold. That means $\mathcal{A}$ outputs $S^* \notin \{S_1, ..., S_q\}$ with non-negligible probability. Then $\mathcal{B}$ plays the following game:

1. Generate $\alpha, a_i, x_i \in_R \mathbb{Z}_p^*$, $i = 1, ..., q$, where a_is are different from one another, then choose $m \in_R \{1, ..., q\}$.
2. Let $x = z - a_m$ ($\mathcal{B}$ does not know x), then the following P, P_{pub}, P_0 are computable by $\mathcal{B}$ from the tuple *challenge*.

$$P = \prod_{i=1, i \neq m}^{q} (z + a_i - a_m)Q$$

$$P_{pub} = xP = (z - a_m) \prod_{i=1, i \neq m}^{q} (z + a_i - a_m)Q$$

$$P_0 = \alpha \prod_{i=1}^{q} (z + a_i - a_m)Q - x_m \prod_{i=1, i \neq m}^{q} (z + a_i - a_m)Q$$

3. Generate $x'_a, x'_b \in_R \mathbb{Z}_p^*$ and $G, H \in_R \mathbb{G}_1$ and give $\mathcal{A}$ the group signature public key $(P, P_0, P_{pub}, H, G, \Theta_a = e(G,G)^{x'_a}, \Theta_b = e(G,G)^{x'_b})$ and the opening key (x'_a, x'_b) (no x'_b, Θ'_b in case of $\mathcal{GS}2$) and simulates a set of possible users.
4. With the capabilities above, $\mathcal{B}$ can simulate oracles $\mathsf{CrptU}(\cdot,\cdot)$, $\mathsf{RReg}(\cdot)$ and $\mathsf{USK}(\cdot)$) that $\mathcal{A}$ needs to access. For $\mathsf{AddU}(\cdot)$ or $\mathsf{SndToI}(\cdot,\cdot)$, $\mathcal{B}$ simulates the addition of an honest or corrupted user i as follows. As playing both sides of the Join, Iss protocol or being able to extract information from $\mathcal{A}$, $\mathcal{B}$ simulates the protocol as specified so that the prepared a_i, x_i above are computed in the protocol to be the corresponding parts of the user i's private signing key. $\mathcal{B}$ can compute S_i as follows:
   - If $i = m$, then $S_m = \frac{1}{a_m+x}(x_mP + P_0) = \alpha \prod_{i=1, i \neq m}^{q} (z + a_i - a_m)Q$. This is computable from the tuple *challenge*.
   - If $i \neq m$, then $S_i = \frac{1}{a_i+x}(x_iP + P_0) = (x_i - x_m) \prod_{j=1, j \neq m, i}^{q} (z + a_j - a_m)Q + \alpha \prod_{j=1, j \neq i}^{q} (z + a_j - a_m)Q$. This is computable from the tuple *challenge*.
5. Get the output $(x^*, a^*, S^*, \Delta^*)$ from $\mathcal{A}$, where $S^* = \frac{1}{a^*+x}(x^*P + P_0) = \frac{1}{z+a^*-a_m}(\alpha z + x^* - x_m) \prod_{i=1, i \neq m}^{q} (z + a_i - a_m)Q$

We can see that the case $\alpha z+x^*-x_m = \alpha(z+a^*-a_m)$ happens with negligible probability, as it results in $S^* = S_m$. So the case $\alpha z + x^* - x_m \not= \alpha(z + a^* - a_m)$ happens with non-negligible probability ϵ_1. Suppose in this case, the probability that $a^* \in \{a_1, ..., a_q\}$ is ϵ_2. Then the probability that $a^* \notin \{a_1, ..., a_q\}\backslash\{a_m\}$ is $\epsilon_1 - \frac{q-1}{q}\epsilon_2$ (as $m \in_R \{1, ..., q\}$), which is also non-negligible if q is polynomially bound by the security parameter l. If $\alpha z + x^* - x_m \not= \alpha(z + a^* - a_m)$ and $a^* \notin \{a_1, ..., a_q\}\backslash\{a_m\}$, then $\frac{1}{z+a^*-a_m}Q$ is computable from the tuple *challenge* and S^* and so $\mathcal{B}$ can compute $(c, \frac{1}{z+c}Q)$, where $c = a^* - a_m$.

4 Variations

4.1 Weak Anonymity Requirement

We introduce this security requirement to account for a class of group signature schemes, including ACJT00 scheme, which can not be proved to achieve Anonymity requirement. Weak Anonymity requirement is defined exactly the same as Anonymity requirement, except that the adversary does not have access to the $\mathsf{Open}(\cdot,\cdot)$ oracle. In practice, when the opener is assumed to be uncorrupted as in Anonymity requirement, it could be hard for the adversary to have access to the Open oracle. As Open oracle is not used in the conventional list of requirements, the same argument as in [4] shows that Weak anonymity, Traceability and Non-frameability are sufficient to imply the conventional list of requirements.

4.2 A Variant Group Signature Scheme, $GS2$

The scheme $\mathcal{GS}2$ is the same as $\mathcal{GS}1$, except that in the signature, Δ_i is encrypted by El Gamal^{BP1} encryption scheme instead of El Gamal^{BP2}. So in GKg, x'_b and Θ_b are not generated and in GSig, Δ_i is encrypted by El Gamal^{BP1} public key (G, Θ_a) as $(E_a = tG, \Lambda_a = \Delta_i\Theta_a^t)$. So there is no E_b, Λ_b or ς in the signature and in the executions of GSig, GVf, Open and Judge algorithms. Security of $\mathcal{GS}2$ is stated in Theorem 6, whose proof is shown in the full version [25].

Theorem 6. *$\mathcal{GS}2$ provides Correctness. $\mathcal{GS}2$ provides Weak Anonymity if the Decisional Bilinear Diffie-Hellman assumption holds. $\mathcal{GS}2$ provides Traceability in the random oracle model if the q-Strong Diffie-Hellman assumption holds, where q is the upper bound of the group size. $\mathcal{GS}2$ provides Non-frameability in the random oracle model if the Discrete Logarithm assumption holds over the group $\mathbb{G}_1$ and the digital signature scheme $(K_S, Sign, Ver)$ is UNF-CMA.*

4.3 Do ACJT00 and $GS2$ Schemes Provide Anonymity?

We first state the security of the ACJT00 scheme in Theorem 7. The ACJT00 scheme refers to the scheme proposed in [1], plus some simple extensions to accommodate the Judge algorithm (defining the UKg algorithm as in our scheme, using $usk[i]$ to sign messages in the Join, Iss protocol, and verifying signatures in the Open and Judge algorithms). The methodology of the proof for Theorem

7 is very similar to the proof of Theorem 6, and the exact details of each step can be extracted from the proofs in [19].

Theorem 7. *The ACJT00 scheme provides Correctness; Weak Anonymity if the DDH-Compo-KF assumption holds; Traceability in the random oracle model if the Strong RSA assumption holds; Non-frameability in the random oracle model if the Discrete Logarithm assumption holds over the quadratic residues group of a product of two known large primes, and the digital signature scheme for* UKg *is UNF-CMA. (See [19] for assumptions used in this theorem).*

It is an open question if the ACJT00 and $\mathcal{GS}2$ schemes provide Anonymity, in line with the open problem whether a combination of an El Gamal encryption (IND-CPA) and a Schnorr proof of knowledge of the plaintext can provide IND-CCA. This combination has been proved to provide IND-CCA in the random oracle model, but the proof has required either another very strong assumption [29] or is in generic model [27]. In ACJT00 and $\mathcal{GS}2$ signatures, the identity-bound information is encrypted by variations of El Gamal encryption and the other part of the signatures proves knowledge of the information. The Open oracle plays a similar role as the Decryption oracle in the model of IND-CCA.

4.4 Variants Based on the DDH Assumption

We can build variants of $\mathcal{GS}1$ and $\mathcal{GS}2$, whose security is based on the DDH assumption over the group $\mathbb{G}_M$ instead of the DBDH (DDHV) assumption. Specifically, Δ_i will be encrypted by the normal El Gamal encryption scheme or the twin-paradigm extension of El Gamal encryption scheme (proposed in [17]). The Open algorithm in these variant schemes requires one less pairing operation than in $\mathcal{GS}1$ and $\mathcal{GS}2$.

We can actually provide a group signature with 4 options, where the users, the issuer and the opener use the same keys for all options. The first two options are $\mathcal{GS}1$ and $\mathcal{GS}2$, offering smaller signature size and more efficient signing and verification. The last two options are the variant schemes based on the normal DDH assumption, with more efficient opening.

5 A Traceable Signature Scheme

We extend $\mathcal{GS}2$ to be a traceable signature scheme $\mathcal{TS}$ =(Setup, Join, Sign, Verify, Open, Reveal, Trace, Claim, $\mathsf{Claim\text{-}Verify}$) with similar advantages over the only other traceable signature scheme [18].

Setup: This is the same as GKg for $\mathcal{GS}2$, but the group public key also includes a $Q \in_R \mathbb{Z}_p^*$. The group public key is $gpk = (P, P_0, P_{pub}, Q, H, G, \Theta_a)$, the issuing key is $ik = x$, and the opening key is $ok = x'_a$. Choose a hash function $\mathcal{H}_3 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ (a random oracle).

Join: This protocol is the same as the Join, Iss protocol in Section 3.2, except for the following. The GM also chooses $\bar{x_i} \in_R \mathbb{Z}_p^*$, computes $S_i = \frac{1}{a_i+x}(P_i+\bar{x_i}Q+P_0)$

at step 5 and sends the user $a_i, S_i, \bar{x}_i$ at step 6. In the last step, the user computes $\Delta_i = e(P, S_i)$, verifies if $e(a_i P + P_{pub}, S_i) = e(P, x_i P + \bar{x}_i Q + P_0)$, and stores the *private signing key* $gsk[i] = (x_i, \bar{x}_i, a_i, S_i, \Delta_i)$. The GM also computes Δ_i and stores it with the protocol's transcript.

Sign: The algorithm for an user i to sign a message $m \in \{0,1\}^*$ is as follows.

1. Compute $E_a = tG$, $\Lambda_a = \Delta_i \Theta_a^t$, $\Upsilon_1 = \Theta_a^{\bar{x}_i r}$, $\Upsilon_2 = \Theta_a^r$, $\Upsilon_3 = \Theta_a^{x_i r'}$ and $\Upsilon_4 = \Theta_a^{r'}$, where $t, r, r' \in_R \mathbb{Z}_p^*$.
2. Generate $r_1, ..., r_3, k_0, ..., k_6 \in_R \mathbb{Z}_p^*$ and compute
   (a) $U = r_1(a_i P + P_{pub})$; $V = r_2 S_i$; $W = r_1 r_2 (x_i P + \bar{x}_i Q + P_0)$; $X = r_2 U + r_3 H$; $T_1 = k_1 P + k_2 P_{pub} + k_0 H$; $T_2 = k_3 P + k_6 Q + k_2 P_0$; $T_3 = k_4 U + k_0 H$; $T_4 = k_5 G - k_4 E_a$; $\Pi = \Theta_a^{k_5} \Lambda_a^{-k_4}$; $\Psi_1 = \Upsilon_1^{-k_2} \Upsilon_2^{k_6}$; $\Psi_2 = \Upsilon_3^{-k_2} \Upsilon_4^{k_3}$.
   (b) $c = \mathcal{H}_3(P||P_0||P_{pub}||H||G||\Theta||E_a||\Lambda_a||E_b||\Lambda_b||\varsigma||U||V||W||X||T_1||...||T_4 ||\Pi||\Psi_1||\Psi_2||m)$.
   (c) Compute in $\mathbb{Z}_p$: $s_0 = k_0 + cr_3$; $s_1 = k_1 + cr_1 r_2 a_i$; $s_2 = k_2 + cr_1 r_2$; $s_3 = k_3 + cr_1 r_2 x_i$; $s_4 = k_4 + cr_2$; $s_5 = k_5 + cr_2 t$; $s_6 = k_6 + cr_1 r_2 \bar{x}_i$
3. Output the signature $(c, s_0, ..., s_6, U, V, W, X, E_a, \Lambda_a, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ for m.

Verify: The verification algorithm for $m, (c, s_0, ..., s_6, U, V, W, X, E_a, \Lambda_a, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ outputs accept if and only if the following two equations hold: (i) $e(U, V) = e(P, W)$ and (ii) $c = \mathcal{H}_3(P||P_0||P_{pub}||H||G||\Theta||E_a||\Lambda_a||E_b||\Lambda_b||\varsigma||U||V||W||X ||s_1 P + s_2 P_{pub} + s_0 H - cX||s_3 P + s_6 Q + s_2 P_0 - cW||s_4 U + s_0 H - cX||s_5 G - s_4 E_a ||\Theta_a^{s_5} \Lambda_a^{-s_4} e(P, cV)||\Upsilon_1^{-s_2} \Upsilon_2^{s_6}||\Upsilon_3^{-s_2} \Upsilon_4^{s_3}||m)$

Open: To open m and its valid signature $(c, s_0, ..., s_5, U, V, W, X, E_a, \Lambda_a, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ to find the signer, the GM computes $\Delta_i = \Lambda_a e(E_a, G)^{-x'_a}$ and finds the corresponding entry i in the table of stored Join transcripts. The GM returns i and a non-interactive zero-knowledge proof ϱ of knowledge of x'_a so that $\Theta_a = e(G, G)^{x'_a}$ and $\Lambda_a / \Delta_i = e(E_a, G)^{x'_a}$ (see [12] for this proof).

Reveal and Trace: Given the Join transcript of user i, the GM recovers the tracing trapdoor $trace_i = \bar{x}_i$. Given $trace_i$ and a message-signature pair, a designated party recovers Υ_1 and Υ_2 and checks if $\Upsilon_1 = \Upsilon_2^{\bar{x}_i}$. If the equation holds, the tracer concludes that user i has produced the signature.

Claim and Claim-Verify: Given a message-signature pair, a user i can claim that he is the signer by recovering Υ_3 and Υ_4 and producing a non-interactive proof of knowledge of the discrete-log of Υ_3 base Υ_4. Any party can run Claim-Verify by verifying the signature and the proof.

Security. The security of $\mathcal{TS}$ is stated in Theorem 8. The proof of this theorem uses techniques similar to those in [18] and arguments similar to the proofs for our group signature schemes.

Theorem 8. *In the random oracle model, $\mathcal{TS}$ provides (i) security against misidentification attacks based on the q-SDH and the DDH assumptions, where q is the upper bound of the group size; (ii) security against anonymity attacks*

based on the DBDH and DDH assumptions; (iii) security against framing attacks based on the DL assumption.

6 Efficiency

The sizes of signatures and keys in our schemes are much shorter than those used in the Strong-RSA-based schemes at a similar level of security. This difference grows when higher level of security is required. In this section, we compare sizes in our new group signature schemes with those in ACJT00 scheme. We assume that our scheme is implemented using an elliptic curve or hyperelliptic curve over a finite field. p is a 170-bit prime, $\mathbb{G}_1$ is a subgroup of an elliptic curve group or a Jacobian of a hyperelliptic curve over a finite field of order p. $\mathbb{G}_M$ is a subgroup of a finite field of size approximately 2^{1024}. A possible choice for these parameters can be found in [8], where $\mathbb{G}_1$ is derived from the curve $E/GF(3^\iota)$ defined by $y^2 = x^3 - x + 1$. We assume that system parameters in ACJT00 scheme are $\epsilon = 1.1$, $l_p = 512$, $k = 160$, $\lambda_1 = 838$, $\lambda_2 = 600$, $\gamma_1 = 1102$ and $\gamma_2 = 840$. We summarize the result in Table 1.

Table 1. Comparison of sizes (in Bytes)

	Signature	gpk	gsk	ik	ok	Security
ACJT00	1087	768	370	128	128	Weak Anonymity
$\mathcal{GS}1$	597	363	192	22	44	Anonymity
$\mathcal{GS}2$	384	235	192	22	22	Weak Anonymity

Acknowledgements. Authors thank anonymous referees of Asiacrypt 2004 for constructive comments and Fangguo Zhang for helpful discussions.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. CRYPTO 2000, Springer-Verlag, LNCS 1880, pp. 255-270.
2. G. Ateniese, and B. de Medeiros. Efficient Group Signatures without Trapdoors. ASIACRYPT 2003, Springer-Verlag, LNCS 2894, pp. 246-268.
3. G. Ateniese, and B. de Medeiros. Security of a Nyberg-Rueppel Signature Variant. Cryptology ePrint Archive, Report 2004/093, http://eprint.iacr.org/.
4. M. Bellare, H. Shi, and C. Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. Cryptology ePrint Archive: Report 2004/077.
5. D. Boneh, and X. Boyen. Short Signatures Without Random Oracles. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 56-73.
6. D. Boneh, and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 223-238.

7. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. CRYPT0 2004, Springer-Verlag, LNCS, to appear.
8. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. ASIACRYPT 2001, Springer-Verlag, LNCS 2248, pp.514-532.
9. J. Camenisch. Efficient and generalized group signatures. EUROCRYPT 1997, Springer-Verlag, LNCS 1233, pp. 465-479.
10. J. Camenisch, and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. CRYPTO 2002, Springer-Verlag, LNCS 2442, pp. 61-76.
11. J. Camenisch, and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. CRYPTO 2004, Springer-Verlag, LNCS, to appear.
12. J. Camenisch, and M. Michels. A group signature scheme with improved efficiency. ASIACRYPT 1998, Springer-Verlag, LNCS 1514.
13. J. Camenisch, and M. Stadler. Efficient group signature schemes for large groups. CRYPTO 1997, Springer-Verlag, LNCS 1296.
14. D. Chaum, and E. van Heyst. Group signatures. CRYPTO 1991, LNCS 547, Springer-Verlag.
15. L. Chen, and T. P. Pedersen. New group signature schemes. EUROCRYPT 1994, Springer-Verlag, LNCS 950, pp. 171-181.
16. A. Fiat, and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. CRYPTO 1986, Springer-Verlag, LNCS 263, pp. 186-194.
17. P. Fouque and D. Pointcheval, Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks, ASIACRYPT 2001, Springer-Verlag, LNCS 2248, pp. 351-368.
18. A. Kiayias, Y. Tsiounis and M. Yung. Traceable Signatures. EUROCRYPT 2004, Springer-Verlag, LNCS 3027, pp. 571-589.
19. A. Kiayias, and Moti Yung. Group Signatures: Provable Security, Efficient Constructions and Anonymity from Trapdoor-Holders. Cryptology ePrint Archive: Report 2004/076.
20. J. Killian, and E. Petrank. Identity escrow. CRYPTO 1998, Springer-Verlag, LNCS 1642, pp. 169-185.
21. S. Kim, S. Park, and D. Won. Convertible group signatures. ASIACRYPT 1996, Springer-Verlag, LNCS 1163, pp. 311-321.
22. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. SAC 1999, Springer-Verlag, LNCS 1758.
23. M. Michels. Comments on some group signature schemes. TR-96-3-D, Department of Computer Science, University of Technology, Chemnitz-Zwickau, Nov. 1996.
24. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. IEICE Trans. Vol. E85-A, No.2, pp. 481-484, 2002.
25. L. Nguyen, and R. Safavi-Naini. Efficient and Provably Secure Trapdoor-free Group Signature Schemes from Bilinear Pairings. Full version.
26. L. Nguyen. Accumulators from Bilinear Pairings and Applications. CT-RSA 2005, Springer-Verlag, LNCS, to appear.
27. P. Schnorr and M. Jakobsson. Security of signed El Gamal encryption. ASIACRYPT 2000, Springer-Verlag, LNCS 1976, pp. 73-89.
28. V. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. DRM Workshop 2003.
29. Y. Tsiounis and M. Yung. On the security of El Gamal based encryption. PKC 1998, Springer-Verlag, LNCS 1431, pp. 117-134.
30. F. Zhang, R. Safavi-Naini and W. Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. PKC 2004, Springer-Verlag, LNCS 2947, pp. 277-290.

On the Security of MOR Public Key Cryptosystem

In-Sok Lee[1,⋆,†], Woo-Hwan Kim[1,⋆,†], Daesung Kwon[2], Sangil Nahm[3,†], Nam-Seok Kwak[1,⋆,†], and Yoo-Jin Baek[4,†]

[1] ISaC, Department of Mathematics, Seoul National Univ., Seoul, 151-747, Korea
{islee, whkim, kwarc}@math.snu.ac.kr
[2] National Security Research Institute (NSRI), Taejon, 305-350, Korea
ds_kwon@etri.re.kr
[3] Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA
snahm@purdue.edu
[4] Multimedia Lab., Samsung Electronics Co., Suwon, 442-742, Korea
yoojin.baek@samsung.com

Abstract. For a finite group G to be used in the MOR public key cryptosystem, it is necessary that the discrete logarithm problem(DLP) over the inner automorphism group $\mathrm{Inn}(G)$ of G must be computationally hard to solve. In this paper, under the assumption that the special conjugacy problem of G is easy, we show that the complexity of the MOR system over G is about $\log|G|$ times larger than that of DLP over G in a generic sense. We also introduce a group-theoretic method, called the group extension, to analyze the MOR cryptosystem. When G is considered as a group extension of H by a simple abelian group, we show that DLP over $\mathrm{Inn}(G)$ can be 'reduced' to DLP over $\mathrm{Inn}(H)$. On the other hand, we show that the reduction from DLP over $\mathrm{Inn}(G)$ to DLP over G is also possible for some groups. For example, when G is a nilpotent group, we obtain such a reduction by the *central commutator attack*.

Keywords: MOR cryptosystem, discrete logarithm problem, group extension, central commutator attack.

1 Introduction

At Crypto 2001, Paeng et al. [8] proposed the MOR public key cryptosystem using finite non-abelian groups. For a group G to be used in the MOR public key cryptosystem, it is necessary that the discrete logarithm problem(DLP) over the inner automorphism group $\mathrm{Inn}(G)$ of G must be computationally hard to solve, and there must be an efficient way to represent group elements as products of the specified generators of G. Furthermore, we expect the security of the MOR system to be something 'mor(e)' than that of DLP over G. Also it should be

⋆ Supported in part by KRF grant #2004-070-C00001 and BK21 Project in 2004.
† Partially supported by NSRI.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 387–400, 2004.

noted that the difficulty of DLP depends not only on the algebraic structure of the group, but also on how elements of the group are represented.

Despite of many cryptographic advantages(see [8]) of the MOR cryptosystem, the groups proposed so far have turned out to be unsatisfactory(see [7, 9, 14]).

In this paper, we are not trying to suggest new candidates for the groups G to be used in the MOR cryptosystem. We would rather intend to reveal the reasons why it is not easy to find *good* candidates for G. Thus, we hope that this paper helps searching for suitable groups for the MOR system.

First, in Section 2, we compute the complexity of finding the secret keys of MOR system in a generic sense. Under the assumption that the special conjugacy problem of G is easy, we show that the complexity of MOR system over G is about $\log|G|$ times larger than that of DLP over G in a generic sense. This result is somewhat unexpected, since our *intuitive* expectation for the generic complexity of MOR system is about $|Z(G)|$ times larger than that of DLP over G.

Next, in Section 3, using the well-known theory of group extensions, we show that it is possible to 'reduce' the problem of finding the secret keys of MOR system over G to that of the MOR system over (smaller) subgroups H of G. Our method is a generalization of various attacks given in [7, 9, 14].

In Section 4, we intend to find a reduction algorithm, which reduces MOR system over G to DLP over G. (If this reduction were efficient enough, MOR system would have less advantage in security than other public key cryptosystem based on DLP over G.) We show that this reduction is possible for the groups which are nilpotent or 'nearly' nilpotent. We call our reduction the *central commutator attack* and we note that this attack is generic.

In this paper, we use the following standard notations: If N is a normal subgroup of G and $g \in G$, the order of g is denoted by $|g|$ and the image of g in G/N is denoted by $\overline{g}$. We let $\mathrm{Inn}(g)$ be the inner automorphism of G induced by g, that is,

$$\mathrm{Inn}(g)(x) = g^{-1}xg, \qquad (x \in G)$$

and we let $\mathrm{Inn}(G) = \{\mathrm{Inn}(g) \mid g \in G\}$ be the subgroup of inner automorphisms in $\mathrm{Aut}(G)$. We note that $\mathrm{Inn}(G) \approx G/Z(G)$, where

$$Z(G) = \{z \in G \mid zg = gz \text{ for all } g \in G\}$$

is the center of G.

2 MOR Cryptosystem

2.1 Description of MOR Cryptosystem

The MOR cryptosystem [8] is described as follows.

– Bob's Public key: $(\mathrm{Inn}(g),\ \mathrm{Inn}(g^s))$
– Bob's Secret key: An integer $s (\mathrm{mod}\,|\overline{g}|)$, where $\overline{g} \in G/Z(G)$

It should be noted that for a fixed generating set $\{\gamma_i \mid i \in I\}$ of G, a public key $(\mathrm{Inn}(g),\ \mathrm{Inn}(g^s)) = (\varphi, \varphi^s)$ is described by the data $\{\varphi(\gamma_i)\}$ and $\{\varphi^s(\gamma_i)\}$.

Encryption

1. Alice chooses a random integer r and computes $(\mathrm{Inn}(g^s))^r = \mathrm{Inn}(g^{sr})$.
2. Alice computes $E = \mathrm{Inn}(g^{sr})(M)$.
3. Alice computes $\mu = (\mathrm{Inn}(g))^r = \mathrm{Inn}(g^r)$.
4. Alice sends (E, μ) to Bob.

Decryption

1. Bob computes $\mu^{-s} = \mathrm{Inn}(g^{-sr})$.
2. Bob recovers $M = \mu^{-s}(E)$.

2.2 MOR Cryptosystem and Related Problems

For simplicity, let us write $\mathrm{DLP}(G)$ for DLP over G. Thus $\mathrm{DLP}(\mathrm{Inn}(G))$ stands for DLP over the inner automorphism group $\mathrm{Inn}(G)$ of G.

The security of MOR system is related with the following problems:

– [Special Conjugacy Problem]: For a given $\varphi \in \mathrm{Inn}(G)$, find $h \in G$ such that $\mathrm{Inn}(h) = \varphi$.

– [$\mathrm{DLP}(\mathrm{Inn}(G))$]: Given $\varphi, \varphi^s \in \mathrm{Inn}(G)$ for some $s \in \mathbb{Z}$, find $s (\mathrm{mod}\, |\varphi|)$.

Throughout this paper, let us assume(agree(?)) that the special conjugacy problems over G are not hard to solve. (Otherwise, one can exploit the cryptosystem using the hardness of the special conjugacy problem over G.) Therefore, for given $\mathrm{Inn}(g)$, we may find $g' \in G$ satisfying $\mathrm{Inn}(g) = \mathrm{Inn}(g')$. It means that $g' = gz$ for some $z \in Z(G)$. In this case, $\mathrm{DLP}(\mathrm{Inn}(G))$ can be restated as follows:

$$\text{Find an integer } s (\mathrm{mod}\, |\overline{g}|) \text{ for given } g,\ g^s z \in G, \text{ where } z \in Z(G),$$

or

$$\text{Find an integer } s (\mathrm{mod}\, |\overline{g}|) \text{ for given } \overline{g},\ \overline{g}^{\,s} \in G/Z(G).$$

It means that $\mathrm{DLP}(\mathrm{Inn}(G))$ is equivalent to $\mathrm{DLP}(G/Z(G))$.

In particular, if $|Z(G)|$ is sufficiently large, there is little possibility that $g^s z$ is contained in the cyclic subgroup $\langle g \rangle$ for a randomly chosen $z \in Z(G)$. Hence, existing algorithms for solving $\mathrm{DLP}(G)$ do not seem to be directly applied to $\mathrm{DLP}(\mathrm{Inn}(G))$. On the contrary, if $|Z(G)|$ is too large, then $\mathrm{Inn}(G)$ becomes too small to be used for MOR system. Therefore, we conclude that the appropriate size of $Z(G)$ is crucial in MOR system.

2.3 Central Attack

The crucial role of $Z(G)$ gives rise to the following *intrinsic* attack against MOR system.

Assume that $|Z(G)| = m$ is known. For given g and $g^s z$ for some $s \in \mathbb{Z}$ and $z \in Z(G)$, we get $h_1 = g^m$ and $h_2 = (g^s z)^m = (g^m)^s$. Now, solving $\mathrm{DLP}(\langle g^m \rangle)$

or DLP(G), we get $s(\mathrm{mod}\,|g^m|)$, which gives a partial information of the secret key s. Of course, g^m may be the identity of G in the extreme case(for example, see [8, p. 477]).

2.4 Complexity of Generic Algorithm on MOR System

Since middle of 90's, a lot of works [11, 4, 5, 6] have been done on generic algorithms for DLP and their lower bounds of complexity. Algorithms which do not exploit any particular property of representations of the group are called generic, and the baby-step giant-step algorithm is one of the generic algorithms for DLP. In generic algorithms for DLP, only group operations and equality tests are used.

Let $\{\gamma_i \mid i \in I\}$ be a given generating set of G for MOR system, and a public key (φ, φ^s) be given by $\{\varphi(\gamma_i)\}$ and $\{\varphi^s(\gamma_i)\}$. Assuming that the special conjugacy problem over G is not difficult as before, we get g and $g^s z$ for some unknown $z \in Z(G)$.

Let $Mul_G(\cdot,\cdot)$, $Inv_G(\cdot)$ and $Equ_G(\cdot,\cdot)$ denote the group operation (multiplication and inversion) oracles and the equality test oracle of G, respectively. Now, consider the factor group $G/Z(G)$. The generic operations of $G/Z(G)$ can be *realized* using those of G as follows.

- Group operation oracle of $G/Z(G)$:

$$Mul_{G/Z(G)}(g_1, g_2) = Mul_G(g_1, g_2),$$
$$Inv_{G/Z(G)}(g) = Inv_G(g).$$

- Equality test oracle of $G/Z(G)$:

$$Equ_{G/Z(G)}(g_1, g_2) = \begin{cases} \texttt{True} & (\text{if } g_1 g_2^{-1}\gamma_i = \gamma_i g_1 g_2^{-1} \text{ for all } i \in I), \\ \texttt{False} & (\text{otherwise}). \end{cases}$$

One equality test in $G/Z(G)$ requires at most $(2|I|+1)$ calls of Mul_G, 1 call of Inv_G and $|I|$ calls of Equ_G. Under the assumption that $|I| = O(\log|G|)$, we have the following result as a direct application of the Pohlig-Hellman algorithm in [10].

Theorem 1. *Let a public key of MOR system* $(\mathrm{Inn}(g),\ \mathrm{Inn}(g^s))$ *be given, and let* $|\overline{g}| = \prod_{i=1}^{k} p_i^{e_i}$, *where* p_i *are distinct primes. Under the assumption that* $|I| = O(\log|G|)$ *and that the special conjugacy problem over* G *is easy, the secret key* s *can be computed by* $O(\sum e_i(\log|\overline{g}| + p_i)\log|G|)$ *group operations and equality tests of group elements. If a memory space for storing* $\lceil\sqrt{p}\rceil$ *group elements(where* p *is the largest prime factor of* $|\overline{g}|$*) is available, the running time can be reduced to* $O(\sum e_i(\log|\overline{g}| + \sqrt{p_i}\log p_i)\log|G|)$.

Proof. By the above discussion, one equality test between two elements of $G/Z(G)$ requires $O(\log|G|)$ group operations and equality tests of elements of G. The second assertion follows directly from [10]. □

Thus, in a generic sense, the complexity of computing the secret key of MOR system is about $\log|G|$ times larger than that of solving DLP(G).

This result is somewhat unexpected, since our *intuitive* expectation for the generic complexity of MOR system is about $|Z(G)|$ times larger than that of DLP over G. (If the equality test oracle of $G/Z(G)$ were; "check if $g_1 = g_2 z$ for each $z \in Z(G)$", then we would obtain the result matching our *intuition*. So, the point is that one equality test between two elements of $G/Z(G)$ requires *only* $O(\log|G|)$ group operations and equality tests of elements of G.)

3 Group Extensions and MOR Cryptosystem

Since it does not seem easy to find a *good* candidate for MOR cryptosystem from the list of well-known finite groups, we consider an *inductive argument* as follows. Suppose that the group G is *good* for MOR system, and suppose that G has the smallest order among *good* candidates. Then we think of G as a group extension of a maximal normal subgroup H of G, which is *not suitable* for MOR system by the hypothesis.

In this section, generalizing the various ideas of [7, 9, 14], we show that it is possible to w-reduce(see the definition below) DLP(Inn(G)) to DLP(Inn(H)), where H is a maximal normal subgroup of G.

Definition 2. Given $\varphi, \varphi^s \in \mathrm{Inn}(G)$ with a secret key $s(\mathrm{mod}\,|\varphi|)$, if we can compute ψ, ψ^s for some $\psi \in \mathrm{Inn}(H)$, we say DLP(Inn(G)) can be *w-reduced* (weakly-reduced) to DLP(Inn(H)). In this case, note that we can recover $s(\mathrm{mod}\,|\psi|)$, provided DLP(Inn(H)) is not hard to solve. (Of course, $|\psi|$ may be 1 in the extreme case.)

Although the theory of group extension(see, for example, [2, § 15.1] or [13, § 2.7]) is quite standard and well-known, we briefly sketch the proofs for some results of group extensions to prepare for our proof of Theorem 10.

3.1 Group Extensions

Definition 3. For given two groups H and F, if $H \triangleleft G$ and $G/H \cong F$, then we call G *a group extension of H by F.*

Theorem 4. *(See [2, 13].) If G is a group extension of H by F, there exist functions $T : F \to \mathrm{Aut}(H)$ and $f : F \times F \to H$ satisfying the following conditions:*

(1) $T(\tau) \circ T(\sigma) = \mathrm{Inn}(f(\sigma,\tau)) \circ T(\sigma\tau)$, *for* $\sigma, \tau \in F$,
(2) $f(\sigma,\tau\rho)\, f(\tau,\rho) = f(\sigma\tau,\rho)\, T(\rho)(f(\sigma,\tau))$, *for* $\sigma,\tau,\rho \in F$,
(3) $f(1,1) = 1$.

Proof. Let $t : F \to G$ give rise to a bijection between F and a complete set of coset representatives of H in G such that $t(1) = 1$ (t is called a transversal). Next, we define two functions $T : F \to \mathrm{Aut}(H)$ and $f : F \times F \to H$ by

(a) $T(\sigma)(h) = t(\sigma)^{-1} h\, t(\sigma)$, for $\sigma \in F,\ h \in H$,
(b) $f(\sigma, \tau) = t(\sigma\tau)^{-1}\, t(\sigma)\, t(\tau)$, for $\sigma, \tau \in F$.

Then, T and f satisfy the conditions (1)–(3). □

Remark 5. If T and f satisfy the conditions (1)–(3) of Theorem 4, then we call f *a factor set belonging to* T. If a factor set f is obtained from G as (a) and (b) in the proof of Theorem 4, then we call f *a factor set associated with the extension* G.

Theorem 6. *(See [2, 13].) Let $f : F \times F \to H$ be a factor set belonging to $T : F \to \mathrm{Aut}(H)$. Then there exists a group G which is a group extension of H by F such that f is a factor set associated with G.*

Proof. Put $G = \{\, t(\sigma)a \mid \sigma \in F,\ a \in H\}$ and define a binary operation $*$ on G by

$$[t(\sigma)a] * [t(\tau)b] = t(\sigma\tau)\, f(\sigma, \tau)\, T(\tau)(a)\, b, \qquad (\sigma, \tau \in F,\ a, b \in H).$$

Then, G becomes a group extension of H by F. Moreover, $t(\sigma)1$ is actually a transversal and (T, f) satisfies the conditions (a) and (b) in the proof of Theorem 4. □

Corollary 7. *(See [2, 13].) The group extension G is uniquely determined by T and f. In this case, we denote $G = [H, F, T, f]$.*

We note that semi-direct products are group extensions with the trivial factor sets. In [7, 9], it is shown that DLP over inner automorphism groups of semi-direct products can be reduced to DLP over inner automorphism groups of individual groups. For group extensions, a similar result can be derived.

Theorem 8. *Assume the group extension data $G = [H, F, T, f]$ is known. If F is non-abelian, then* $\mathrm{DLP}(\mathrm{Inn}(G))$ *can be w-reduced to* $\mathrm{DLP}(\mathrm{Inn}(F))$.

Proof. Let $\varphi = \mathrm{Inn}(g)$ and $g = t(\sigma)a$, where $\sigma \in F,\ a \in H$. For any $x = t(\tau)b \in G$, we have

$$\begin{aligned}
\varphi(x) &= [(t(\sigma)a)^{-1}] * [t(\tau)b] * [t(\sigma)a] \\
&= [t(\sigma^{-1})d] * [t(\tau)b] * [t(\sigma)a], \quad (\text{where } T(\sigma)(d) = f(\sigma^{-1}, \sigma)^{-1} a^{-1}) \\
&= [t(\sigma^{-1}\tau)\, f(\sigma^{-1}, \tau)\, T(\tau)(d)\, b] * [t(\sigma)a] \\
&= t(\sigma^{-1}\tau\sigma)\, f(\sigma^{-1}\tau, \sigma) \cdot T(\sigma)(f(\sigma^{-1}, \tau)\, T(\tau)(d)\, b) \cdot a.
\end{aligned}$$

Similarly there exists $A \in H$ such that $\varphi^s(x) = t(\sigma^{-s}\tau\sigma^s)A$. Let $\Psi = \mathrm{Inn}(\sigma)$. Then, the problem of finding s from given $\varphi, \varphi^s \in \mathrm{Inn}(G)$ can be w-reduced to that of finding s from $\Psi, \Psi^s \in \mathrm{Inn}(F)$. □

Theorem 8 implies that the smaller order $\overline{\sigma} \in F/Z(F)$ has, the less information about s is exposed. Therefore, it is reasonable to take F to be abelian. The next theorem is useful when we investigate group extensions by finite cyclic groups.

Theorem 9. *(See [2, § 15.3].) If G is a group extension of H by $\mathbb{Z}_n$, then G is uniquely determined by $\chi \in \mathrm{Aut}(H)$ and $\alpha \in H$ satisfying the following conditions:*

(1) $\chi^n = \mathrm{Inn}(\alpha) \in \mathrm{Inn}(H)$,
(2) $\chi(\alpha) = \alpha$.

Proof. Write $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$. We choose a coset representative $\overline{1}$ of 1, and define a transversal $t : \mathbb{Z}_n \to G$ by $t(i) = \overline{1}^i$ for $0 \le i \le n-1$. Then, $\overline{1}^n = \alpha$ for some $\alpha \in H$. Therefore $\chi := \mathrm{Inn}(\overline{1})|_H \in \mathrm{Aut}(H)$. Then χ and α satisfy conditions (1) and (2). Conversely, if χ and α are given, we define $T : \mathbb{Z}_n \to \mathrm{Aut}(H)$ and $f : \mathbb{Z}_n \times \mathbb{Z}_n \to H$ by

$$T(i) = \chi^i, \qquad (0 \le i \le n-1)$$
$$f(i,j) = \begin{cases} 1 & \text{if } i+j < n, \\ \alpha & \text{if } i+j \ge n. \end{cases}$$

Then T and f satisfy the conditions (1)–(3) of Theorem 4. □

3.2 MOR System and Group Extensions

Let G be given by a group extension of H by F. The case, for which F is non-abelian, is not desirable since $\mathrm{DLP}(\mathrm{Inn}(G))$ can be w-reduced to $\mathrm{DLP}(\mathrm{Inn}(F))$ by Theorem 8.

Furthermore, since every finite group has a composition series, we may regard G as a group extended by finite simple groups for finitely many times. Therefore, in this section, we analyze the case when $F = \mathbb{Z}_p$ for some prime p. Now we have the main result of the present section.

Theorem 10. *If the group extension data $G = [H, \mathbb{Z}_p, T, f]$ is known, then* $\mathrm{DLP}(\mathrm{Inn}(G))$ *can be w-reduced to* $\mathrm{DLP}(\mathrm{Inn}(H))$.

Proof. Let $G = [H, \mathbb{Z}_p, T, f]$. Then, by Theorem 9, there exist $\chi \in \mathrm{Aut}(H)$ and $\alpha \in H$ satisfying the following conditions:

$$T(i) = \chi^i, \qquad (0 \le i < p),$$
$$f(i,j) = \begin{cases} 1 & \text{if } i+j < p, \\ \alpha & \text{if } i+j \ge p, \end{cases}$$
$$\chi^p = \mathrm{Inn}(\alpha) \in \mathrm{Inn}(H).$$

Now, we compute $Z(G)$. If $t(i)a \in Z(G)$, then for all $j \in \mathbb{Z}_p$ and $b \in H$, we have

$$[t(i)a] * [t(j)b] = [t(j)b] * [t(i)a].$$

Therefore,

$$t(i+j)\, f(i,j)\, \chi^j(a)\, b = t(j+i)\, f(j,i)\, \chi^i(b)\, a$$

and hence this implies $\chi^j(a) = a$ and $b = a^{-1}\,\chi^i(b)\,a$. Note that this is equivalent to $\chi(a) = a$ and $\chi^i = \mathrm{Inn}(a^{-1})$. Hence we conclude that

$$Z(G) = \{\, t(i)a \mid \chi(a) = a,\ \chi^i = \mathrm{Inn}(a^{-1})\}.$$

Since $\chi^p = \mathrm{Inn}(\alpha) \in \mathrm{Inn}(H)$ and p is prime, we note that the order of $\overline{\chi}$ in $\mathrm{Out}(H) = \mathrm{Aut}(H)/\mathrm{Inn}(H)$ is 1 or p.

Case 1. $|\overline{\chi}| = 1$.

We prove this case by showing that there is a *computable* isomorphism between $G/Z(G)$ and $H/Z(H)$. If $|\overline{\chi}| = 1$, then $\chi = \mathrm{Inn}(h)$ for some $h \in H$. Since $\chi^i = \mathrm{Inn}(h^i) = \mathrm{Inn}(a^{-1})$, there exists $z_i \in Z(H)$ such that $h^i = a^{-1}z_i$ (i.e., $h^i a \in Z(H)$). Then h commutes with a and thus $\chi(a) = \mathrm{Inn}(h)(a) = a$. Therefore,

$$Z(G) = \{\, t(i)a \mid h^i a \in Z(H)\}$$

and we have

$$|Z(G)| \geq |Z(H)|.$$

Next, we find an isomorphism between $G/Z(G)$ and $H/Z(H)$. Since $\chi^p = \mathrm{Inn}(h^p) = \mathrm{Inn}(\alpha)$, we have $\alpha = h^p z$ for some $z \in Z(H)$. We define $\Psi : G \to H/Z(H)$ by

$$\Psi(t(i)a) = \overline{h^i a}, \qquad (a \in H,\ i \in \mathbb{Z}_p).$$

Then we can show the followings.

1. Ψ is a group homomorphism:

$$\begin{aligned}
&\Psi([t(i)a] * [t(j)b]) = \Psi\left(t(i+j)\, f(i,j)\, \chi^j(a)\, b\right) \\
&= \begin{cases} \overline{h^{i+j}\,\chi^j(a)\,b} = \overline{h^{i+j}\,h^{-j}\,a\,h^j\,b} = \overline{h^i a}\ \overline{h^j b}, & \text{if } i+j < p \\ \overline{h^{i+j-p}\,h^p\,z\,\chi^j(a)\,b} = \overline{z\,h^i\,a\,h^j\,b} = \overline{h^i a}\ \overline{h^j b}, & \text{if } i+j \geq p \end{cases} \\
&= \Psi(t(i)a)\,\Psi(t(j)b).
\end{aligned}$$

2. Ψ is surjective: For $\overline{g} \in H/Z(H)$, where $g \in H$, we have

$$\Psi(t(i)h^{-i}g) = \overline{h^i h^{-i} g} = \overline{g}.$$

3. $\mathrm{Ker}\,\Psi = Z(G)$: $t(i)a \in \mathrm{Ker}\,\Psi \Leftrightarrow h^i a \in Z(H) \Leftrightarrow t(i)a \in Z(G)$.
Hence, by the first isomorphism theorem, we have

$$\overline{\Psi} : G/Z(G) \xrightarrow{\approx} H/Z(H).$$

Note that $\overline{\Psi}$ is *computable* since h can be derived from $\chi = \mathrm{Inn}(h)$.

Case 2. $|\overline{\chi}| = p$.

If $|\overline{\chi}| = p$, i should be 0 in order that the equation $\chi^i = \mathrm{Inn}(a^{-1})$ holds. Moreover, since $\chi^0(b) = b = aba^{-1}$ for all $b \in H$, a must be contained in $Z(H)$. Therefore, we have

$$Z(G) = \{\, t(0)a \mid \chi(a) = a,\ a \in Z(H)\} \leq Z(H).$$

For given $\mathrm{Inn}(t(i)a)$ and $\mathrm{Inn}\left((t(i)a)^s\right)$, under the assumption that the special conjugacy problem of G is easy, we can find $t(j)c$ and $t(l)d$ such that $\mathrm{Inn}(t(i)a) = \mathrm{Inn}(t(j)c)$ and $\mathrm{Inn}\left((t(i)a)^s\right) = \mathrm{Inn}(t(l)d)$. Then we must have $i \equiv j \,(\mathrm{mod}\, p)$ and $c = az$ for some $z \in Z(H)$ with $\chi(z) = z$. Similarly, we get $is \equiv l \,(\mathrm{mod}\, p)$. Consequently, we obtain $s \equiv r' \,(\mathrm{mod}\, p)$ and thus we may put $s = pr + r'$ for some integer r. Since

$$\begin{aligned}
(t(i)a)^p &= \overbrace{[t(i)a] * [t(i)a] * \cdots * [t(i)a]}^{p\text{-times}} \\
&= \overbrace{[t(i)a] * [t(i)a] * \cdots * [t(i)a]}^{(p-1)\text{-times}} * [t(2i)\, f(i,i)\, T(i)(a)\, a] \\
&= \overbrace{[t(i)a] * \cdots * [t(i)a]}^{(p-2)\text{-times}} * [t(3i)\, f(i,2i)\, T(2i)(a)\, f(i,i)\, T(i)(a)\, a] \\
&= t(0) \prod_{j=0}^{p-1} f(i,ij)\, T(ij)(a) \\
&= t(0)\, \Phi,
\end{aligned}$$

where $\Phi = \prod_{j=0}^{p-1} f(i,ij)\, T(ij)(a)$, we have

$$\mathrm{Inn}\left((t(i)a)^p\right) = \mathrm{Inn}(t(0)\, \Phi)$$

and

$$\mathrm{Inn}\left((t(i)a)^s\right) \circ \mathrm{Inn}\left((t(i)a)^{-r'}\right) = \mathrm{Inn}\left((t(i)a)^{pr}\right) = \mathrm{Inn}\left(t(0)\, \Phi^r\right).$$

We may consider $\mathrm{Inn}(t(0)\, \Phi)|_H$ and $\mathrm{Inn}\left(t(0)\, \Phi^r\right)|_H$ as elements of $\mathrm{Inn}(H)$, and we conclude that $\mathrm{DLP}(\mathrm{Inn}(G))$ is w-reduced to $\mathrm{DLP}(\mathrm{Inn}(H))$. □

Example 11. Let Λ be the graph automorphism of order 2 of $\mathrm{SL}_n(q)$(see [12, § 10]). The group extension $G = [\mathrm{SL}_n(q), \mathbb{Z}_2, \Lambda, 1]$ belongs to Case 2. In this case, the order of $Z(G)$ is the same as that of $\mathrm{SL}_n(q)$.

Example 12. A metacyclic group(for example, see [3, p. 99]) is a semi-direct product and belongs to Case 2. In this case, the order of the center of the group decreases.

In Case 1, since we can find a *computable* isomorphism

$$\overline{\Psi} : G/Z(G) \xrightarrow{\approx} H/Z(H),$$

we see that $\mathrm{DLP}(\mathrm{Inn}(G))$ can be *completely* reduced to $\mathrm{DLP}(\mathrm{Inn}(H))$ in this case.

Example 13. (See [8].) Let $G = \mathrm{SL}_2(p) \times_\theta \mathbb{Z}_p$, where

$$\theta = \mathrm{Inn} \circ \theta_1 : \mathbb{Z}_p \to \mathrm{Aut}(\mathrm{SL}_2(p)),$$

and θ_1 is an isomorphism from Z_p to $\langle \alpha \rangle$, $\alpha \in \mathrm{SL}_2(p)$. Then

$$Z(G) = \{\, t(i)a \mid h^i a = \pm I,\ a \in \mathrm{SL}_2(p)\}.$$

Note that $|Z(G)| > |Z(H)|$ and hence this example belongs to Case 1. Therefore, we have

$$G/Z(G) \cong \mathrm{SL}_2(p)/Z(\mathrm{SL}_2(p)) \cong \mathrm{PSL}_2(p).$$

Remark 14. Moreover, all semi-direct products using inner automorphisms are of Case 1. This is the reason why the authors of [7, 9] search for outer automorphisms.

Remark 15. As in [8, 9], even when the message space is restricted to $\{\, t(0)h \mid h \in H\}$, a similar reduction is possible and we omit the proof.

Remark 16. Since we can only w-reduce $\mathrm{DLP}(\mathrm{Inn}(G))$ to $\mathrm{DLP}(\mathrm{Inn}(H))$, we may not succeed in recovering full information about the secret keys. However, we note that there are many choices of maximal normal subgroups H in G. Thus, we may conclude that the group extension data $G = [H, \mathbb{Z}_p, T, f]$ should not be easily obtained in order to have a secure MOR system. This should be kept in mind when we search for suitable groups for MOR system.

4 Central Commutator Attack

As we have mentioned in Section 2, $\mathrm{DLP}(\mathrm{Inn}(G))$, which is the underlying problem of MOR system, depends a lot on the center $Z(G)$ of G. We are thus naturally led to consider the lower central series of G. Especially, we are interested in the nilpotent groups of which the length of lower central series are finite.

In this section, we show that there is a reduction algorithm for MOR system on a nilpotent group.

4.1 Central Commutator Attack

As before, for $g \in G$, we assume a public key $(\mathrm{Inn}(g),\ \mathrm{Inn}(g^s)) = (\varphi, \varphi^s)$ is given.

Lemma 17. *Suppose we can find $h, z \in G$ such that $z = \varphi(h^{-1})h = g^{-1}h^{-1}gh \neq 1$ and $\varphi(z^{-1})z = g^{-1}z^{-1}gz = 1$, then z^s can be computed from φ^s.*

Proof. Observe the following computation:

$$\varphi^s(h^{-1})h = g^{-s}h^{-1}g^s h = g^{-s}(h^{-1}gh)^s = g^{-s}(gz)^s = z^s. \quad \square$$

Thus, if we can find such h and z and can solve DLP($\langle z \rangle$) from z and z^s, we get $s (\mathrm{mod}\, |z|)$. To find such h and z, assume G is nilpotent and consider the lower central series of G;

$$G = G^0 > G^1 > \cdots > G^{k-1} > G^k = \langle 1 \rangle,$$

where $G^i = [G, G^{i-1}]$. We have $k \geq 2$ because we are assuming G is non-abelian. Since $G^{k-2} \not\leq Z(G)$ and $G^{k-1} \leq Z(G)$, there exists $h \in G^{k-2} \backslash Z(G)$. Letting $z = g^{-1}h^{-1}gh \in G^{k-1}$, z is contained in $G^{k-1} \leq Z(G)$ and thus z commutes with g. This technique is called the *central commutator attack*, since z and $z^s \in Z(G)$ are central commutators.

However, when z is the identity of G, we do not get any information about s, and the condition $z \neq 1$ is not guaranteed here. The next algorithm settles this problem and it can be applied to any nilpotent group.

Lemma 18. *Let G be a nilpotent group of nilpotency $(k-1)$ with $k \geq 2$. Then the Algorithm-1 below outputs z and z^s with $z \neq 1$ (and n in the Algorithm-1 satisfies $n \leq k$).*

Algorithm-1	
Input:	$\varphi = \mathrm{Inn}(g)$ and $\varphi^s = \mathrm{Inn}(g^s)$ such that $\varphi \neq 1$.
Step 1:	Define $\sigma(x) := \varphi(x^{-1})x = g^{-1}x^{-1}gx$ and choose x_0 such that $\sigma(x_0) \neq 1$.
Step 2:	For $m \in \mathbb{N}$, define $x_m := \sigma(x_{m-1})$ and let n be the smallest integer such that $x_n = 1$.
Step 3:	Put $h = x_{n-2},\ z = x_{n-1}$ and compute $z^s = \varphi^s(h^{-1})h$.
Output:	z and z^s with $z \neq 1$.

Proof. For $\mathrm{Inn}(g)$ to be used for an encryption, there should exist x_0 which is not trivially encrypted, i.e., $\varphi(x_0) \neq x_0$ and $g^{-1}x_0\, g\, {x_0}^{-1} \neq 1$. Since G is a nilpotent group of nilpotency $(k-1)$, we have the following lower central series of G;

$$G = G^0 > G^1 > \cdots > G^{k-1} > G^k = \langle 1 \rangle,$$

where $G^i = [G, G^{i-1}]$. Define σ and x_m as in the Algorithm-1. We note that $x_m \in G^m$ for $m = 1, \ldots, k$ and thus $x_k = 1$. Therefore we see that $n \leq k$. Since n is the smallest integer such that $x_n = 1$, we have $z = x_{n-1} \neq 1$. Now, if we put $h = x_{n-2}$, then h and z satisfy the conditions of Lemma 17 and thus we get $\varphi^s(h^{-1})h = z^s$. □

Thus by solving DLP($\langle z \rangle$), one can compute some partial information of the secret, i.e., $s (\mathrm{mod}\, |z|)$. Moreover, we will show that one can recover s completely, if DLP over prime order subgroups of G are easy.

Let $m = |\overline{g}| = \prod_{i=1}^{k} p_i^{e_i}$ be the order of $\overline{g}$ in $G/Z(G)$, where p_i are distinct primes. Then the following algorithm is nothing but an application of the Pohlig-Hellman algorithm [10] to MOR system.

- Step A: For a fixed i, compute $s(\mathrm{mod}\, p_i^j)$ for $j = 1, \ldots, e_i$, inductively.
- Step B: Compute $s(\mathrm{mod}\, {p_i}^{e_i})$ for each $i = 1, \ldots, k$.
- Step C: Using the Chinese remainder theorem, compute $s(\mathrm{mod}\, m)$.

We note that only the Step A is essential here: Fix a prime factor p of m, and let e be the exponent of p in m. Let

$$s(\mathrm{mod}\, p^e) = \sum_{j=0}^{e-1} s_j p^j, \qquad (0 \le s_j \le p-1).$$

First, compute

$$\psi := (\mathrm{Inn}(g))^{m/p} = \mathrm{Inn}(g^{m/p})$$

and

$$\psi_0 := (\mathrm{Inn}(g^s))^{m/p} = \mathrm{Inn}(g^{m/p})^s = \mathrm{Inn}(g^{m/p})^{s_0} = \psi^{s_0}.$$

Since $g^{m/p}$ is not contained in $Z(G)$, we have $\psi(\gamma_i^{-1})\gamma_i \neq 1$ for some$i$, where $\{\gamma_i \mid i \in I\}$ is a given generating set of G. Applying the Algorithm-1 to ψ and ψ_0, we get h, z and z^{s_0} such that

$$z = (g^{-m/p})h^{-1}(g^{m/p})h \quad \text{and} \quad (g^{-m/p})z^{-1}(g^{m/p})z = 1.$$

Observe that $|z| = p$. Solving DLP($\langle z \rangle$), we obtain s_0. Now, assume that we have obtained $s_0, \ldots, s_{\ell-1}$ for some $\ell < e$. Next, we compute

$$\begin{aligned}\psi_\ell :=& (\mathrm{Inn}(g^s) \circ \mathrm{Inn}(g)^{-\sum_{j=0}^{\ell-1} s_j p^j})^{m/p^{\ell+1}} \\ =& (\mathrm{Inn}(g^{s-\sum_{j=0}^{\ell-1} s_j p^j}))^{m/p^{\ell+1}} \\ =& \mathrm{Inn}(g^{m/p})^{s_\ell}.\end{aligned}$$

Again applying the Algorithm-1 to ψ and ψ_ℓ, and solving DLP($\langle z \rangle$), we obtain s_ℓ. By induction we can compute $s(\mathrm{mod}\, p^e)$. In summary, we have the following result.

Theorem 19. *Let G be a finite nilpotent group. For given* Inn(g) *and* Inn(g^s)*, by solving DLP over prime order subgroups of G, one can recover $s(mod\, |\overline{g}|)$ completely. In other words,* DLP(Inn(G)) *can be completely reduced to DLP over prime order subgroups of G.*

We mention here that the central commutator attack is generic in the sense that the algorithm does not use particular property of representations of the group but uses only group operations and equality tests of group elements.

Even when G is not nilpotent, the Algorithm-1 can be applied. First, observe the following.

Lemma 20. *For $x \in G$ define $\tau_x : G \to G$ by*

$$\tau_x(y) = x^{-1}y^{-1}xy, \qquad (y \in G).$$

Then $G/Z(G)$ has nontrivial center if and only if there exists $x \in G \backslash Z(G)$ such that $\tau_x(G) \subseteq Z(G)$.

Proof. Elementary(see, for example, [1, p. 70]).

When the center of $G/Z(G)$ is non-trivial, there exists $x \in G$ such that $[x, G] \subseteq Z(G)$. Thus, given $\varphi = \mathrm{Inn}(g)$, we have $\tau_x(g) = x^{-1}\varphi(x) \in Z(G)$ and $\varphi(x^{-1})\, x \in Z(G)$. Now we see that Algorithm-1 works. Therefore, we might say that Algorithm-1 is valid if G is 'nearly' nilpotent.

When the center of $G/Z(G)$ is trivial, G has the *trivial upper central series* and perhaps is secure against the central commutator attack. But we expect this kind of groups would be 'similar' to simple groups or semi-simple linear groups which are usually not suitable for MOR system.

5 Conclusion

The security of the MOR cryptosystem using a group G is based on the hardness of $\mathrm{DLP}(\mathrm{Inn}(G))$ and is related with the size of $Z(G)$. In a generic sense, the complexity of $\mathrm{DLP}(\mathrm{Inn}(G))$ is about $\log |G|$ times larger than that of $\mathrm{DLP}(G)$, since Pohlig-Hellman or the baby-step giant-step algorithm can be applied to MOR system, provided the special conjugacy problem of G is easy.

Since every finite group G has a composition series, we may regard G as a group extended by finite simple groups for finitely many times. This leads us to analyze a group extension G of H by $\mathbb{Z}_p$ for some prime p, and it is shown that $\mathrm{DLP}(\mathrm{Inn}(G))$ can be w-reduced to $\mathrm{DLP}(\mathrm{Inn}(H))$.

We note that there are many choices of maximal normal subgroups H in G. Thus, we may conclude that the group extension data $G = [H, \mathbb{Z}_p, T, f]$ should not be easily obtained in order to have a secure MOR system. This should be kept in mind when we search for suitable groups for MOR system.

We also analyzed MOR systems on finite nilpotent groups. If G is nilpotent, or $Z(G/Z(G)) \neq 1$, using central commutator attacks, it is shown that $\mathrm{DLP}(\mathrm{Inn}(G))$ can be completely reduced to $\mathrm{DLP}(G)$.

Finally, it should be noted again that MOR system and DLP highly depend on the representations (or presentations) of groups.

References

1. M. L. Curtis, *Matrix groups*, Springer-Verlag, New York, 1979.
2. M. Hall, *The theory of groups*, The Macmillan company, 1959.
3. T. Hungerford, *Algebra*, Springer-Verlag, 1974.
4. U. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, in *Advances in Cryptology - Crypto 1994*, *Lecture Notes in Comput. Sci.*, 839, Springer-Verlag, New York, 1994, pp. 271–281.
5. U. Maurer and S. Wolf, The Diffie-Hellman protocol, in *Des. Codes Cryptography*, 19(2), 2000, pp. 147–171.
6. U. Maurer and S. Wolf, Lower bounds on generic algorithms in groups, in *Advances in Cryptology - Eurocrypt 1998*, *Lecture Notes in Comput. Sci.*, 1403, Springer-Verlag, New York, 1998, pp. 72–84.

7. S. Paeng, On the security of cryptosystem using automorphism groups, in *Inf. Process. Lett.*, 88(6), 2003, pp. 293–298.
8. S. Paeng, K. Ha, J. Kim, S. Chee and C. Park, New public key cryptosystem using finite nonabelian groups, in *Advances in Cryptology - Crypto 2001*, *Lecture Notes in Comput. Sci.*, 2139, pp. 470–485.
9. S. Paeng, D. Kwon, K. Ha and J. Kim, Improved public key cryptosystem using finite nonabelian groups, Cryptology ePrint Archive, Report 2001/066, `http://eprint.iacr.org/2001/066/`.
10. S. Pohlig and M. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, *IEEE Trans. Inform. Theory*, 24, 1978, pp. 106–110.
11. V. Shoup, Lower bounds for discrete logarithms and related problems, in *Advances in Cryptology - Eurocrypt 1995*, *Lecture Notes in Comput. Sci.*, 1233, Springer-Verlag, New York, 1997, pp. 256–266.
12. R. Steinberg, *Lectures on Chevalley groups*, Yale University, 1967.
13. M. Suzuki, *Group theory I*, Springer-Verlag, 1977.
14. C. Tobias, Security analysis of the MOR cryptosystem, in *Proceedings of PKC 2003, Lecture Notes in Comput. Sci.*, 2567, Springer-Verlag, 2003, pp. 175–186.

Cryptanalyzing the Polynomial-Reconstruction Based Public-Key System Under Optimal Parameter Choice

Aggelos Kiayias[1] and Moti Yung[2]

[1] Department of Computer Science and Engineering,
University of Connecticut, Storrs, CT, USA
aggelos@cse.uconn.edu
[2] Department of Computer Science,
Columbia University, New York, NY, USA
moti@cs.columbia.edu

Abstract. Recently, Augot and Finiasz presented a coding theoretic public key cryptosystem that suggests a new approach for designing such systems based on the Polynomial Reconstruction Problem. Their cryptosystem is an instantiation of this approach under a specific choice of parameters which, given the state of the art of coding theory, we show in this work to be sub-optimal. Coron showed how to attack the Augot and Finiasz cryptosystem. A question left open is whether the general approach suggested by the cryptosystem works or not. In this work, we show that the general approach (rather than only the instantiation) is broken as well. Our attack employs the recent powerful list-decoding mechanisms.

1 Introduction

Recently, in Eurocrypt 2003 [AF03], Augot and Finiasz presented a public-key cryptosystem that was based on the Polynomial Reconstruction problem (PR). This scheme suggests a general approach for designing such cryptosystems; their cryptosystem is an instantiation of this approach based on a specific choice of parameters.

Let us first review PR, which is a curve-fitting problem that has been studied extensively especially in the coding theoretic setting, where it corresponds to the Decoding Problem of Reed-Solomon Codes.

Definition 1 (Polynomial Reconstruction (PR)). *Given a set of points over a finite field* $\{\langle z_i, y_i\rangle\}_{i=1}^{n}$*, and parameters* $[n, k, w]$*, recover all polynomials* p *of degree less than* k *such that* $p(z_i) \neq y_i$ *for at most* w *distinct indexes* $i \in \{1, \ldots, n\}$.

Regarding the solvability of PR, we remark that unique solution can only be guaranteed when $w \leq \frac{n-k}{2}$ (the error-correction bound of Reed-Solomon

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 401–416, 2004.

Codes). For such parameter choices, the Berlekamp-Welch Algorithm [BW86] can be used to recover the solution in polynomial-time. When the number of errors w exceeds this bound, unique solution is not necessarily guaranteed. In this range, a decoding algorithm may output a list of polynomials that satisfy the constraints. This is called list-decoding and recently some breakthrough results have been achieved in this field. The most powerful list-decoding algorithm is the one by Guruswami and Sudan, [GS98]. The algorithm will work for any number of errors such that $w < n - \sqrt{(k-1)n}$. For choice of parameters beyond the Guruswami-Sudan solvability bound, no known efficient algorithm exists that solves PR (and [GS98] gives some indication why such an algorithm is not likely to be found).

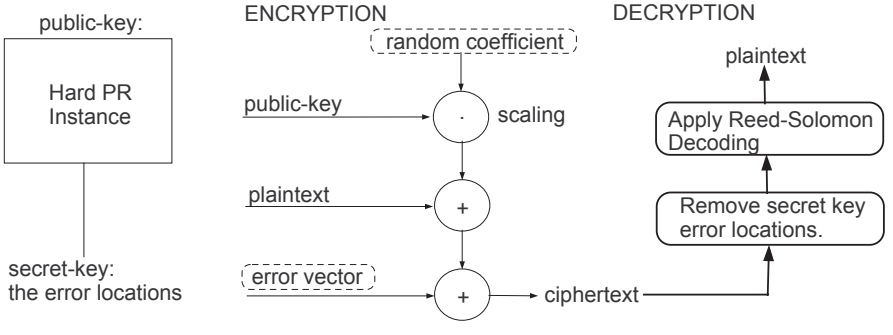


Fig. 1. The Augot and Finiasz general approach for designing a pk-cryptosystem using the hardness of RS decoding

Augot and Finiasz's general approach (see figure 1) is to use a PR instance which is hard to solve (i.e., a highly noisy instance) as a public key, and to encrypt a message by scaling the given public key (i.e., multiplying the polynomial values by a scalar) and adding to the scaled instance the message which is represented as a slightly lower degree second PR instance which is solvable, yielding a PR instance representing the ciphertext. The receiver who knows the noise locations in the public key can recover the message. The approach allows key sizes that are much smaller than the traditional coding theoretic based public-key systems (i.e., the McEliece cryptosystem [McE78]). Further, direct use of the above mentioned decoding and list-decoding methods do not apply to breaking the cryptosystem (directly). To implement the approach of figure 1 one needs to specify: (i) the structure of the public-key, (ii) the structure of the error-vector, and in accordance (iii) the decoding method employed in decryption.

What we noticed is that while the public-key structure was chosen to be an unsolvable PR instance, the choice of the error-vector and the associated decoding method was sub-optimal considering the state-of-the-art of Coding Theory. The scheme was in fact, based on unique decoding (and not list decoding

techniques) and did not consider probabilistic analysis to maximize the allowed entropy of the error-vector.

The scheme of [AF03] was recently broken by Coron [Cor03a, Cor03c] (without affecting the solvability of PR). The elegant attack presented in [Cor03c] is in fact a ciphertext-only attack that is built on the Berlekamp-Welch method and recovers the message, given knowledge only of the public-key and a ciphertext. A further modification of the scheme, using extension fields but essentially the same system, was suggested recently [AFL03] and was shown by Coron [Cor03b] to be vulnerable to essentially the same attack.

Coding Theoretic Motivation. The Augot-Finiasz cryptosystem employed unique decoding techniques rather than list-decoding techniques (assuming that unique decoding is what is needed for a correct cryptosystem — an assumption we refute herein). Moreover, they consider only worst-case analysis in the selection of the code parameters. Thus, their cryptosystem is sub-optimal in the above respects given the general approach outlined above.

This leaves open the question of whether this general approach works in principle, i.e., when one uses the optimal coding theoretic techniques and probabilistic analysis for the parameter selection.

Our Results: In this work we investigate the above question. In particular, we maximize the rate of the error vector used during encryption and choose state-of-the-art list-decoding techniques to implement the Reed-Solomon decoding step for decryption. Regarding the optimization (maximization) of the error-rate we make two key observations (1) the system of [AF03] employs a worst-case approach in selecting this parameter; a probabilistic approach (that we perform in this work) allows higher values. (2) the system of [AF03] employs Berlekamp-Welch RS-decoding for the decryption operation. We emphasize that more powerful decoding techniques can be employed that allow larger values for the error-rate parameter. Our methodology is to use an extended set of tools both for design and analysis in order to get the best possible instantiations of the general approach. The tools include "list decoding" rather than unique decoding techniques (which we show to be still good for decryption, since decoding to a unique value is assured with extremely high probability over a large enough field, even when ambiguous decoding is allowed, cf. Lemma 1).

We develop our presentation as a ping-pong game between a cryptosystems designer and a cryptanalyst. To avoid any misunderstanding our goal is not to design a new cryptosystem, but rather using the design and cryptanalysis steps as a methodology for exploring the general approach.

First Step. Regarding our key-observation (1) we employ the tails of the hypergeometric distribution to show that the original scheme allowed too few errors in the error-vector to be used by the message encryption process. Thus the error-rate can be increased high enough to aid the designer to achieve instances of the cryptosystem where Coron's analysis does not work. But, nevertheless we provide an alternative probabilistic analysis showing that the original attack of

Coron would work almost always even in this modified (more noisy) version, thus aiding the cryptanalyst.

Second Step. Combining our key-observations (1) and (2) above, we discover the optimal setting for the sender error-parameter ("optimal" under the assumption that the Guruswami-Sudan list-decoding algorithm [GS98] represents the best possible decoding algorithm against Polynomial Reconstruction). We show that the optimal parameter setting, helps the designer and in this case Coron's attack fails. To answer our question about the limit of the approach, we then present a new attack that is based on the Sudan and Guruswami-Sudan algorithms [Sud97, GS98]. Our attack, with overwhelming probability, breaks even the optimal parameter setting. This means that the general approach, outlined in figure 1, taken by Augot and Finiasz (rather than merely their non-optimal instantiation) breaks.

We believe that our results demonstrate how design and analysis of Coding theory based cryptography, must employ probabilistic methods and state of the art decoding techniques. Furthermore, our results and the attack of Coron demonstrate that PR-based cryptosystems that lack formal proofs of security by concrete reduction arguments, even when they seem to be related to PR, are potentially susceptible to coding theoretic attacks that do not imply any weakness in the PR problem itself. Note that, the private-key cryptosystem based on PR suggested by the authors in [KY02] was shown to be semantically secure under an intractability decisional assumption that bears upon the average-case PR (for choices of the parameters beyond the Guruswami-Sudan solvability bound). This cryptosystem (as well as the other cryptographic primitives in [KY02]) are not affected by the techniques of the present paper and of Coron's [Cor03a, Cor03b, Cor03c] and breaking these designs seems to require significant advances in RS decodability.

Due to space constraints proofs are omitted from the present abstract; the full-version with all details is available in [KY04].

2 Background: The Recent Polynomial-Based Public-Key Cryptosystem

We review the recent developments, while setting up the necessary notations and interesting points regarding our investigation.

2.1 The Cryptosystem of [AF03]

The cryptosystem of [AF03] can be described in high level as follows:

1. The public-key is a PR-instance of parameters $[n, k+1, W]$ for which (i) the hidden polynomial p is monic; (ii) solving the instance is considered hard. The public-key is a sequence of values in $(\mathbb{F} \times \mathbb{F})^n$ (while the locations of the error points is the secret key).

2. Encryption operates by first transposing (i.e., scaling the polynomial of) the public-key using a random value $\alpha \in \mathbb{F}$ (the encryption coefficient), and then adding to the transposed public-key the message (evaluated as a second polynomial represented as pairs of points using the same first coordinates as the points of the public key PR instance, with no errors), and finally adding some additional w errors. (In other words, the message is embedded in a second PR instance with w errors and added to the transposed public key). It follows that a ciphertext is a sequence of values in $(\mathbb{F} \times \mathbb{F})^n$.
3. Decryption removes the points that correspond to public-key errors, i.e., W points of the ciphertext. Decryption relies on the following two facts: (i) the remaining $n - W$ points can be decoded into a polynomial p^*; (ii) due to the fact that the message polynomial is selected to be of degree less than the degree of the monic polynomial p hidden in the public-key, it follows that the recovery of p^* implies the recovery of the encryption coefficient α. The message polynomial can be recovered as $p_{msg}(x) = p^*(x) - \alpha p(x)$.

We note that the points over which the polynomials are evaluated in a PR instance can be publicly known (thus the public-key and the ciphertext can be considered to be of size only $|\mathbb{F}|^n$).

In more detail, let $z_1, \ldots, z_n \in \mathbb{F}$ be arbitrary distinct elements of the underlying field, where $n \in \mathbb{N}$ is a security parameter. The public-key of the system is a PR-instance that is generated as follows: first a random tuple $\langle E_1, \ldots, E_n \rangle$ is selected that has exactly W non-zero randomly selected elements from $\mathbb{F}$. Second, a random polynomial p of degree less than k is selected. The public-key is set to $\mathsf{pk} := \{\langle z_i, y_i \rangle\}_{i=1}^n$ where $y_i = p(z_i) + E_i + z_i^k$ for $i = 1, \ldots, n$.

Remark. Observe that $\{\langle z_i, y_i - z_i^k \rangle\}_{i=1}^n$ is a random PR-instance with parameters n, k, W.

The encryption operation is defined with domain $\mathbb{F}^k$ and general range the set $(\mathbb{F} \times \mathbb{F})^n$. The message msg is encoded as a polynomial of degree less than k, denoted by $p_{msg}(x)$; a random tuple $\langle e_1, \ldots, e_n \rangle$ is selected so that it has exactly w non-zero randomly selected field elements; a random element $\alpha \in \mathbb{F}$ is selected as well. The ciphertext that corresponds to msg is the sequence of pairs $\{\langle z_i, y_i' \rangle\}_{i=1}^n$ defined as follows $y_i' = \alpha y_i + p_{msg}(z_i) + e_i$, for $i = 1, \ldots, n$.

So far, the above represents a general approach. The exact choice of parameters (as a function of n, say) gives the specific system of [AF03].

The decryption operates as follows: let $I \subseteq \{1, \ldots, n\}$ be such that $|I| = n - W$ and for all $i \in I$ it holds that $E_i = 0$ (from the selection of the public-key). Observe now that the sequence of pairs $C = \{\langle z_i, y_i' \rangle\}_{i \in I}$ can be seen as a PR-instance with parameters $[n - W, k + 1, w]$. Now suppose that,

$$\text{Condition \#1}: \quad w \leq \frac{n - W - k - 1}{2} \Rightarrow n \geq 2w + W + k + 1$$

This condition implies that the PR-instance has a unique solution that can be recovered by the unique decoding technique of Berlekamp-Welch algorithm.

Given such solution $p^*(x)$ it follows that the leading coefficient of p^* will be equal to α (by construction, we have that the polynomial hidden into the public-key is monic and of degree k while the degree of the message polynomial is at most $k-1$). Then, the transmitted message can be recovered as follows $p_{msg}(x) = p^*(x) - \alpha(x^k + p(x))$.

A second condition is that W should be large, beyond the known bounds of list-decoding, to assure that a third party cannot simply get the error locations of the public key (and thus decrypt all ciphertexts). This condition is the base of the presumed security of the scheme.

2.2 A Cryptanalytic Framework

The Cryptanalytic problem that is the basic building block for mounting a ciphertext-only attack on the Public-Key Cryptosystem of [AF03] as described above is defined as follows:

Definition 2 Ciphertext-Only Attack Problem *(CAP).* *Given two sequences of tuples* $X_1 := \{\langle z_i, y_i\rangle\}_{i=1}^n$ *and* $X_2 := \{\langle z_i, y_i'\rangle\}_{i=1}^n$ *and parameters* n, k, w, W *that satisfy the following conditions*

i. $w \leq \frac{n-W-k-1}{2}$ *and* $W \geq n - \sqrt{n(k-1)}$.
ii. $\{\langle z_i, y_i - z_i^k\rangle\}_{i=1}^n$ *is a random PR-instance with parameters* $[n, k, W]$.
iii. $\exists \alpha \in \mathbb{F}$ *such that* $\{\langle z_i, y_i' - \alpha y_i\rangle\}_{i=1}^n$ *is a random PR-instance with parameters* $[n, k, w]$.

Goal. *Find a list of values of polynomial-length that contains the value* α.

Any algorithm that solves CAP in polynomial-time can be turned into a ciphertext-only attack against the cryptosystem of [AF03], as the following proposition reveals.

Proposition 1. *Let* $\mathcal{A}$ *be an algorithm that solves CAP in polynomial-time. Then any message encrypted in the cryptosystem of [AF03] can be decrypted without knowledge of the secret-key in polynomial-time in the security parameter.*

2.3 Coron's Attack

In [Cor03c], Coron presented an elegant ciphertext-only attack against the cryptosystem of [AF03]. We explain the attack briefly below and we show that in fact it can be seen as an algorithm to solve CAP (in fact our formulation of CAP above is motivated by the original attack and by further extensions of this idea in the sequel).

Let X_1, X_2 be an instance of CAP, with $X_1 = \{\langle z_i, y_i\rangle\}_{i=1}^n$ $X_2 = \{\langle z_i, y_i'\rangle\}_{i=1}^n$ and parameters k, w, W, n. Due to condition *iii* of definition 2 it follows that there exist $p \in \mathbb{F}[x]$ of degree less than k and $\alpha \in \mathbb{F}$, so that $p(z_i) \neq y_i' - \alpha y_i$ for at most w indexes i.

The attack modifies the Berlekamp-Welch algorithm: Let $E(x)$ be a monic polynomial of degree w such that $E(z_i) = 0$ for exactly those indexes i for which

$p(z_i) \not= y'_i - \alpha y_i$. The existence of this polynomial is guaranteed due to the condition *iii* of definition 2. Let $N(x) = p(x)E(x)$ be a polynomial of degree less than $k + w$.

Now consider the following system of equations

$$\Big[E(z_i)(y'_i - \lambda y_i) = N(z_i)\Big]_{i=1}^{n} \quad \text{(system 1)}$$

that has as unknowns the $2w + k$ coefficients of the polynomials E, N. Observe that the above system (with λ as a parameter) is not homogeneous (due to the fact that E is monic). Recall that all steps up to this point follow exactly the Berlekamp-Welch algorithm (modulo the unknown λ value).

Now consider the slightly extended system below:

$$\Big[E'(z_i)(y'_i - \lambda y_i) = N(z_i)\Big]_{i=1}^{n} \qquad \text{(system 2)}$$

where $E'(x)$ is a non-monic polynomial that has the same properties as E (i.e. E' and E have the same roots). It follows that system 2 defined above is homogeneous with $2w + k + 1$ unknowns. Let $A_2[\lambda]$ be the $n \times (2w + k + 1)$-matrix of system 2.

Due to condition i of definition 2 the number of equations n satisfies

$$n \geq 2w + k + 1$$

and thus system 2 has at least as many equations as unknowns.

Case 1 of the Attack. $\text{rank}(A_2[0]) = 2w + k + 1$ (i.e., $A_2[0]$ is of full rank). It follows that there are $2w + k + 1$ linearly independent equations in system 2 for $\lambda = 0$ (and their locations can be recovered e.g. by Gaussian elimination). Without loss of generality let us assume that these are the equations on locations $1, \ldots, 2w + k + 1$. We eliminate the remaining $n - (2w + k + 1)$ equations from system 2, to make it a square homogeneous system, and we call the remaining equations system 3.

It follows that if we substitute the value α for λ in the matrix of the system 3, the matrix is singular since it accepts a solution (the polynomials E', N) that is non-trivial. As a result the matrix of system 3, denoted by $A_3[\lambda]$, has the following property:

$$\exists \alpha \in \mathbb{F} : \det(A_3[\alpha]) = 0$$

Now observe that the determinant of system 3 is a polynomial $f(\lambda) := \det(A_3[\lambda])$ that is of degree at most $w + 1$ (because λ is only involved in the part of the matrix of system 3 that corresponds to the polynomial E').

Further observe that $f(0) = \det(A_3[0]) \not= 0$ because of our selection of$A_3[\lambda]$ to have the property that $A_3[0]$ is the full rank minor of the matrix $A_2[0]$. Thus, the value α is among the $w+1$ roots of f and the output will be the list of roots of f. It follows that the above algorithm gives an efficient solution for the CAP problem.

Case 2 of the Attack. $\mathrm{rank}(A_2[0]) < 2w + k + 1$. In this case one can find a non-trivial solution of the system $A_2[0]$ which defines two non-zero polynomials E', N such that

$$[E'(z_i)y'_i = N(z_i)]_{i=1}^{n}$$

Since $y'_i = \alpha(p(z_i) + z_i^k + E_i) + p_{msg}(z_i) + e_i$ it follows that

$$[E'(z_i)(\alpha(p(z_i) + z_i^k + E_i) + p_{msg}(z_i) + e_i) = N(z_i)]_{i=1}^{n}$$

Let I be the subset of $\{1, \ldots, n\}$ for which it holds that $i \in I \iff (e_i = 0) \wedge (E_i = 0)$. It follows that

$$[E'(z_i)p^*(z_i) = N(z_i)]_{i \in I}$$

where $p^*(x) = \alpha(p(x) + x^k) + p_{msg}(x)$. Recall that the degree of the polynomial N is less than $k+w$ and E' is a polynomial of degree w; it follows that $E'(x)p^*(x)$ is a polynomial of degree $w + k$.

Observe that $|I|$ is a random variable (denoted by η) ranging from $n-w-W$ to $n - \max\{w, W\}$. Next consider this relation:

$$\eta > w + k \quad \text{(Sufficient Condition for Case 2)}$$

Under the above relation, it follows that $|I| \geq w + k + 1$ and as a result the polynomials $E'(x)p^*(x)$ and $N(x)$ are equal. It follows immediately that $p^* = \frac{N}{E'}$; naturally given p^* we recover α immediately and non-ambiguously (in fact, in this case we will even be able to recover the value of the secret-key).

Performing a worst-case analysis of the above, we know that $\eta \geq n - w - W$ and as a result the attack would go through as long as $n-w-W > w+k \iff n > 2w+W+k$ something that matches condition #1 of the [AF03]-cryptosystem (cf. section 2.1) and thus the case 2 of the attack can be carried for the parameters of the cryptosystem (without even taking into account that η would be somewhat larger than its lower bound $n - w - W$).

On the other hand, it would be of interest to us to find a necessary condition for case 2 of the attack (the reason for this will become clear in section 3). This can be found by setting η to its highest possible value and requiring this to be greater than $w + k$: $\eta := n - \max\{w, W\} > w + k$; this is equivalent to:

$$n > w + \max\{w, W\} + k \quad \text{(Necessary condition for Case 2)}$$

3 The Increased Error Case

The cryptosystem of [AF03] mandates that the number of errors introduced by the sender in the formation of the ciphertext is less or equal to $\frac{n-W-k-1}{2}$ (condition #1 of section 2.1), to ensure unique decoding in the reduced PR-instance that is obtained after removing the W locations that contain the errors of the Public-Key.

We observe that the bound on w is unreasonably low, for the following reason: many of the errors introduced by the sender will fall into the error-area of the public-key, and thus they will not affect the decryption operation (i.e., introducing a new error in an already erroneous location is a case where $1 + 1 = 1$).

To see this better, we can think of the sender in the cryptosystem to be playing the following game: he selects w points out of n and randomizes them. Since W of these points will be discarded by the receiver it follows that the number of the good points (out of the total $n - W$ of good points) that will be randomized by the sender follow a *hypergeometric* distribution with mean value $\frac{n-W}{n}$. It follows that the expected number of good points that will be randomized by the sender are $w\frac{n-W}{n}$.

In order to ensure decoding for the decryption operation it suffices to force $\tilde{e} \leq \frac{n-W-k-1}{2}$ where $\tilde{e}$ is a random variable that follows the hypergeometric distribution with mean $\frac{n-W}{n}$. Let $w = \frac{1}{\frac{n-W}{n}+\epsilon}\frac{n-W-k-1}{2}$, for some $\epsilon > 0$. Using the Chvátal bound for the hypergeometric distribution, [Chv79], we have that

$$\mathbf{Prob}[\tilde{e} > (\frac{n-W}{n} + \epsilon)w] \leq e^{-2\epsilon^2 w} \Longrightarrow \mathbf{Prob}[\tilde{e} > \frac{n-W-k-1}{2}] \leq e^{-2\epsilon^2 w}$$

From the above, as long as $\epsilon < W/n$, if we set $w = \frac{1}{\frac{n-W}{n}+\epsilon}\frac{n-W-k-1}{2}$ it follows that the probability $\mathbf{Prob}[\tilde{e} > \frac{n-W-k-1}{2}] \leq e^{-2\epsilon^2 w}$, and thus condition # 1 of section 2.1 will be satisfied in the probabilistic sense and decryption will succeed with probability $1 - e^{-2\epsilon^2 w}$.

We will concentrate on parameters s.t. $W > w$ and w is selected as above. Consider for example the assignment $n = 2000$, $k = 100$, $W \geq 1556$ (to avoid an attack with [GS98] on the public-key), e.g. we set $W = 1600$, and $\epsilon = 1/6$; now observe that $W/n = 0.8 > 1/6$. The equation for w mentioned above yields $w = 407$. It follows that the probability of correct decryption is $1 - e^{-2\frac{407}{36}} = 1 - e^{-22} \approx 1 - 2^{-31}$. Observe now that case 2 of Coron's attack would be foiled since the necessary condition fails:

$$n > w + \max\{w, W\} + k \Longleftrightarrow 2000 > 1600 + 407 + 100 \Longleftrightarrow \mathsf{false}$$

Thus, by merely increasing the number of errors that the sender of the cryptosystem introduces during encryption (relying on randomization to allow decryption with very high probability), we are capable of thwarting the analysis of Coron's attack (in particular the analysis of case 2 of the attack). Observe that this is possible without any other modification of the cryptosystem whatsoever.

Nevertheless, this is only a temporary comfort as we will prove in the next section.

4 With High Probability Modified Coron's Attack Succeeds Against Increased Errors

Next, we use another probabilistic analytical tool to show that, in fact, in spite of the increased errors, the attack actually works with high probability.

First, observe the error-increase we introduced in section 3 does not apply to case 1 of Coron's attack. Indeed, one can show for any $\epsilon > 0$ that

$$w = \frac{1}{\frac{n-W}{n} + \epsilon} \frac{n - W - k - 1}{2} \leq \frac{n-k-1}{2}$$

and the condition $w \leq (n-k-1)/2$ is sufficient for case 1 to go through (that is, if we can apply it). Recall that case 1 of the attack only applies to the case $\det(A_3[0]) \not= 0$.

We will show that this in fact happens most of the times (a fact observed in practice in [Cor03c] but not proved). This means that the attack works even in the increased error setting of the previous section. Let us recall the matrix of system 2, as defined in section 2.3.

$$A_2[\lambda] = \left(B_2 \; C_2[\lambda] \right) =$$

$$= \begin{pmatrix} 1 & z_1 & \dots & z_1^{w+k-1} & y'_1 - \lambda y_1 & (y'_1 - \lambda y_1) z_1 & \dots & (y'_1 - \lambda y_1) z_1^w \\ 1 & z_2 & \dots & z_2^{w+k-1} & y'_2 - \lambda y_2 & (y'_2 - \lambda y_2) z_2 & \dots & (y'_2 - \lambda y_2) z_2^w \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & z_n & \dots & z_n^{w+k-1} & y'_n - \lambda y_n & (y'_n - \lambda y_n) z_n & \dots & (y'_n - \lambda y_n) z_n^w \end{pmatrix}$$

where B_2 is a Vandermonde matrix of dimension $w+k$ over the elements $z_1, \dots, z_n$; B_2 corresponds to the coefficients of $N(x)$; $C_2[\lambda]$ is a Vandermonde matrix of dimension $w+1$ over the elements $z_1, \dots, z_n$ where its i-th row is multiplied by $y'_i - \lambda y_i$, for $i = 1, \dots, n$; C_2 corresponds to the coefficients of $E'(x)$. Recall that $A_2[\lambda]$ is a $n \times (2w+k+1)$ matrix. We would like to prove that $\mathrm{rank}(A_2[0]) = 2w+k+1$ with overwhelming probability.

If $\mathrm{rank}(A_2[0]) < 2w+k+1$ then it follows that any $(2w+k+1)$-minor of $A_2[0]$ is singular. Below we show that this event can only happen with very small probability (assuming that the underlying finite field $\mathbb{F}$ is large — something that is assumed in [AF03]) thus we deduce that the first case of the attack would work almost always.

Theorem 1. *Let* $\mathbf{P} = \mathbf{Prob}[\mathrm{rank}(A_2[0]) < 2w+k+1]$ *be the probability that the rank of* $A_2[0]$ *is less than* $2w+k+1$ *where the probability is taken over all possible choices for the given CAP instance out of which we construct* $A_2[\lambda]$. *It holds that* $\mathbf{P} \leq 2w/|\mathbb{F}|$ *and the proof works even if the first inequality of condition i of definition 2 is relaxed to only* $w \leq (n-k-1)/2$.

5 The Most General AF System Avoids Coron's Attack

5.1 An "Optimal Variant" of the Cryptosystem

In this section we show that the number of errors w introduced by the sender can, in fact, be increased further beyond the improved bound that we describe in section 3, by employing the proper decoding method for decryption (cf. figure

1). In particular, we make the following crucial observation: [AF03] requires that w is below the error-correction bound of Reed-Solomon Codes, so that the decryption (decoding) is unique. Nevertheless the introduction of random errors in a large enough finite field (such fields are utilized in [AF03]) suggests that uniqueness of decoding can be ensured far beyond the error-correction bound.

In the lemma below we show that randomly selected PR instances that can accept two different decodings are unlikely. This probabilistic analysis allow us to resort to modern list-decoding techniques in the sequel.

Lemma 1. *Let* $\{\langle z_i, y_i\rangle\}_{i=1}^{n}$ *be a RS-Code codeword of a random message* $p \in \mathbb{F}[x]$ *with* $\text{degree}(p) < k$ *that has* e *errors uniformly random distributed over* $\mathbb{F}$*, s.t.* $e < n - k$*. Then, the probability that it accepts another decoding* $p' \in \mathbb{F}[x]$ *with* $p \neq p'$ *is at most* $\binom{n}{t}^2/(|\mathbb{F}|^{n-e-k})$ *(the probability is taken over all possible messages and noise corruptions).*

Now observe that if the "message rate" is $\kappa := k/n$ and the "error-rate" is $\epsilon := e/n$, with $\kappa, \epsilon \in \mathbf{Q}^+$ then it follows that the probability in lemma 1 is less than $\frac{4^n}{|\mathbb{F}|^{(1-\epsilon-\kappa)n}}$. As a result, provided that $\mathbb{F}$ satisfies $|\mathbb{F}|^{1-\epsilon-\kappa} > 4$ it follows that the probability of proposition 1 is "negligible."

Optimal Parameter Setting and Modifications for the Cryptosystem of [AF03]. Taking advantage of the above Lemma in conjunction with the observation of section 3, we can increase the error-parameter w further. We refer to our choice as optimal with respect to figure 1 under the basic assumption that the list-decoding algorithm of [GS98] represents the state of the art in RS-decodability.

Below we assume that the sender employs the algorithm of [GS98] for decryption. For this algorithm to work it should hold that $\tilde{e} < (n - W) - \sqrt{(n - W)k}$ where $\tilde{e}$ is the number of errors introduced in the area of good points of the public-key due to the encryption operation. As argued in section 3, $\tilde{e}$ is a random variable following a hypergeometric distribution with mean $w\frac{n-W}{n}$. In our analysis below we will simply substitute $\tilde{e}$ for the expected number of errors. Note that this does not guarantee that the receiver will be capable of recovering the transmitted message "most of the times." To guarantee this we would have to show that the probability $\mathbf{Prob}[\tilde{e} < (n - W) - \sqrt{(n - W)k}]$ is overwhelming (as we did in section 3), something that cannot simply be inferred from the fact that the mean of $\tilde{e}$ is less than $(n - W) - \sqrt{(n - W)k}$; in order for the receiver to be able to decrypt most of the times we would instead require that the mean of $\tilde{e}$ is sufficiently lower than the bound $(n-W) - \sqrt{(n - W)k}$ and then employ the Chvátal bound on the tails of the hypergeometric distribution to bound the error probability by a negligible fraction, [Chv79] (as in section 3).

Nevertheless, since we intend to cryptanalyze the resulting cryptosystem, we will opt for simply substituting $\tilde{e}$ for its mean, as this would only make our attack stronger. On the other hand observe that a public-key cryptosystem that works, say, half the times is still quite useful. Thus, substituting $\tilde{e}$ for $w\frac{n-W}{n}$ we obtain

$$w\frac{n-W}{n} < (n-W) - \sqrt{(n-W)k} \Longrightarrow w < n - n\sqrt{\frac{k}{n-W}}$$

We conclude that the optimal selection would allow the parameter w to be selected as high as:

$$w < n(1 - \sqrt{\frac{k}{n-W}})$$

The new bound above increases the number of errors that we can allow the sender to introduce, as long as W is selected appropriately:

Proposition 2. *There are choices for W such that $W \geq n - \sqrt{n(k-1)}$ and $n(1 - \sqrt{\frac{k}{n-W}}) > \frac{n-k-1}{2}$, as long as $k \geq 5$ and $k/n \leq 1/16$.*

Now recall that the necessary condition for Coron's attack (both cases) is $w \leq \frac{n-k-1}{2}$. It follows from the proposition above that our analysis puts the parameter w beyond the range of Coron's attack, provided that W is properly selected. To illustrate this concretely, suppose that $n = 2500$ and $k = 101$. Then, W should be selected in the range $[2000, \ldots, 2126]$; if we make the choice $W = 2063$ then we can set w to be as high as 1298, whereas Coron's attack would correct any value of w only up to 1199. Note that the gap of 99 elements between the bound of Coron's attack and the assignment $w = 1298$ ensures that the application of the attack by removing 99 points at random would only succeed with probability less than $(0.52)^{99} \approx 2^{-96}$ (since the ratio of the sender-introduced error points is ≈ 0.52).

Corollary 1. *Coron's attack cannot be applied against the [AF03]-cryptosystem in the optimal parameter setting.*

To draw a parallel to our exposition in section 2.2, we introduce the problem CAP+ to stand for the ciphertext-only attack problem of the optimal variant of Augot and Finiasz Cryptosystem, (the only difference from CAP being in the choice of w):

Definition 3 Ciphertext-Only Attack Problem in the Optimal Parameter Setting *(CAP+).* *Given two sequences of tuples $X_1 := \{\langle z_i, y_i \rangle\}_{i=1}^n$ and $X_2 := \{\langle z_i, y_i' \rangle\}_{i=1}^n$ and parameters k, w, W that satisfy the following conditions*

i. $w < n(1 - \sqrt{\frac{k}{n-W}})$, *and* $W \geq n - \sqrt{n(k-1)}$.
ii. $\{\langle z_i, y_i - z_i^k \rangle\}_{i=1}^n$ *is a random PR-instance with parameters* $[n, k, W]$.
iii. $\exists \alpha \in \mathbb{F}$ *such that* $\{\langle z_i, y_i' - \alpha y_i \rangle\}_{i=1}^n$ *is a random PR-instance with parameters* $[n, k, w]$.

Goal. *Find a list of values of polynomial-length that contains the value α.*

As before we show that any algorithm that solves CAP+ can be used to mount a ciphertext-only attack on the cryptosystem of [AF03] (but now in the optimal parameter setting):

Proposition 3. *Let $\mathcal{A}$ be an algorithm that solves CAP+ in polynomial-time. Then any message encrypted in the cryptosystem of [AF03] in the optimal parameter setting can be decrypted without knowledge of the secret-key in polynomial-time in the security parameter.*

In the Lemma below we give an upper bound on the value of w (that is independent of W).

Lemma 2. *For any CAP+ instance, it holds that $n - w > \sqrt[4]{n^3(k-1)}$.*

6 The Attack Against the General System Employing List-Decoding

The results we present in this section (essentially an algorithm for solving CAP+) is based on Sudan's list-decoding algorithm, [Sud97] and Guruswami Sudan [GS98] algorithms (for both there are efficient polynomial-time algorithms, see [McE03]).

6.1 The Attack

Let $n, k, w, W \in \mathbf{Z}$ and $X_1 = \{\langle z_i, y_i\rangle\}_{i=1}^n, X_2 = \{\langle z_i, y_i'\rangle\}_{i=1}^n$ be an instance of CAP+. We denote $\hat{y}_i := y_i' - \lambda y_i$ for $i = 1, \ldots, n$, where λ is an unspecified parameter (free variable); to set the parameter λ to a specific value α we will write $\hat{y}_i[\alpha]$.

According to the definition of a CAP+ instance we know that there exists a value $\alpha \in \mathbb{F}$ (the "encryption coefficient") and a polynomial $p \in \mathbb{F}[x]$ of degree less than k (the "message polynomial") that agrees with $n - w$ of the points $\langle z_i, \hat{y}_i[\alpha]\rangle$. Define $l := n - w - 1$. Next we consider the following system of equations on a set of unknowns $\{q_{j_1,j_2}\}_{j_1 \geq 0, j_2 \geq 0, j_1 + (k-1)j_2 < l}$ (called system 4):

$$\forall i \in \{1, \ldots, n\} \qquad \sum_{j_1 \geq 0, j_2 \geq 0, j_1 + (k-1)j_2 < l} q_{j_1,j_2} z_i^{j_1} \hat{y}_i^{j_2} = 0 \quad \text{(system 4)}$$

Observe that any solution to system 4 above defines a bivariate polynomial $Q(x, y)$ that satisfies the property $\text{degree}_{Q,x} + (k-1)\text{degree}_{Q,y} < l$.

Lemma 3. *The number of unknowns of system 4, is at least $\frac{l(l-1)}{2(k-1)}$.*

Recall that from proposition 2 we know that we only consider parameter choices that satisfy $k/n \leq 1/16$. For such range of parameters (and sufficiently large n) we, in fact, show:

Lemma 4. *System 4 is not overdefined provided that $n \geq 19$ and $k/n \leq 1/9$.*

Subsequently we omit the appropriate number of unknowns from system 4, to equalize the number of unknowns and equations. This results in a square homogeneous system of n equations and unknowns that we call system 5. We denote the matrix of system 5 by $A[\lambda]$.

Theorem 2. *Let $\alpha \in \mathbb{F}$ be the "encryption coefficient" for a CAP+ instance as defined in item iii of definition 3. The matrix $A[\alpha]$ as constructed above is singular.*

Since $A[\alpha]$ is singular, it follows that if we define the polynomial $f(\lambda) := \mathrm{Det}(A[\lambda])$, α will be among the solutions of $f(\lambda)$. Thus we can solve CAP+ by computing all the (polynomialy many, by degree constraint) roots of $f(\lambda)$.

Theorem 3. *The probability* $\mathbf{P}$ *that the polynomial $f(\lambda) = \mathrm{Det}(A[\lambda])$ is the zero-polynomial satisfies $\mathbf{P} \leq 2s(n-l)/|\mathbb{F}|$, where $s = \lfloor \frac{l-1}{k-1} \rfloor$ (the maximum degree of the y-variable in any of the columns of $A[\lambda]$).*

7 Summary

In this section, we summarize our cryptanalytic results.

Given an instance of CAP+ $\{\langle z_i, y_i\rangle\}_{i=1}^n$, $\{\langle z_i, y_i'\rangle\}_{i=1}^n$ with parameters n, k, w, W.
0. Set $l := n - w - 1$.
1. Select $D \subseteq \mathbb{N} \times \mathbb{N}$ so that $|D| = n$ and for all $\langle j_1, j_2\rangle \in D$, $j_1 + (k-1)j_2 < l$
2. Let $\boldsymbol{D} = \left\langle \langle j_1[1], j_2[1]\rangle, \ldots, \langle j_1[n], j_2[n]\rangle \right\rangle$, a lexicographic ordering of D.
3. Construct a $(n \times n)$-matrix A so that its (i, i')-entry equals $z_i^{j_1[i']}(y_i' - \lambda y_i)^{j_2[i']}$
4. Compute $f(\lambda) := \det(A[\lambda])$ symbolically to obtain the polynomial f on λ.
5. Output all roots of f.

Fig. 2. The algorithm that solves CAP+

First in figure 2 we overview the CAP+ algorithm that was presented in the previous section. Using this, the general cryptosystem based on Augot and Finiasz [AF03], even under the optimal choice of parameters is broken under ciphertext-only attacks. The breaking algorithm is summarized in figure 3.

Given the public-key and a ciphertext of the [AF03]-cryptosystem with parameters n, k, w, W.
1. if $w \leq \frac{n-k-1}{2}$ invoke case 1 of Coron's attack.
2. else invoke the CAP+ algorithm of figure 1, and recover the plaintext using Guruswami-Sudan algorithm (as described in proposition 3).

Fig. 3. The attack against the Generalized Version of [AF03]-Cryptosystem

Note that the attack outlined above is probabilistic and is guaranteed to work with very high probability as we have shown in theorem 1 (for case 1 of Coron's attack), and theorem 3 (for CAP+ algorithm).

References

[AF03] Daniel Augot and Matthieu Finiasz, *A Public Key Encryption Scheme Based on the Polynomial Reconstruction Problem*, Eli Biham (Ed.): Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science 2656 Springer 2003, pp. 229-240.

[AFL03] Daniel Augot, Matthieu Finiasz and Pierre Loidreau, *Using the Trace Operator to repair the Polynomial Reconstruction based Cryptosystem presented at Eurocrypt 2003*, Cryptology ePrint Archive, Report 2003/209, 2003, `http://eprint.iacr.org/`.

[BW86] Elwyn R. Berlekamp and L. Welch, *Error Correction of Algebraic Block Codes.* U.S. Patent, Number 4,633,470, 1986.

[Chv79] Vasek Chvátal, *The tail of the hypergeometric distribution*, Discrete Math, Vol. 25, pp. 285-287, 1979.

[Cor03a] Jean-Sebastien Coron, *Cryptanalysis of a public-key encryption scheme based on the polynomial reconstruction problem*, Cryptology ePrint Archive, Report 2003/036. `http://eprint.iacr.org/`.

[Cor03b] Jean-Sebastien Coron, *Cryptanalysis of the Repaired Public-key Encryption Scheme Based on the Polynomial Reconstruction Problem*, Cryptology ePrint Archive, Report 2003/219. `http://eprint.iacr.org/`.

[Cor03c] Jean-Sebastien Coron, *Cryptanalysis of a public-key encryption scheme based on the polynomial reconstruction problem*, in Feng Bao, Robert H. Deng, Jianying Zhou (Eds.): Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Lecture Notes in Computer Science 2947 Springer 2004, pp. 14-27.

[GS98] Venkatesan Guruswami and Madhu Sudan, *Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes.* In the Proceedings of the 39th Annual Symposium on Foundations of Computer Science, Palo Alto, California, November 8-11, IEEE Computer Society, pp. 28–39, 1998.

[KY02] Aggelos Kiayias and Moti Yung, *Cryptographic Hardness based on the Decoding of Reed-Solomon Codes*, in the Proceedings of ICALP 2002, Lecture Notes in Computer Science, vol. 2380, Malaga, Spain, July 8-13, pp. 232-243.

[KY04] Aggelos Kiayias and Moti Yung, *Cryptanalyzing the Polynomial-Reconstruction based Public-Key System Under Optimal Parameter Choice*, Cryptology ePrint Archive, Report 2004/217. `http://eprint.iacr.org/`.

[McE78] Robert J. McEliece. *A public key cryptosystem based on algebraic coding theory. Jet Propulsion Lab, DSN Progress Report*, 42(44), pp. 114–116, Jan-Feb 1978.

[McE03] Robert J. McEliece, The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes, IPN Progress Report 42-153, May 15, 2003. http://ipnpr.jpl.nasa.gov/tmo/progress_report/42-153/153F.pdf.

[Sch80] Jacob T. Schwartz, *Fast Probabilistic Algorithms for Verifications of Polynomial Identities*, Journal of the ACM, Vol. 27(4), pp. 701–717, 1980.

[Sud97] Madhu Sudan, *Decoding of Reed Solomon Codes beyond the Error-Correction Bound.* Journal of Complexity 13(1), pp. 180–193, 1997.

Colluding Attacks to a Payment Protocol and Two Signature Exchange Schemes

Feng Bao

Institute for Infocomm Research,
21 Heng Mui Keng Terrace, Singapore 119613
baofeng@i2r.a-star.edu.sg

Abstract. An untraceable fair network payment protocol is proposed by Wang in Asiacrypt'03, which employs the existent techniques of the offline untraceable cash and a new technique called restrictive confirmation signature scheme (RCSS). It is claimed that the fair payment protocol has both the fairness such that the buyer obtains the digital goods if and only if the merchant gains the digital cash and the untraceability and unlinkability such that no one can tell who is the original owner of the money. In this paper we show that the fairness is breached under a simple colluding attack, by which a dishonest merchant can obtain the digital money without the buyer obtaining the goods. We also apply the attack to some of the schemes of fair exchange of digital signatures proposed by Ateniese in ACM CCS'99. Our study shows that two of them are subjected to the attack. A countermeasure against the attack is proposed for the fair exchange of digital signatures. However, we are unable to fix the fair payment protocol if the untraceability and unlinkability are the required features.

1 Introduction

In Asiacrypt 2003, Wang proposed an untraceable fair network payment protocol, which is claimed to have untraceability, unlinkability and fairness [25]. The protocol is for online purchasing of digital goods with digital money. A buyer withdraws untraceable and unlinkable digital cash from a bank and buys some digital goods from an online merchant with the digital cash. The fairness is a feature that prevents either the buyer or the merchant from taking the advantage of the other. It guarantees that the buyer can obtain the goods if and only if the merchant gains the money. The protocol combines the techniques of the untraceable offline e-coin ([8], [9]) and a new primitive called restrictive confirmation signature scheme (RCSS). By RCSS, a signature confirmed by a designated confirmer can only convince some specified verifiers. In this paper we present a colluding attack where a dishonest merchant can breach the fairness such that he can obtain the money without the buyer obtaining the goods. The problem with the protocol is that the money is the untraceable and unlinkable e-coin, which has no link with the buyer's ID and hence can be separated from the RCSS-signed order agreement. That is the vulnerable point our attack

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 417–429, 2004.

exploits. The attack does not work if the digital money is internally linked to the buyer's ID. But the untraceability and unlinkability would be lost in that case.

We can also apply a similar colluding attack to the schemes of fair exchange of digital signature proposed by Ateniese in [4]. There are six schemes in [4] for fair exchange of 1) RSA signatures; 2) Gennaro-Halevi-Rabin signatures; 3) Cramer-Shoup signatures; 4) Guillou-Quisquater signatures; 5) Schnorr (or Poupard-Stern) signatures; and 6) ElGamal (or DSA) signatures, respectively. We show that 5) and 6) are subject to the attack, while 1), 2), 3) and 4) are not. The schemes 1), 2) and 3) have the same principle as Boyd-Foo scheme in [7], where TTP performs different converting functions for different users. The colluding attack does not apply to such schemes. The scheme 4) is an improved version of Bao-Deng-Mao scheme in [5]. A flaw of [5], which was first pointed out in [7], is removed in 4). The reason why 5) and 6) are subject to the attack is that Schnorr signatures and ElGamal signatures have a special feature, which Guillou-Quisquater signatures do not have. The feature does not affect the security requirements of digital signature. However, it is the key point in determining whether the attack works. The feature will be discussed later in this paper. The attack works only if the system allows new users to register at any time.

The rest of the paper is organized as follows. In Section 2, we describe the untraceable fair payment protocol proposed in Asiacrypt'03. In Section 3, we present a colluding attack breaching the fairness of the protocol and explain why the attack works. In Section 4, we describe the two schemes of fair exchange of digital signatures proposed in ACM CCS'99. In Section 5, we discuss the special feature of digital signatures that we exploit and present the colluding attack to the two schemes. We also give the countermeasure against the attack. Section 6 concludes the paper.

2 Untraceable Fair Network Payment Protocol

In this section we describe the untraceable fair network payment protocol proposed in [25]. For simplicity, we skip the details of the building-block RCSS (restrictive confirmation signature scheme) and put it in the Appendix A for interested readers. We also simplify the description of the protocol by assuming that the payment is in one e-coin instead of n e-coins as in [25].

Entities

$\mathcal{U}$ — the buyer, who buys soft goods from the merchant.
$\mathcal{M}$ — the merchant, who sells the soft goods to the buyer.
$\mathcal{B}$ — the bank, who issues e-coins to the buyers.
TTP — the trusted third party, who resolves dispute in payment protocol.

System Parameters and Cryptographic Keys
p, q, g — p, q large primes, $q|p-1$, g a generator of the subgroup G_q of order q of $\mathbb{Z}_p^*$.

$y_{\mathcal{B}}, x_{\mathcal{B}}$ — $\mathcal{B}$'s public and private keys, $y_{\mathcal{B}} = g^{x_{\mathcal{B}}} \mod p$.
y_{TTP}, x_{TTP} — TTP's public and private keys, $y_{TTP} = g^{x_{TTP}} \mod p$.
g_1, g_2 — two elements of G_q published by $\mathcal{B}$, for e-coin scheme.

Two Building-Block Techniques
RCSS — restrictive confirmation signature scheme. In RCSS, a signature signed by a signer S can be confirmed by a confirmer C, and C can convince only some specified verifiers $\mathcal{G}$ that the signature is valid and truly signed by S. RCSS is the main technique designed for the fair payment protocol in [25]. It is denoted by $Sign_{RCSS}(S, C, \mathcal{G}, m)$.

BP — interactive bi-proof of equality. In BP either $\log_\alpha Y = \log_\beta Z$ or $\log_\alpha Y \neq \log_\beta Z$ is proved. The proof system is denoted by $BP(\alpha, Y, \beta, Z)$ in [25], where no detailed description of BP is presented but the reader is referred to [16] and [19].

The untraceable fair network payment protocol consists of five processes, namely account opening, withdrawal, payment, dispute and deposit. The details are as follows.

Account Opening
The buyer $\mathcal{U}$ randomly selects $u_1 \in \mathbb{Z}_q$ and transmits $I = {g_1}^{u_1} \mod p$ to $\mathcal{B}$ if $Ig_2 \neq 1$. The identifier I used to uniquely identify $\mathcal{U}$ can be regarded as the account number of $\mathcal{U}$. Then $\mathcal{B}$ publishes ${g_1}^{x_{\mathcal{B}}}$ (we omit $\mod p$ here) and ${g_2}^{x_{\mathcal{B}}}$ so that $\mathcal{U}$ can compute $z = (Ig_2)^{x_{\mathcal{B}}} = ({g_1}^{x_{\mathcal{B}}})^{u_1} {g_2}^{x_{\mathcal{B}}}$ for himself.

Withdrawal
The buyer $\mathcal{U}$ performs the following protocol to withdraw an e-coin from the bank:

1. $\mathcal{B}$ randomly selects $w \in \mathbb{Z}_q^*$ and sends $e_1 = g^w$ and $e_2 = (Ig_2)^w$ to $\mathcal{U}$.
2. $\mathcal{U}$ randomly selects $s, x_1, x_2 \in \mathbb{Z}_q^*$ and computes $A = (Ig_2)^s$, $B = {g_1}^{x_1} {g_2}^{x_2}$ and $z' = z^s$. $\mathcal{U}$ also randomly selects $u, v, t_c \in \mathbb{Z}_q^*$ and computes $e_1' = {e_1}^u g^v, e_2' = {e_2}^{su} A^v$ and $(a_c, b_c) = (g^{t_c}, {y_{TTP}}^{t_c})$. Then $\mathcal{U}$ sends $c = c'/u \mod q$ to $\mathcal{B}$, where $c' = \mathcal{H}(A, B, z', e_1', e_2', b_c) + a_c \mod q$, where $\mathcal{H}$ is a collision-free hash function to $\mathbb{Z}_q^*$. Note that (a_c, b_c) is a pair of confirmation parameters.
3. $\mathcal{B}$ sends $r = cx_{\mathcal{B}} + w \mod q$ to $\mathcal{U}$.
4. $\mathcal{U}$ verifies whether $g^r = {y_{TTP}}^c e_1$ and $(Ig_2)^r = z^c e_2$. If the verification holds, $\mathcal{U}$ accepts and computes $r' = ru + v \mod q$. Note that $< A, B, (z', e_1', e_2', r', a_c, b_c) >$ represents a pseudo e-coin.

Payment
The Buyer $\mathcal{U}$ and the merchant $\mathcal{M}$ exchange the e-coin and the soft goods in this protocol. In the original protocol multiple e-coins are traded for the soft goods. We present a simplified version of one e-coin without loss of generality.

1. $\mathcal{U}$ selects goods and signs an order agreement $\theta = Sign_{RCSS}(\mathcal{U}, \mathcal{M}, TTP, OA)$, where $OA = \{ID_{\mathcal{U}}, ID_{\mathcal{M}}$, purchase data/information, goods description, $(A, B)\}$.

2. $\mathcal{U}$ sends the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c) >$ and θ to $\mathcal{M}$
3. $\mathcal{M}$ verifies the pseudo e-coin and θ. If all of them are valid and $A \neq 1$, then he sends $d = \mathcal{H}(A, B, ID_{\mathcal{M}},$ date/time) to $\mathcal{U}$.
4. $\mathcal{U}$ sends $k_1 = du_1 s + x_1 \mod q$ and $k_2 = ds + x_2 \mod q$ to $\mathcal{U}$. In addition, $\mathcal{U}$ must run the interactive protocol of bi-proof $BP(g, a_c, y_{TTP}, b_c)$ with $\mathcal{M}$ to show $\log_g a_c = \log_{y_{TTP}} b_c$.
5. $\mathcal{M}$ accepts the pseudo e-coin and the payment transcripts $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ if the following verifications hold:
$$g^{r'} = y_{\mathcal{B}}{}^{\mathcal{H}(A,B,z',e'_1,e'_2,b_c)+a_c} e'_1$$
$$A^{r'} = z'{}^{\mathcal{H}(A,B,z',e'_1,e'_2,b_c)+a_c} e'_2$$
$$g_1{}^{k_1} g_2{}^{k_2} = A^d B$$
If the above verifications pass, $\mathcal{M}$ sends the soft goods to the buyer $\mathcal{U}$.
6. $\mathcal{U}$ checks the soft goods delivered by $\mathcal{M}$. If it matches the description in OA, $\mathcal{U}$ releases t_c to $\mathcal{M}$. Since each one can check $a_c = g^{t_c}$ and $b_c = y_{TTP}{}^{t_c}$ by himself, the coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c, t_c), (d, k_1, k_2) >$ (i.e., the pseudo e-coin plus t_c) denotes a true e-coin that can be directly cashed from the bank.

Disputes

If $\mathcal{U}$ refuses to send t_c to the merchant $\mathcal{M}$, $\mathcal{M}$ begins the dispute process in which TTP can convert the pseudo e-coin into the true e-coin.

1. $\mathcal{M}$ sends the order agreement OA, the RCSS signature θ, the soft goods and the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ to TTP.
2. The TTP checks the validity of the soft goods, pseudo e-coin and signature θ. If the pseudo e-coin is constructed properly, the soft goods from $\mathcal{M}$ is consistent with the description in OA, and θ is valid, TTP sends $\mathcal{M}$ a transformation certificate $TCer = (E_c, T_c)$, where $E_c = a_c{}^{\sigma}$ (σ is a random number selected by TTP) and $T_c = \sigma + x_{TTP} F(a_c, E_c) \mod q$ (F is a public collision-free hash function). The transformation certificate can be used to verify the relation of a_c and b_c by the following equation:
$$a_c{}^{T_c} = E_c b_c{}^{F(a_c, E_c)}$$
3. TTP sends the soft goods to the buyer $\mathcal{U}$.

Deposit

In a normal case, $\mathcal{M}$ forwards the payment transcript and the true e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c, t_c), (d, k_1, k_2) >$ to the bank for deposit. Nevertheless, if $\mathcal{U}$ maliciously aborts the payment process, $\mathcal{M}$ can start the dispute process to acquire the $TCer$ from TTP. In this situation, the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ plus $TCer = (E_c, T_c)$ can be the valid token for deposit. We can also regard $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2), (E_c, T_c) >$ as a true e-coin with different form.

3 Analysis of the Fair Payment Protocol

Before presenting our analysis, we copy the claimed security features of the fair payment protocol, which are expressed in the form of propositions and lemmas in [25].

Unforgeability. *No one except* $\mathcal{U}$ *can create his own pseudo e-coin* $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$.

Indistinguishability. *No one can distinguish between a valid pseudo e-coin and a simulated one without the help of the buyer or TTP.*

Convertibility. *If* $\mathcal{M}$ *accepts the pseudo e-coin, it is guaranteed that TTP can later convert the pseudo e-coins into the true e-coins which can be directly deposited in the bank.*

Fairness. *If the above unforgeability, indistinguishability and convertibility hold for the proposed payment protocol, it can be guaranteed that at the end of the transaction, the buyer* $\mathcal{U}$ *can obtain the soft goods if and only if the merchant* $\mathcal{M}$ *can gain the equivalent true e-coin.*

Untraceability. *No one except* $\mathcal{M}$ *and TTP can confirm the signature* θ. *That means only* $\mathcal{M}$ *and TTP can be convinced that the order agreement* OA *is valid.*

Unlinkability. *The bank or other parties cannot link a coin* $< A, B, (z', e'_1, e'_2, r', a_c, b_c) >$ *to the original owner.*

Idea of the Colluding Attack

In the fair payment protocol, the merchant $\mathcal{M}$ colludes with his conspirator $\mathcal{C}$. After $\mathcal{M}$ receives the pseudo e-coin from the buyer $\mathcal{U}$, $\mathcal{M}$ brings the pseudo e-coin to TTP but claims that the trade is between $\mathcal{C}$ and $\mathcal{M}$. Then the TTP will convert the e-coin to an equivalent true e-coin for $\mathcal{M}$ and send the soft goods to $\mathcal{C}$, while $\mathcal{U}$ will gain nothing. Next we present the attack in details and explain why there is no solution against the attack.

Attack Details and Explanation

1. The malicious merchant $\mathcal{M}$ honestly implements the Payment protocol till step 5. After the verifications pass, he halts the protocol. That is, he obtains the valid pseudo e-coin without giving the soft goods.
2. Then $\mathcal{M}$ asks his conspirator $\mathcal{C}$ to sign a forged order agreement between $\mathcal{M}$ and $\mathcal{C}$, $\theta' = Sign_{RCSS}(\mathcal{C}, \mathcal{M}, TTP, OA')$ where $OA' = \{ID_{\mathcal{C}}, ID_{\mathcal{M}}$, purchase data/information, goods description, $(A, B)\}$.
3. $\mathcal{M}$ starts the Dispute process by sending the order agreement OA', the RCSS signature θ' on OA', the soft goods and the pseudo e-coin $< A, B, (z', e'_1, e'_2, r', a_c, b_c), (d, k_1, k_2) >$ to TTP. Note that TTP has no way to tell whether OA' and θ' are consistent with the pseudo e-coin or not because of the unlinkability and untraceability. Note that the d in the pseudo e-coin is $d = \mathcal{H}(A, B, ID_{\mathcal{M}}$, date/time) instead of $d = \mathcal{H}(A, B, ID_{\mathcal{M}}, ID_{\mathcal{U}}$, date/time). If d is replaced with $d = \mathcal{H}(A, B, ID_{\mathcal{M}}, ID_{\mathcal{U}}$, date/time), the attack does not work anymore but the unlinkability and untraceability would disappear.
4. TTP converts the pseudo e-coin into a true e-coin for $\mathcal{M}$ and forwards the soft goods to $\mathcal{C}$. The buyer $\mathcal{U}$ is left without obtaining anything.
5. The problem of the fair payment protocol is that the e-money is in the form of digital cash, which is generated with the bank's private key and has no

link with the buyer $\mathcal{U}$. If the e-money is in the digital cheque form that is generated with $\mathcal{U}$'s private key, the attack would not work.

6. The protocol cannot be fixed by asking $\mathcal{U}$ to sign a pre-contract to indicate that the trade is between $\mathcal{U}$ and $\mathcal{M}$ or by any other means. The conspirator $\mathcal{C}$ can just simulate $\mathcal{U}$ by doing everything $\mathcal{U}$ does. No one can distinguish $\mathcal{C}$ from $\mathcal{U}$ since the money is unlinkable and untraceable.

4 Fair Exchange of Digital Signatures

Fair exchange protocols have been studied by many researchers in recent years in [1], [2], [3], [4], [5], [6], [7], [11] [12], [15], [20], [26] and many other papers. Among them, [2], [4] and [5] have the same principle in employing verifiable encryption schemes (VES) of digital signatures. In [4], six schemes are proposed in its sections 4.1, 4.2, 4.3, 4.4, 4.6 and 4.7, respectively. The first three schemes are actually not by VES but by the same method of [7]. The latter three schemes exploit the VES of Guillou-Quisquater signatures, VES of Schnorr signatures and VES of ElGamal signatures, respectively. We describe the latter two here in the same denotations as in [4]. Before our description, we introduce a technique that is the main building-block for the schemes.

Building-Block $EQ\_DLOG(m; {g_1}^x, {g_2}^x; g_1, g_2)$
$EQ\_DLOG(m; y_1, y_2; g_1, g_2)$ is a non-interactive proof system for proving $Dlog_{g_1} y_1 = Dlog_{g_2} y_2$ without disclosing the value $x = Dlog_{g_1} y_1 = Dlog_{g_2} y_2$. The proof is associated with a message m. Here $g_1, y_1 \in$ group G_1, $g_2, y_2 \in$ group G_2, and at least one of G_1 and G_2 has an unknown order with bit-length l. Let $\mathcal{H}$ be a hash function $\{0,1\}^* \rightarrow \{0,1\}^k$ and $\epsilon > 1$ be a security parameter. The proof $EQ\_DLOG(m, y_1, y_2; g_1, g_2)$ is implemented as follows.

Prover: randomly choose $t \in [-2^{\epsilon(l+k)}, 2^{\epsilon(l+k)}]$, compute $c = \mathcal{H}(m||y_1||y_2||g_1||g_2||{g_1}^t||{g_2}^t)$ and $s = t - cx$ (in integer $\mathbb{Z}$). (s, c) is the proof/signature of $EQ\_DLOG(m; y_1, y_2; g_1, g_2)$.

Verifier: given (s, c) and (m, y_1, y_2, g_1, g_2), check if $c = \mathcal{H}(m||y_1||y_2||g_1||g_2||{g_1}^s {y_1}^c||{g_2}^s {y_2}^c)$ and $s \in [-2^{\epsilon(l+k)}, 2^{\epsilon(l+k)}]$. If both hold, the verification is passed.

4.1 Fair Exchange of Schnorr Signatures

Settings

• System parameters: The system parameters are p, q and α, where p, q are primes and $q|p-1$, α is an element of order q of $\mathbb{Z}_p^*$.

• TTP: TTP has a pair of public/private keys (n, g)/(factors of n) for either Naccache-Stern encryption scheme [18] (or Okamoto-Uchiyama encryption scheme [21]). The encryption of M under the public key (n, g) is $g^M \mod n$ (or $h^r g^M \mod n$). M can be computed if the factors of n are known. It is claimed in [4] that both schemes can be adopted but for the sake of simplicity Naccache-Stern encryption scheme is employed.

• Alice: Alice has a pair of public/private keys y/a for Schnorr signature scheme where $y = \alpha^a \mod p$.

• Bob: Bob also has a pair of public/private keys for signature. Bob's signature on message M is denoted by $\mathbf{S}_{Bob}(M)$.

• Message: m is a message, on which Alice and Bob are exchanging their signatures.

Fair Exchange by Verifiable Encryption

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, e)$, where $r = \alpha^k \mod p$, $e = \mathcal{H}(m||r)$ and $s = k + ea \mod q$, for a randomly chosen k from $\mathbb{Z}_q$. The verification of $\mathbf{S}_{Alice}(m)$ is to check if $e = \mathcal{H}(m||\alpha^s y^{-e})$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. The e is left in plaintext, from which no one else can compute s. Since Alice knows s, she can implement $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$. Then Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob. That is (C, e, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $e = \mathcal{H}(m||Vy^{-e})$, if valid, sends $\mathbf{S}_{Bob}(m)$ to Alice, otherwise does nothing.
3. Alice verifies Bob's signature and, if valid, sends $\mathbf{S}_{Alice}(m)$ to Bob.
4. If Bob does not receive anything or if Alice's signature is invalid, then he sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ and $\mathbf{S}_{Bob}(m)$ to TTP. This provides a vehicle for TTP to understand whether the protocol was correctly carried out. If this is the case, TTP sends $\mathbf{S}_{Alice}(m)$ to Bob and $\mathbf{S}_{Bob}(m)$ to Alice.

4.2 Fair Exchange of ElGamal Signatures

The settings are exactly the same as in the fair exchange of Schnorr signatures in Section 4.1. The scheme of fair exchange of ElGamal signatures is as follows.

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, r)$, where $r = \alpha^k \mod p$ and $s = k\mathcal{H}(m) + ar \mod q$, for a randomly chosen k from $\mathbb{Z}_q$. The verification of $\mathbf{S}_{Alice}(m)$ is to check if $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. The r is left in plaintext, from which no one else can compute s. Since Alice knows s, she can implement $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$. Then Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob. That is (C, r, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $V = r^{\mathcal{H}(m)} y^r \mod p$, if valid, sends $\mathbf{S}_{Bob}(m)$ to Alice, otherwise does nothing.
3. Alice verifies Bob's signature and, if valid, sends $\mathbf{S}_{Alice}(m)$ to Bob.
4. If Bob does not receive anything or if Alice's signature is invalid, then he sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ and $\mathbf{S}_{Bob}(m)$ to TTP. This provides a vehicle for TTP to understand whether the protocol was correctly carried out. If this is the case, TTP sends $\mathbf{S}_{Alice}(m)$ to Bob and $\mathbf{S}_{Bob}(m)$ to Alice.

5 Colluding Attacks and Countermeasures

5.1 A Feature of Digital Signatures

The digital signatures we consider here are those that consist of two parts, such as the (s, e) of Schnorr scheme, the (s, r) of ElGamal scheme, the (d, D) of Guillou-Quisquater scheme and similar signatures of many other schemes. Let us denote a signature on message m with public/private keys PK/SK by (X, Y), and the verification formula of the signature by

$$\mathbf{Vef}(m, X, Y, PK) = 1 \tag{1}$$

The security requirement of digital signature demands that, for given PK, it is infeasible to compute m and (X, Y) such that (1) holds without knowing SK. (That is the unforgeability in passive attack model. In active attack model, the security requirment is that it is infeasible to forge a valid signature without knowing SK even given a signing oracle.) However, the following feature is not prohibited for the security of signatures, while it plays an important role in our colluding attack.

> **Feature.** Given $m, (X, Y), PK$ that satisfy (1), it is easy to find $Y' \neq Y$ and $PK' \neq PK$ such that $\mathbf{Vef}(m, X, Y', PK') = 1$ without knowing SK.

Schnorr Signature. Given a signature (s, e) on message m such that $e = \mathcal{H}(m||\alpha^s y^{-e})$, we can always find $e' \neq e$ and $y' \neq y$ such that $e' = \mathcal{H}(m||\alpha^s y'^{-e'})$. We just take $e' = \mathcal{H}(m||\alpha^s \alpha^t)$ for a randomly chosen $t \in \mathbb{Z}_q$, and then set $x' = -t/e' \mod q$ and $y' = \alpha^{x'} \mod p$. Hence Schnorr signatures have the feature.

ElGamal Signature. Given a signature (s, r) on message m such that $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$, we can find $r' \neq r$ and $y' \neq y$ such that $\alpha^s = r'^{\mathcal{H}(m)} y'^{r'} \mod p$. We take $r' = (\alpha^s/\alpha^t)^{(1/\mathcal{H}(m) \mod q)}$ for a randomly chosen $t \in \mathbb{Z}_q$, and then set $x' = t/r' \mod q$ and $y' = \alpha^{x'} \mod p$. Hence ElGamal signatures have the feature.

Guillou-Quisquater Signature. In Guillou-Quisquater scheme, $n = pq$ is generated by a trusted center, where p and q are safe primes. A large prime v is selected, and n and v are published as system parameters. The p, q are recommended to be destroyed after that. (It is also allowed that n is generated by each signer. In that case different signer has different n, v.) The public/private keys J/B have relation $B^v J = 1 \mod n$. A signature (d, D) can be generated with the private key B by setting $T = r^v$, $d = \mathcal{H}(m||T)$ and $D = rB^d$, where r is randomly chosen from $\mathbb{Z}_n$. The verification of (d, D) is $d = \mathcal{H}(m||D^v J^d)$. To generate $d' \neq d, J' \neq J$ such that $d' = \mathcal{H}(m||D^v J'^{d'})$ is not as simple as the problems for Schnorr and ElGamal signatures. We cannot solve the problem of computing the d'th-root $\mod n$ since the factorization of n is not known.

5.2 Attack to Fair Exchange of Schnorr and ElGamal Signatures

Attack to Fair Exchange of Schnorr Signature by Dishonest Bob

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, e)$, where $e = \mathcal{H}(m||r)$ and $s = k + ea \mod q$ for randomly chosen $k \in \mathbb{Z}_q$ and $r = \alpha^k \mod p$. The verification formula of $\mathbf{S}_{Alice}(m)$ is $e = \mathcal{H}(m||\alpha^s y^{-e})$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. Then she implements $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$ and $C = g^s \mod n$. After that Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob, i.e., (C, e, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $e = \mathcal{H}(m||Vy^{-e})$, if valid, halts the protocol. Then he computes e', y' such that $e' = \mathcal{H}(m||Vy'^{-e'})$, i.e., $e' = \mathcal{H}(m||V\alpha^t)$ for $t \in_R \mathbb{Z}_q$ and $y' = \alpha^{(-t/e' \mod q)} \mod p$, and asks his conspirator Cathy to register y' as her public key. Bob can ask Cathy to sign (C, e', V) and any other things that could be signed by Alice for any possible authentication.
3. Bob sends (C, e', V), the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $\mathbf{S}_{Bob}(m)$ to TTP and claims that the exchange is between Cathy and Bob.
4. TTP first verifies the proof of $EQ\_DLOG(m; V, C; \alpha, g)$, then decrypts s and verifies whether (s, e') is a valid signature of Cathy and whether $\mathbf{S}_{Bob}(m)$ is a valid signature of Bob. If all the verifications pass, TTP sends $\mathbf{S}_{Cathy}(m) = (s, e')$ to Bob and $\mathbf{S}_{Bob}(m)$ to Cathy. Hence Bob obtains Alice's signature (s, e) without Alice obtaining anything.

Attack to Fair Exchange of ElGamal Signature by Dishonest Bob

1. Alice generates her signature $\mathbf{S}_{Alice}(m) = (s, r)$, where $r = \alpha^k \mod p$ and $s = k\mathcal{H}(m) + ar \mod q$ for a randomly chosen k from $\mathbb{Z}_q$. The verification formula of $\mathbf{S}_{Alice}(m)$ is $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$. Alice encrypts s with TTP's public key (n, g) by setting the ciphertext to be $C = g^s \mod n$. Then she implements $EQ\_DLOG(m; V, C; \alpha, g)$ for $V = \alpha^s \mod p$. Finally Alice sends the verifiable encryption of $\mathbf{S}_{Alice}(m)$ to Bob, which is (C, r, V) plus the proof of $EQ\_DLOG(m; V, C; \alpha, g)$.
2. Bob checks the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $\alpha^s = r^{\mathcal{H}(m)} y^r \mod p$, if valid, halts the protocol. Then he computes r', y' such that $\alpha^s = r'^{\mathcal{H}(m)} y'^{r'} \mod p$, i.e., $r' = (V/\alpha^t)^{(1/\mathcal{H}(m) \mod q)}$ for $t \in_R \mathbb{Z}_q$ and $y' = \alpha^{(t/r' \mod q)} \mod p$, and asks his conspirator Cathy to register y' as her public key. Bob can ask Cathy to sign (C, r', V) and any other things that could be signed by Alice for authentication.
3. Bob sends (C, r', V), the proof of $EQ\_DLOG(m; V, C; \alpha, g)$ and $\mathbf{S}_{Bob}(m)$ to TTP and claims that the exchange is between Cathy and Bob.
4. TTP first verifies the proof of $EQ\_DLOG(m; V, C; \alpha, g)$, then decrypts s and verifies if (s, r') is a valid signature of Cathy and if $\mathbf{S}_{Bob}(m)$ is a valid signature of Bob. If the verifications all pass, TTP sends $\mathbf{S}_{Cathy}(m) = (s, r')$ to Bob and $\mathbf{S}_{Bob}(m)$ to Cathy. Hence Bob obtains Alice's signature (s, r) without Alice obtaining anything.

For some applications it is possible that message m implies the two parties to be Alice and Bob instead of Cathy and Bob. However TTP is not supposed to semantically understand the content of m. TTP only confirms that the m in the EQ_DLOG proof is identical to the m in the signatures.

5.3 Countermeasures

Before presenting our countermeasure, we show an interesting fact that the attack does not apply to DSA signatures. In DSA scheme, a signature (s, r) under public key y satisfies $r^s = \alpha^{\mathcal{H}(m)} y^r \mod p$. Although it is also simple to compute r', y' such that $r'^s = \alpha^{\mathcal{H}(m)} y'^{r'} \mod p$, the attack does not work anymore because the commitment V is different from that of ElGamal signatures. In ElGamal scheme $V = \alpha^s$ while in DSA $V = r^s$. Therefore, in DSA the proof is $EQ\_DLOG(m; V, C; r, g)$. Recall that the proof of $EQ\_DLOG(m; V, C; r, g)$ is (c, σ) satisfying $c = \mathcal{H}(m||V||C||r||g||r^\sigma V^c||g^\sigma C^c)$. That is, r is included in the verification of (c, σ). An $r' \neq r$ would make (c, σ) fail to pass the verification. Forging a new proof of $EQ\_DLOG(m; V, C; r', g)$ is impossible since it is equivalent to knowing s.

Now it is easy to see that the countermeasure is quite simple: Alice includes her ID (or her public key) into the proof, i.e., $EQ\_DLOG(m||ID_{Alice}; V, C; \alpha, g)$. Even better, she includes more detailed information $\mathcal{I}$ about the exchange in the proof, i.e., $EQ\_DLOG(m||\mathcal{I}; V, C; \alpha, g)$. In such case, $\mathcal{I}$ is like a label that cannot be removed and replaced. While attaching $\mathbf{S}_{Alice}(\mathcal{I})$ is like a label stick from outside and can be replaced, and therefore is useless. The ASW fair exchange scheme in [2] is not subject to the attack since a similar label is adopted.

Such label technique would destroy the untraceability and unlinkability, therefore cannot be adopted to fix the fair payment protocol.

6 Conclusions

In this paper we present a colluding attack to breach the fairness of an untraceable fair payment protocol and two schemes of fair exchange of digital signatures. Their fairness actually has no problem in the situation where only the entities described in the protocols exist. The cryptographic techniques employed are also secure and efficient. However, the security flaws appear if we consider the real situation where more entities exist.

As many security experts have pointed out, security does not equal to cryptography and good cryptographic algorithms do not automatically guarantee the security of application systems. Every component is secure does not necessarily mean that the whole system is secure. For complex systems, security should be studied under various attacks from various angles very carefully. It takes long time and big effort before being able to make an assertion.

Another viewpoint reflected from the result of this paper is that the concrete implementation is very critical to security. We show that a tiny difference, such

as whether to include r in $EQ\_DLOG(m; V, C; r, g)$, could make a big difference in security. Hence engineers who implement the security schemes should be very carefully in following every step of the schemes.

References

1. N. Asokan, M. Schunter, and M. Waidner, "Optimistic protocol for fair exchange", Proceedings of the 4th ACM Conference on Computer and Communication Security, pp. 8-17, ACM Press, 1997.
2. N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures", Advances in Cryptology - Proceedings of Eurocrypt'98, LNCS 1403, pp. 591-606, Springer-Verlag, 1998.
3. N. Asokan, V. Shoup and M. Waidner, "Asynchronous protocols for optimistic fair exchange", Proceedings of the 1998 IEEE Symposium on Security and Privacy, IEEE Computer Press, Oakland, CA, 1998.
4. G. Ateniese, "Efficient verifiable encryption and fair exchange of digital signatures", Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), pp. 138-146, 1999.
5. F. Bao, R. H. Deng and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP", Proceedings of 1998 IEEE Symposium on Security and Privacy, pp. 77-85, IEEE Computer Press, 1998.
6. M. Ben-Or, O. Goldreich, S. Micali and R. Rivest, "A fair protocol for signing contracts", IEEE Transactions on Information Theory, IT-36(1):40–46, January 1990.
7. C. Boyd and E. Foo, "Off-line fair payment protocols using convertible signature", Proceedings of Asiacrypt'98, LNCS 1514, pp. 271-285, Springer-Verlag, 1998.
8. S. Brands, "Untraceable off-line cash in wallets with observers", Proceedings of Crypto'93, LNCS 773, pp. 302-318, Springer-Verlag, 1993.
9. D. Chaum, "Blind signature for untraceable payments", Advances in Cryptology – Proc. of Crypto'82, Plenum Press, pp. 199-03, 1983.
10. D. Chaum, "Designated confirmer signatures", Proceedings of Eurocrypt'94, LNCS 950, pp. 86-91, Springer-Verlag, 1994.
11. L. Chen, "Efficient fair exchange with verifiable confirmation of signatures", Proceedings of Asiacrypt'98, LNCS 1514, pp. 286-299, Springer-Verlag, 1998.
12. R. H. Deng, L. Gong, A. A. Lazar and W. Wang, "Practical protocol for certified electronic mail", Journal of Network and Systems Management, 4(3), pp. 279-297, 1996.
13. S. Even, O. Goldreich and A. Lempel, "A randomized protocol for signing contracts", CACM, Vol. 28, No. 6, pp.637-647, 1985.
14. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Transactions on Information Theory, IT-31(4):469-472, 1985.
15. M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party", Proceedings of the 4th ACM Conferences on Computer and Communications Security, pp. 1-5, April 1-4, 1997, Zurich, Switzerland.
16. A. Fujioka, T. Okamoto and K. Ohta, "Interactive bi-proof systems and undeniable signature schemes", Proceedings of Eurocrypt'91, LNCS 547, pp. 243-256, Springer-Verlag, 1992.
17. L. C. Guillou and J. J. Quisquater, "A paradoxical identity-based signature scheme resulting from zero-knowledge", Advances in Cryptology - Crypto'88, LNCS 403, Springer-Verlag, pp. 216-231.

18. D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues", Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS), pp. 59-66, 1998.
19. M. Michels and M. Stadler, "Generic constructions for secure and efficient confirmer signature schemes", Proceedings of Eurocrypt'98, LNCS 1403, pp. 406-421, Springer-Verlag, 1998.
20. T. Okamoto and K.Ohta, "How to simultaneously exchange secrets by general assumption", Proceedings of 2nd ACM Conference on Computer and Communications Security, pp. 184-192, 1994.
21. T. Okamoto and S. Uchiyama, "A new public key cryptosystem as secure as factoring", Proceedings of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 308-318, 1998.
22. C. Pomerance, "On the Role of Smooth Numbers in Number Theoretic Algorithms." In Proc. Internat. Congr. Math., Zrich, Switzerland, 1994, Vol. 1 (Ed. S. D. Chatterji). Basel: Birkh?user, pp. 411-422, 1995.
23. C. P. Schnorr, "Efficient signature generation for smart cards", Proceedings Crypto'89, LNCS, Springer-Verlag, pp.225-232, 1990.
24. M. Stadler, "Publicly verifiable secret sharing", Proceedings of Eurocrypt'96, LNCS 1070, Springer-Verlag, pp.190-199, 1996
25. C.-H. Wang, "Untraceable fair network payment protocols with off-line TTP", Proceedings of Asiacrypt 2003, LNCS 2894, pp. 173-187, Springer-Verlag, 2003.
26. J. Zhou and D. Gollmann, "A fair non-repudiation protocol", Proceedings of the 1996 IEEE Symposium on Security and Privacy, IEEE Computer Press, pp. 55-61, Oakland, CA, 1996.

A Restrictive Confirmation Signature Scheme

In [25], RCSS is designed as follows.

- **System Setup.** The parameters p, q and g, where p, q are primes such that $q|p-1$ and g is an element of $\mathbb{Z}_p^*$ of order q. F_1, F_2 are two collision resistant functions. The private/public key pairs of the signer S, the confirmer C, the recipient R and the verifier V are $(x_S, y_S^{x_S} \mod p)$, $(x_C, y_C^{x_C} \mod p)$, $(x_R, y_R^{x_R} \mod p)$ and $(x_V, y_V^{x_V} \mod p)$, respectively.
- **Signing Protocol.** Assume the signer has signed an undeniable signature (a, b, δ) on message m related to the confirmer's public key, i.e., $a = g^t \mod p, b = {y_C}^t \mod p$ and $\delta = (F_1(m||a)+b)^{x_S} \mod p$, where t is randomly chosen by S. For delegating C the ability of confirming this signature, the signer randomly selects k, u, v_1, v_2 and constructs a proof of

$$(w, z, u, v_1, v_2) = Proof_{DVLogEQ}(c, g, y_S, F_1(m||a) + b, \delta, y_V),$$

where $c = (c_1||c_2), c_1 = g^u {y_V}^{v_1} \mod p, c_2 = g^u {y_C}^{v_2} \mod p, w = F_2(c||g||y_S||F_1(m||a) + b||\delta||g^k||(F_1(m||a) + b)^k)$ and $z = k - x_S(w + u) \mod q$. Thus, the RCSS on m denotes $Sign_{RCSS}(S, C, V, m) = (a, b, u, v_1, v_2, w, z, \delta)$.

- **Proof by the Signer.** The confirmer C also plays the role of the recipient R. That means C will be convinced that he is able to prove the validity of the signature to V in this procedure. C checks the proof by computing $c = ((g^u {y_V}^{v_1} \mod p)||(g^u {y_C}^{v_2} \mod p))$ and verifying if

$$w = F_2(c||g||y_S||F_1(m||a) + b||\delta||g^z {y_S}^{(w+u)}||(F_1(m||a) + b)^z \delta^{(w+u)})$$

  To prove the relation of a and b, the signer needs to run the interactive protocol of bi-proof $BP(g, a, y_C, b)$ to show $\log_g a = \log_{y_C} b$.
- **Confirmation Protocol.** The confirmer C can prove the validity of the signature to V by running the interactive protocol bi-proof $BP(g, y_C, a, b)$ with V to show $\log_g y_C = \log_a b$. The verifier V needs to check whether the signature $(a, b, u, v_1, v_2, w, z, \delta)$ is created properly, and he can be convinced that the signature is valid if he accepts the proof of $BP(g, y_C, a, b)$.
- **Conversion Protocol.** The confirmer can convert the designated confirmer signature to a general non-interactive undeniable signature. Since the signer has constructed the designated verifier proof in a non-interactive way, V can check the validity of the signature by himself. The verifier V no longer needs to ask C to help him verify the signautre. Here, C randomly selects $\sigma \in \mathbb{Z}_q^*$ and computes $E = a^\sigma \mod p$ and $T = \sigma + x_C F(a, E) \mod q$, where F is also a hash function. The confirmer sends (E, T) to the verifier V, thus, V can verify if $a^T = E b^{F(a,E)}$ [10].

Information Security in Korea IT839 Strategy

Ho-Ick Suk

Ministry of Information and Communication, Korea
hisuk@mic.go.kr

Korea now has world-class IT infrastructure such as broadband Internet and mobile communications, and produces high-quality products based on broadband networks and IT technologies including semiconductors, mobile handsets, digital TV, etc. The achievement was made possible thanks to new services that create demand, establishment of infrastructure that enables the provision of new services, and enhanced manufacturing capability. To formulate a new virtuous cycle, the Ministry of Information and Communication (MIC) developed the IT839 Strategy.

The IT839 strategy is composed of 8 services, 3 infrastructures and 9 new growth engines. 8 services are WiBro (Wireless Broadband) Service, DMB (Digital Multimedia Broadcasting) Service, Home Network Service, Telematics Service, RFID-based Service, W-CDMA Service, Terrestrial Digital TV, and Internet Telephony (VoIP). 3 infrastructures are Broadband Convergence Network (BcN), Ubiquitous Sensor Network (USN), and Next-Generation Internet Protocol (IPv6). And 9 new growth engines are Next-Generation Mobile Communications, Digital TV, Home Network, IT System on Chip (SoC), Next-Generation PC, Embedded SW, Digital Contents, Telematics, and Intelligent Service Robot. The success of the Strategy will enhance the quality of our lives and bring us into ubiquitous society.

But, with the advance of new services, intelligent devices such as telematics, home networking, and digital TV, the adverse effect of information society would become one of the major concerns in forthcoming information society. Users living in ubiquitous society propelled by IT839 Strategy will be very sensitive to security and privacy issues. We anticipate possible new information security threats. These are the diffusion of threats caused by network convergence, a sheer of collection and disclosure of personal information through pervasive devices, unestablished authentication framework for emerging transaction devices, and transition from the threats of the cyberspace into ones of the real world.

If we fail to prepare for adequate and timely policies and related technical solutions to cope with such security and privacy challenges, IT839 Strategy would not be successfully implemented in our society. For IT839 Strategy to be successful, we need the proper security policies to overcome the anticipated threats of the future. First, to develop trustworthy convergent network, we will develop cryptography and authentication technologies for secure network connection, agent technology for rapid hand-off, and will standardize interface technologies for secure interoperability among different networks. Secondly, to ensure ubiqui-

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 430–431, 2004.

tous service's safety, we will establish safety criteria for new intelligent devices, develop lightweight cryptography technologies for privacy protection, and develop DRM (Digital Right Management) technologies for protection of illegal digital contents distribution. Lastly, we will improve legal system in preparation for the future IT environments, and will try to formulate security culture.

It is obvious that a higher level of information security will be required to effectively sustain the ubiquitous society. For information security to be more effective, we should take not only technological countermeasures but also social and legal ones. MIC will try to build secure Korea by means of considering information security from the initial stage of IT839 Strategy implementation.

How Far Can We Go Beyond Linear Cryptanalysis?

Thomas Baignères, Pascal Junod, and Serge Vaudenay

EPFL
http://lasecwww.epfl.ch

Abstract. Several generalizations of linear cryptanalysis have been proposed in the past, as well as very similar attacks in a statistical point of view. In this paper, we define a rigorous general statistical framework which allows to interpret most of these attacks in a simple and unified way. Then, we explicitely construct optimal distinguishers, we evaluate their performance, and we prove that a block cipher immune to classical linear cryptanalysis possesses some resistance to a wide class of generalized versions, but not all. Finally, we derive tools which are necessary to set up more elaborate extensions of linear cryptanalysis, and to generalize the notions of bias, characteristic, and piling-up lemma.

Keywords: Block ciphers, linear cryptanalysis, statistical cryptanalysis.

1 A Decade of Linear Cryptanalysis

Linear cryptanalysis is a known-plaintext attack proposed in 1993 by Matsui [21, 22] to break DES [26], exploiting specific correlations between the input and the output of a block cipher. Namely, the attack traces the statistical correlation between one bit of information about the plaintext and one bit of information about the ciphertext, both obtained linearly with respect to $\mathrm{GF}(2)^L$ (where L is the block size of the cipher), by means of *probabilistic linear expressions*, a concept previously introduced by Tardy-Corfdir and Gilbert [30].

Soon after, several attempts to generalize linear cryptanalysis are published: Kaliski and Robshaw [13] demonstrate how it is possible to combine several independent linear correlations depending on the same key bits. In [31], Vaudenay defines another kind of attack on DES, called χ^2-attack, and shows that one can obtain an attack slightly less powerful than a linear cryptanalysis, but without the need to know precisely what happens in the block cipher. Harpes, Kramer, and Massey [7] replace the linear expressions with so-called I/O sums, i.e., balanced binary-valued functions; they prove the potential effectiveness of such a generalization by exhibiting a block cipher secure against conventional linear cryptanalysis but vulnerable to their generalization. Practical examples are the attack of Knudsen and Robshaw [15] against LOKI91 and the one of Shimoyama and Kaneko [28] against DES which both use non-linear approximations.

In [8], Harpes and Massey generalize the results of [7] by considering *partitions pairs* of the input and output spaces. Let $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ and

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 432–450, 2004.

$\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_n\}$ be *partitions* of the input and output sets respectively, where $\mathcal{X}_i$ and $\mathcal{Y}_i$ are called *blocks*. The pair $(\mathcal{X}, \mathcal{Y})$ is called a *partition-pair* if all blocks of $\mathcal{X}$ (respectively $\mathcal{Y}$) contain the same number of plaintexts (respectively ciphertexts). A partitioning cryptanalysis exploits the fact that the probabilities $\Pr\left[(X, \mathsf{f}_k(X)) \in (\mathcal{X}, \mathcal{Y})\right]$ may not be uniformly distributed for a block cipher f_k when the plaintext X is uniformly distributed. In order to characterize the non-uniformity of a sample distribution, Harpes and Massey consider two "measures" called *peak imbalance* and *squared Euclidean imbalance*. Furthermore, they observe on toy-examples that the latter seems to lead to more successful attacks. These results are completed by Jakobsen and Harpes in [10, 9], where they develop useful bounds to estimate the resistance of block ciphers to partitioning cryptanalysis, with the help of spectral techniques; these bounds are relative to the squared Euclidean imbalance only, but this choice is not motivated in a formal way. To the best of our knowledge, the first practical example of partitioning cryptanalysis breaking a block cipher is the attack known as "stochastic cryptanalysis" [24] proposed by Minier and Gilbert against Crypton [17, 18].

In recent papers, Junod and Vaudenay [12, 11] consider linear cryptanalysis in a purely statistical framework, as it was done for the first time by Murphy et al. [25], for deriving optimal key ranking procedures and asymptotic bounds on the success probability of optimal linear distinguishers. A somewhat similar approach is chosen by Coppersmith et al. [1], except that it is adapted to stream ciphers. One can note that tight results about optimal distinguishers allow furthermore to derive useful security criteria.

Finally, the NESSIE effort resulted in a few papers investigating the power of linear (or non-linear) approximations based on different algebraic structures, like $\mathbb{Z}_4$. For instance, Parker [27] shows how to approximate constituent functions of an S-box by *any* linear function over *any* weighted alphabet. However, Parker observes that it is not straightforward to piece these generalized linear approximations together. In [29], Standaert et al. take advantage of approximations in $\mathbb{Z}_4$ by *recombining* the values in order to reduce the problem to the well-known binary case; they obtain more interesting biases comparatively to a classical linear cryptanalysis.

Notation. Throughout this paper, random variables $X, Y, \ldots$ are denoted by capital letters, whilst their realizations $x \in \mathcal{X}, y \in \mathcal{Y}, \ldots$ are denoted by small letters. The cardinal of a set $\mathcal{X}$ is denoted $|\mathcal{X}|$. The probability function of a random variable X following a distribution D is denoted $\Pr_{\mathsf{D}}[x]$ or abusively $\Pr_X[x]$, when the distribution is clear from the context. For convenience, sequence $X_1, X_2, \ldots, X_n$ of n random variables is denoted $\boldsymbol{X}^n$. Similarly, a sequence $x_1, x_2, \ldots, x_n$ of realizations is denoted $\boldsymbol{x}^n$. We call *support* of a distribution D the set of all $x \in \mathcal{X}$ such that $\Pr_{\mathsf{D}}[x] \neq 0$. As usual, "iid" means "independent and identically distributed". The transpose of a linear function h is denoted ${}^t h$. $\mathbb{1}_{\mathrm{A}}$ is 1 if the predicate A is true, 0 otherwise. Finally, "$\cdot$" denotes

the inner product. The distribution function of the standard normal distribution is denoted

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{1}{2}u^2} du \ .$$

2 Optimal Distinguisher Between Two Sources

In this section, we shall consider a source generating a sequence of n iid random variables $\boldsymbol{Z}^n$ following a distribution D and taking values in a set $\mathcal{Z}$. We wonder whether $\mathsf{D} = \mathsf{D}_0$ or $\mathsf{D} = \mathsf{D}_1$ (where D_1 is referred to as an "ideal distribution"), knowing that one of these two hypotheses is true. An algorithm which takes a sequence of n realizations $\boldsymbol{z}^n$ as input and outputs either 0 or 1 is known as a *distinguisher* limited to n samples. It can be defined by an *acceptance region* $\mathcal{A} \subset \mathcal{Z}^n$ such that the distinguisher outputs 0 (respectively 1) when $\boldsymbol{z}^n \in \mathcal{A}$ (respectively $\boldsymbol{z}^n \notin \mathcal{A}$). The ability to distinguish a distribution from another is known as the *advantage* of the distinguisher and is defined by

$$\mathrm{Adv}^n_{\mathcal{A}} = \left| \Pr_{\mathsf{D}_0^n}[\mathcal{A}] - \Pr_{\mathsf{D}_1^n}[\mathcal{A}] \right| \ ,$$

which is a quantity an adversary would like to maximize. The distinguisher can make two types of mistakes: it can either output 0 when $\mathsf{D} = \mathsf{D}_1$ or output 1 when $\mathsf{D} = \mathsf{D}_0$. We denote α and β the respective error probabilities and $\mathrm{P_e} = \frac{1}{2}(\alpha+\beta)$ the *overall probability of error*. We can assume without loss of generality that $\mathrm{P_e} \leq \frac{1}{2}$; we easily obtain that $\mathrm{Adv}^n_{\mathcal{A}} = 1 - 2\mathrm{P_e}$.

2.1 Deriving an Optimal Distinguisher

We describe here how to derive an optimal distinguisher for the scenario described below [1, 11]. Clearly, $\mathrm{P_e} = \frac{1}{2} - \frac{1}{2}\sum_{\boldsymbol{z}^n \in \mathcal{A}} \left(\Pr_{\mathsf{D}_0^n}[\boldsymbol{z}^n] - \Pr_{\mathsf{D}_1^n}[\boldsymbol{z}^n]\right)$, and therefore that the set minimizing[1] $\mathrm{P_e}$ is

$$\mathcal{A} = \{\boldsymbol{z}^n \in \mathcal{Z}^n \ : \ \mathrm{LR}(\boldsymbol{z}^n) \geq 1\} \quad \text{where} \quad \mathrm{LR}(\boldsymbol{z}^n) = \frac{\Pr_{\mathsf{D}_0^n}[\boldsymbol{z}^n]}{\Pr_{\mathsf{D}_1^n}[\boldsymbol{z}^n]} \tag{1}$$

stands for *likelihood ratio*[2]. It defines an optimal distinguisher, i.e., with maximum advantage given a bounded number of samples and with no assumption on the computational power of the adversary.

In order to take a decision, a distinguisher defined by (1) has to keep in memory the results of the n queries, which is not feasible in practice if n grows. Fortunately, it is possible to derive an equivalent distinguisher with $|\mathcal{Z}|$ counter values $\mathrm{N}(a|\boldsymbol{z}^n)$, each one counting the number of occurrence of a certain symbol a of $\mathcal{Z}$ in the sequence $\boldsymbol{z}^n$. We summarize this in the following result.

[1] Note that we could have equivalently chosen a strict inequality in (1).
[2] The likelihood ratio builds the core of the Neyman-Pearson lemma [2–Ch. 12].

Proposition 1 (*Optimal Distinguisher*). *The optimal acceptance region to test* $\mathsf{D} = \mathsf{D}_0$ *against* $\mathsf{D} = \mathsf{D}_1$ *is* $\mathcal{A}_{\text{opt}} = \{\boldsymbol{z}^n \in \mathcal{Z}^n \ : \ \text{LLR}(\boldsymbol{z}^n) \geq 0\}$ *where*

$$\text{LLR}(\boldsymbol{z}^n) = \sum_{\substack{a \in \mathcal{Z} \\ \text{s.t. } \text{N}(a|\boldsymbol{z}^n)>0}} \text{N}(a|\boldsymbol{z}^n) \log \frac{\Pr_{\mathsf{D}_0}[a]}{\Pr_{\mathsf{D}_1}[a]}$$

is the logarithmic likelihood ratio*, with the convention that* $\log \frac{0}{p} = -\infty$ *and* $\log \frac{p}{0} = +\infty$ *(the* $\log \frac{0}{0}$ *case can be ignored), and where* $\text{N}(a|\boldsymbol{z}^n)$ *is the number of times the symbol* a *occurs in the sequence* $\boldsymbol{z}^n \in \mathcal{Z}^n$*.*

Given the number of realizations n, we can compute the exact advantage of the optimal distinguisher. Let $[\mathsf{D}_0]^n$ and $[\mathsf{D}_1]^n$ be the vectors defined by

$$[\mathsf{D}_j]^n_{(z_1,z_2,\ldots,z_n)} = \Pr_{\mathsf{D}_j^n}[z_1, z_2, \ldots, z_n] \quad \text{with} \quad j \in \{0,1\} \ ,$$

which are a specific case of n-wise distribution matrices of the Decorrelation Theory [33] in a simplified case as we have no input here, only outputs z_i. The probability that the distinguisher outputs 0 when $\mathsf{D} = \mathsf{D}_j$ is $\sum_{\boldsymbol{z}^n \in \mathcal{A}} [\mathsf{D}_j]^n_{\boldsymbol{z}^n}$, for $j \in \{0,1\}$. The advantage is thus $\left|\sum_{\boldsymbol{z}^n \in \mathcal{A}} ([\mathsf{D}_0]^n_{\boldsymbol{z}^n} - [\mathsf{D}_1]^n_{\boldsymbol{z}^n})\right|$. Since $\mathcal{A}_{\text{opt}}$ maximizes the sum, we obtain

$$\text{Adv}^n_{\mathcal{A}_{\text{opt}}} = \frac{1}{2} \parallel [\mathsf{D}_0]^n - [\mathsf{D}_1]^n \parallel_1 \ ,$$

where the norm $\| \cdot \|_1$ of a vector $\boldsymbol{A}$ is defined by $\| \boldsymbol{A} \|_1 = \sum_i |\boldsymbol{A}_i|$. Note that the statistical framework of Coppersmith et al. [1] is based on this norm.

2.2 Complexity Analysis

In this section, we compute the number of queries the optimal distinguisher needs in order to distinguish D_0 from D_1, given a fixed error probability P_e.

Definition 2. *The* relative entropy *or* Kullback-Leibler distance *between two distributions* D_0 *and* D_1 *is defined as*

$$D(\mathsf{D}_0 \parallel \mathsf{D}_1) = \sum_{z \in \mathcal{Z}} \Pr_{\mathsf{D}_0}[z] \log \frac{\Pr_{\mathsf{D}_0}[z]}{\Pr_{\mathsf{D}_1}[z]} \ ,$$

with the convention that $0 \log \frac{0}{p} = 0$ *and* $p \log \frac{p}{0} = +\infty$ *for* $p > 0$*.*

We will refer to this notion using the term relative entropy as, being non-symmetric, it is not exactly a distance. Nevertheless, it is always positive since $-\log$ is convex. Using this notation, the following proposition can be proved.

Proposition 3. *Considering that* $Z_1, Z_2, \ldots$ *is a sequence of iid random variables of distribution* D *and that* D_0 *and* D_1 *share the same support,*

$$\Pr\left[\frac{\text{LLR}(\boldsymbol{Z}^n) - n\mu}{\sigma\sqrt{n}} < t\right] \xrightarrow{\quad n \to \infty \quad} \Phi(t) \ , \tag{2}$$

assuming that $\mu = \mu_j$ *with* $\mu_0 = D(\mathsf{D}_0 \parallel \mathsf{D}_1) \geq 0$ *and* $\mu_1 = -D(\mathsf{D}_1 \parallel \mathsf{D}_0) \leq 0$, *and that* σ^2 *is*

$$\sigma_j{}^2 = \sum_{z \in \mathcal{Z}} \Pr_{\mathsf{D}_j}[z] \left(\log \frac{\Pr_{\mathsf{D}_0}[z]}{\Pr_{\mathsf{D}_1}[z]} \right)^2 - \mu_j^2 \ , \tag{3}$$

when $\mathsf{D} = \mathsf{D}_j$ *for* $j \in \{0, 1\}$.

Proof. We first note that the logarithmic likelihood ratio can be expressed as a sum $\mathrm{LLR}(\boldsymbol{Z}^n) = R_1 + \cdots + R_n$ where

$$R_i = \sum_{z \in \mathcal{Z}} \mathbb{1}_{Z_i = z} \log \frac{\Pr_{\mathsf{D}_0}[z]}{\Pr_{\mathsf{D}_1}[z]} \ ,$$

where every Z_i follows distribution D_j (so that the R_i's are iid). The Central Limit Theorem then states that $\Pr\left[(\mathrm{LLR}(\boldsymbol{Z}^n) - n\mu_j)/(\sigma_j\sqrt{n}) < t\right]$ converges in distribution towards $\Phi(t)$, where $\mu_j = \mathrm{E}_{\mathsf{D}_j}[R_i]$ and $\sigma_j{}^2 = \mathrm{Var}_{\mathsf{D}_j}[R_i]$. Some straightforward computations lead to the announced result. Note that the assumption that both distributions share the same support is necessary for μ_j and σ_j to be well defined. □

We now assume that the distributions D_0 and D_1 are close to each other, since it is the usual encountered case in practice.

Assumption 4. *Considering that* D_0 *is close to* D_1, *we can write*

$$\forall z \in \mathcal{Z} \ : \ \Pr_{\mathsf{D}_0}[z] = p_z + \epsilon_z \quad \textit{and} \quad \Pr_{\mathsf{D}_1}[z] = p_z \quad \textit{with} \quad |\epsilon_z| \ll p_z \ .$$

Note that in such a case we can approximate $\mathrm{LLR}(\boldsymbol{z}^n)$ by $\sum_a \mathrm{N}(a|\boldsymbol{z}^n)\epsilon_a/p_a$. Proposition 3 can now be simplified using Taylor series.

Proposition 5. *Under the hypothesis of Proposition 3 and of Assumption 4 we have, at order two:*

$$\mu_0 \approx -\mu_1 \approx \frac{1}{2} \sum_{z \in \mathcal{Z}} \frac{\epsilon_z^2}{p_z} \quad \textit{and} \quad \sigma_0{}^2 \approx \sigma_1{}^2 \approx \sum_{z \in \mathcal{Z}} \frac{\epsilon_z^2}{p_z} \ .$$

We can finally derive a heuristic theorem giving the number of samples the distinguisher needs, together with the implied probability of error, in order to distinguish close distributions with same support.

Theorem 6. *Let* $Z_1, \ldots, Z_n$ *be iid random variables over the set* $\mathcal{Z}$ *of distribution* D, D_0 *and* D_1 *be two distributions of same support which are close to each other, and* n *be the number of samples of the best distinguisher between* $\mathsf{D} = \mathsf{D}_0$ *or* $\mathsf{D} = \mathsf{D}_1$. *Let* d *be a real number such that*

$$n \ = \ \frac{d}{\displaystyle\sum_{z \in \mathcal{Z}} \frac{\epsilon_z^2}{p_z}} \ \approx \ \frac{d}{2D(\mathsf{D}_0 \parallel \mathsf{D}_1)} \tag{4}$$

(where $p_z = \Pr_{\mathsf{D}_1}[z]$ *and* $p_z + \epsilon_z = \Pr_{\mathsf{D}_0}[z]$*). Then, the overall probability of error is* $\mathrm{P_e} \approx \Phi(-\sqrt{d}/2)$.

Proof. If d is such that $\tilde{\mu} = \frac{1}{2}\sqrt{d/n}\ \tilde{\sigma}$, where $\tilde{\mu}$ and $\tilde{\sigma}$ respectively denote the approximation of μ_0 and σ_0 at order 2, we obtain (4). By definition $\mathrm{P_e} = \frac{1}{2}(1 - \Pr_{\mathsf{D}_1}[\mathrm{LLR} < 0] + \Pr_{\mathsf{D}_0}[\mathrm{LLR} < 0])$. However

$$\Pr_{\mathsf{D}_j}[\mathrm{LLR} < 0] = \Pr_{\mathsf{D}_j}\left[\frac{\mathrm{LLR} - n\mu_j}{\sigma_j\sqrt{n}} < -\frac{\sqrt{n}\mu_j}{\sigma_j}\right] \approx \Phi\left(-\frac{\sqrt{n}\mu_j}{\sigma_j}\right) \ ,$$

where we make the usual approximation that the left hand side of (2) can be approximated by $\Phi(t)$. Therefore, as Proposition 5 states that $\mu_0 \approx -\mu_1 \approx \tilde{\mu}$ and that $\sigma_0 \approx \sigma_1 \approx \tilde{\sigma}$, we have $\mathrm{P_e} \approx \frac{1}{2}\left(1 - \Phi(\sqrt{d}/2) + \Phi(-\sqrt{d}/2)\right) = \Phi(-\sqrt{d}/2)$. □

Note that it may be possible to obtain strict tight bounds instead of an approximation for $\mathrm{P_e}$ using, for instance, Chernoff bounds.

2.3 Case Where the Ideal Source Is Uniform

From now on, we assume that D_1 is the uniform distribution. When D_0 is a distribution whose support is $\mathcal{X}$ itself and which is close to D_1, Theorem 6 can be rewritten with

$$n = \frac{d}{|\mathcal{Z}| \sum_{z \in \mathcal{Z}} \epsilon_z^2} \ .$$

This shows that the distinguishability can be measured by means of the Euclidean distance between D_0 and D_1. In the very specific case where $\mathcal{Z} = \{0, 1\}$, we have $\epsilon_0 = -\epsilon_1 = \epsilon$ and one can see that n is proportional to ϵ^{-2}. It is a well accepted fact that the complexity of linear cryptanalysis is linked to the inverse of the square of the bias [21] which is, as we can see, a consequence of Theorem 6. We now recall what appears to be the natural measure of the bias of a distribution, considering the needed number of samples and Assumption 4.

Definition 7. *Let* $\epsilon_z = \Pr_{\mathsf{D}_0}[z] - \frac{1}{|\mathcal{Z}|}$. *The* Squared Euclidean Imbalance[3] *(*SEI*)* $\Delta(\mathsf{D}_0)$ *of a distribution* D_0 *of support* $\mathcal{Z}$ *from the uniform distribution is defined by*

$$\Delta(\mathsf{D}_0) = |\mathcal{Z}| \sum_{z \in \mathcal{Z}} \epsilon_z^2 \ .$$

It is well-known (see [6, 14]) that a χ^2 cryptanalysis needs $O(1/\Delta(\mathsf{D}_0))$ queries to succeed, which is by no means worse, up to a *constant term*, than an optimal distinguisher. Junod observed [11] that a χ^2 statistical test is asymptotically equivalent to a generalized likelihood-ratio test developed for a multinomial distribution; although such tests are not optimal in general, they usually perform reasonably well. Our results confirm this fact: a cryptanalyst will not loose any

[3] Although this appellation coincide with the one of [7], note that the definitions slightly differ.

essential information in the case she can describe *only one* of the two distributions, but the precise knowledge of *both* distributions allows to derive an optimal attack. In other words, when it is impossible to derive both probability distributions, or when an attack involves many different distributions and only one is known, the best practical alternative to an optimal distinguisher seems to be a χ^2 attack, as proposed in [31]. This fact corroborates the intuition stipulating that χ^2 attacks are useful when one does not know precisely what happens in the attacked block cipher.

2.4 Case Where the Source Generates Boolean Vectors

We assume here that random variables are bitstrings[4], so that $\mathcal{Z} = \{0,1\}^\ell$.

Definition 8. *Following the notations of Assumption 4, let* D_0 *be the distribution defined by the set* $\{\epsilon_z\}_{z\in\mathcal{Z}}$, D_1 *being the uniform distribution on* $\mathcal{Z}$. *We define the Fourier transform of* D_0 *at point* $u \in \mathcal{Z}$ *as*

$$\widehat{\epsilon}_u = \sum_{z\in\mathcal{Z}} (-1)^{u\cdot z}\, \epsilon_z \ . \tag{5}$$

The involution property of the Fourier transform leads to

$$\epsilon_z = \frac{1}{2^\ell} \sum_{u\in\mathcal{Z}} (-1)^{u\cdot z}\, \widehat{\epsilon}_u \ . \tag{6}$$

The next property can be compared to Parseval's Theorem.

Proposition 9. *In the case where* D_1 *is the uniform distribution over* $\mathcal{Z} = \{0,1\}^\ell$, *the* SEI *and the Fourier coefficients are related by:*

$$\Delta(\mathsf{D}_0) = \sum_{u\in\mathcal{Z}} \widehat{\epsilon}_u^2 \ .$$

We now recall the definition of the linear probability [23], which plays a central role in the context of linear cryptanalysis.

Definition 10. *The* linear probability *of a boolean random variable* B *is*

$$\mathsf{LP}(B) = (\Pr\,[B=0] - \Pr\,[B=1])^2 = (2\Pr\,[B=0]-1)^2 = \left(\mathrm{E}\left[(-1)^B\right]\right)^2 \ .$$

Proposition 11. *Let* $\mathcal{Z} = \{0,1\}^\ell$. *If* $Z \in \mathcal{Z}$ *is a random variable of distribution* D_0, *the* SEI *and the linear probability are related by:*

$$\Delta(\mathsf{D}_0) \ = \sum_{w\in\mathcal{Z}\setminus\{0\}} \mathsf{LP}(w\cdot Z) \ .$$

[4] Note that all the study below extends in a straightforward way to $\mathcal{Z} = \mathrm{GF}(p)^\ell$ for a prime p by replacing (-1) by $e^{\frac{2i\pi}{p}}$ and by using the conjugates of ϵ_z and $\widehat{\epsilon}_z$ in (5) and (6) respectively. For simplicity we restrict ourselves to GF(2).

Proof. By using (5) we have $\widehat{\epsilon}_u = \mathrm{E}_{\mathsf{D}_0}\left[(-1)^{u\cdot Z}\right] - \mathbb{1}_{u=0}$. Proposition 9 gives $\mathit{\Delta}(\mathsf{D}_0) = \sum_{u\in\mathcal{Z}\setminus\{0\}} \left(\mathrm{E}_{\mathsf{D}_0}\left[(-1)^{u\cdot Z}\right]\right)^2 = \sum_{w\in\mathcal{Z}\setminus\{0\}} \mathsf{LP}(w\cdot Z)$. □

Corollary 12. *Let Z be a random variable over $\mathcal{Z} = \{0,1\}^\ell$ of distribution D_0 and let*[5] $\mathsf{LP}^Z_{\max}$ *be the maximum of* $\mathsf{LP}(w\cdot Z)$ *over* $w \in \mathcal{Z}\setminus\{0\}$. *We have*

$$\mathit{\Delta}(\mathsf{D}_0) \;\le\; \left(2^\ell - 1\right)\; \mathsf{LP}^Z_{\max}\;.$$

Theorem 6 and Corollary 12 together mean that the complexity of the best distinguisher between two distributions of random bit strings can decrease with a factor up to 2^ℓ when compared to the best linear distinguisher. It is interesting to note that there are cases where this bound is tight. For example if D_0 is such that $\Pr_{\mathsf{D}_0}[z]$ is $\frac{1}{2^\ell} + \left(1 - \frac{1}{2^\ell}\right)\gamma$ if $z = 0$, and $\frac{1}{2^\ell} - \frac{1}{2^\ell}\gamma$ otherwise (where γ is a positive constant), it can be shown that $\mathsf{LP}(w\cdot Z) = \gamma^2$ for all $w \not= 0$. Hence $\mathit{\Delta}(\mathsf{D}_0) = (2^\ell - 1)\gamma^2$ and $\mathsf{LP}_{\max} = \gamma^2$.

2.5 Statistical Distinguishers

In the last section, we have been trying to distinguish two random variables following two distinct distributions in a set $\mathcal{Z} = \{0,1\}^\ell$ where ℓ should not be too large from an implementation point of view. If we try to distinguish two random variables distributed in some set $\{0,1\}^L$ of large cardinality (e.g. where $L = 128$), we won't be able to implement the best distinguisher of Proposition 1 as the memory requirement would be too high. Instead, we can reduce the source space to a smaller space $\mathcal{Z} = \{0,1\}^\ell$ by means of a *projection*[6] $h : \{0,1\}^L \to \mathcal{Z}$ defining, for a random variable $S \in \{0,1\}^L$ of distribution $\widetilde{\mathsf{D}}$, a random variable $Z = h(S)$ of distribution D. Here we consider that h is a balanced function and that $\widetilde{\mathsf{D}}_1$ is a uniform distribution, so that D_1 is a uniform distribution as well. This is a typical construction in a real-life block cipher cryptanalysis, where the block length is quite large. Now, even though we know which distinguisher is the best to use in order to distinguish D_0 from D_1, it is still not clear how the projection h has to be chosen. Probably the most classical example arises when $\ell = 1$ and $h(S) = a\cdot S$ for some non-zero $a \in \{0,1\}^\ell$. We then talk about a *linear distinguisher*. In this case, we note that $\mathit{\Delta}(\mathsf{D}_0) = \mathsf{LP}(a\cdot S) \le \mathsf{LP}^S_{\max}$. Modern ciphers protect themselves against that type of distinguisher by bounding the value of $\mathsf{LP}^S_{\max}$. A natural extension of the previous scheme would be to consider any linear projection onto wider spaces, e.g. to consider $h(S) \in \mathcal{Z} = \{0,1\}^\ell$ (where $\ell > 1$ is still small) such that h is GF(2)-linear. We then talk about an *extended linear distinguisher*. It seems natural to wonder about the complexity gap between linear cryptanalysis and this extension. The following theorem proves that if a cipher provably resists classical linear cryptanalysis, it is (to some extent) protected against extended linear cryptanalysis.

[5] We make a slight abuse of notation since $\mathsf{LP}^Z_{\max}$ is not a random variable depending on Z, but a real value depending on the *distribution* of Z.

[6] We borrow this appellation from Vaudenay [31]; the same expression is used within Wagner's unified view of block cipher cryptanalysis [34] as well.

Theorem 13. *Let S be a random variable over $\{0,1\}^L$. Whenever the source space is reduced by a projection $h : \{0,1\}^L \to \{0,1\}^\ell$ in a* GF(2)*-linear way, we have $\Delta(h(S)) \leq (2^\ell - 1)\mathsf{LP}^S_{\max}$.*

Proof. We use Proposition 11 and the fact that $w \cdot h(S) = {}^t h(w) \cdot S$. □

A classical example of a linear space reduction arises when considering *concatenation* of several projections. For example, denoting $\mathsf{D}_0^{(i)} = h^{(i)}(\widetilde{\mathsf{D}}_0)$ for $i \in \{1, \ldots, \ell\}$ where $h^{(i)} : \{0,1\}^L \to \{0,1\}$ is linear, we consider $h(S) = \big(h^{(1)}(S), \ldots, h^{(n)}(S)\big)$. This corresponds to the works of Kaliski and Robshaw [13] (where different linear characteristics involving *identical* key bits are merged) and of Junod and Vaudenay [12] (where different linear characteristics involving *different* key bits are merged). In the latter situation, if no assumption is made about the dependency among the $\Delta(\mathsf{D}_0^{(i)})$'s, Theorem 13 tells us $\Delta(\mathsf{D}_0^{(1)} \times \cdots \times \mathsf{D}_0^{(\ell)}) \leq (2^\ell - 1)\mathsf{LP}^S_{\max}$. The following proposition tells us what happens in general when the $\mathsf{D}_0^{(i)}$'s are independent but do not necessarily come from a linear projection nor a Boolean projection.

Proposition 14. *Consider the case where $\mathsf{D}_0 = \mathsf{D}_0^{(1)} \times \cdots \times \mathsf{D}_0^{(\ell)}$. If $\mathsf{D}_0^{(1)}, \ldots, \mathsf{D}_0^{(\ell)}$ are* independent *distributions, then $\Delta(\mathsf{D}_0)+1 = \prod_{i=1}^{\ell} \Big(\Delta(\mathsf{D}_0^{(i)}) + 1\Big)$. Therefore, $\Delta(\mathsf{D}_0)$ can be approximated by the sum of the $\Delta(\mathsf{D}_0^{(i)})$'s.*

Proof. For the sake of simplicity, we restrict this proof to the case where $\mathsf{D}_0 = \mathsf{D}_0^{(a)} \times \mathsf{D}_0^{(b)}$. Let $Z = (A, B)$ where A and B are two independent random variable following distributions $\mathsf{D}_0^{(a)}$ and $\mathsf{D}_0^{(b)}$ respectively. As in Proposition 11, we have

$$\begin{aligned}
\Delta(\mathsf{D}_0^{(a)} \times \mathsf{D}_0^{(b)}) &= \sum_{(v,w) \in \mathcal{Z}^2 \setminus \{\mathbf{0}\}} \left(\mathrm{E}\left[(-1)^{v \cdot A \oplus w \cdot B}\right]\right)^2 \\
&= \sum_{(v,w) \in \mathcal{Z}^2 \setminus \{\mathbf{0}\}} \left(\mathrm{E}\left[(-1)^{v \cdot A}\right]\right)^2 \left(\mathrm{E}\left[(-1)^{w \cdot B}\right]\right)^2 \\
&= \Big(\Delta(\mathsf{D}_0^{(a)}) + 1\Big)\Big(\Delta(\mathsf{D}_0^{(b)}) + 1\Big) - 1 \ .
\end{aligned}$$

□

This result tells us that merging ℓ independent biases should only be considered when their respective amplitudes are within the same order of magnitude.

In the light of the preceeding discussion, the cryptanalyst may wonder if it is possible to find a distinguisher with a high advantage even though the value of $\mathsf{LP}^S_{\max}$ is very small. We provide an example for which it is indeed the case.

Example. Consider a source generating a random variable $S = (X_1, \ldots, X_{n+1}) \in \mathbb{Z}_4^{n+1}$, where n is some odd large integer, and we represent $\mathbb{Z}_4$ by $\{0,1\}^2$ in binary. Here we have $L = 2n+2$. If the source follows distribution D_0, then $X_1, \ldots, X_n \in \mathbb{Z}_4$ are uniform iid random variables and $X_{n+1} = (Y + \sum_{i=1}^n X_i) \bmod 4$, where $Y \in \{0,1\}$ is a uniformly distributed random variable independent of $X_1, \ldots, X_n$.

If the source follows distribution D_1, $S \in \mathbb{Z}_4^{n+1}$ is uniformly distributed. It can be shown (see Appendix A) that $\mathsf{LP}^S_{\max} = 2^{-(n+1)}$. On the other hand, if we let $h : \mathbb{Z}_4^{n+1} \to \mathbb{Z}_2$ be such that $h(S) = \mathrm{msb}\left((X_{n+1} - \sum_{i=0}^{n} X_i) \bmod 4\right)$ (where msb stands for most significant bit), we have $\ell = 1$ and a SEI equal to 1, so that $\Delta(\mathsf{D}_0) \gg \mathsf{LP}^S_{\max}$: D_0 can be distinguished from D_1 despite $\mathsf{LP}^S_{\max}$ is small.

This example shows that Theorem 13 tells us nothing about the SEI whenever the plaintext space is reduced by a non-linear projection. Therefore, even though $\mathsf{LP}^S_{\max}$ is very low, there may exist some tricky non-linear projections which lead to significant breakdown of the complexity of distinguishers, i.e., there may be something beyond linear cryptanalysis.

3 Optimal Distinguisher Between Two Oracles

So far we discussed how to distinguish random sources. Now we investigate applications for distinguishing random oracles, such as block ciphers, and how to transform this into the previous problem.

We consider the random variable Z taking values in $\mathcal{Z}$ to be a couple of random variables (X, Y) taking values in $\mathcal{X} \times \mathcal{Y}$. As discussed in Sect. 2.5, the couple (X, Y) can be seen like the image of a plaintext/ciphertext couple (P, C) by some balanced projections ϕ and ψ (which actually define the statistical cryptanalysis in use); in other words, the adversary queries the oracle for known-plaintext pairs and compute the projections ϕ and ψ to sample (X, Y). For simplicity reasons, we focus our study on known-plaintext attacks (such as linear cryptanalysis) and thus, we consider that X is uniformly distributed. The distribution of Y is defined by a transition matrix $\boldsymbol{T}$ such that

$$[\boldsymbol{T}]_{x,y} = \Pr\,[Y = y | X = x] = \Pr\,[\psi(C) = y | \phi(P) = x] \quad .$$

The transition matrix $\boldsymbol{T}$ can either be $\boldsymbol{T}_0$ or $\boldsymbol{T}_1$, where $\boldsymbol{T}_1$ is the uniform transition matrix (i.e., $[\boldsymbol{T}_1]_{x,y} = \frac{1}{|\mathcal{Y}|}$). The distribution D of Z depends on the transition matrix $\boldsymbol{T}$. We will denote it D_0 (respectively D_1) when $\boldsymbol{T} = \boldsymbol{T}_0$ (respectively $\boldsymbol{T} = \boldsymbol{T}_1$). We can see that if $\boldsymbol{T} = \boldsymbol{T}_1$, as X is uniformly distributed, the distribution D_1 of Z is also uniform. Therefore, all the results presented so far can be applied to the particular case we study here. Indeed, if we note that

$$\Pr_{\mathsf{D}}[z] = \Pr\,[X = x, Y = y] = [\boldsymbol{T}]_{x,y} \Pr\,[X = x] \quad .$$

We can express Proposition 1 in terms of the transition matrices.

Proposition 15 (*Optimal Binary Hypothesis Test with Transition Matrices*). *The optimal acceptance region to test* $\mathsf{D} = \mathsf{D}_0$ *against* $\mathsf{D} = \mathsf{D}_1$ *(where* D_1 *is the uniform distribution), that is to test* $\boldsymbol{T} = \boldsymbol{T}_0$ *against* $\boldsymbol{T} = \boldsymbol{T}_1$*, is*

$$\mathcal{A}_{\mathrm{opt}} = \{(\boldsymbol{x}^n, \boldsymbol{y}^n) \in \mathcal{X}^n \times \mathcal{Y}^n \ : \ \mathrm{LLR}(\boldsymbol{x}^n, \boldsymbol{y}^n) \geq 0\}$$

where

$$\mathrm{LLR}(\boldsymbol{x}^n, \boldsymbol{y}^n) = \sum_{\substack{(x,y)\in\mathcal{X}\times\mathcal{Y} \\ \text{s.t. } \mathrm{N}((x,y)|\boldsymbol{z}^n)>0}} \mathrm{N}((x,y)|\boldsymbol{z}^n) \log \frac{[\boldsymbol{T}_0]_{x,y}}{[\boldsymbol{T}_1]_{x,y}}$$

with the conventions used in Proposition 1.

In the next sections, we derive the complexity of this distinguisher, discuss the relationship between our model and previous work, and study how Matsui's *Piling-up Lemma* [21] extends to our model.

3.1 Cryptanalysis Complexity

We introduce the notion of *bias matrix* $\boldsymbol{B} = \boldsymbol{T}_0 - \boldsymbol{T}_1$. Note that $\sum_{x\in\mathcal{X}}[\boldsymbol{B}]_{x,y} = 0$ when X is uniformly distributed and that $\sum_{y\in\mathcal{Y}}[\boldsymbol{B}]_{x,y} = 0$ in any case. Similarly to Definition 8, the Fourier transform $\widehat{\boldsymbol{B}}$ of the bias matrix $\boldsymbol{B}$ is such that

$$\left[\widehat{\boldsymbol{B}}\right]_{u,v} = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} (-1)^{u\cdot x \oplus v\cdot y} [\boldsymbol{B}]_{x,y} \ .$$

Furthermore, we define LPM, the *linear probability matrix*, by $[\mathsf{LPM}]_{u,v} = 0$ if $u = v = 0$ and by $[\mathsf{LPM}]_{u,v} = \mathsf{LP}(u\cdot X \oplus v\cdot Y)$ otherwise. It can be noted that $\left[\widehat{\boldsymbol{B}}\right]^2_{u,v} = |\mathcal{X}|^2 [\mathsf{LPM}]_{u,v}$. With the notations we just introduced, it is possible to derive the complexity of the best distinguisher between two oracles as a simple consequence of Theorem 6 and of Proposition 11.

Proposition 16. *Let n be the number of queries of the best distinguisher between $\boldsymbol{T}_0$ and $\boldsymbol{T}_1$, which are supposed to be close to each other and of same support. Then the overall probability of error is $\mathrm{P_e} \approx 1 - \Phi(\sqrt{d}/2)$, where d is a real number such that $n = d/\Delta(\mathsf{D}_0)$. Furthermore, as*

$$\Delta(\mathsf{D}_0) \ = \ \frac{|\mathcal{Y}|}{|\mathcal{X}|} \parallel \boldsymbol{B} \parallel_2^2 = \ \frac{1}{|\mathcal{X}|^2} \parallel \widehat{\boldsymbol{B}} \parallel_2^2 = \sum_{(u,v)\in\mathcal{X}\times\mathcal{Y}} [\mathsf{LPM}]_{u,v} \ ,$$

n can be equivalently expressed in terms of the bias matrix, of its Fourier transform, or of the linear probability matrix (and thus, of the linear probabilities).

Matsui's linear expressions are a very particular case of the transition matrices we have defined at the beginning of Sect. 3. Indeed, choosing balanced *linear* projections $\phi, \psi : \{0,1\}^L \to \{0,1\}$ is equivalent to choose input/output masks on the plaintext/ciphertext bits. The respective shapes of the corresponding bias matrix, of its Fourier transform, and of the LPM matrix are

$$\boldsymbol{B} = \begin{pmatrix} \epsilon & -\epsilon \\ -\epsilon & \epsilon \end{pmatrix} \quad , \quad \widehat{\boldsymbol{B}} = \begin{pmatrix} 0 & 0 \\ 0 & 4\epsilon \end{pmatrix} \quad \text{, and} \quad \mathsf{LPM} = \begin{pmatrix} 0 & 0 \\ 0 & 4\epsilon^2 \end{pmatrix} \ ,$$

where ϵ is nothing but the bias of Matsui's linear expressions. According to Proposition 16, we see that the complexity of the distinguishing attack is proportional to $\parallel \boldsymbol{B} \parallel_2^{-2}$, which is a well known result in linear cryptanalysis, for which $\parallel \boldsymbol{B} \parallel_2^2 = 4\epsilon^2$.

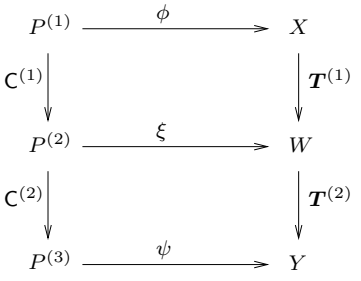


Fig. 1. Two rounds of an iterated block cipher

There is an intuitive link between linear probability matrices and *correlation matrices* [3]. Recall that the correlation matrix of a Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ is the $2^m \times 2^n$ matrix $\mathsf{C}^{(f)}$ such that $\left[\mathsf{C}^{(f)}\right]_{u,v} = 2\Pr\left[u \cdot f(P) \oplus v \cdot P\right] - 1$, where the probability holds over the uniform distribution of P, so that $\left[\mathsf{C}^{(f)}\right]^2_{u,v} = \mathsf{LP}(u \cdot f(P) \oplus v \cdot P)$. We see that correlation matrices are strongly related to the linear probability matrices in the specific case where ϕ and ψ are identity functions (i.e., no reduction is performed on the plaintext space).

3.2 Piling-Up Transition Matrices

A distinguishing attack on an iterated cipher is practical on condition that the cryptanalyst knows a transition matrix spanning several rounds. In practice, the cryptanalyst will derive a transition matrix on each round and, provided that the projections were chosen carefully, pile them in order to obtain a transition matrix on several rounds of the cipher.

We consider the scenario where a block cipher is represented by a random permutation C over $\{0,1\}^L$ (L denotes the block size of the cipher), where the randomness comes from the key. Moreover we suppose that the block cipher is made of two rounds corresponding to the succession of two random permutations $\mathsf{C}^{(1)}$ and $\mathsf{C}^{(2)}$. In other words $\mathsf{C} = \mathsf{C}^{(2)} \circ \mathsf{C}^{(1)}$. We denote $P^{(1)}, P^{(2)} \in \{0,1\}^L$ the respective inputs of $\mathsf{C}^{(1)}$ and $\mathsf{C}^{(2)}$, whereas $P^{(3)}$ denote the output of $\mathsf{C}^{(2)}$. The random variables X, W, and Y respectively denote $\phi(P^{(1)})$, $\xi(P^{(2)})$, and $\psi(P^{(3)})$, where ϕ, ξ, and ψ are projections onto $\mathcal{X}$, $\mathcal{W}$, and $\mathcal{Y}$, respectively. With these notations, the respective transition matrices of $\mathsf{C}^{(1)}$, $\mathsf{C}^{(2)}$, and C are

$$\left[\boldsymbol{T}^{(1)}\right]_{x,w} = \Pr_{W|X}\left[w \mid x\right] \ , \ \left[\boldsymbol{T}^{(2)}\right]_{w,y} = \Pr_{Y|W}\left[y \mid w\right] \ ,$$
$$\text{and} \ \ \left[\boldsymbol{T}\right]_{x,y} = \Pr_{Y|X}\left[y \mid x\right] \ .$$

This situation is represented on Fig. 1. Note that we use a representation which is very similar to Wagner's commutative diagrams [34]. Under the assumption that $X \leftrightarrow W \leftrightarrow Y$ is a Markov chain (as in [34]), it can easily be shown that successive transition matrices are multiplicative, i.e., $\boldsymbol{T} = \boldsymbol{T}^{(1)} \times \boldsymbol{T}^{(2)}$. Note that this situation is idealistic as, even under the classical assumption that

$P^{(1)} \leftrightarrow P^{(2)} \leftrightarrow P^{(3)}$ is a Markov chain [16, 32], $X \leftrightarrow W \leftrightarrow Y$ may not be a Markov chain unless the projection are chosen with care. Nevertheless, under the suitable suppositions, the following lemma shows how the Piling-Up Lemma extends to our model.

Lemma 17. *Let $\boldsymbol{B}^{(1)}$, $\boldsymbol{B}^{(2)}$, and $\boldsymbol{B}$ be the bias matrices associated with $\boldsymbol{T}^{(1)}$, $\boldsymbol{T}^{(2)}$, and $\boldsymbol{T}$ respectively, such that $\boldsymbol{T} = \boldsymbol{T}^{(1)} \times \boldsymbol{T}^{(2)}$. In the case of a known-plaintext attack, $\boldsymbol{B} = \boldsymbol{B}^{(1)} \times \boldsymbol{B}^{(2)}$ and $\widehat{\boldsymbol{B}} = \frac{1}{|\mathcal{W}|} \widehat{\boldsymbol{B}^{(1)}} \times \widehat{\boldsymbol{B}^{(2)}}$. Therefore, $\| \boldsymbol{B} \|_2^2 \leq \| \boldsymbol{B}^{(1)} \|_2^2 \| \boldsymbol{B}^{(2)} \|_2^2$, with equality if, and only if we can write $\left[\boldsymbol{B}^{(1)}\right]_{x,w} = \alpha_x \gamma_w$ and $\left[\boldsymbol{B}^{(2)}\right]_{w,y} = \gamma_w \beta_y$, for some $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{X}|}$, $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{Y}|}$ and $\boldsymbol{\gamma} \in \mathbb{R}^{|\mathcal{W}|}$.*

Proof. As $\boldsymbol{T} = \boldsymbol{T}^{(1)} \times \boldsymbol{T}^{(2)}$, we have

$$[\boldsymbol{B}]_{x,y} = [\boldsymbol{T}]_{x,y} - \frac{1}{|\mathcal{Y}|} = \sum_{w \in \mathcal{W}} \left(\left[\boldsymbol{B}^{(1)}\right]_{x,w} + \frac{1}{|\mathcal{W}|} \right) \left(\left[\boldsymbol{B}^{(2)}\right]_{w,y} + \frac{1}{|\mathcal{Y}|} \right) - \frac{1}{|\mathcal{Y}|} \ .$$

As $\sum_w \left[\boldsymbol{B}^{(1)}\right]_{x,w} = 0$, we obtain $[\boldsymbol{B}]_{x,y} = \left[\boldsymbol{B}^{(1)} \times \boldsymbol{B}^{(2)}\right]_{x,y} + \frac{1}{|\mathcal{W}|} \sum_w \left[\boldsymbol{B}^{(2)}\right]_{w,y}$. The fact that $P^{(1)}$ is uniformly distributed implies that $P^{(2)}$ and $P^{(3)}$ are uniformly distributed and thus, as ϕ, ξ, and ψ are balanced, that X, Z, and Y are also uniformly distributed. In that case, we know that $\sum_{w \in \mathcal{W}} \left[\boldsymbol{B}^{(2)}\right]_{w,y} = 0$, which proves that $\boldsymbol{B} = \boldsymbol{B}^{(1)} \times \boldsymbol{B}^{(2)}$. We also have

$$\begin{aligned}
\left[\widehat{\boldsymbol{B}^{(1)}} \times \widehat{\boldsymbol{B}^{(2)}}\right]_{u,v} &= \sum_{a \in \mathcal{W}} \left[\widehat{\boldsymbol{B}^{(1)}}\right]_{u,a} \left[\widehat{\boldsymbol{B}^{(2)}}\right]_{a,v} \\
&= \sum_{\substack{(x,w) \in \mathcal{X} \times \mathcal{W} \\ (w',y) \in \mathcal{W} \times \mathcal{Y}}} (-1)^{u \cdot x \oplus v \cdot y} \left[\boldsymbol{B}^{(1)}\right]_{x,w} \left[\boldsymbol{B}^{(2)}\right]_{w',y} \sum_{a \in \mathcal{W}} (-1)^{a \cdot (w \oplus w')} \\
&= |\mathcal{W}| \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} (-1)^{u \cdot x \oplus v \cdot y} \sum_{w \in \mathcal{W}} \left[\boldsymbol{B}^{(1)}\right]_{x,w} \left[\boldsymbol{B}^{(2)}\right]_{w,y} \\
&= |\mathcal{W}| \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} (-1)^{u \cdot x \oplus v \cdot y} [\boldsymbol{B}]_{x,y} \\
&= |\mathcal{W}| \left[\widehat{\boldsymbol{B}}\right]_{u,v} \ ,
\end{aligned}$$

which proves that $\widehat{\boldsymbol{B}} = \frac{1}{|\mathcal{W}|} \widehat{\boldsymbol{B}^{(1)}} \times \widehat{\boldsymbol{B}^{(2)}}$. Finally, from Cauchy-Schwarz inequality:

$$\begin{aligned}
\| \boldsymbol{B}^{(1)} \times \boldsymbol{B}^{(2)} \|_2^2 &= \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \left(\sum_{w \in \mathcal{W}} \left[\boldsymbol{B}^{(1)}\right]_{x,w} \left[\boldsymbol{B}^{(2)}\right]_{w,y} \right)^2 \\
&\leq \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} \left(\sum_{w \in \mathcal{W}} \left[\boldsymbol{B}^{(1)}\right]_{x,w}^2 \right) \left(\sum_{w' \in \mathcal{W}} \left[\boldsymbol{B}^{(2)}\right]_{w',y}^2 \right) \\
&= \| \boldsymbol{B}^{(1)} \|_2^2 \| \boldsymbol{B}^{(2)} \|_2^2 \ ,
\end{aligned}$$

with equality if, and only if, for all $x, y \in \mathcal{X} \times \mathcal{Y}$ there exists some $\lambda_{x,y}$ such that $\left[\boldsymbol{B}^{(1)}\right]_{x,w} = \lambda_{x,y}\left[\boldsymbol{B}^{(2)}\right]_{w,y}$, so that $\left[\boldsymbol{B}^{(1)}\right]_{x,w} = \lambda_{x,0}\left[\boldsymbol{B}^{(2)}\right]_{w,0} = \alpha_x\gamma_w$. Taking β_y equal to $\alpha_0/\lambda_{0,y}$ when $\lambda_{0,y} \not= 0$ and to zero otherwise leads to the announced result. □

How to find projections ϕ, ψ and ξ on larger spaces exhibiting such a Markovian property in a given block cipher remains however an open question to us. We may hope to *approximate* such a Markovian process.

4 Distinguishers Versus Key Recovery

In this section we show that our framework can adapt to key recovery instead of distinguishers. Let us consider a process which generates *independent* random variables $Z_{1,K}, \ldots, Z_{n,K}$ depending on some key $K \in \{0,1\}^k$. We assume that for one unknown value $K = K_0$ all $Z_{i,K}$'s follow distribution D_0, whereas when $K \not= K_0$ all $Z_{i,K}$'s follow distribution D_1. We consider the simple key ranking procedure which, for all possible $K \in \{0,1\}^k$, instantiates $\boldsymbol{z}_K^n$ and ranks K according to the grade $\mathrm{G}_K = \mathrm{LLR}(\boldsymbol{z}_K^n)$. For any $K \not= K_0$ we obtain (similarly to what we had in Theorem 6) that $\mathrm{G}_{K_0} - \mathrm{G}_K$ is approximatively normally distributed with expected value $n\Delta(\mathsf{D}_0)$ and standard deviation $\sqrt{2n\Delta(\mathsf{D}_0)}$. Hence we obtain $\mathrm{G}_{K_0} < \mathrm{G}_K$ (i.e., a wrong key K has a better rank than the right key K_0) with probability approximatively $\Phi\left(-\sqrt{n\Delta(\mathsf{D}_0)/2}\right)$. Let d be such that $n = d/\Delta(\mathsf{D}_0)$. This probability becomes $\Phi\left(-\sqrt{d/2}\right)$ which is approximatively $e^{-d/4}/\sqrt{2\pi}$ when d is large. So K_0 gets the highest grade with probability approximatively equal to $\left(1 - e^{-d/4}/\sqrt{2\pi}\right)^{2^k-1} \approx \exp\left(-2^k \cdot e^{-d/4}/\sqrt{2\pi}\right)$, which is high provided that $d \geq 4k \log 2$. Hence we need

$$n \geq \frac{4k \log 2}{\Delta(\mathsf{D}_0)} .$$

This formula is quite useful to estimate the complexity of many attacks, e.g. [19, 20][7]. We can finally note that the expected rank of K_0 (from 1 up to 2^k) is $1 + (2^k - 1)\Phi\left(-\sqrt{n\Delta(\mathsf{D}_0)/2}\right)$.

5 Conclusion

Most modern block ciphers are proven to be resistant to linear cryptanalysis in some sense. In this paper, we wonder how this resistance extends to (both known and unknown) generalizations of linear cryptanalysis. For this, we define a sound

[7] Note that [19, 20] use slightly different notations: $\Delta(\mathsf{D}_0)$ denotes the Euclidean Imbalance instead of the *Squared* Euclidean Imbalance.

and rigorous statistical framework which allows us to interpret most of these attacks in a simple and unified way; secondly, we develop a set of useful statistical tools to describe such attacks and to analyze their performance. Recently, our results on GF(2)-linear projections were exploited by [20] to obtain a small improvement factor in an attack on E0, and by [19] in another attack against two-level E0 [19]. In the sequel of this paper, we observe that resistance to linear cryptanalysis implies (a somewhat weaker) resistance to generalizations based on GF(2)-*linear projections*; however this resistance does not extend to *all* statistical cryptanalysis, as demonstrated by our example exploiting correlations in $\mathbb{Z}_4$, which confirms observations of Parker and Standaert et al. [27, 29]. The next natural step, which we hope to have rendered easier, will be to exhibit such a practical statistical cryptanalysis against a block cipher immune to linear cryptanalysis, like AES [4].

References

[1] D. Coppersmith, S. Halevi, and C. Jutla. Cryptanalysis of stream ciphers with linear masking. In M. Yung, editor, *Advances in Cryptology* - CRYPTO *'02*, volume 2442 of *LNCS*, pages 515–532. Springer-Verlag, 2002.

[2] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications. John Wiley & Sons, 1991.

[3] J. Daemen, R. Govaerts, and J. Vandewalle. Correlation matrices. In B. Preneel, editor, *Fast Software Encryption* - FSE *'94*, volume 1008 of *LNCS*, pages 275–285. Springer-Verlag, 1995.

[4] J. Daemen and V. Rijmen. *The Design of Rijndael.* Information Security and Cryptography. Springer-Verlag, 2002.

[5] W. Feller. *An Introduction to Probability Theory and Its Applications.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, third edition, 1968.

[6] H. Handschuh and H. Gilbert. χ^2 cryptanalysis of the SEAL encryption algorithm. In E. Biham, editor, *Fast Software Encryption* - FSE *'97*, volume 1267 of *LNCS*, pages 1–12. Springer-Verlag, 1997.

[7] C. Harpes, G. Kramer, and J. Massey. A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma. In L.C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology* - EUROCRYPT *'95*, volume 921 of *LNCS*, pages 24–38. Springer-Verlag, 1995.

[8] C. Harpes and J. Massey. Partitioning cryptanalysis. In E. Biham, editor, *Fast Software Encryption* - FSE *'97*, volume 1267 of *LNCS*, pages 13–27. Springer-Verlag, 1997.

[9] T. Jakobsen. *Higher-order cryptanalysis of block ciphers.* PhD thesis, Department of Mathematics, Technical University of Denmark, 1999.

[10] T. Jakobsen and C. Harpes. Non-uniformity measures for generalized linear cryptanalysis and partitioning cryptanalysis. In J. Pribyl, editor, PRAGOCRYPT *'96*. CTU Publishing House, 1996.

[11] P. Junod. On the optimality of linear, differential and sequential distinguishers. In E. Biham, editor, *Advances in Cryptology* - EUROCRYPT *'03*, volume 2656 of *LNCS*, pages 17–32. Springer-Verlag, 2003.

[12] P. Junod and S. Vaudenay. Optimal key ranking procedures in a statistical cryptanalysis. In T. Johansson, editor, *Fast Software Encryption* - FSE*'03*, volume 2887 of *LNCS*, pages 235–246. Springer-Verlag, 2003.

[13] B. Kaliski and M. Robshaw. Linear cryptanalysis using multiple approximations. In Y.G. Desmedt, editor, *Advances in Cryptology* - CRYPTO*'94*, volume 839 of *LNCS*, pages 26–39. Springer-Verlag, 1994.

[14] J. Kelsey, B. Schneier, and D. Wagner. mod$n$ cryptanalysis, with applications against RC5P and M6. In L. Knudsen, editor, *Fast Software Encryption* - FSE*'99*, volume 1636 of *LNCS*, pages 139–155. Springer-Verlag, 1999.

[15] L. Knudsen and M. Robshaw. Non-linear approximations in linear cryptanalysis. In U. Maurer, editor, *Advances in Cryptology* - EUROCRYPT*'96*, volume 1070 of *LNCS*, pages 224–236. Springer-Verlag, 1996.

[16] X. Lai, J. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In D.W. Davies, editor, *Advances in Cryptology* - EUROCRYPT*'91*, volume 547 of *LNCS*, pages 17–38. Springer-Verlag, 1991.

[17] C.H. Lim. CRYPTON: A new 128-bit block cipher. In *The First AES Candidate Conference*. National Institute for Standards and Technology, 1998.

[18] C.H. Lim. A revised version of CRYPTON: CRYPTON V1.0. In L. Knudsen, editor, *Fast Software Encryption* - FSE*'99*, volume 1636 of *LNCS*, pages 31–45. Springer-Verlag, 1999.

[19] Y. Lu and S. Vaudenay. Cryptanalysis of Bluetooth Keystream Generator Two-level E0. In *Advances in Cryptology* - ASIACRYPT*'04*, LNCS. Springer-Verlag, 2004.

[20] Y. Lu and S. Vaudenay. Faster correlation attack on Bluetooth keystream generator E0. In M. Franklin, editor, *Advances in Cryptology* - CRYPTO*'04*, volume 3152 of *LNCS*, pages 407–425. Springer-Verlag, 2004.

[21] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology* - EUROCRYPT*'93*, volume 765 of *LNCS*, pages 386–397. Springer-Verlag, 1993.

[22] M. Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Y.G. Desmedt, editor, *Advances in Cryptology* - CRYPTO*'94*, volume 839 of *LNCS*, pages 1–11. Springer-Verlag, 1994.

[23] M. Matsui. New structure of block ciphers with provable security against differential and linear cryptanalysis. In D. Gollman, editor, *Fast Software Encryption* - FSE*'96*, volume 1039 of *LNCS*, pages 205–218. Springer-Verlag, 1996.

[24] M. Minier and H. Gilbert. Stochastic cryptanalysis of Crypton. In B. Schneier, editor, *Fast Software Encryption* - FSE*'00*, volume 1978 of *LNCS*, pages 121–133. Springer-Verlag, 2000.

[25] S. Murphy, F. Piper, M. Walker, and P. Wild. Likelihood estimation for block cipher keys. Technical report, Information Security Group, University of London, England, 1995.

[26] National Institute of Standards and Technology, U. S. Department of Commerce. *Data Encryption Standard*, NIST FIPS PUB 46-2, 1993.

[27] M. Parker. Generalized S-Box linearity. Technical report `nes/doc/uib/wp5/020/a`, NESSIE Project, 2003. Available on `https://www.cryptonessie.org`.

[28] T. Shimoyama and T. Kaneko. Quadratic relation of S-Box and its application to the linear attack of full round DES. In H. Krawczyk, editor, *Advances in Cryptology* - CRYPTO *'98*, volume 1462 of *LNCS*, pages 200–211. Springer-Verlag, 1998.

[29] F.-X. Standaert, G. Rouvroy, G. Piret, J.-J. Quisquater, and J.-D. Legat. Key-dependent approximations in cryptanalysis: an application of multiple Z4 and non-linear approximations. In *24th Symposium on Information Theory in the Benelux*, 2003.

[30] A. Tardy-Corfdir and H. Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In J. Feigenbaum, editor, *Advances in Cryptology* - CRYPTO *'91*, volume 576 of *LNCS*, pages 172–182. Springer-Verlag, 1992.

[31] S. Vaudenay. An experiment on DES statistical cryptanalysis. In *3rd ACM Conference on Computer and Communications Security*, pages 139–147. ACM Press, 1996.

[32] S. Vaudenay. On the security of CS-cipher. In L. Knudsen, editor, *Fast Software Encryption* - FSE *'99*, volume 1636 of *LNCS*, pages 260–274. Springer-Verlag, 1999.

[33] S. Vaudenay. Decorrelation: a theory for block cipher security. *Journal of Cryptology*, 16(4):249–286, 2003.

[34] D. Wagner. Towards a unifying view of block cipher cryptanalysis. In B. Roy and W. Meier, editors, *Fast Software Encryption* - FSE *'04*, volume 3017 of *LNCS*, pages 16–33. Springer-Verlag, 2004.

A A Strange Distribution

We consider a source generating a random variable $S = (X_1, \ldots, X_{n+1}) \in \mathbb{Z}_4^{n+1}$, where n is some large integer, which follows either D_0 or D_1 (the uniform distribution). The distribution D_0 is such that the $X_1, \ldots X_n$ are uniformly distributed iid random variables and $X_{n+1} = (Y + \sum_{i=1}^{n} X_i) \bmod 4$, where $Y \in \{0, 1\}$ is uniformly distributed and independent of $X_1, \ldots, X_n$. We claim that the linear probability of the best linear distinguisher with one query is very small (equal to $2^{-(n+1)}$) whereas it is still possible to find a projection h such that $Z = h(S)$ has a high SEI. In order to simplify the proof, we will suppose that $n + 1$ is a multiple of 4.

Proposition 18. *Let* $h : \mathbb{Z}_4^{n+1} \to \mathbb{Z}_2$ *be defined by* $h(S) = \mathrm{msb}\left((X_{n+1} - \sum_{i=1}^{n} X_i) \bmod 4\right)$. *Then the* SEI *of* $Z = h(S)$ *is 1.*

The following lemmas will be used to prove that the best linear distinguisher is drastically less powerful than the distinguisher of Proposition 18.

Lemma 19. *Let* $\boldsymbol{u} = u_1 u_2 \ldots u_n$ *be a string of* n *bits. If we denote* w *the Hamming weight of* $\boldsymbol{u}$ *then we have*

$$\sum_{1 \le j < k \le n} u_j u_k = \frac{w(w-1)}{2} \quad .$$

Lemma 20. *For any positive integer N, we have:*

$$\sum_{j=0}^{\lfloor N/4 \rfloor} \binom{N}{4j} = \frac{1}{4}\left(2^N + (1+i)^N + (1-i)^N\right) \quad \textit{and}$$

$$\sum_{j=0}^{\lfloor (N-1)/4 \rfloor} \binom{N}{4j+1} = \frac{1}{4}\left(2^N - i(1+i)^N + i(1-i)^N\right) \ ,$$

where i is the imaginary unit equal to $\sqrt{-1}$.

Proposition 21. *When S follows D_0 we have $\mathsf{LP}^S_{max} = 2^{-(n+1)}$.*

Proof. Each X_i is in $\mathbb{Z}_4$ so that it can be described by two bits, denoted $X_i^{\mathrm{H}} X_i^{\mathrm{L}}$. If S is considered like a bit string, a linear distinguisher will be defined by a hash function h such that

$$h(S) = \left(\bigoplus_{j=1}^{n+1} a_j X_j^{\mathrm{L}}\right) \oplus \left(\bigoplus_{j=1}^{n+1} b_j X_j^{\mathrm{H}}\right) \ ,$$

where $a_1, \ldots, a_{n+1}, b_1, \ldots, b_{n+1} \in \{0,1\}$ with at least one non-zero value. We easily prove that

$$X_{n+1}^{\mathrm{L}} \oplus Y = \bigoplus_{j=1}^{n} X_j^{\mathrm{L}} \ \text{ and } \ X_{n+1}^{\mathrm{H}} = \left(\bigoplus_{j=1}^{n} X_j^{\mathrm{H}}\right) \oplus \left(\bigoplus_{j<k\le n} X_j^{\mathrm{L}} X_k^{\mathrm{L}}\right) \oplus \left(\bigoplus_{j=1}^{n} X_j^{\mathrm{L}} Y\right).$$

Thus, if B denotes the value of the bit $h(S)$, we have

$$B = \left(\bigoplus_{j=1}^{n} (a_j \oplus a_{n+1}) X_j^{\mathrm{L}}\right) \oplus \left(\bigoplus_{j=1}^{n} (b_j \oplus b_{n+1}) X_j^{\mathrm{H}}\right) \oplus a_{n+1} Y$$
$$\oplus \left(b_{n+1} \bigoplus_{1\le j<k\le n} X_j^{\mathrm{L}} X_k^{\mathrm{L}}\right) \oplus \left(b_{n+1} \bigoplus_{j=1}^{n} X_j^{\mathrm{L}} Y\right) \ .$$

If $b_{n+1} = 0$ we can see that (as at least one of the $a_1, \ldots, a_{n+1}, b_1, \ldots, b_n$ is strictly positive) $\Pr_{\mathsf{D}_0}[B = 0] = \frac{1}{2}$, hence $\mathsf{LP}(B) = 0$. If $b_{n+1} = 1$, we have

$$B = \left(\bigoplus_{j=1}^{n} (a_j \oplus a_{n+1}) X_j^{\mathrm{L}}\right) \oplus \left(\bigoplus_{j=1}^{n} \overline{b_j} X_j^{\mathrm{H}}\right) \oplus a_{n+1} Y$$
$$\oplus \left(\bigoplus_{1\le j<k\le n} X_j^{\mathrm{L}} X_k^{\mathrm{L}}\right) \oplus \left(\bigoplus_{j=1}^{n} X_j^{\mathrm{L}} Y\right) \ .$$

If one of the $\overline{b_j}$'s is non-zero, then B is uniformly distributed so $\mathsf{LP}(B) = 0$. We now assume that $b_j = 1$ for all $j = 1, \ldots, n$. We have

$$B = \left(\bigoplus_{j=1}^{n} (a_j \oplus a_{n+1}) X_j^{\mathrm{L}} \right) \oplus a_{n+1} Y \oplus \left(\bigoplus_{1 \le j < k \le n} X_j^{\mathrm{L}} X_k^{\mathrm{L}} \right) \oplus \left(\bigoplus_{j=1}^{n} X_j^{\mathrm{L}} Y \right) .$$

Let us define $U_j = X_j^{\mathrm{L}} \oplus a \oplus a_j$ for $j \in \{0, \ldots, n\}$ and $U_j = Y \oplus a$ for $j = n+1$, with $a = \bigoplus_{j=1}^{n+1} a_j$. We can show that

$$B = \left(\bigoplus_{1 \le j < k \le n+1} U_j U_k \right) \oplus c \,,$$

where $c \in \{0, 1\}$ is a constant. Using Lemma 19 and denoting W the Hamming weight of the random string of bits $U_1, \ldots, U_{n+1}$ we obtain

$$\begin{aligned} \Pr\,[B = c] &= \Pr\left[\frac{W(W-1)}{2} \equiv 0 \pmod 2 \right] = \Pr\,[W \bmod 4 = 0 \text{ or } 1] \\ &= \frac{1}{2^{n+1}} \sum_{j=0}^{\frac{n+1}{4}} \binom{n+1}{4j} + \frac{1}{2^{n+1}} \sum_{j=0}^{\lfloor \frac{n}{4} \rfloor} \binom{n+1}{4j+1} . \end{aligned}$$

Using Lemma 20 we deduce

$$\Pr\,[B = c] = \frac{1}{2} + \frac{(1+i)^n + (1-i)^n}{4 \times 2^n} = \frac{1}{2} + \frac{\cos\left(\frac{n\pi}{4}\right)}{2^{\frac{n}{2}+1}} = \frac{1}{2} + \frac{(-1)^{\frac{n+1}{4}}}{2^{\frac{n+3}{2}}} \,,$$

where we used the fact that $n+1$ is a multiple of 4. Finally, $\mathsf{LP}_{\max}^{S} = 2^{-(n+1)}$. □

The Davies-Murphy Power Attack

Sébastien Kunz-Jacques, Frédéric Muller, and Frédéric Valette

DCSSI Crypto Lab 51, Boulevard de Latour-Maubourg,
75700 Paris, 07 SP France
{Sebastien.Kunz-Jacques, Frederic.Muller,
Frederic.Valette}@sgdn.pm.gouv.fr

Abstract. In this paper, we introduce a new power analysis attack against DES. It is based on the well known Davies-Murphy attack. As for the original attack, we take advantage of non-uniform output distributions for two adjacent S-boxes. We show how to detect these biased distributions by power analysis on any DES inner round and thus obtain one bit of information about the key.

An advantage of this new attack is that no information about DES inputs or outputs is required. Therefore it is likely to defeat many actual countermeasures, in particular the popular masking techniques.

1 Introduction

Side-channel attacks have been developed in parallel to "classical" attack techniques since about 10 years. The initial publication by Kocher [13, 14] of Simple Power Analysis (SPA) and Differential Power Analysis (DPA) has been a major breakthrough in the domain. The general idea in this new family of attacks is to use "non-conventional" sources of information. Typically, the situation is we have a cryptographic device manipulating secret key or data which is protected against physical intrusion (we can think of this device as a smart-card, for instance). Then an attacker tries to obtain these secrets by measuring some external elements of information about the device. A leakage can result from the electric consumption of the device, its electromagnetic radiations, or simply by timing measurements. Some related attacks are also based on analyzing faults during the execution of the cryptographic computations [8].

Side channel attacks using the electric consumption are generally called "Power Attacks". It is widely believed that power consumption is always somehow correlated to the manipulated data. The question is thus to find appropriate countermeasures in order to thwart all known attacks. Power Attacks have been developed without distinction to secret and public key primitives. However in this paper, we mostly focus on the analysis of block ciphers. In this particular context, the most popular family of attacks are DPA [14] and its extended version, Higher-Order DPA [17, 21]. Advanced attacks usually revisit some techniques of "classical" cryptanalysis, like collision attacks [20] or differential attacks [15].

The goal of this paper is to propose a new power attack. We revisit the well-known Davies-Murphy cryptanalysis of DES [5, 11] and transform it into a power

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 451–467, 2004.

analysis attack. The "classical" attack uses non-uniform output distributions for each pair of adjacent S-boxes in DES. This property results from the duplication of some state bits by the expansion function. Non-uniform distributions result in detectable imbalance in electric consumption, and we propose several techniques to detect and exploit this imbalance. We call our new attack the Davies-Murphy Power Attack (DMPA).

First we discuss the model used to describe the correlation between intermediate data and power consumption. Then we recall the principles of DES, the Davies-Murphy attack and investigate some additional properties. In Section 5, the general principle of DMPA is exposed and we propose some tricks to apply it to various scenarios and different kinds of implementation. The final sections are dedicated to discussing the advantages and the extensions of DMPA.

2 The Power Consumption Model

In power analysis attacks, the basic assumption is that power consumption is somehow correlated with some data handled during the execution of an instruction. A classical assumption is the **Hamming weight model** [1, 9] where we suppose that the power consumption is proportional to the hamming weight of the manipulated data D. Let W be the power consumption and H the hamming weight function. We suppose that

$$W = \lambda\, H(D \oplus R) + \theta$$

where θ is a term of noise, λ a scalar and R a reference state from which we measure the number of bits flipped. For instance, R is often seen as a constant, unknown machine word (but R is not necessarily zero). The underlying assumption for electric consumption is that flipping a bit from 0 to 1 or flipping it from 1 to 0 costs almost the same thing, while keeping a bit unchanged costs almost nothing.

Many papers on side channel attacks [7, 10, 14, 18] observed empirically this correlation between the consumption of a smart card and the hamming weight of the operands. This model has also been verified more formally (see [9] for instance). Although a finer analysis has revealed that an extended linear model was sometimes more appropriate [1], it is still widely believed in practice that the Hamming weight model is a reasonable approximation. Actually it seems particularly well suited to model circuits based on the widely used CMOS technology, while it may be less appropriate for other technologies.

In the following, we suppose that the Hamming weight model is verified. We stress out that this model is not specifically helpful for our attack. We choose it because it is frequently used in the literature, and from our experience of cryptographic hardware, we believe it is very often appropriate. However our attack could probably be adapted to another model, as long as an actual correlation exists between W and D.

It is classically known that implementations can be subject to power analysis attacks when one of the following condition holds :

- the intermediate data D depends only on the plaintext and a small portion of key bits. This is the fundamental hypothesis for *Differential Power Analysis (DPA)*.
- a simple function of several intermediate data $D_1, \ldots, D_t$ depends only on the plaintext and a small portion of key bits. This is the fundamental hypothesis for *Higher-Order Differential Power Analysis (HO-DPA)*[1].

Then an attacker would use the correlation between intermediate data and power consumption to detect a correct guess of the key bits. Recent implementations take into account this threat by protecting all inner instructions. For instance a popular family of countermeasures consists in masking the manipulated data [2–4]. The underlying idea is that intermediate values should look random, even when the plaintext is known. However, most countermeasures do not take into account the fact that intermediate data may be biased **independently of the plaintext**. In the case of DES, this is actually the case because of Davies' observation about pairs of adjacent S-boxes [11]. In the next section, we focus on DES and recall the well-known Davies-Murphy attack.

3 DES and the Davies-Murphy Attack

The Data Encryption Standard (DES) is one of the most popular block cipher. Since it was selected as a standard by the NBS in 1977 [19], it has been the target of many research on cryptanalysis. Among all the results against DES, three attacks have emerged :

- Differential Cryptanalysis (DC) [6] was proposed by Biham and Shamir in 1990. It has been a major breakthrough and many applications to other algorithms have been demonstrated thereafter. Since then, it was revealed that the principle of DC was already known by the designers of DES.
- Linear Cryptanalysis (LC) [16] was proposed by Matsui in 1993. Like DC, it became quickly very popular and was applied successfully to other algorithms. In addition, this attack was practically implemented by Matsui in the case of DES. This technique was presumably not known by the designers of DES.
- The Davies-Murphy Cryptanalysis [5, 11] is a dedicated attack against DES. It takes advantage of biased distributions for two adjacent S-boxes. Although less generic than the previous two, Davies-Murphy cryptanalysis is a concern for Feistel ciphers with a non-bijective round function.

First we remind the general structure of DES (see Figure 1). We call F the round function, iterated 16 times in this case.

F is represented in more details in Figure 2. The general idea of Davies-Murphy attack is to look at two adjacent S-boxes (say S_1 and S_2). Because of the expansion phase, two bits of the input have been duplicated and are shared by the inputs of S_1 and S_2. These two bits are the two rightmost bits of S_1 and

[1] Here it is t-th order DPA, since t intermediate data are considered.

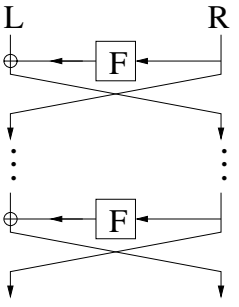


Fig. 1. The general structure of DES

the two leftmost bits of S_2. Consequently the output distributions for S_1 and for S_2 are not independent. A precise analysis shows that the joint distribution is not uniform. Moreover, depending on one key bit[2], two distributions (both non uniform) can be observed. Theoretically this allows an attacker to learn the sum of the 4 key bits corresponding to the shared positions in S_1 and S_2 (see [5, 11]).

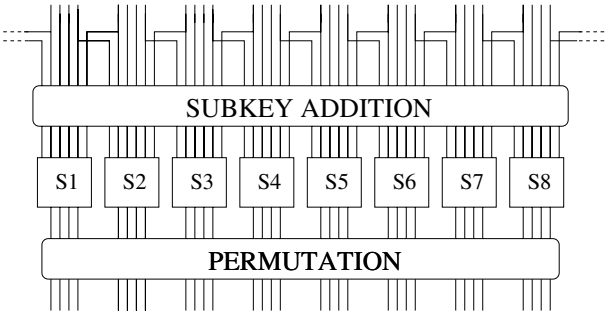


Fig. 2. The round function of DES

To give an illustration of the Davies-Murphy biased distributions, we focus on the S-boxes S_1 and S_2. We denote by (k_1, k_2, k_3, k_4) the 4 subkey bits corresponding to the "shared" positions of S_1 and S_2, and we call $k = k_1 \oplus k_2 \oplus k_3 \oplus k_4$ the sum of these 4 bits. In Table 1 we represent the output distributions for both cases $k = 0$ and $k = 1$. y_1 and y_2 represent respectively the outputs of S_1 and S_2. These distributions were simply obtained by looping on all possible inputs of S_1 and S_2.

This kind of imbalance was initially observed by Davies [11]. At first, it was thought that the attack could not be extended to the full DES. Indeed the previous observation extends to 16 rounds by composing 8 times the distributions.

[2] Actually, it is one linear combination of key bits.

Table 1. Biased Distributions for S_1 and S_2 (all elements in the table should be divided by 2^{10})

y_2 \ y_1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
01	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
02	2	2	4	6	4	4	6	4	6	4	0	4	4	2	6	6	6	6	4	2	4	4	2	4	2	4	8	4	4	6	2	2
03	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
04	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
05	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
06	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
07	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
08	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
09	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
10	6	6	4	2	4	4	2	4	2	4	8	4	4	6	2	2	2	2	4	6	4	4	6	4	6	4	0	4	4	2	6	6
11	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
12	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5
13	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
14	3	3	4	5	4	4	5	4	5	4	2	4	4	3	5	5	5	5	4	3	4	4	3	4	3	4	6	4	4	5	3	3
15	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
	Case $k = 0$																Case $k = 1$															

The XOR of plaintext and ciphertext is therefore non uniform and it turns out things depend only on one combination of key bits. Unfortunately the resulting imbalance is too small to be detected. Later on, Biham and Biryukov demonstrated how to improve this attack to obtain an attack faster than exhaustive search for the full DES [5]. In this paper we focus on Davies-Murphy's biased distributions for just one round.

4 Extension of Davies-Murphy to the Hamming Weight

The key observation of Davies-Murphy attack is that, for any DES inner round, intermediate data are not distributed uniformly, for randomly-chosen inputs. However in a power attack we do not have access directly to the intermediate data but to the power consumption (which is hopefully correlated to the data). Since we assume the Hamming weight model, this correlation depends on the Hamming weight of the S-box output. Hence it is natural to consider how the Davies-Murphy property translates to the Hamming weight.

As a first example, we consider the S-boxes S_1 and S_2 and look at the joint distribution of $(h_1, h_2) = (\mathrm{H}(S_1(x_1)), \mathrm{H}(S_2(x_2)))$ where x_1 and x_2 are uniformly chosen. The resulting distribution is given in Table 2.

Four values are biased in Table 2 (the corresponding positions are $(h_1, h_2) = (0, 2)$, $(4, 2)$, $(0, 3)$ and $(4, 3)$). Hence the imbalance exists but is not huge. Still, we hope to make it exploitable but we need to introduce appropriate statistical tools.

Definition 1. *Let $\mathcal{D}_1$, $\mathcal{D}_2$ be two distributions over some finite domain X. The* ***statistical distance*** *between $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined as*

$$|\mathcal{D}_1 - \mathcal{D}_2| = \sum_{x \in X} |\mathcal{D}_1(x) - \mathcal{D}_2(x)|$$

Table 2. Distributions of output hamming weight for S_1 and S_2 (all elements in the table should be divided by 2^{10})

Random Distribution						Case $k = 0$						Case $k = 1$					
h_2 \ h_1	0	1	2	3	4	h_2 \ h_1	0	1	2	3	4	h_2 \ h_1	0	1	2	3	4
0	4	16	24	16	4	0	4	16	24	16	4	0	4	16	24	16	4
1	16	64	96	64	16	1	16	64	96	64	16	1	16	64	96	64	16
2	24	96	144	96	24	2	26	96	144	96	22	2	22	96	144	96	26
3	16	64	96	64	16	3	14	64	96	64	18	3	18	64	96	64	14
4	4	16	24	16	4	4	4	16	24	16	4	4	4	16	24	16	4

Using this definition, we can compute the statistical distance between the previous distributions. Let $\mathcal{U}$ be the distribution of hamming weight for uniformly chosen inputs. $\mathcal{D}_i$ denotes the distribution in the case $k = i$. For S-boxes S_1 and S_2, we can easily compute :

$$|\mathcal{D}_0 - \mathcal{U}| = \frac{1}{128}$$
$$|\mathcal{D}_1 - \mathcal{U}| = \frac{1}{128}$$
$$|\mathcal{D}_1 - \mathcal{D}_0| = \frac{1}{64}$$

The imbalance for S-boxes S_1 and S_2 is not the best we can obtain. We repeated the same experience with different pairs of S-box and obtained better results. This is summarized in Table 3.

When using random inputs, all pairs of adjacent S-boxes present an imbalance regarding the output hamming weight. The best ones are obtained for (S_2,S_3), (S_7,S_8) and (S_8,S_1). One can also notice that

Table 3. Statistical distance between distributions $\mathcal{U}$, $\mathcal{D}_0$ and $\mathcal{D}_1$

S-boxes	$\lvert\mathcal{D}_0 - \mathcal{U}\rvert$	$\lvert\mathcal{D}_1 - \mathcal{U}\rvert$	$\lvert\mathcal{D}_1 - \mathcal{D}_0\rvert$
S_1 and S_2	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{64}$
S_2 and S_3	$\frac{3}{64}$	$\frac{3}{64}$	$\frac{3}{32}$
S_3 and S_4	$\frac{1}{256}$	$\frac{1}{256}$	$\frac{1}{128}$
S_4 and S_5	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{64}$
S_5 and S_6	$\frac{1}{128}$	$\frac{1}{128}$	$\frac{1}{64}$
S_6 and S_7	$\frac{3}{128}$	$\frac{3}{128}$	$\frac{3}{64}$
S_7 and S_8	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{16}$
S_8 and S_1	$\frac{1}{32}$	$\frac{1}{32}$	$\frac{1}{16}$

$$|\mathcal{D}_0 - \mathcal{U}| = |\mathcal{D}_1 - \mathcal{U}| = 0.5 \times |\mathcal{D}_1 - \mathcal{D}_0|$$

always holds due to the symmetry property :

$$\mathcal{D}_0(x) + \mathcal{D}_1(x) = 2 \cdot \mathcal{U}(x)$$

Therefore we have exhibited a Hamming weight version of the Davies-Murphy imbalance on DES, and we are confident that the electric consumption for adjacent S-boxes is biased, even **for randomly-chosen plaintexts**.

5 The Davies-Murphy Power Attack

In this section, we want to turn the imbalance of Hamming weight into a powerful side channel attack against DES. First, we need to specify which specific assumptions we make about the power consumption of the cryptographic device.

5.1 Assumptions

As mentioned previously, our general assumption is the Hamming weight model. However, before describing an attack, we need to precise more specifically this model.

A first and crucial question is to determine what the reference state R corresponds to in practice. In [9], some experiments were conducted on different hardwares to answer this question. Depending on the chips, different results were obtained. In many cases, R corresponded to the address of the input value or to the opcode of the current instruction. For other chips, R was always 0, presumably because these chips clear the bus between each instruction. Overall it is reasonable to consider that each instruction corresponds to a unique constant R.

More formally, we make the assumption that **there is a constant R_i, independent of the round, such that the electric consumption W_i of S-box S_i** is

$$W_i = \lambda \, \mathrm{H}(y_i \oplus R_i) + \theta$$

with the same notations than in Section 3.

Moreover, we suppose that **all S-box computations are done separately**, hence we can observe any W_i separately by looking at an appropriate portion of the power consumption curves. This assumption is reasonable, but may be subject to discussions, depending on the implementation. Indeed some computations might be done in parallel (for instance, on a 8-bit architecture, it is likely that pairs of adjacent S-boxes are executed simultaneously, thus we could observe only $W_{2i} + W_{2i-1}$).

We further explore these different scenarios in Section 6. Here, we explore only the case where all S-boxes are computed sequentially. This is convenient to describe a basic attack.

5.2 The Principle of the Attack

We have already seen that $(H(y_i), H(y_{i+1}))$ is biased *a priori* for random plaintext, depending just on one key bit k. Actually what we observe also depends on the unknown constants R_i, R_{i+1} and on the noisy term θ. The general idea of the attack is decomposed in 3 steps:

- First, we observe that the distribution of $(H(y_i \oplus R_i), H(y_{i+1} \oplus R_{i+1}))$ is, in general, still biased for most constants R_i, R_{i+1}.
- Secondly, we build an empirical distribution of $(H(y_i \oplus R_i), H(y_{i+1} \oplus R_{i+1}))$ by encrypting a set of randomly chosen plaintexts. Hence we need to identify the portion of curves corresponding to W_i and W_{i+1}, then to counter the influence of the noisy term. The resulting empirical distribution is then matched with theoretical results.
- Finally, a good method to perform this matching is proposed. Our strategy is to compare distributions for two different inner rounds (not necessarily consecutive).

1 - Adding the Constants in the Distributions. To analyze the influence of constants R_i, we simply explored all possible cases. Hence we have looked at distributions $(H(y_i \oplus R_i), H(y_{i+1} \oplus R_{i+1}))$ for various pairs of S-boxes, with all possible constants (R_i, R_{i+1}). These results are summarized in Table 4. As before, distribution $\mathcal{D}_i$ corresponds to the case $k = i$. Besides the column "constant $= 0$" corresponds to the previous results (see Table 3).

Table 4. Statistical distances with constant R_i's

S-boxes	Statistical Distance $\lvert\mathcal{D}_1 - \mathcal{D}_0\rvert$			
	constant $= 0$	worst constant	best constant	average value
(S_1,S_2)	$\frac{1}{64}$	0	$\frac{5}{32}$	$\frac{1.5}{32}$
(S_2,S_3)	$\frac{3}{32}$	$\frac{3}{32}$	$\frac{7}{32}$	$\frac{3.656}{32}$
(S_3,S_4)	$\frac{1}{128}$	0	$\frac{9}{128}$	$\frac{0.473}{32}$
(S_4,S_5)	$\frac{1}{64}$	0	$\frac{9}{64}$	$\frac{0.984}{32}$
(S_5,S_6)	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{3}{32}$	$\frac{1.195}{32}$
(S_6,S_7)	$\frac{3}{64}$	$\frac{1}{64}$	$\frac{9}{128}$	$\frac{1.262}{32}$
(S_7,S_8)	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{25}{128}$	$\frac{3.094}{32}$
(S_8,S_1)	$\frac{1}{16}$	$\frac{1}{128}$	$\frac{3}{32}$	$\frac{0.711}{32}$

Clearly, we observe that the average distance is quite significant for all pairs (it ranges from $\frac{0.473}{32} \simeq \frac{1}{64}$ to $\frac{3.656}{32} \simeq \frac{1}{8}$). We also observe that there are "good" and "bad" constants, but in average an imbalance is expected.

2 - Getting Rid of the Noise. In the second phase, our goal is to build empirical distributions. More precisely, we encrypt a set of M randomly chosen

plaintexts and we monitor the electric consumption. We target the appropriate portion of the curves to observe (W_i, W_{i+1}). Our goal is, from these observations, to decide the underlying value of $(H(y_i \oplus R_i), H(y_{i+1} \oplus R_{i+1}))$ for each sample, despite the noise. Hence we obtain an empirical distribution over M samples. It is well known that when M grows, the empirical distribution converges to the theoretical distribution. More precisely, to get rid of the noise, two situations must be distinguished :

1. Suppose we can repeat each experiments. Typically, we can obtain twice from the cryptographic device the same encryption and the same execution. This assumption is commonly used in power analysis attacks. In this case the noise is eliminated by multiplying the samples for each trace and computing the average consumption.
2. Suppose we cannot repeat any experiments. Typically, any encryption corresponds to a random plaintext. This can result from masking countermeasures (with a fresh mask for each block !) or from a randomized mode of operation (CBC plus IV for instance).

In the first hypothesis, there is just an extra workload per message to make the noise arbitrarily small. Typically, if we have a Gaussian noise with expected value 0 and standard deviation σ, we expect to reduce the standard deviation by a factor $\sqrt{M}$ if we repeat M times each experiment. Therefore, since our model is:

$$W_i = \lambda \, \mathrm{H}(y_i \oplus R_i) + \theta$$

we consider the noise is sufficiently small when $\lambda \gg \frac{\sigma}{\sqrt{M}}$, *i.e.* the noise is negligible compared to the data-dependent term.

In the second hypothesis, we cannot eliminate the noise by averaging methods. However we hope that it will only slightly perturb our empirical distributions. Hence we suppose $\lambda \gg \sigma$, so when making a decision for each hamming weight, we have a small probability p of making a mistake. Our practical experiments on a smart card confirmed this supposition (see Section 6). A justification is that the data-dependent terms represent the consumption of bus lines which is generally dominant in a chip. More precisely, our decisions are made using thresholds:

$$t - \frac{1}{2} < \frac{W_i}{\lambda} < t + \frac{1}{2} \Rightarrow \mathrm{H}(y_i \oplus R_i) = t$$

For example if $\frac{W_i}{\lambda}$ is in the range [2.5 - 3.5], we decide $\mathrm{H}(y_i \oplus R_i) = 3$. If the noise is indeed negligible, we are successful in predicting the hamming weight with overwhelming probability. This threshold strategy is summarized in Figure 3 and further analyzed in Appendix A. Of course, in practice, λ is not known but we can set up thresholds experimentally to fit to the observations. We stress out that this analysis requires a good knowledge of the electric behavior of the chip.

3 - Comparing Two Inner Rounds. After the step 2, we construct an empirical distribution of hamming weight from power consumption curves. We know this is biased depending on one round key bit k. However since the key is fixed,

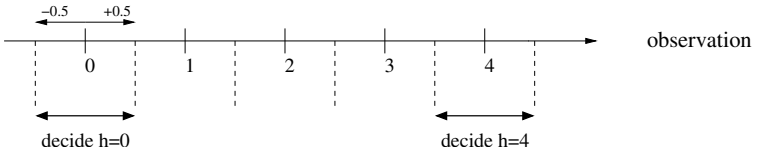


Fig. 3. Threshold rules of decision when observing $\frac{W_i}{\lambda}$

we have nothing to compare it with. Besides it is impossible to tell the value of k just by looking at the distribution, because it highly depends on the unknown R_i.

However an attack is still possible by **looking at two different inner rounds of DES** (not necessarily consecutive rounds). For instance, suppose we encrypt random plaintexts and compare the consumptions of round 1 and 2 for the adjacent S-boxes (S_2, S_3). At round 1 we observe (W_2, W_3), which is distributed differently depending on whether some first round key bit k is 0 or 1. These distributions are respectively called $\mathcal{D}_1$ and $\mathcal{D}_0$. A similar observation holds for round 2 with a second round key bit k'. Thus

- If $k = k' = i$, we have distributions $\mathcal{D}_i$ for both rounds.
- If $k \not= k'$, we have distributions $\mathcal{D}_0$ for one round and $\mathcal{D}_1$ for the other.

$\mathcal{D}_1$ and $\mathcal{D}_0$ depend on the constants R_i, but we have seen that, in average

$$|\mathcal{D}_1 - \mathcal{D}_0| = \frac{3.656}{32} \simeq 0.114$$

and even in the worst case, this value is $\frac{3}{32}$. So, in theory, if the number of samples M is sufficient (typically $M \geq \frac{1}{0.114^2} \simeq 100$), we should be able to tell if $k = k'$ or $k \not= k'$ and thus learn one bit of information about the key. In practice, we retrieve two empirical distributions $\mathcal{E}^0$ and $\mathcal{E}^1$. We must decide whether these distributions are the same $(k = k')$ or if they are different $(k \not= k')$. Because of the symmetry property exhibited in Section 4, we use the following indicator :

$$I = \sum_x (\mathcal{E}^0(x) - \mathcal{U}(x)) \times (\mathcal{E}^1(x) - \mathcal{U}(x))$$

Basically, we have normalized the empirical distributions by subtracting the $\mathcal{U}$ distribution, and then we compute a scalar product. If $k = k' = i$, then this indicator is positive :

$$I_{k=k'} = \sum_x (\mathcal{D}_i(x) - \mathcal{U}(x))^2$$

Otherwise, if $k \not= k'$, then the indicator should be negative

$$\begin{aligned} I_{k \not= k'} &= \sum_x (\mathcal{D}_0(x) - \mathcal{U}(x)) \times (\mathcal{D}_1(x) - \mathcal{U}(x)) \\ &= -\sum_x (\mathcal{D}_0(x) - \mathcal{U}(x))^2 \\ &= -\sum_x (\mathcal{D}_1(x) - \mathcal{U}(x))^2 \end{aligned}$$

because of the symmetry property described in Section 4. Therefore

$$I_{k=k'} = -I_{k \neq k'}$$

and these values are sufficiently large to be detected in practice.

5.3 Simulations

We ran some simulations of the previous distinguisher to evaluate our ability to predict correctly whether $k = k'$, and we obtained the results summarized in Table 5. Four intensity of noise where considered (see Appendix A for the role of the probability p) as well as several values for the number of samples M. We repeated the attack about 1000 times in each case, with a random choice of constants R_i.

Table 5. Simulation results

	Probability of success in Deciding if $k = k'$								
$p =$	0			0.1			0.25		
$M =$	256	4000	40000	256	4000	40000	256	4000	100000
(S_1, S_2)	0.5	0.65	0.98	0.5	0.59	0.90	0.5	0.51	0.69
(S_2, S_3)	0.67	0.99	1	0.58	0.96	1	0.52	0.61	0.99
(S_3, S_4)	0.5	0.54	0.83	0.5	0.54	0.72	0.5	0.5	0.54
(S_4, S_5)	0.5	0.59	0.94	0.5	0.56	0.80	0.5	0.51	0.59
(S_5, S_6)	0.5	0.59	0.93	0.5	0.56	0.81	0.5	0.51	0.63
(S_6, S_7)	0.51	0.61	0.96	0.5	0.57	0.84	0.5	0.51	0.65
(S_7, S_8)	0.70	0.99	1	0.58	0.95	1	0.51	0.59	0.99
(S_8, S_1)	0.5	0.57	0.91	0.5	0.56	0.77	0.5	0.51	0.58

It appears from Table 5 that the best pairs of S-boxes are (S_2, S_3) and (S_7, S_8), as predicted in Section 5.2. Hence, for our basic attack we will use any of these two pairs. For the variation attack with an 8-bit architecture, we can only use pairs with index of the form $(2i, 2i-1)$. Fortunately we can use the pair $(7, 8)$ here, which is strongly biased.

6 Some Variations of the Attack

In the previous section, we considered a simple hypothesis where all S-boxes were computed separately. Therefore we could identify portions of the power consumption curves corresponding to each S-box. In practice, the implementations are often more complex and we need to investigate if our attack applies to other situations

6.1 A Real-Life Situation : 8-Bit Architecture

As an example of our attack, we have considered a recent smart-card running a software DES implementation. The card also featured some usual hardware countermeasure (but no software countermeasure like masking). These countermeasures included a variable internal clock and some random peaks of power. Despite these protections, we managed to identify the power consumption corresponding to each portion of the DES execution. This analysis required first to understand well the behavior of the card. The trickiest part was to eliminate the random peaks of power but it turned out they were not "that" random and their presence was strongly correlated with the external clock.

In addition, we realized that two S-boxes are executed simultaneously by the card, *i.e.* each pair of adjacent S-boxes (S_1 and S_2, S_3 and S_4, etc ...). Therefore, the power consumption observed is $W_{2i} + W_{2i-1}$ for $i = 1 \dots 4$. It is strongly correlated with the sum of the hamming weights:

$$\begin{aligned} &\mathrm{H}(y_{2i} \oplus R_{2i}) + \mathrm{H}(y_{2i-1} \oplus R_{2i-1}) \\ &= \mathrm{H}((y_{2i} \oplus R_{2i}) || ((y_{2i-1} \oplus R_{2i-1}))) \end{aligned}$$

Accordingly, we expect to observe 9 groups of curves locally, if we have many samples (corresponding to hamming weight ranging from 0 to 8). In fact, due to the noise influence, it is difficult to make the groups appear very distinctly, but if we display a few curves (see Figure 4), a clear distinction starts to appear depending on the hamming weight. Low hamming weight correspond to low consumption (few bits are flipped from the reference states), while high hamming weights curves are located at the top of this Figure. In addition, these experiments illustrate the fact that some noise θ is indeed present, but it is relatively small compared to the data-dependent term, since the Hamming weight distinction appears clearly.

In this scenario, we can only observe the sum of power consumption for certain pairs of S-boxes. We computed theoretically the expected imbalance for these pairs (see Table 6). Here the distributions considered are over the sum of hamming weights $h_1 + h_2$ and not the joint distribution (h_1, h_2) as previously.

Hence, the statistical distances are still relatively high for the four pairs of S-boxes. To perform the Davies-Murphy power attack here, we can use again the trick of comparing two different inner rounds, like in Section 5.2.

6.2 Case When More S-Boxes are Computed Simultaneously

When considering software implementations of DES, we believe the most common situation are those where 1 or 2 S-boxes at most are computed simultaneously. This was developed in Section 5 and Section 6.1. To our knowledge, no software implementation presents a higher degree of parallelization than that.

However, when turning to DES hardware implementations, more than two S-boxes are often computed at the same time. Things are thus more complex

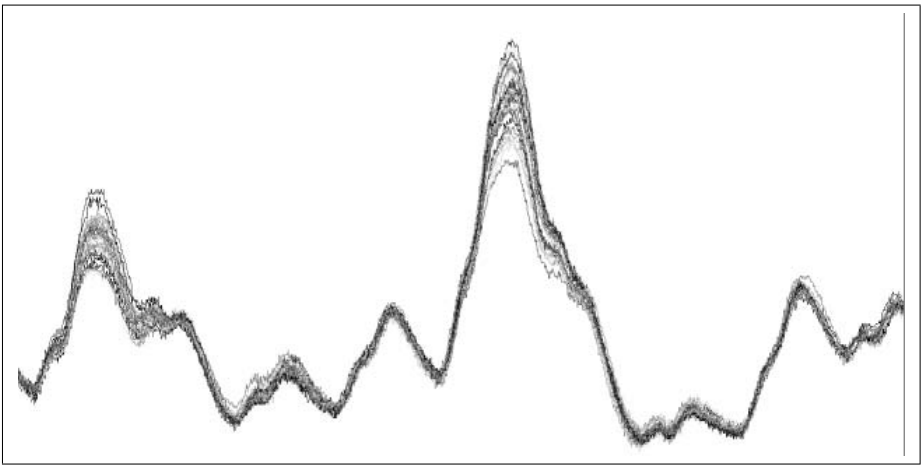


Fig. 4. Distinction of Curves According to the Hamming Weight

Table 6. Statistical distances in the 8-bit scenario

S-boxes	Statistical Distance $\lvert\mathcal{D}_1 - \mathcal{D}_0\rvert$			
	constant $= 0$	worst constant	best constant	average value
(S_1,S_2)	$\frac{1}{64}$	0	$\frac{1}{8}$	$\frac{1.097}{32}$
(S_3,S_4)	$\frac{1}{128}$	0	$\frac{3}{64}$	$\frac{0.406}{32}$
(S_5,S_6)	$\frac{1}{64}$	$\frac{1}{64}$	$\frac{21}{256}$	$\frac{1.076}{32}$
(S_7,S_8)	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{39}{256}$	$\frac{2.627}{32}$

because we observe many biased distributions simultaneously, and this depends on many key bits. We are currently investigating some refined version of our attack in this case. We believe an attack can be achieved since the imbalance is detectable in theory, but it will probably not be very efficient.

7 Impact on DES Implementations

Modern countermeasures against Side Channel Attacks are often focused against DPA. Accordingly they try to make intermediate data handled during the block cipher computation as random and unpredictable as possible. Two main techniques have receive a huge interest in recent years

– *Masking Techniques* [2–4] where the idea is to ensure that critical intermediate data are equal to the "true" data XOR some random mask. Masking the

round input has clearly no effect, since we process randomly-chosen data. However masking the output is problematic since the Davies imbalance now depends on the mask. But, for consistency, masking countermeasures generally require some unmasked round output (see [4]). This was not believed to be critical because the goal was to thwart DPA and HO-DPA. However DMPA will still work here, except we need to choose two inner rounds with unmasked output.

If all round outputs were masked with the same value, an higher-order version of DMPA could be envisaged[3]. So the best protection consists in masking all round outputs with a distinct value. But this is probably too expensive in practice (actual countermeasures use one or two masking values at most).

- *Duplication Techniques* [12] where all intermediate data are split into two parts, using the secret sharing principle. However, by analyzing simultaneously the behavior of both parts, the Davies imbalance should still be observed. Since everything is duplicated, the analysis is probably more complicated because 4 S-boxes need to be considered instead of just 2.

Therefore the Davies-Murphy Power Attack (DMPA) is likely to defeat most "software" countermeasures. In fact, this new attack is not fundamentally different from classical DPA : both gather power traces and sort them according to some intermediate data, the goal being to verify a guess on a few key bits. However, while DPA focuses on predicting some data from the plaintext and a few key bits, DMPA does not require the knowledge of the plaintext. The analysis is based only on the internal structure of DES and we can predict intermediate data (actually a bias on intermediate data), only from a few key bits. The advantage is that we can focus our analysis on any inner round, while DPA usually focuses on the first (or last) rounds of DES.

An other advantage is that countermeasures designed specifically to thw art the family of DPA attack (like masking, duplication, or others ...) are unlikely to be very efficient as a protection against DMPA. The main drawback of the attack is that it is rather expensive in terms of messages encrypted. Moreover it requires a fine analysis of the electric behavior of the target cryptographic hardware, in order to find an appropriate power consumption model and to identify each portion of the DES execution. So there is a lot of preliminary analysis to do before applying the attack.

Finally DMPA proves that even slight weakness or small "non-random" behavior of a cipher can be exploited to mount a side channel attack. Software countermeasures are helpful to complicate the task of the attacker, but a better protection against power attacks will be obtained if

- the cipher behaves as randomly as possible.
- efficient hardware countermeasures are implemented, to limit the information leaked in the electric consumption.

[3] Actually the trick from Section 5.2 of using power traces of any two inner rounds is already, by definition, a second-order attack.

8 Extensions

All our analysis has focused on the case of DES. Indeed the principle of Davies-Murphy attack was initially developed specifically against DES. However, more generally, for any Feistel cipher with a non-bijective round function, some imbalance in the round output necessarily exists. In this case, the requirements for DMPA are

- Express the output imbalance with a small number of key bits.
- Find a correlation between the non-randomly distributed data and the electric consumption

The first requirement depends on the cipher, while the second depends on the cryptographic hardware considered. We did not explore further to find applications on other algorithms but we believe it is an interesting topic for further research.

9 Conclusion

We have proposed a new side channel attack against DES, the Davies-Murphy Power Attack. It is based on the well known Davies-Murphy attack. Like its predecessor, our attack uses non-uniform output distributions of adjacent S-boxes. Then we detect this imbalance using electric consumption curves.

DMPA is very powerful, because it requires no information about the plaintext and can be performed on any inner rounds of DES. Therefore we believe it can defeat software countermeasures, which do not take into account this type of threat. However DMPA is rather expensive : good knowledge of the device behavior regarding power consumption is required, and the data processing complexity is rather high. For a non-protected implementation of DES, simpler side-channel attacks (like DPA) should be preferred.

References

1. M.-L. Akkar, R. Bevan, P. Dischamp, and D. Moyart. Power Analysis, What Is Now Possible ... In T. Okamoto, editor, *Advances in Cryptology – Asiacrypt'00*, volume 1976 of *Lectures Notes in Computer Science*, pages 489–502. Springer, 2000.
2. M.-L. Akkar, R. Bevan, and L. Goubin. Two Power Analysis Attacks against One-Mask Methods. In B. Roy and W. Meier, editors, *Fast Software Encryption – 2004*, pages 308–325, 2004. Pre-proceedings Version.
3. M.-L. Akkar and C. Giraud. An Implementation of DES and AES Secure against Some Attacks. In Ç. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 2162 of *Lectures Notes in Computer Science*, pages 309–318. Springer, 2001.
4. M.-L. Akkar and L. Goubin. A Generic Protection against High-Order Differential Power Analysis. In T. Johansson, editor, *Fast Software Encryption – 2003*, volume 2887 of *Lectures Notes in Computer Science*, pages 192–205. Springer, 2003.

5. E. Biham and A. Biryukov. An Improvement of Davies' Attack on DES. In A. De Santis, editor, *Advances in Cryptology – Eurocrypt'95*, volume 950 of *Lectures Notes in Computer Science*, pages 461–467. Springer, 1995.
6. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology – Crypto'90*, volume 537 of *Lectures Notes in Computer Science*, pages 2–21. Springer, 1990.
7. E. Biham and A. Shamir. Power Analysis of the Key Scheduling of the AES Candidates. In *Second AES Candidate Conference*, 1999.
8. D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In W. Fumy, editor, *Advances in Cryptology – Eurocrypt'97*, volume 1233 of *Lectures Notes in Computer Science*, pages 37–51. Springer, 1997.
9. E. Brier, C. Clavier, and F. Olivier. Optimal Statistical Power Analysis, 2003. Available on Eprint : `http://eprint.iacr.org/2003/152/`.
10. C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 1965 of *Lectures Notes in Computer Science*, pages 252–263. Springer, 2000.
11. D. Davies and S. Murphy. Pairs and Triplets of DES S-Boxes. *Journal of Cryptology*, 8(1):1–25, 1995.
12. L. Goubin and J. Patarin. DES and Differential Power Analysis, The "Duplication" Method. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *Lectures Notes in Computer Science*, pages 158–172. Springer, 1999.
13. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Others Systems. In N. Koblitz, editor, *Advances in Cryptology – Crypto'96*, volume 1109 of *Lectures Notes in Computer Science*, pages 104–113. Springer, 1996.
14. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – Crypto'99*, volume 1666 of *Lectures Notes in Computer Science*, pages 388–397. Springer, 1999.
15. H. Ledig, F. Muller, and F. Valette. Enhancing Collision Attacks. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2004. to appear.
16. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In T. Helleseth, editor, *Advances in Cryptology – Eurocrypt'93*, volume 765 of *Lectures Notes in Computer Science*, pages 386–397. Springer, 1993.
17. T. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant software. In Ç. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 1965 of *Lectures Notes in Computer Science*, pages 238–251. Springer, 2000.
18. T. Messerges, E. Dabbish, and R. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *USENIX Workshop on Smartcard Technology 1999*, 1999. Available at `http://www.usenix.org/`.
19. NIST FIPS PUB 46-3. *Data Encryption Standard*, 1977.
20. K. Schramm, T. Wollinger, and C. Paar. A New Class of Collision Attacks and its Application to DES. In T. Johansson, editor, *Fast Software Encryption – 2003*, volume 2887 of *Lectures Notes in Computer Science*. Springer, 2003.
21. J. Waddle and D. Wagner. Towards Efficient Second Order Power Analysis. In *Cryptographic Hardware and Embedded Systems (CHES)*, 2004. to appear.

A Statistical Influence of the Noise

In Section 5, we are interested in distinguishing two distributions, $\mathcal{D}_0$ and $\mathcal{D}_1$. Our analysis of DES revealed that the statistical distance $d = |\mathcal{D}_0 - \mathcal{D}_1|$ was sufficiently large to distinguish between these two distributions. But in practice we need to build an empirical observation of these distributions in the presence of noise. As we argued, this noise is generally small, but still it may result in errors of prediction in the threshold technique of Figure 3. There is a small probability p that the noise is larger than $0.5\ \lambda$ and thus we predict $h+1$ or $h-1$ instead of the "true" h. We call $\mathcal{D}'_i$ the new distribution obtained when the noise is taken into account. We have :

$$\mathcal{D}'_i(h) = p\ \mathcal{D}_i(h-1) + p\ \mathcal{D}_i(h+1) + (1-2p)\ \mathcal{D}_i(h)$$

Thus

$$\Delta(h) = \mathcal{D}'_i(h) - \mathcal{D}_i(h) = p \cdot [\mathcal{D}_i(h-1) + \mathcal{D}_i(h+1) - 2\ \mathcal{D}_i(h)]$$

$\Delta(h)$ is the difference of probability of deciding h, resulting from the noise influence. We see that if $p \ll 1$, then $\Delta(h) \ll 1$ for all $h \in \{0, \ldots, 4\}$. Therefore

$$|\mathcal{D}'_i - \mathcal{D}_i| = \sum_h |\Delta(h)| = \varepsilon \ll 1$$

for $i = 0, 1$. Hence, it is still possible to make the difference between the two distributions since

$$\begin{aligned} |\mathcal{D}'_1 - \mathcal{D}'_0| &\leq |\mathcal{D}'_1 - \mathcal{D}_1| + |\mathcal{D}_1 - \mathcal{D}_0| + |\mathcal{D}'_0 - \mathcal{D}_0| \\ &\leq |\mathcal{D}_1 - \mathcal{D}_0| + 2\ \varepsilon \end{aligned}$$

Therefore as long as the noise results in small probabilities of incorrect decisions, we can still apply the same methods.

Time-Memory Trade-Off Attacks on Multiplications and T-Functions

Joydip Mitra[1] and Palash Sarkar[2]

[1] Management Development Institute, Post Box No. 60, Mehrauli Road, Sukhrali Gurgaon 122001, Haryana, India
joydip@mdi.ac.in
[2] Cryptology Research Group, Applied Statistics Unit, Indian Statistical Institute, 203, B.T. Road, Kolkata, India 700108
palash@isical.ac.in

Abstract. T–functions are a new class of primitives which have recently been introduced by Klimov and Shamir. The several concrete proposals by the authors have multiplication and squaring as core nonlinear operations. Firstly, we present time-memory trade-off algorithms to solve the problems related to multiplication and squaring. Secondly, we apply these algorithms to two of the proposals of multi-word T-functions. For the proposal based on multiplication we can recover the 128 unknown bits of the state vector in 2^{40} time whereas for the proposal based on squaring the 128 unknown bits can be recovered in 2^{21} time. The required amount of key stream is a few (less than five) 128-bit blocks. Experimental data from implementation suggests that our attacks work well in practice and hence such proposals are not secure enough for stand-alone usage. Finally, we suggest the use of conjugate permutations to possibly improve the security of T–functions while retaining some attractive theoretical properties.

Keywords: stream cipher, T–functions, multiplication, cryptanalysis, time-memory trade-off.

1 Introduction

Stream ciphers are a fundamental primitive in cryptography. Encryption is performed by XORing the message bit sequence with a pseudo-random bit sequence while decryption is performed by XORing the cipher bit sequence once more with the same pseudo-random bit sequence.

The cryptographic strength of a stream cipher depends on the unpredictability of the pseudo-random bit sequence. The other important issue is efficiency of the pseudo-random generator. Most practical proposals for stream ciphers strive to achieve a good balance between speed and security. Typically stream ciphers are built out of linear feedback shift registers, nonlinear Boolean functions and S-boxes. See [4] for various models of stream ciphers.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 468–482, 2004.

Recently Klimov and Shamir [1–3] have proposed a new class of primitives for design of stream ciphers. They call their primitive T–functions and have developed a nice theory for analysing T–functions. From an efficiency point of view, T–functions are extremely attractive, since they can be built using fast and easily available operations on most processors. From a security point of view, there are many nice features including the single cycle property of the underlying permutation.

Klimov and Shamir [3] have also introduced multi-word T–functions and have extended their theory to cover such functions. In [3], they present several concrete constructions of multi-word T–functions. A key constituent of their proposals is multiplication modulo 2^{64}.

Our Contributions: In the first part of the paper, we study the following problem related to multiplication. Suppose x, y and z are n-bit integers satisfying $xy \bmod 2^n = z$. Further, suppose the m most significant bits of x, y and z are known. The problem is to compute all possible combinations of the $(n-m)$ least significant bits of x and y such that the multiplication holds.

We present a time-memory trade-off algorithm to solve this problem and make a detailed study of the effectiveness of the algorithm under different scenarios. We also study the related problem of squaring, i.e., when $x = y$. It turns out that the algorithm for multiplication is not efficient for squaring and hence we develop a separate algorithm to solve this problem. Apart from the application to T–functions, our algorithm can possibly be used for analysing other ciphers based on multiplication.

The second part of the paper consists of analysing the security of two concrete proposals of multi-word T–functions from [3]. The first proposal involves multiplication and the T-function operates on a state vector consisting of four 64-bit words. The pseudo-random bit sequence obtained from the state vector consists of the 32 most significant bits of each of the four 64-bit words. Thus the state vector has 128 unknown bits. We perform a detailed analysis of this T-function. The major step in the analysis consists of an application of (a modification) of the algorithm to solve multiplication as mentioned above. The final result that we obtain is that the 128 unknown bits can be computed in 2^{40} time which makes this proposal unsafe for stand-alone use as a pseudo-random generator.

The second proposal that we consider also operates on a state vector of four 64-bit words and produces 128 bits as before. The difference is that this proposal involves squaring instead of multiplication. Consequently, our analysis of this proposal involves the algorithm to solve squaring. In this case, we obtain an algorithm that determines the 128 unknown bits in 2^{21} time. Hence this proposal is much more insecure than the one based on multiplication.

The required amount of known pseudo-random key stream for both the above attacks is only a few (less than five) 128-bit consecutive key stream blocks. In most cases, we expect the attack to work with only three 128-bit consecutive key stream blocks. This shows that these two proposals, and probably other similar proposals, are not secure enough for stand-alone usage.

One possibility for improving the security is to extract less number of bits from each state vector. We consider this possibility for the multiplication based

T–function mentioned above, where only 16 most significant bits of each 64-bit word of the state vector is produced as output. Thus a total of 64 bits are produced from each state vector and 192 bits are unknown. Our attack also applies to this situation and the 192 unknown bits can be obtained in 2^{112} time. Though infeasible in practice, this constitutes a theoretical attack on the system.

We have implemented the algorithm to solve multiplication and our estimate of the expected run-time is supported by experimental data. We have also implemented the attack on a scaled down version of the multiplication based T–function. Instead of a state vector consisting of four 64-bit words we have worked with four 32-bit words. In this case, we can actually recover the 64 unknown bits of the state vector. This shows that our attack works quite well in practice. We have also implemented the algorithm to solve the squaring problem and the corresponding attack on the 64-bit version of the squaring based T-function proposal. Experiments show that the attack performs as predicted by the theoretical analysis.

Finally, we suggest a method based on conjugate permutations to possibly improve the security of T-functions while maintaining some desirable features such as the single cycle property.

2 Multiplication

We consider the following problem. Suppose two n-bit integers x and y are multiplied modulo 2^n to obtain an n-bit integer z. The m most significant bits (MSBs) of x, y and z are known and we have to find all possible solutions for the $(n-m)$ least significant bits (LSBs) of x and y such that $xy \bmod 2^n = z$. This problem can be stated more precisely as follows:

Problem: Mult
Input: Three integers $x^{(1)}$, $y^{(1)}$ and $z^{(1)}$ such that, $0 \leq x^{(1)}, y^{(1)}, z^{(1)} < 2^m$.
Task: Find all pairs of integers $(x^{(0)}, y^{(0)})$ such that, $0 \leq x^{(0)}, y^{(0)} < 2^{n-m}$, $x = 2^{n-m}x^{(1)} + x^{(0)}$, $y = 2^{n-m}y^{(1)} + y^{(0)}$ and

$$\left\lfloor \frac{xy \bmod 2^n}{2^{n-m}} \right\rfloor = \left\lfloor \frac{xy}{2^{n-m}} \right\rfloor \bmod 2^m = z^{(1)}. \tag{1}$$

Note that the operation $x \bmod 2^t$ returns the t LSBs of x and the operation $\lfloor x/2^t \rfloor$ returns $x \gg t$, i.e. the binary representation of x right shifted t times. The number of unknown bits in the pair $(x^{(0)}, y^{(0)})$ is $2(n-m)$ and the m known bits on the right hand side of (1) imposes m restrictions on these unknowns. Hence, *on an average*, one should expect $2^{2(n-m)-m} = 2^{2n-3m}$ distinct pairs of $(x^{(0)}, y^{(0)})$ to be solutions to Mult. See Section 5 for an emperical justification of this statement.

We first consider the naive approaches to solve Mult. There are $(2n-2m)$ unknown bits and one approach is to try all possible combinations of these unknown bits. This approach requires 2^{2n-2m} time. The second naive approach using an offline table computation can be described as follows. For each possible pair of n-bit integers (x, y) compute the product $z = xy \bmod 2^n$. Store

in $\mathsf{Tab}[x^{(1)}, y^{(1)}, z^{(1)}]$ the set of all pairs $(x^{(0)}, y^{(0)})$ which are solutions to Mult for the instance $x^{(1)}, y^{(1)}, z^{(1)}$. This table takes 2^{2n} time to prepare and store. The preparation of the table can be done offline. Given a particular instance $x^{(1)}, y^{(1)}, z^{(1)}$ of Mult the solutions can be directly obtained from the entries of the row $\mathsf{Tab}[x^{(1)}, y^{(1)}, z^{(1)}]$. Since, on an average, there are 2^{2n-3m} solutions, at least this amount of online time will be required in producing the solutions.

Thus the online time will be at least 2^{2n-3m} (requiring 2^{2n} precomputation time and a table of size 2^{2n}) and at most 2^{2n-2m} (using exhaustive online search but without using any look-up table). We describe solutions to Mult whose online time complexity is between the two extreme values and which uses a table of moderate size. Thus our algorithms can be considered to be time-memory trade-off algorithms.

To improve readability, we will use the same notation for an integer and its binary representation. Also the length of a binary string will be denoted by $|.|$. Thus a binary string x of length $|x| = k$ denotes an integer $x \in \{0, \ldots, 2^k - 1\}$. For two binary strings x_1 and x_2, by (x_2, x_1) we denote the binary string x obtained by concatenating x_2 and x_1. Using the integer representation of x_1, x_2 and x we have $x = 2^{|x_1|}x_2 + x_1$.

Using this notation, we write $x = x^{(1)}2^{n-m} + x^{(0)}$, $y = y^{(1)}2^{n-m} + y^{(0)}$ and $z = z^{(1)}2^{n-m} + z^{(0)}$, where $|x| = |y| = |z| = n$, $|x^{(1)}| = |y^{(1)}| = |z^{(1)}| = m$ and $|x^{(0)}| = |y^{(0)}| = |z^{(0)}| = n - m$. We now introduce parameters n_0, n_1 and n_2 defined by the following equations.

$$\left.\begin{aligned} x &= X^{(2)}2^{n_1+n_0} + X^{(1)}2^{n_0} + X^{(0)} \\ y &= Y^{(1)}2^{n_1} + Y^{(0)} \\ z &= Z^{(1)}2^{n_1+n_0} + Z^{(0)} \end{aligned}\right\} \tag{2}$$

where $|X^{(2)}| = n_2$, $|X^{(1)}| = n_1$, $|X^{(0)}| = n_0$, $|Y^{(1)}| = n - n_1$, $|Y^{(0)}| = n_1$, $|Z^{(1)}| = n_2$ and $|Z^{(0)}| = n_1 + n_0$. We require these parameters to satisfy certain conditions. These conditions are given below.

1. $n_0 + n_1 + n_2 = n$: This is required since x, y and z are n-bit integers.
2. $n_0 \leq n - m$: This ensures that $X^{(0)}$ is a suffix of $x^{(0)}$.
3. $n_2 \leq m$: This ensures that $X^{(2)}$ is a prefix of $x^{(1)}$.
4. $n_1 \leq n - m$: This ensures that $Y^{(0)}$ is a suffix of $y^{(0)}$.
5. $n_1 \leq n_2$: This ensures that the expected number of entries in each row of $\mathsf{Tab}[\,]$ (see later) is one. The case $n_1 > n_2$ is also feasible but does not provide better results.
6. $n_2 + n_1 > m$: The case $n_2 + n_1 \leq m$ is also feasible, but does not provide better results and hence we do not consider it.

We now define binary strings $U^{(1)}, U^{(2)}$ and $V^{(1)}$ in the following manner. The strings $U^{(1)}$ and $U^{(2)}$ are such that $x^{(0)} = (U^{(1)}, X^{(0)})$ and $X^{(1)} = (U^{(2)}, U^{(1)})$, where $|U^{(1)}| = n - m - n_0$, $|U^{(2)}| = m - n_2$. Then $x^{(1)} = (X^{(2)}, U^{(2)})$. The string $V^{(1)}$ is such that $y^{(0)} = (V^{(1)}, Y^{(0)})$, where $|V^{(1)}| = n - m - n_1$. Then $Y^{(1)} = (y^{(1)}, V^{(1)})$. Note that the portion $U^{(2)}$ of $X^{(1)}$ is provided as part of the

input whereas the part $U^{(1)}$ of $X^{(1)}$ has to be determined. Also the string $V^{(1)}$ is part of $y^{(0)}$ and has to be determined. These substrings are shown in Figure 1.

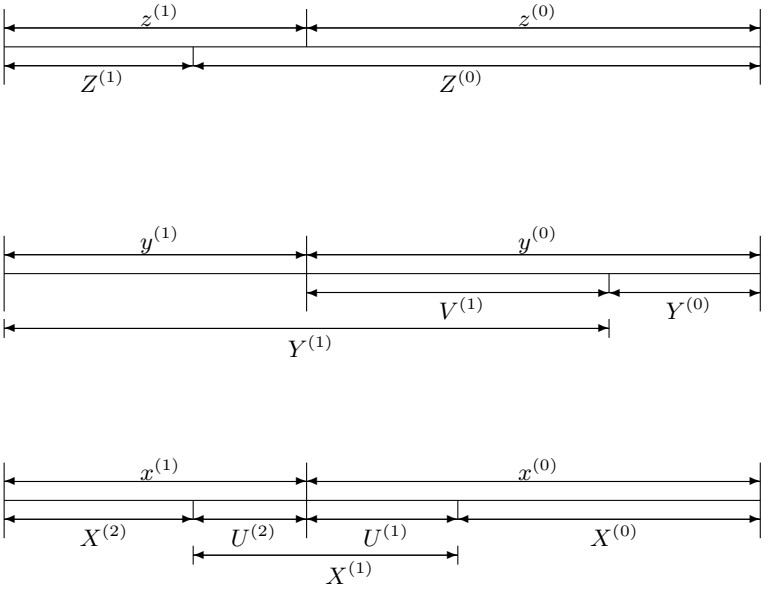


Fig. 1. Definitions of substrings

Our algorithm is based on the following result.

Proposition 1. $\left\lfloor \frac{xy}{2^{n-n_2}} \right\rfloor = r + d + c$, *where* $r = \left\lfloor \frac{Y^{(1)} x 2^{n_1}}{2^{n-n_2}} \right\rfloor$, $d = Y^{(0)}X^{(2)} + \left\lfloor \frac{Y^{(0)}X^{(1)}2^{n_0}}{2^{n-n_2}} \right\rfloor$ *and* $c \in \{0, 1, 2\}$.

Proof: We write

$$\left\lfloor \frac{xy}{2^{n-n_2}} \right\rfloor = \left\lfloor \frac{xY^{(1)}2^{n_1}}{2^{n-n_2}} + Y^{(0)}X^{(2)} + \frac{Y^{(0)}X^{(1)}2^{n_0}}{2^{n-n_2}} + \frac{Y^{(0)}X^{(0)}}{2^{n-n_2}} \right\rfloor.$$

Note that $r = \left\lfloor \frac{xY^{(1)}2^{n_1}}{2^{n-n_2}} \right\rfloor = \left\lfloor X^{(2)}Y^{(1)}2^{n_1} + X^{(1)}Y^{(1)} + \frac{X^{(0)}Y^{(1)}2^{n_1}}{2^{n_0+n_1}} \right\rfloor.$

Now $\left\lfloor \frac{X^{(0)}Y^{(1)}2^{n_1}}{2^{n_0+n_1}} + \frac{Y^{(0)}X^{(1)}2^{n_0}}{2^{n-n_2}} + \frac{Y^{(0)}X^{(0)}}{2^{n-n_2}} \right\rfloor$

$$= \left\lfloor \frac{X^{(0)}Y^{(1)}2^{n_1}}{2^{n_0+n_1}} \right\rfloor + \left\lfloor \frac{Y^{(0)}X^{(1)}2^{n_0}}{2^{n-n_2}} \right\rfloor + \left\lfloor \frac{Y^{(0)}X^{(0)}}{2^{n-n_2}} \right\rfloor + c$$

for some $c \in \{0, 1, 2\}$. Further, since $Y^{(0)} < 2^{n_1}$, $X^{(0)} < 2^{n_0}$, we have $X^{(0)}Y^{(0)} < 2^{n_0+n_1} = 2^{n-n_2}$ and hence $\left\lfloor Y^{(0)}X^{(0)}/2^{n-n_2} \right\rfloor = 0$. Putting all these together gives us the required result. □

Based on Proposition 1 we have the following algorithm to solve Mult. The algorithm uses a table Tab[ ] which is prepared in the first phase and is used to solve Mult in the second phase.

Algorithm 1
Input: $x^{(1)}, y^{(1)}$ and $z^{(1)}$.

1. Write $x^{(2)} = (X^{(2)}, U^{(2)})$, where $|X^{(2)}| = n_2$ and $|U^{(2)}| = m - n_2$;
2. set $Z^{(1)}$ to the n_2 most significant bits of $z^{(1)}$;
3. for $U^{(1)} \in \{0,1\}^{n-m-n_0}$
4. set $X^{(1)} = (U^{(2)}, U^{(1)})$;
5. for $Y^{(0)} \in \{0,1\}^{n_1}$
6. compute $d^{(1)} = Y^{(0)}X^{(2)} + \lfloor (Y^{(0)}X^{(1)}2^{n_0})/2^{n-n_2} \rfloor \bmod 2^{n_2}$;
7. $\mathsf{Tab}[d^{(1)}] = \mathsf{Tab}[d^{(1)}] \cup \{Y^{(0)}\}$;
8. end for;
9. for $(X^{(0)}, V^{(1)}) \in \{0,1\}^{n_0} \times \{0,1\}^{n-m-n_1}$
10. set $Y^{(1)} = (y^{(1)}, V^{(1)})$; set $x = (X^{(2)}, X^{(1)}, X^{(0)})$;
11. compute $r^{(1)} = \lfloor (xY^{(1)}2^{n_1})/2^{n-n_2} \rfloor \bmod 2^{n_2}$;
12. for $c \in \{0,1,2\}$
13. compute $d^{(2)} = Z^{(1)} - r^{(1)} - c \pmod{2^{n_2}}$;
14. for each $Y^{(0)} \in \mathsf{Tab}[d^{(2)}]$
15. set $y = (Y^{(1)}, Y^{(0)})$;
16. if $(\lfloor (xy)/2^{n-m} \rfloor \bmod 2^m = z^{(1)})$ then
17. set $x^{(0)} = (U^{(1)}, X^{(0)})$ and $y^{(0)} = (V^{(1)}, Y^{(0)})$;
18. output $(x^{(0)}, y^{(0)})$;
19. end if;
20. end for;
21. end for;
22. end for;
23. end for;

2.1 Complexity of Algorithm 1

The space complexity of Algorithm 1 is the space required to store Tab[ ]. By construction Tab[ ] has $2^{|d^{(1)}|} = 2^{n_2}$ rows and a total of $2^{|Y^{(0)}|} = 2^{n_1}$ entries in all the rows. By Condition 5 after Equation (2) we have $n_1 \leq n_2$ and hence on an average the number of entries in each row of Tab[ ] is at most one.

The time required by Algorithm 1 depends on the number of entries in a row of Tab[ ]. The expected number of such entries is one and this allows us to obtain the expected run-time R of Algorithm 1:

$$\left.\begin{aligned} R &= 2^{|U^{(1)}|}\left(2^{|Y^{(0)}|} + 3 \times 2^{|X^{(0)}|} \times 2^{|V^{(1)}|}\right) \\ &= 2^{n-m-n_0}\left(2^{n_1} + 3 \times 2^{n-m-n_1+n_0}\right) \\ &= 2^{n-m-n_0+n_1} + 3 \times 2^{2(n-m)-n_1} \end{aligned}\right\} \tag{3}$$

We now consider two cases and obtain the value of R in each case.

Case 1: $n_2 = m$. Hence $n_0 + n_1 = n - m$. In this case $R = 2^{2n_1} + 3 \times 2^{2(n-m)-n_1}$.

Subcase 1a: $n = 64$ and $m = 32$. Then $R = 2^{2n_1} + 3 \times 2^{64-n_1}$. This expression is minimized when $n_1 = 22$, whence $R = 2^{44} + 3 \times 2^{42} = 7 \times 2^{42}$.

Subcase 1b: $n = 64$ and $m = 16$. Then $R = 2^{2n_1} + 3 \times 2^{96-n_1}$. Since $n_1 \leq n_2 = m = 16$, the maximum value of n_1 is 16. Choosing $n_1 = 16$ gives $R = 2^{32} + 3 \times 2^{80}$.

Case 2: $n_2 < m$. This case is more complicated to analyse and we first perform a special case analysis by setting $n_2 = n_1$. Then $n_0 = n - 2n_1$ and $R = 2^{3n_1 - m} + 3 \times 2^{2(n-m)-n_1}$.

Subcase 2a: $n = 64$ and $m = 32$. Choosing $n_1 = 24$ we have $R = 2^{40} + 3 \times 2^{40} = 2^{42}$.

In general $n_2 \neq n_1$. However, we have verified that for $n = 64$ and $m = 32$ and for all possible distinct values of n_0, n_1 and n_2, the value of R is minimized for $n_2 = n_1 = 24$ and $n_0 = 16$. Thus the special case is also optimal for the general case. In fact, for $n = 64$ and $m = 32$, $R = 2^{42}$ is the minimum possible expected run-time for Algorithm 1.

2.2 Offline Table Preparation

The expected run-time of Algorithm 1 can be made optimal by using a larger table which can be prepared offline. We describe this idea for $n = 64$ and $m = 32$. Write $x = 2^{32}x^{(1)} + x^{(0)}$ and $y = 2^{32}y^{(1)} + y^{(0)}$. We write $\lfloor (xy)/2^{32} \rfloor = d + r$, where $d = \lfloor (xy^{(0)})/2^{32} \rfloor$ and $r = xy^{(1)}$.

In the offline table preparation phase, for each $(x^{(1)}, x^{(0)}, y^{(0)}) \in \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32}$, we compute $d = \lfloor (xy^{(0)})/2^{32} \rfloor \bmod 2^{32}$ and set $\mathsf{Tab}[x^{(1)}, x^{(0)}, d] = \mathsf{Tab}[x^{(1)}, x^{(0)}, d] \cup \{y^{(0)}\}$.

In the online phase, we are given $x^{(1)}, y^{(1)}$ and $z^{(1)}$. For each possible value of $x^{(0)} \in \{0,1\}^{32}$, we compute $r = xy^{(1)} \bmod 2^{32}$; $d = r - z^{(1)} \bmod 2^{32}$ and for each $y^{(0)} \in \mathsf{Tab}[x^{(1)}, x^{(0)}, d]$ output $(x^{(0)}, y^{(0)})$.

The run-time for table preparation is 2^{96}; the space required to store $\mathsf{Tab}[\,]$ is also 2^{96} and the (expected) runtime of the online phase is 2^{32}. Since there are 2^{32} solutions, the online run-time is the minimum possible. This comes at an expense of huge offline processing time and space.

3 Squaring

In the case $x = y$, the problem Mult reduces to squaring which can be formally stated as follows.

Problem: Sqr
Input: Two integers $x^{(1)}$ and $z^{(1)}$ such that, $0 \leq x^{(1)}, z^{(1)} < 2^m$.
Task: Find all integers $x^{(0)}$ such that, $0 \leq x^{(0)} < 2^{n-m}$, $x = 2^{n-m}x^{(1)} + x^{(0)}$ and

$$\left\lfloor \frac{x^2 \bmod 2^n}{2^{n-m}} \right\rfloor = \left\lfloor \frac{x^2}{2^{n-m}} \right\rfloor \bmod 2^m = z^{(1)}. \tag{4}$$

Note that there are $(n-m)$ unknown bits and m constraints. Hence the expected number of solutions is $\max(1, 2^{n-2m})$. If $n = 2m$, then the expected number of solutions is one. Algorithm 1 is not very efficient for Sqr so that we have to deal with the problem separately.

Let n_0, n_1 be such that $n_0 + n_1 = n - m$ and $x = 2^{n-m}X^{(2)} + 2^{n_0}X^{(1)} + X^{(0)}$, where $|X^{(2)}| = m$, $|X^{(1)}| = n_1$ and $|X^{(0)}| = n_0$ with $n_0 \leq m$.

Proposition 2. $\left\lfloor \frac{x^2}{2^{n_1+2n_0}} \right\rfloor = \left\lfloor \frac{z^{(1)}}{2^{n_0}} \right\rfloor = r + d + c$, *where* $r = 2^{n_1}\left(X^{(2)}\right)^2 + \left\lfloor \frac{\left(X^{(1)}\right)^2}{2^{n_1}} \right\rfloor + 2X^{(2)}X^{(1)}$, $d = \left\lfloor \frac{2X^{(2)}X^{(0)}}{2^{n_0}} \right\rfloor$ *and* $c \in \{0, 1, 2, 3\}$.

Based on Proposition 2, we have the following algorithm to solve Sqr.

Algorithm 2
Input: $x^{(1)}, z^{(1)}$.

1. set $k = n - (n_1 + 2n_0) = m - n_0$;
2. for $X^{(0)} \in \{0,1\}^{n_0}$
3. compute $d = \lfloor (2X^{(2)}X^{(0)})/2^{n_0} \rfloor \bmod 2^k$;
4. $\mathsf{Tab}[d] = \mathsf{Tab}[d] \cup \{X^{(0)}\}$;
5. end for;
6. for $X^{(1)} \in \{0,1\}^{n_1}$
7. compute $r = 2^{n_1}(X^{(2)})^2 + \lfloor (X^{(1)})^2/2^{n_1} \rfloor + 2X^{(2)}X^{(1)} \bmod 2^k$;
8. for each $c \in \{0, \ldots, 3\}$
9. compute $d = \lfloor z^{(1)}/2^{n_0} \rfloor - r - c \bmod 2^k$;
10. for each $X^{(0)} \in \mathsf{Tab}[d]$
11. if $(\lfloor x^2/2^{n-m} \rfloor \bmod 2^m = z^{(1)})$ then output $(X^{(1)}, X^{(0)})$;
12. end for;
13. end for;
14. end for;

The space complexity of Algorithm 2 is 2^{m-n_0} and the (expected) time complexity is $R = 2^{n_0} + 4 \times 2^{n_1}$.

Case 1: $n = 64$, $m = 32$. In this case we choose $n_0 = n_1 = 16$. Then the space complexity is 2^{16} and the time complexity is $R = 2^{16} + 4 \times 2^{16} = 5 \times 2^{16}$. This particular choice of n_0 and n_1 minimizes the value of R.

Case 2: $n = 64$, $m = 16$. Choosing $n_0 = 8$ and $n_1 = 40$ gives a run-time $R = 2^8 + 4 \times 2^{40}$.

4 Attacks on T-Functions

We consider two specific proposals of multiword T-functions from [3] and describe attacks on them. These T–functions operate on an internal state vector which consists of four 64-bit words. Applying a T–function once to the state

vector changes the value of each of the four 64-bit words. As suggested in [3], the extracted output consists of the most significant 32 bits of each of the four 64-bit words of the state vector. Thus applying the T–function repeatedly to the state vector produces a sequence of 128-bit (four 32-bit words) output blocks. These output blocks are treated as the generated pseudo-random sequence. The secret key consists of the initial 256-bit (four 64-bit words) value of the state vector.

For the attack we will assume that several consecutive output blocks are known. We actually require only two consecutive output blocks to perform the attack and a few more to verify the correctness. The goal of our attack is to obtain the complete 256-bit (four 64-bit words) value of the internal state vector at some point of time.

For a 64-bit word w, let $\mathsf{msb}(w)$ (resp. $\mathsf{lsb}(w)$) denote the 32 most (resp. least) significant bits of w. Let (x_0, x_1, x_2, x_3) be the internal state vector at some point of time. Let (y_0, y_1, y_2, y_3) be the state vector after application of T to (x_0, x_1, x_2, x_3), i.e., $(y_0, y_1, y_2, y_3) = T(x_0, x_1, x_2, x_3)$. The outputs corresponding to (x_0, x_1, x_2, x_3) and (y_0, y_1, y_2, y_3) are $(\mathsf{msb}(x_0), \mathsf{msb}(x_1), \mathsf{msb}(x_2), \mathsf{msb}(x_3))$ and $(\mathsf{msb}(y_0), \mathsf{msb}(y_1), \mathsf{msb}(y_2), \mathsf{msb}(y_3))$ respectively. We assume that these outputs are known and our attack is to compute $(\mathsf{lsb}(x_0), \mathsf{lsb}(x_1), \mathsf{lsb}(x_2), \mathsf{lsb}(x_3))$.

There are a total of 128 unknown bits in (x_0, x_1, x_2, x_3) and a method to obtain them in time less than 2^{128} constitutes an attack on the system. Our algorithms are much more efficient – the attacks in Section 4.1 and Section 4.2 require time 2^{40} and 2^{21} respectively to compute the 128 unknown bits.

4.1 Attack on Multiplication Based T–Function

Consider the following T–function:

$$T\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_0 \oplus s & \oplus (2(x_1 \vee C_1)x_2) \\ x_1 \oplus (s \wedge a_0) & \oplus (2x_2(x_3 \vee C_3)) \\ x_2 \oplus (s \wedge a_1) & \oplus (2(x_3 \vee C_3)x_0) \\ x_3 \oplus (s \wedge a_2) & \oplus (2x_0(x_1 \vee C_1)) \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \tag{5}$$

where $a_0 = x_0$, $a_i = a_{i-1} \wedge x_i$, $1 \leq i < 4$, $s = a_3 \oplus (a_3 + C_0)$. Also, C_0 is odd and known, $C_1 = (12481248)_{16}$ and $C_3 = (48124812)_{16}$ (Equation (13) in Klimov and Shamir [3]). Each of C_1 and C_3 are considered to be 64-bit words where the leading 32 bits are all zeros.

During use of this T-function as pseudo-random generator, the quantities $\mathsf{msb}(x_i), \mathsf{msb}(y_i)$ are known for $i = 0, 1, 2, 3$. Our attempt will be to obtain $\mathsf{lsb}(x_i)$ for $i = 0, 1, 2, 3$. This proceeds in several steps.

Step 1:
First note that $\mathsf{msb}(w_1 \oplus w_2) = \mathsf{msb}(w_1) \oplus \mathsf{msb}(w_2)$ and $\mathsf{msb}(w_1 \wedge w_2) = \mathsf{msb}(w_1) \wedge \mathsf{msb}(w_2)$. Hence we have $\mathsf{msb}(a_0) = \mathsf{msb}(x_0)$, $\mathsf{msb}(a_1) = \mathsf{msb}(x_0) \wedge \mathsf{msb}(x_1)$, $\mathsf{msb}(a_2) = \mathsf{msb}(x_0) \wedge \mathsf{msb}(x_1) \wedge \mathsf{msb}(x_2)$ and $\mathsf{msb}(a_3) = \mathsf{msb}(x_0) \wedge \mathsf{msb}(x_1) \wedge \mathsf{msb}(x_2) \wedge \mathsf{msb}(a_3)$. The quantity s involves an addition mod 2^{64} and cannot be directly tackled in this manner. However, we can determine the upper part of s with only one bit of uncertainty in the following manner. First note that we

have, $\mathsf{msb}(a_3 + C_0 \bmod 2^{64}) = \mathsf{msb}(a_3) + \mathsf{msb}(C_0) + \epsilon \bmod 2^{32}$ where ϵ is the carry of $\mathsf{lsb}(a_3) + \mathsf{lsb}(C_0)$ and hence $\epsilon = 0, 1$. Thus

$$\mathsf{msb}(s) = \mathsf{msb}(a_3) \oplus \left(\mathsf{msb}(a_3) + \mathsf{msb}(C_0) + \epsilon \bmod 2^{32}\right)$$

and hence $\mathsf{msb}(s)$ can take only two values as determined by ϵ.

Thus, with respect to the known 32 most significant bits, equation 5 reduces to

$$\begin{pmatrix} \mathsf{msb}\left(2(x_1 \vee C_1)x_2\right) \\ \mathsf{msb}\left(2x_2(x_3 \vee C_3)\right) \\ \mathsf{msb}\left(2(x_3 \vee C_3)x_0\right) \\ \mathsf{msb}\left(2x_0(x_1 \vee C_1)\right) \end{pmatrix} = \begin{pmatrix} \mathsf{msb}(y_0) \oplus \mathsf{msb}(x_0) \oplus \mathsf{msb}(s) \\ \mathsf{msb}(y_1) \oplus \mathsf{msb}(x_1) \oplus (\mathsf{msb}(s) \wedge \mathsf{msb}(a_0)) \\ \mathsf{msb}(y_2) \oplus \mathsf{msb}(x_2) \oplus (\mathsf{msb}(s) \wedge \mathsf{msb}(a_1)) \\ \mathsf{msb}(y_3) \oplus \mathsf{msb}(x_3) \oplus (\mathsf{msb}(s) \wedge \mathsf{msb}(a_2)) \end{pmatrix}. \quad (6)$$

Equation (6) gives a relation between known quantities. Let $W_0 = 2x_0$, $W_1 = (x_1 \vee C_1)$, $W_2 = 2x_2$ and $W_3 = (x_3 \vee C_3)$. Also let $K_0 = \mathsf{msb}(y_0) \oplus \mathsf{msb}(x_0) \oplus \mathsf{msb}(s)$, $K_1 = \mathsf{msb}(y_1) \oplus \mathsf{msb}(x_1) \oplus (\mathsf{msb}(s) \wedge \mathsf{msb}(a_0))$, $K_2 = \mathsf{msb}(y_2) \oplus \mathsf{msb}(x_2) \oplus (\mathsf{msb}(s) \wedge \mathsf{msb}(a_1))$ and $K_3 = \mathsf{msb}(y_3) \oplus \mathsf{msb}(x_3) \oplus (\mathsf{msb}(s) \wedge \mathsf{msb}(a_2))$. Our next step is to solve for W_0, W_1, W_2 and W_3 such that

$$\mathsf{msb}(W_1W_2) = K_0, \mathsf{msb}(W_2W_3) = K_1, \mathsf{msb}(W_3W_0) = K_2, \mathsf{msb}(W_0W_1) = K_3. \quad (7)$$

Since there are two choices of ϵ, the rest of the steps have to be carried out for each value of ϵ.

Step 2:
We use Algorithm 1 to solve (7). There are, however, a few adjustments, which improve the run-time of Algorithm 1. Note that due to masking with C_1 and C_3, eight bits of each of $\mathsf{lsb}(W_1)$ and $\mathsf{lsb}(W_3)$ are fixed and known. Also $W_0 = 2x_0$. We know $\mathsf{msb}(x_0)$ which means we do not know the last bit of $\mathsf{msb}(W_0)$ which is equal to the first bit of $\mathsf{lsb}(x_0)$. To apply Algorithm 1 we have to know all the 32 bits of $\mathsf{msb}(W_0)$. This means that we have to guess the last bit of $\mathsf{msb}(W_0)$. On the other hand, since $W_0 = 2x_0$, the last bit of W_0 (and hence of $\mathsf{lsb}(W_0)$) is zero. Similar considerations hold for W_2.

Now suppose we are solving for $\mathsf{lsb}(W_0)$ and $\mathsf{lsb}(W_1)$ from the equation $\mathsf{msb}(W_0W_1) = K_3$. While invoking Algorithm 1, we let W_1 play the role of x and W_0 play the role of y. Further, we choose $n_2 = n_1 = 24$ (Subcase 2a in Section 2.1). Then in Algorithm 1, $|U^{(1)}| = 16$, $|Y^{(0)}| = 24$, $|X^{(0)}| = 16$ and $|V^{(1)}| = 8$. As mentioned before, due to the masking of $W^{(1)}$ with C_1, four bits of each of $U^{(1)}$ and $X^{(0)}$ are fixed to be one. Hence the number of choices of $U^{(1)}$ in Step 3 and $X^{(0)}$ in Step 9 of Algorithm 1 both reduces to 2^{12} from 2^{16}. The last bit of $Y^{(0)}$ is zero and hence the number of choices of $Y^{(0)}$ in Step 5 of Algorithm 1 reduces to 2^{23} from 2^{24}. The length of $V^{(1)}$ is eight. However, we also need to guess the last bit of $\mathsf{msb}(W_0)$, which is the next bit after $V^{(1)}$. Thus the number of possible choices of $V^{(1)}$ in Step 9 of Algorithm 1 increases to 2^9 from 2^8.

In Step 18, Algorithm 1 produces $(\mathsf{lsb}(W_1), \mathsf{lsb}(W_0))$ as output. The modification described above also determines the last bit of $\mathsf{msb}(W_1)$. Suppose this bit

is b. By definition, the last bit of $\mathsf{lsb}(W_0)$ is zero. Then $\mathsf{lsb}(x_0)$ is obtained by prefixing b to $\mathsf{lsb}(W_0)$ and dropping the last bit. We assume that the modified Algorithm 1 produces $(\mathsf{lsb}(W_1), \mathsf{lsb}(x_0))$ as output. Similar considerations hold for the other equations in (7).

Recall from Equation (3) that the original expression for the expected runtime of Algorithm 1 is $R = 2^{|U^{(1)}|}\left(2^{|Y^{(0)}|} + 3 \times 2^{|X^{(0)}|} \times 2^{|V^{(1)}|}\right)$. Due to the changes in the number of possible choices of $U^{(1)}, Y^{(0)}, X^{(0)}$ and $V^{(1)}$, as explained above, this expression reduces to

$$\begin{aligned} R &= 2^{16-4}(2^{24-1} + 3 \times 2^{16-4} \times 2^9) \\ &= 2^{35} + 3 \times 2^{33} = 7 \times 2^{33} < 2^{36}. \end{aligned}$$

The time for solving one equation in (7) is approximately 2^{36} and hence the total time to solve all four equations is 2^{38}. The solutions to (7) are stored in separate lists, as we explain below. Define $w_i = \mathsf{lsb}(W_i)$ for $i = 1, 3$ and $w_i = \mathsf{lsb}(x_i)$ for $i = 0, 2$.

- Lst10 stores (w_1, w_0), sorted on w_1, such that $\mathsf{msb}(W_0W_1) = K_3$.
- Lst12 stores (w_1, w_2), sorted on w_1, such that $\mathsf{msb}(W_1W_2) = K_0$.
- Lst30 stores (w_3, w_0), sorted on w_3, such that $\mathsf{msb}(W_3W_0) = K_2$.
- Lst32 stores (w_3, w_2), sorted on w_3, such that $\mathsf{msb}(W_2W_3) = K_1$.

Step 3:
The next task is to "merge" the four lists to obtain solutions (w_0, w_1, w_2, w_3) which are consistent with all four equations. This is done as follows.

- Merge Lst10 and Lst12 on w_1 to obtain list Lst102 containing pairs of the form (w_0, w_2, w_1).
- Merge Lst30 and Lst32 on w_1 to obtain list Lst302 containing pairs of the form (w_0, w_2, w_3).
- Sort each of Lst102 and Lst302 on (w_0, w_2).
- Merge Lst102 and Lst302 on (w_0, w_2) to obtain a list Fin which contains tuples of the form (w_0, w_1, w_2, w_3) which are solutions to (7).

The time for merging and sorting (ignoring logarithmic factors) is 2^{32} and hence the above steps can be completed in approximately 2^{34} steps.

We consider the expected number of solutions to (7). There are 24 unknown bits in each of $\mathsf{lsb}(W_1)$ and $\mathsf{lsb}(W_3)$. On the other hand, there are 31 unknown bits in each of $\mathsf{lsb}(W_0)$ and $\mathsf{lsb}(W_2)$. In addition, we have to determine the last bits of both $\mathsf{msb}(W_0)$ and $\mathsf{msb}(W_2)$. Thus there are a total of 112 unknown bits in (7). Each of the equations in (7) provide 32 restrictions on these unknown bits. Hence there are a total of 128 restrictions on these 112 unknown bits. Thus, on an average, we can expect the solution to (7) to be unique. See Section 5 for an emperical justification of this statement.

Step 4:
The list Fin contains the possible solutions (w_0, w_1, w_2, w_3). Now $w_i = \mathsf{lsb}(W_i)$ for $i = 1, 3$ and we want $\mathsf{lsb}(x_i)$. As mentioned before, the masking of x_1 and x_3

by C_1 and C_3 respectively fixes 8 bits each of W_1 and W_3. Thus from $\mathsf{lsb}(W_1)$ and $\mathsf{lsb}(W_3)$ we do not obtain the values of these 16 bits of $\mathsf{lsb}(x_1)$ and $\mathsf{lsb}(x_3)$. Instead, for each possible solution (w_0, w_1, w_2, w_3) in Fin and each possible value of these 16 bits, we construct a possible solution $(\mathsf{lsb}(x_0), \mathsf{lsb}(x_1), \mathsf{lsb}(x_2), \mathsf{lsb}(x_3))$ and verify it using the definition of the T–function given in (5) and a few more outputs of the pseudo-random generator. The expected number of solutions $(\mathsf{lsb}(x_0), \mathsf{lsb}(x_1), \mathsf{lsb}(x_2), \mathsf{lsb}(x_3))$ is also one and the complexity of this step is 2^{16}.

This completes the description of the attack. By combining all the complexities, we see that the complexity of the attack is less than 2^{40} in determining the 128 unknown bits of $(\mathsf{lsb}(x_0), \mathsf{lsb}(x_1), \mathsf{lsb}(x_2), \mathsf{lsb}(x_3))$. This makes the attack quite practical and suggests that this T–function should not be used as a stand-alone pseudo-random generator.

4.2 Attack on Squaring Based T–Function

Consider the following T–function:

$$T\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_0 \oplus s \quad\quad\quad \oplus x_1^2 \wedge M \\ x_1 \oplus s \wedge a_0 \oplus x_2^2 \wedge M \\ x_2 \oplus s \wedge a_1 \oplus x_3^2 \wedge M \\ x_3 \oplus s \wedge a_2 \oplus x_0^2 \wedge M \end{pmatrix} \tag{8}$$

where, $a_0 = x_0$, $a_i = a_{i-1} \wedge x_i$, $1 \le i < 4$, $s = a_3 \oplus (a_3 + 1)$, and $M = 1 \ldots 1110_2$. This is Equation (10) in [3]. Since, $\mathsf{msb}(a_0) = \mathsf{msb}(x_0)$ and $\mathsf{msb}(a_i) = \mathsf{msb}(a_{i-1}) \wedge \mathsf{msb}(x_i)$, $1 \le i < 4$, we know $\mathsf{msb}(a_i)$ for $0 \le i < 4$. Now,

$$\begin{aligned} s &= \big(\mathsf{msb}(a_3) \times 2^{32} + \mathsf{lsb}(a_3)\big) \oplus \big(\mathsf{msb}(a_3) \times 2^{32} + \mathsf{lsb}(a_3) + 1\big) \\ &= 2^{32}\left\{\mathsf{msb}(a_3) \oplus (\mathsf{msb}(a_3) + \epsilon)\right\} + \left\{\mathsf{lsb}(a_3) \oplus \big(\mathsf{lsb}(a_3) + 1 \bmod 2^{32}\big)\right\} \end{aligned}$$

where ϵ is the carry of $\mathsf{lsb}(a_3)+1$. If $\epsilon = 1$, then $\mathsf{lsb}(a_3)$ equals $1 \ldots 111_2 = 2^{32} - 1$. But, $\mathsf{lsb}(a_3) = \mathsf{lsb}(x_0) \wedge \mathsf{lsb}(x_1) \wedge \mathsf{lsb}(x_2) \wedge \mathsf{lsb}(x_3)$ and hence $\mathsf{lsb}(x_i) = 1 \ldots 111_2$ for $0 \le i < 4$. In other words, $\epsilon = 1 \Rightarrow \mathsf{lsb}(x_i) = 1 \ldots 111_2$ for $0 \le i < 4$ and we can verify if this is indeed the case.

If this is not the case, then $\epsilon = 0$ and so, $\mathsf{msb}(s) = \mathsf{msb}(a_3) \oplus \mathsf{msb}(a_3) = 0 \ldots 000_2$. But then, $\mathsf{msb}(s \wedge a_i) = \mathsf{msb}(s) \wedge \mathsf{msb}(a_i) = 0 \ldots 000_2$. Also, the 32 most significant bits of M are all ones and hence, $\mathsf{msb}(x \wedge M) = \mathsf{msb}(x)$ for all x. Hence, with respect to the 32 most significant bits, Equation (8) reduces to

$$T\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_0 \oplus x_1^2 \\ x_1 \oplus x_2^2 \\ x_2 \oplus x_3^2 \\ x_3 \oplus x_0^2 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \tag{9}$$

with $\mathsf{msb}(x_i)$ and $\mathsf{msb}(y_i)$ known for $0 \le i < 4$. Let $z_i = \mathsf{msb}(x_{i-1}) \oplus \mathsf{msb}(y_{i-1})$, where the computation on the subscripts is done modulo 4. Then z_i, $0 \le i \le 3$ are known and we need to solve for (x_0, x_1, x_2, x_3) such that the following equation holds for $i = 0, 1, 2, 3$.

$$\mathsf{msb}\left(x_i^2 \bmod 2^{64}\right) = z_i \tag{10}$$

We use Algorithm 2 to solve these four equations. Solving each equation takes time $5 \times 2^{16} < 2^{19}$ (see Case 1 of Section 3) and hence the time to solve all four equations is less than 2^{21}. The possible solutions for $\mathsf{lsb}(x_0), \mathsf{lsb}(x_1), \mathsf{lsb}(x_2)$ and $\mathsf{lsb}(x_3)$ are kept in four lists L_0, L_1, L_2 and L_3 respectively. Since $n = 64 = 2 \times 32 = 2m$, the expected number of entries in each list is one. We then form a list Fin which contains tuples $(\mathsf{lsb}(x_0), \mathsf{lsb}(x_1), \mathsf{lsb}(x_2), \mathsf{lsb}(x_3))$ such that $\mathsf{lsb}(x_i)$ is in L_i. Then for each entry in Fin, we verify the solution by evolving the T–function in the forward direction a few times and comparing the output with the already available pseudo-random bits.

Thus, we get an algorithm to determine the 128 unknown bits in the input of Equation (8). It is easy to verify that the entire attack takes 2^{21} time. This shows that the T-function based on squaring is completely insecure as a stand-alone pseudo-random generator.

4.3 Extracting Lesser Bits

In this subsection, we use the notation $\mathsf{msb}_k(x)$ (resp. $\mathsf{lsb}_k(x)$) to denote the k most (resp. least) significant bits of x. The state vector for the T–function in (5) is (x_0, x_1, x_2, x_3). Suppose that instead of producing 128-bit output only the 64 bits $(\mathsf{msb}_{16}(x_0), \mathsf{msb}_{16}(x_1), \mathsf{msb}_{16}(x_2), \mathsf{msb}_{16}(x_3))$ are produced as output. Thus there are 192 unknown bits in (x_0, x_1, x_2, x_3) which have to be determined. We consider the effectiveness of our attack for this situation. The attack described in Section 4.1 goes through for this case.

The complexity of the total attack depends on the complexity of solving Equation 7. We use Subcase 1b of Section 2.1 along with the modification described in Step 2 of Section 4.1. Then the run-time of the modified Algorithm 1 becomes $2^{16-4}(2^{16-1} + 3 \times 2^{32-4} \times 2^{33}) = 2^{27} + 3 \times 2^{73}$. The number of unknown bits in (7) is $2 \times (48 - 8) + 2 \times 48 = 176$. The number of constraints in (7) is $4 \times 16 = 64$. Hence the expected number of solutions in Fin is 2^{112}. The correct solution can be determined by iterating the T–function and comparing the output with the available pseudo-random string. Thus the time taken to determine the 192 unknown bits will be 2^{112}. Though this is infeasible in practice, it still constitutes a theoretical attack on the system.

5 Implementation

We have performed some experiments to verify some of the assumptions about the average case behaviour. In this section, we briefly describe these results.

The first thing to consider is the expected number of solutions to Mult. As mentioned in Section 2, the expected number of solutions is 2^{2n-3m}. We describe some experimental results for $n = 16$ and $m = 8$. The expected number of solutions is $2^8 = 256$. The total number of possible instances $(x^{(1)}, y^{(1)}, z^{(1)})$ is 2^{24}. The number of instances such that the number of solutions is at most 256 is around 55% of the total number of instances while the number of instances such that the number of solutions is at most 512 is more than 99%. The

maximum number of solutions occurs for the case $(x^{(1)}, y^{(1)}, z^{(1)}) = (0, 0, 0)$ and such pathological situations are extremely rare.

We have implemented the attack on a reduced version of the multiplication based T–function described in Section 4.1. We have chosen the state vector to be four 32-bit words instead of four 64-bit words. Correspondingly, we have extracted the top 16 bits of each word. Also the constants C_1 and C_3 have been suitably scaled down. The attack has been implemented on randomly chosen instances and in each case the size of Fin was found to be one. This provided 2^8 choices for the state vector and the unique one could be found using only one more block of the available pseudo-random bit string. Thus the attack worked extremely well with only three consecutive blocks of output. We expect the attack to scale up quite well when applied to the T–function having state vector consisting of four 64-bit words.

For the problem on squaring, we have implemented Algorithm 2. In the case $n = 64$ and $m = 32$, the complexity of Algorithm 2 is 5×2^{16}. Our experiments confirm this theoretical result and hence the attack on the squaring based T–function in Section 4.2 works as expected.

6 Possible Countermeasure

Our attacks show that T–functions are probably not secure enough for stand-alone use especially when half of the bits of the state vector are produced as output. As suggested by Klimov and Shamir, T–functions can be used in conjunction with S-boxes for design of stream ciphers. We provide one suggestion for possibly improving the security of T–functions while retaining some of the nice theoretical properties.

There is a large and easily identifiable subclass $\mathcal{C}$ of T–functions such that any function in $\mathcal{C}$ defines a single cycle permutation on the state space. This is an attractive theoretical property. In our suggestion, we would like to preserve this property. To do this we apply the notion of conjugate permutations. (A similar idea has been used in the context of one-way permutations [5].) If π and τ are any two permutations of a set S, then $\sigma = \tau^{-1} \circ \pi \circ \tau$ has the same cycle structure as π; further, σ and π are called conjugate permutations.

We apply it to the context of T–functions in the following manner. Suppose π is the permutation on the set of all state vectors induced by a T–function from $\mathcal{C}$. Then π has a single cycle and any conjugate of π also has a single cycle. Note that this property does not depend on the choice of the permutation τ. Hence we can choose τ so as to improve the security of the overall mapping.

In our attack, the basic weakness that we exploit is that there is insufficient intermixing of higher and lower bits. One simple operation which can help in improving such intermixing is the circular shift (which is not a T–function). Thus we can construct a permutation τ on the state space by using circular shifts and other nonlinear operations. These operations can be arbitrarily chosen (in particular they need not be T–functions) to ensure higher security as long as τ is a permutation and that they are efficient to apply.

One penalty for introducing this countermeasure will be reduction in speed. The exact amount of speed reduction will depend on the concrete proposal. Developing such a concrete proposal based on our guideline is a future research problem.

7 Conclusion

In this paper, we studied multiplication, squaring and T–functions. In the first part of the paper, we presented a time-memory trade-off algorithm to solve the problems of multiplication and squaring. These algorithms are used in the second part of the paper to analyse two concrete proposal of multi-word T–functions from [3]. For the proposal based on multiplication, the 128 unknown bits of the state vector can be determined in 2^{40} time while for the proposal based on squaring, these bits can be determined in 2^{21} time. Experimental results from our implementation suggests that our attack works well in practice. Hence one can conclude that these two (and other similar) constructions of T–functions are not secure enough for stand-alone use. We also suggest the use of conjugate permutations for possibly improving the security of T–functions while maintaining some nice theoretical properties.

Notes: An anonymous reviewer of the paper has suggested that the problems Mult and Sqr can be formulated as closest vector problems in a two-dimensional lattice. Using this approach, the time complexity of Algorithm 1 will be 2^{32} with minimal storage space. At the time of preparing this final version, we have not been able to obtain the details of such an algorithm. We hope to present such details in a later communication.

References

1. A. Klimov and A. Shamir. A New Class of Invertible Mappings, *Proceedings of CHES 2002*, LNCS, 2002, pp 470–483.
2. A. Klimov and A. Shamir. Cryptographic Applications of T–functions, *Proceedings of SAC 2003*, LNCS.
3. A. Klimov and A. Shamir. New Cryptographic Primitives Based on Multiword T–functions, *Proceedings of FSE 2004*, LNCS, to appear.
4. A. Menezes, P. C. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.
5. M. Naor, O. Reingold. Constructing Pseudo-Random Permutations with a Prescribed Structure, *Journal of Cryptology*, 15(2): 97-102 (2002).

Cryptanalysis of Bluetooth Keystream Generator Two-Level E0

Yi Lu* and Serge Vaudenay

EPFL
http://lasecwww.epfl.ch

Abstract. In this paper, we carefully study both distinguishing and key-recovery attacks against Bluetooth two-level E0 given many short frames. Based on a flaw in the resynchronization of Bluetooth E0, we are able to fully exploit the largest bias of the finite state machine inside E0 for our attacks. Our key-recovery attack works with 2^{40} simple operations given the first 24 bits of 2^{35} frames. Compared with all existing attacks against two-level E0, this is the best one so far.

1 Background

The short-range wireless technology Bluetooth uses the keystream generator E0 to produce the keystream for encryption. After the earlier results [10, 9, 6] of correlation (also called bias) properties inside the Finite State Machine (FSM) towards the one-level E0, most recently, [12] systematically studied the biases and proved two previously known large biases to be the only largest up to 26 consecutive bits of the FSM output sequences. Attacks against E0 mostly focus on one-level E0 only and the best attacks [12, 1, 5] work on one impractically long frame of keystream without exception. Nevertheless, a few attacks [15, 11, 7–9] apply to two-level E0; compared with feasible attack complexities on one-level E0, attack complexities on two-level E0 are extremely high and make the practical Bluetooth E0 unbroken.

The main contribution of this paper is that first based on one of the two largest biases inside the FSM within one-level E0, we identify the bias at two-level E0 due to a resynchronization flaw in Bluetooth E0. Unlike the traditional approach to find the bias, the characterized bias *does not* involve the precomputation of the multiple polynomial with low weight. Second, to utilize the identified bias, we develop a novel attack to directly recover the original encryption key for two-level E0 without reconstructing the initial state of E0 at the second level. Our key-recovery attack works with 2^{40} simple operations given the first 24 bits of 2^{35} frames. Compared with all existing attacks [15, 11, 7–9] against two-level E0, this is the best so far.

* supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the Swiss National Science Foundation under the grant number 5005-67322.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 483–499, 2004.

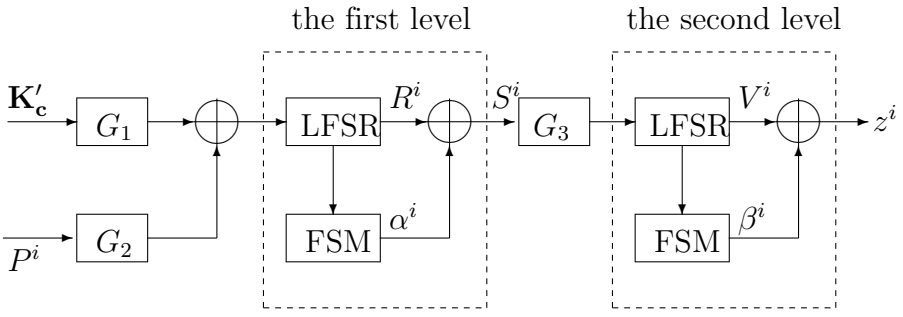


Fig. 1. Diagram of two-level E0 keystream generation

The rest of the paper is structured as follows. In Section 2 we review description of two-level E0. In Section 3 we study the attack against one-level E0. Then, we investigate the E0 resynchronization flaw, which allows to develop the basic attack of previous section into the distinguishing and key-recovery attacks against two-level E0 in Section 4; we further extend our key-recovery attack in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

2.1 The Core of Bluetooth E0

To briefly outline, the core of E0 (both dashed boxes in Fig. 1) can be viewed as a nonlinear filtering generator. The filtering generator consists of four LFSRs $(R_1, \ldots, R_4)$ which are equivalent to a single L-bit LFSR with connection polynomial[1] $p(x)$ and a 4-bit FSM, where $L = 128$. The keystream bit of the generator is obtained by xoring the output bit of the regularly-clocked LFSR with that of the FSM, which takes the current state of the LFSR as input and emits one bit (denoted by c_t^0 in Bluetooth specification) out of its 4-bit memory.

2.2 Review on Two-Level Bluetooth E0

Let $\mathbf{K_c}$ be the L-bit secret key computed by the key generation algorithm E3 [3, p783]. According to [3], the effective key $\mathcal{K}$ of length 8ℓ $(1 \le \ell \le 16)$ is computed by

$$\mathcal{K}(x) = \mathbf{K_c}(x) \mod g_1^{(\ell)}(x),$$

where the polynomial $g_1^{(\ell)}(x)$ is specified in [3, p770] and has degree 8ℓ. Bluetooth two-level E0 (depicted in Fig. 1) uses two L-bit inputs: one is the known nonce[2]

[1] Note that the connection polynomial of the equivalent single LFSR equals the product of those of the four LFSRs.

[2] By convention, hereafter we always use the superscript i to indicate the context of the i-th frame.

P^i, the other is the linearly expanded L-bit key $\mathbf{K}'_{\mathbf{c}}$ from $\mathcal{K}$ by $\mathbf{K}'_{\mathbf{c}}(x) = g_2^{(\ell)}(x) \cdot \mathcal{K}(x)$, where the polynomial $g_2^{(\ell)}(x)$ is also specified in [3, p770] and has degree no larger than $L - 8\ell$; or equivalently, we can rewrite it as

$$\mathbf{K}'_{\mathbf{c}} = E(\mathcal{K}), \tag{1}$$

where E is a linear mapping. After initialization of the equivalent LFSR for the first level E0, we can express its initial state[3] $R^i_{[-199,\ldots,L-200]} = (R^i_{-199}, \ldots, R^i_{L-200})$ as

$$R^i_{[-199,\ldots,L-200]} = G_1(\mathbf{K}'_{\mathbf{c}}) \oplus G_2(P^i), \tag{2}$$

for $i = 1, \ldots, m$, where G_1 and G_2 are affine transformations over $GF(2)^L$. Next comes the so-called two-level E0:

- During the first level, with the FSM initial state preset to zero, E0 runs L clocks producing 200-bit output $S^i_t = R^i_t \oplus \alpha^i_t$ and updating $R^i_{[t,\ldots,t+L-1]}$ by $R^i_{[t+1,\ldots,t+L]} = M(R^i_{[t,\ldots,t+L-1]})$ for $t = -199, \ldots, 0$, where M is the linear mapping over $GF(2)^L$ that corresponds to the companion matrix associated with $p(x)$. Note that first, α^i_t is the output bit of the FSM fed with $R^i_{[t,\ldots,t+L-1]}$; second, the last L-bit output at the first level E0 is $S^i_{[1-L,\ldots,0]}$; last, $R^i_{[1-L,\ldots,0]} = (M^{72} \circ G_1)(\mathbf{K}'_{\mathbf{c}}) \oplus (M^{72} \circ G_2)(P^i)$.
- At the beginning of the second level, the equivalent single LFSR is initialized by $V^i_{[1,\ldots,L]} = G_3(S^i_{[1-L,\ldots,0]})$, where $G_3 : GF(2)^{128} \to GF(2)^{128}$ is another affine transformation (see [3, p772]); the FSM initial state at the second level remains the same as the one in the end of the first level. Note that the present time is $t = 1$.
- During the second level, for $t = 1, \ldots, 2745$, E0 produces the keystream $z^i_t = V^i_t \oplus \beta^i_t$ for encryption of the i-th frame and updates $V^i_{[t,\ldots,t+L-1]}$ by $V^i_{[t+1,\ldots,t+L]}$.

2.3 An Important Note on G_3

We observe that G_3 is implemented in such a simple way[4] that the last L-bit output sequence of the first level E0 is *byte-wise* reloaded into the four component LFSRs in parallel at the second level E0 with only a few exceptions, which turns out to be a flaw as introduced later in Section 4. For completeness, Table 1 lists in time order the first 24 output bits of $R_1, \ldots, R_4$ individually at the beginning of E0 level two, in terms of the L-bit input $v_0, \ldots, v_{L-1}$.

2.4 The Bias Inside the FSM

Our starting point would be the bias inside the FSM, which was discovered by [9, 6] and further proved in [12] to be the the largest bias up to 26-bit output

[3] Throughout the rest of the paper, we use the unified notation $\Omega^i_{[a,\ldots,b]}$ with the formatted subscript to denote the vector $(\Omega^i_a, \ldots, \Omega^i_b)$.

[4] It is believed to help increase the rate of keystream generation.

Table 1. The first 24 output bits of LFSRs at E0 level two

LFSR	output bits
R_1	$v_{71} \cdots v_{64}$, $v_{39} \cdots v_{32}$, $v_7 \cdots v_0$
R_2	$v_{79} \cdots v_{72}$, $v_{47} \cdots v_{40}$, $v_{15} \cdots v_8$
R_3	$v_{111} \cdots v_{104}$, $v_{87} \cdots v_{80}$, $v_{55} \cdots v_{48}$
R_4	$v_{119} \cdots v_{112}$, $v_{95} \cdots v_{88}$, $v_{63} \cdots v_{56}$

sequence of the FSM involving the smallest number of consecutive bits. Let $\lambda = \frac{25}{256}$, we have

$$\Pr(c_t^0 \oplus c_{t+1}^0 \oplus c_{t+2}^0 \oplus c_{t+3}^0 \oplus c_{t+4}^0 = 1) = \frac{1}{2} + \frac{\lambda}{2},$$

for any integer t, assuming that the $L + 4 = 132$-bit initial state of E0 is random and uniformly distributed. Hereafter, we analyze as exactly described in Bluetooth specification [3]. For convenience, we denote c_t^0 used for the first and second level keystream generation by α_t^i, β_t^i respectively. Therefore $\{\alpha_t^i\}, \{\beta_t^i\}$ being separated sequences of $\{c_t^0\}$ both satisfy the same statistical property:

$$\Pr(\alpha_t^i \oplus \alpha_{t+1}^i \oplus \alpha_{t+2}^i \oplus \alpha_{t+3}^i \oplus \alpha_{t+4}^i = 1) = \frac{1}{2} + \frac{\lambda}{2}, \tag{3}$$

$$\Pr(\beta_t^i \oplus \beta_{t+1}^i \oplus \beta_{t+2}^i \oplus \beta_{t+3}^i \oplus \beta_{t+4}^i = 1) = \frac{1}{2} + \frac{\lambda}{2}, \tag{4}$$

for any t and any i.

3 Security Analysis on E0 Level One

The goal of the attacker in this section is to recover the effective 8ℓ-bit encryption key $\mathcal{K}$ with knowledge of m L-bit output sequences $S^i_{[1-L,\ldots,0]}$ of the first level E0 for $i = 1, \ldots, m$ and the corresponding m nonces $P^1, \ldots, P^m$.

3.1 Finding the Closest Sequences with Fixed Differences

We begin with a very simple problem: given $2m$ L-bit sequences $s^1, \ldots, s^m$ and $\delta^1, \ldots, \delta^m$, where $\delta^1 = \mathbf{0}$ and $\delta^i \neq \delta^j$ for all $i \neq j$, find the L-bit sequence r^1 that maximizes $N(r^1) = \sum_{i=1}^m \sum_{t=1}^L (s_t^i \oplus r_t^i)$ where $r_t^i = r_t^1 \oplus \delta_t^i$ for $i = 1, \ldots, m$ and $t = 1, \ldots, L$.

Similar to the well-known approach (see [9, p251]), the solution based on the idea of minority vote goes fairly easy. We have

$$N(r^1) = \sum_{t=1}^{L} \sum_{i=1}^{m} (s_t^i \oplus r_t^1 \oplus \delta_t^i).$$

Thus, in order to maximize $N(r^1)$, we must have

$$r_t^1 = \text{minority}\{s_t^i \oplus \delta_t^i : i = 1, \ldots, m\}$$

for all $t = 1, \ldots, L$. Note that in case of a tie for r_t^1, we have two answers for this t-th bit regardless of all the other bits. We finally obtain all the answers that achieve the same maximal $N(r^1)$. The time and memory complexities of the above algorithm both equal the data complexity $O(mL)$.

3.2 Attack Against E0 Level One

Let $\Delta^i_{[1-L,\ldots,0]} = R^1_{[1-L,\ldots,0]} \oplus R^i_{[1-L,\ldots,0]}$ for $i = 1, \ldots, m$. By Eq.(2) we have $\Delta^i_{[1-L,\ldots,0]} = (M^{72} \circ G_2)(P^1 \oplus P^i)$. We further set

$$r_t^i = \bigoplus_{j=0}^{4} R_{t+j}^i,$$

$$\delta_t^i = \bigoplus_{j=0}^{4} \Delta_{t+j}^i,$$

$$s_t^i = \bigoplus_{j=0}^{4} S_{t+j}^i,$$

for $i = 1, \ldots, m$ and $t = 1-L, \ldots, -4$. Note that $s_t^i \oplus r_t^i = \bigoplus_{j=0}^{4} \alpha_{t+j}^i$ follows the biased distribution by Eq.(3). As long as $\sum_{i=1}^{m} \sum_{t=1-L}^{-4} (s_t^i \oplus r_t^i)$ is the maximal and Δ_t^i, S_t^i are known, we can apply the preceding algorithm to recover $(L-4)$ bits of r^1 followed by an exhaustive search on the remaining 4 bits, next solve R^1, then $\mathbf{K'_c}$ by Eq.(2), and finally deduce $\mathcal{K}$ from $\mathbf{K'_c}$ by Eq.(1). The time/memory/data complexities all equal $O(mL + 2^4 L)$, i.e. $O((m+16)L)$. No precomputation is needed.

About the minimal m to guarantee the valid precondition $\sum_{i=1}^{m} \sum_{t=1-L}^{-4} (s_t^i \oplus r_t^i)$ is the maximal, we use the result in [12, Eq.(10)] based on [2] that says regardless of the value of L and ℓ, we need the minimum

$$m \approx \frac{4 \log 2}{\lambda^2} \text{ (frames)}. \tag{5}$$

Consequently, we require $m = 512$ to recover $\mathcal{K}$ from S^i and P^i for $i = 1, \ldots, m$. This results in the time/data/memory complexities all the same as $O(2^{16})$. To verify this, we ran experiments on 512 frames of the randomly-chosen 132-bit E0 initial state 2^{25} times. It turned out that we had 1.5 errors and 0.4 tie in average, which means we can easily correct all errors by an extra checking step in the end in negligible time. Finally, Table 2 compares our result with the only known[5] four attacks [7, 8, 11, 15] working on frames of L-bit consecutive keystreams. Note that existing attacks [7, 8, 11, 15] directly apply to two-level E0

[5] In the similar approached paper [9], m is chosen as 45 for E0 level one without the complexity estimate, because the authors focused on the two-level E0 and traded m with the time complexity of E0 level one, whose time complexity is negligible with that of the E0 level two.

as well with the level-by-level key-recovery scheme[6]; in contrast, our attack is completely disabled against two-level E0 with this scheme as the attack is based on a naive assumption that we directly observe the output of E0 level one[7]. In the next section, we introduce a resynchronization flaw in Bluetooth E0 that leads to a shortcut extended attack against the two-level E0.

Table 2. Comparison of our attack with existing attacks against E0 level one given frames of L bits

Attack Type	Precomputation	Time	Frames	Data	Memory
Divide & Conquer [15]	-	2^{93}	1	2^7	-
BDD [11]	-	2^{77}	1	2^7	-
Algebraic Attack [7]	-	2^{51}	2	2^8	2^{51}
Algebraic Attack [8]	-	$2^{23.4}$	3	$2^{8.6}$	$2^{23.4}$
Our Correlation Attack	-	2^{16}	2^9	2^{16}	2^{16}

4 Security Analysis on Two-Level E0

4.1 The Resynchronization Flaw in Bluetooth Two-Level E0

Define

$$U^i = (U_1^i, \ldots, U_L^i) = G_3(R^i_{[1-L,\ldots,0]}). \tag{6}$$

Following the description of G_3 in Subsection 2.3, we can easily verify that

$$\begin{aligned}
V_t^i &= U_t^i \oplus \alpha^i_{-56-t} \oplus \alpha^i_{-48-t} \oplus \alpha^i_{-16-t} \oplus \alpha^i_{-8-t}, && \text{for } t = 1, \ldots, 8,\\
V_t^i &= U_t^i \oplus \alpha^i_{-80-t} \oplus \alpha^i_{-72-t} \oplus \alpha^i_{-32-t} \oplus \alpha^i_{-24-t}, && \text{for } t = 9, \ldots, 16,\\
V_t^i &= U_t^i \oplus \alpha^i_{-104-t} \oplus \alpha^i_{-96-t} \oplus \alpha^i_{-56-t} \oplus \alpha^i_{-48-t}, && \text{for } t = 17, \ldots, 24.
\end{aligned}$$

From the above equations, we summarize the characteristics of V_t^i by

$$V_t^i = U_t^i \oplus \alpha^i_{a_t} \oplus \alpha^i_{a_t+8} \oplus \alpha^i_{b_t} \oplus \alpha^i_{b_t+8}, \tag{7}$$

for $t = 1, \ldots, 24$, where $a_t = -t + \text{const}_{\lfloor \frac{t-1}{8} \rfloor}$ and $b_t = -t + \text{const}'_{\lfloor \frac{t-1}{8} \rfloor}$. Note that Eq.(7) is our crucial observation about Bluetooth E0 resynchronization flaw which enables a shortcut attack throughout the two levels of E0. Now, we express the output bit z_t^i of the second level E0 keystream by

$$z_t^i = U_t^i \oplus \alpha^i_{a_t} \oplus \alpha^i_{a_t+8} \oplus \alpha^i_{b_t} \oplus \alpha^i_{b_t+8} \oplus \beta^i_t, \tag{8}$$

[6] namely, the initial state at the first level is reconstructed after the initial state at the second level is recovered.

[7] As a matter of fact, according to [3, p763], Bluetooth takes the correlation properties into account and adopts the two-level scheme of keystream generation in practice on purpose.

for $t = 1, \ldots, 24$. Let $u_t^i = \bigoplus_{j=0}^{4} U_{t+j}^i$ and $s_t^i = \bigoplus_{j=0}^{4} z_{t+j}^i$. From Eq.(8), we have that

$$s_t^i \oplus u_t^i = \bigoplus_{j=0}^{4} \alpha_{a_t-j}^i \oplus \bigoplus_{j=0}^{4} \alpha_{a_t-j+8}^i \oplus \bigoplus_{j=0}^{4} \alpha_{b_t-j}^i \oplus \bigoplus_{j=0}^{4} \alpha_{b_t-j+8}^i \oplus \bigoplus_{j=0}^{4} \beta_{t+j}^i \quad (9)$$

for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$. Therefore, we deduce an important correlation concerning the practical implementation of E0 from Eq.(3,4,9):

Theorem 1. *Assuming independence of α_t^i's and β_t^i's, we have*

$$\Pr(s_t^i \oplus u_t^i = 1) = \frac{1}{2} + \frac{\lambda^5}{2}$$

for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$.

4.2 A Near-Practical Distinguishing Attack Against Two-Level E0

Using the standard technique of linear cryptanalysis, we expect that $s_t^i \oplus u_t^i$ equals one most of the time for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$ with a total of $\lambda^{-10} \approx 2^{34}$ samples. Since the difference

$$U^i \oplus U^j = G_3(R^i_{[1-L,\ldots,0]}) \oplus G_3(R^j_{[1-L,\ldots,0]}) = (G_3 \circ M^{72} \circ G_2)(P^i \oplus P^j),$$

is known for all i and j by Eq.(2,6), we apply the algorithm in Subsection 3.2 to recover the bit u_1^1 separately with two sets of 2^{34} frames sharing only one common frame denoted as the first frame for both sets. If we get a unique solution, we conclude the keystreams are generated by E0; otherwise, we accept them as truly random sources. The time/data complexities are $O(2 \times 2^{34} \times 5)$, i.e. $O(2^{37})$. In contrast to the conventional treatment based on finding a multiple polynomial with low weight, no precomputation is needed in our scenario. So far, this is the only known near-practical attack against the full two-level E0. We can further improve the distinguisher by recovering u_t^1 for $t = 1, \ldots, 4$ with two different sets of frames of the first 8 bits. Comparing two sets of solutions for the four bits, if we get a majority of identical bits, then we conclude the keystreams are generated by E0, otherwise we accept them as truly random sources. The number of frames we need is $2 \times 2^{34}/4 = 2^{33}$. This results in time/data complexities $O(2^{33} \times 8)$, i.e. $O(2^{36})$.

4.3 The Key-Recovery Attack Against Two-Level E0

From last subsection and Theorem 1, we know that with 2^{34} frames of keystreams, we can recover twelve bits, i.e. u_t^1 for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$. After that, we try exhaustively for the remaining $|\mathcal{K}| - 12$ bits assuming linear independency of the twelve bits[8]. Note that we have

$$U^i = (G_3 \circ M^{72} \circ G_1)(\mathbf{K}'_{\mathbf{c}}) \oplus (G_3 \circ M^{72} \circ G_2)(P^i), \quad (10)$$

[8] We tested and found that the twelve bits are linearly independent for all choices of effective keylength $|\mathcal{K}| = 8\ell$ except for $|\mathcal{K}| = 8$ in which case our attack is worse than the brute force attack and becomes meaningless anyway.

by Eq.(6,2). So, we deduce from Eq.(10,1) that

$$U^i = (G_3 \circ M^{72} \circ G_1 \circ E)(\mathcal{K}) \oplus (G_3 \circ M^{72} \circ G_2)(P^i),$$

which means U^i is an affine transformation of $\mathcal{K}$ given P^i and so is u^i. Thus, we can ultimately solve the effective key $\mathcal{K}$ from u^i. The total time complexity of our attack is computed as

$$2^{34} + 2^{|\mathcal{K}|-13} = \begin{cases} 2^{34}, & |\mathcal{K}| < 48 \\ 2^{|\mathcal{K}|-13}, & |\mathcal{K}| \geq 48 \end{cases}$$

The data complexity of our attack is $(2^{34} - 1) \cdot 24 + 128$, i.e. $O(2^{38.6})$, as we need $2^{34} - 1$ frames of the first 24 bits plus one frame of 128 bits. Table 3 compares our attack with existing attacks [15, 11, 7, 8, 9] against the two-level Bluetooth E0. Note that the number of required frames completely depends on the frame size in [7] to meet the requirement of data amount. This is, to our best knowledge, the first non-trivial[9] attack against practical E0 with various key length. Notice that when $40 \leq |\mathcal{K}| \leq 80$, our attack offers the best performance over the others.

Table 3. Comparison of our attack with existing attacks against two-level Bluetooth E0

Attack	Precomputation	Time	Frames	Data	Memory
exhaustive search	-	$2^{|\mathcal{K}|-1}$	1	$|\mathcal{K}|$	-
[15]	-	2^{93}	1	2^7	-
[11]	-	2^{113}	1	2^7	-
[7]	-	2^{73}	-	2^{43}	2^{51}
[8]	2^{80}	2^{65}	2	$2^{12.4}$	2^{80}
[9]	2^{80}	2^{70}	45	2^{17}	2^{80}
Our Attack	-	$2^{|\mathcal{K}|-13} + 2^{34}$	2^{34}	$2^{38.6}$	2^{34}

Remark 2. Note that our attack is based on one of the largest two (linearly dependent) biases which is introduced in Eq.(3). As time/data tradeoff, we might also expect to have some other linearly independent biases to be large enough so that the time is decreased at somewhat reasonably increasing cost of data/memory complexities. Nonetheless, using the computation formula of [12], we find none such bias that leads to the data complexity of less than 2^{50}.

Remark 3. As the nonces P^i's are affine transformation of a 26-bit clock and a master device address, our attack requiring much more than 2^{26} frames of keystreams still remains impractical unfortunately.

[9] in contrast to the brute force attack.

5 Extended Key-Recovery Attack Against Two-Level E0

5.1 A Partial Key-Recovery Attack

Notice that on one hand, each of the four leftmost biased bits on the right-hand side of Eq.(9) is computed only with a certain subset of fixed key bits, the known nonce and the unknown variable FSM initial state; on the other hand, the value of Eq.(9) can be easily predetermined from the left-hand side, after we recover u_t^i's with 2^{34} frames by the distinguisher in Subsection 4.2. Consequently, the well-known technique of *statistical cryptanalysis* leads us to the following approach to advance our key-recovery attack: supposing we manage to guess one of those four biased bits for all frames by guessing only the related key bits, then, for each frame, we XOR the guess on the biased bit with the predetermined value of Eq.(9) to obtain one bit. Thanks to Eq.(9), this bit shows bias for the right guess and almost balancedness for the wrong guess (which is also called statistical distinguishable); we're able to spot out the right guess of all guesses finally.

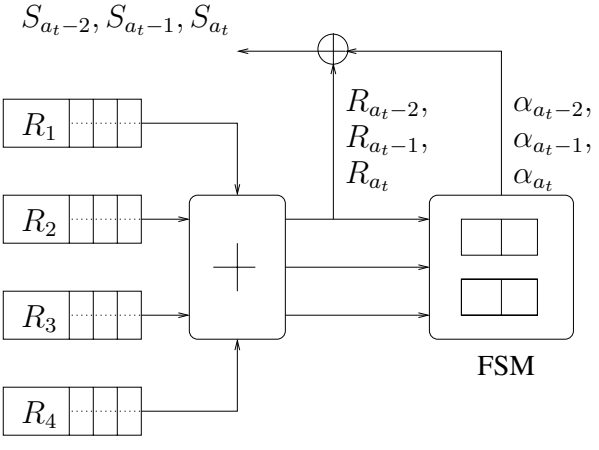


Fig. 2. Computation diagram of $S_{a_t-2}, S_{a_t-1}, S_{a_t}$

More specifically, we observe two important points about E0 FSM state: first, the FSM state at time t always contains the two bits c_t^0, c_{t-1}^0; second, the 4-bit FSM state is updated by its current state together with four current output bits of LFSRs. Therefore, for fixed $t \in \{8k+1, \ldots, 8k+4\}$ and $k \in \{0, 1, 2\}$, the bit[10] $\bigoplus_{j=0}^{4} \alpha_{a_t-j}^i$ computed from 5 consecutive bits $\alpha_{a_t}^i, \ldots, \alpha_{a_t-4}^i$, is derived from the same subset of $4 \times 3 = 12$ bits of the shared key $\mathcal{K}$ in all frames given P^i together with the unknown frame-dependent FSM state at time $a_t - 3$ (see

[10] For our convenience, we discuss the first biased bit on the right-hand side of Eq.(9) from now on; however, due to symmetry of the subscripts on the right-hand side of Eq.(9), our discussion also applies to the other three biased bits but the last.

Fig. 2). We can compute all the possible sequences[11] $\alpha'_{a_t}, \ldots, \alpha'_{a_t-2}$ in according to every possible FSM state for each frame i with $t \in \{8k+1, \ldots, 8k+4\}$ and $k \in \{0,1,2\}$. Within one frame, of all the choices of 12-bit K' of $\mathcal{K}$ and the FSM state, the sequence computed with the right shared 12-bit K and the right FSM state yields $\bigoplus_{j=0}^{4} \alpha^i_{a_t-j}$, which equals $\bigoplus_{j=0}^{4}(\alpha^i_{a_t-j+8} \oplus \alpha^i_{b_t-j} \oplus \alpha^i_{b_t-j+8} \oplus \beta^i_{t+j})$ with bias λ^4 when xoring with the computable bit $s^i_t \oplus u^i_t$ by Eq.(9); meanwhile, the sequence obtained with the wrong FSM state and/or the wrong shared 12-bit K' is expected to produce a new biased bit $\bigoplus_{j=0}^{4} \alpha'_{a_t-j}$ (with bias λ) which when xoring with $s^i_t \oplus u^i_t$ finally generates a bit with much smaller bias λ^6 that could be approximated by a randomly and uniformly distributed bit. Therefore, we estimate that for every frame, the 12-bit guess K' would yield 2^4 randomly and uniformly distributed bits, except for the correct guess that produces $2^4-1=15$ randomly and uniformly distributed bits as well as one biased bit (with bias λ^4).

Alternatively, for every frame i, we can guess $4(2+\tau)$ bits K' of $\mathcal{K}$ together with the FSM state at time $a_t - \tau - 2$ to compute consecutively τ bits $\bigoplus_{j=0}^{4} \alpha'_{a_t-j}, \ldots, \bigoplus_{j=0}^{4} \alpha'_{a_t-j-\tau+1}$ with $\tau \le 5 - (t \mod 8)$ for fixed $t \in \{8k+1, \ldots, 8k+4\}$ and $k \in \{0,1,2\}$. Denote the parameter m as the required number of frames to be discussed later. For the same reason as before, when we xor the τ bits with $s^i_t \oplus u^i_t, \ldots, s^i_{t+\tau-1} \oplus u^i_{t+\tau-1}$, we expect the $16m$ sequences to comply with a truly random distribution $\mathcal{D}_0$ of τ-bit vectors for all wrong guesses K', and the $16m$ sequences for the right guess K to comply with the biased distribution $\mathcal{D}_1$ of τ-bit vectors approximated by

$$\mathcal{D}_1 \approx \frac{\mathcal{D}' + 15\mathcal{D}_0}{16}, \tag{11}$$

where $\mathcal{D}' \stackrel{\text{def}}{=} \mathcal{D}^{\otimes 4}$ with $\otimes$ representing the regular convolutional product (see [12]), and $\mathcal{D}$ is the distribution of $\bigoplus_{j=0}^{4} c^0_{t-j}, \ldots, \bigoplus_{j=0}^{4} c^0_{t-\tau-j+1}$. Note that Eq.(11) means all the biases in $\mathcal{D}'$ dwindles 16 times in $\mathcal{D}_1$, i.e. we have the following relation between the two Walsh coefficients $\hat{\mathcal{D}}_1(x), \hat{\mathcal{D}}'(x)$ of any nonzero τ-bit vector x:

$$\hat{\mathcal{D}}_1(x) = \frac{1}{16}\hat{\mathcal{D}}'(x). \tag{12}$$

Let $f : GF(2)^\tau \to \mathbf{R}$ be a weighted grade for those resultant sequences $\chi^1_{K'}, \ldots, \chi^{16m}_{K'}$ from the guess K'. We accordingly grade each guess K' by

$$G_{K'} = \sum_{j=1}^{16m} f(\chi^j_{K'}). \tag{13}$$

Using analysis of [16] and [2] (see Appendix for complete treatment), we show that with minimal

$$m \approx \frac{\tau+2}{2\tau-1} \cdot 2^{34.5},$$

[11] We omit the superscript i and use α'_t to denote the candidate for α^i_t.

the score G_K of the right guess tops the chart by choosing $f(x) = \mathcal{D}_1(x) - \frac{1}{16}$ for all $x \in GF(2)^\tau$. Note that with $f(x) = \mathcal{D}'(x)$ we obtain equivalently the same resultant score $G_{K'}$ for all K'. Also, recall that to predetermine $u_t^i, \ldots, u_{t+\tau-1}^i$ we need 2^{34} frames for the distinguisher. Thus we must have

$$m \approx \max\left(2^{34}, \frac{\tau+2}{2\tau-1} \cdot 2^{34.5}\right). \tag{14}$$

Algorithm 1 details the above partial key-recovery attack for $4(2+\tau)$ bits.

Algorithm 1 The extended attack against two-level E0 to recover $4(2+\tau)$ bits

Parameters:
1: $\mathcal{D}_1$ by Eq.(11)
2: $\tau \in [1, 4]$
3: m by Eq.(14)

Input:
4: keystreams $z_1^i \cdots z_{24}^i$ generated by the same $\mathcal{K}$ together with P^i under two-level E0 for $i = 1, \ldots, m$

Processing:
5: choose $k \in \{0, 1, 2\}$ and $t \in \{8k+1, \ldots, 8k+4\}$ with $\tau \le 5 - (t \mod 8)$
6: **for** $4(2+\tau)$ bits K' of $\mathcal{K}$ that are used to compute $\alpha'_{a_t-\tau-1}, \ldots, \alpha'_{a_t}$ and are consistent with previously recovered bits **do**
7: initialize $G_{K'}$ to zero
8: **for** $i = 1, \ldots, m$ **do**
9: initialize counters $\mu_0, \mu_1, \ldots, \mu_{2^\tau-1}$ to zero
10: **for all** 4-bit FSM state at time $a_t - \tau - 2$ **do**
11: compute $\alpha'_{a_t-\tau-1}, \ldots, \alpha'_{a_t}$
12: compute $b = b_0, \ldots, b_{\tau-1}$ with $b_n = u_{t+n}^i \oplus s_{t+n}^i \oplus \bigoplus_{j=0}^4 \alpha'_{a_t-j-n}$
13: increment μ_b
14: **end for**
15: increment $G_{K'}$ by $\sum_b \mu_b(\mathcal{D}_1(b) - \frac{1}{16})$
16: **end for**
17: add the pair $(G_{K'}, K')$ into the list
18: **end for**
19: find the largest G_K in the list and output K

5.2 Complexities and Optimization Issues

About the performance of the partial key-recovery attack, it is seen from Algorithm 1 that to recover $4(2+\tau)$ bits, it runs $2^{4(2+\tau)} \cdot m \cdot 2^4(2+\tau) = m(2+\tau) \cdot 2^{12+4\tau}$ times to compute each α'_t for grading $2^{4(2+\tau)}$ candidates. In total, we have to perform $T = m(2+\tau) \cdot 2^{12+4\tau}$ operations, where m is set by Eq.(14).

Additionally, the loop from Line 9 to Line 15 can be done by one operation after precomputation as detailed below, which makes $T = m \cdot 2^{4(2+\tau)}$. During preprocessing, we run through every $y_{a_t-1}, \ldots, y_{a_t-\tau-2}$ (where y_t denotes the Hamming weight of the original four component LFSRs' output bits at time t) to compute the 2^4 possible sequences $\alpha'_{a_t}, \ldots, \alpha'_{a_t-\tau-1}$ which yield

2^4 sequences $b' = \bigoplus_{j=0}^{4} \alpha'_{a_t-j}, \ldots, \bigoplus_{j=0}^{4} \alpha'_{a_t-j-\tau+1}$ accordingly; and then for each τ-bit b'', we increment the counter $\mu^{b''}_{b'\oplus b''}$; last, we build up a table to store $h(y_{a_t-1}, \ldots, y_{a_t-\tau-2}; b'') = \sum_{b'} \mu^{b''}_{b'\oplus b''}(\mathcal{D}_1(b' \oplus b'') - \frac{1}{16})$. The precomputation needs memory $2^\tau \cdot 5^{2+\tau} \approx 2^{3.32\tau+4.6}$ and time $2^4 \cdot 5^{2+\tau} \cdot (2+\tau) \cdot 2^\tau \approx (\tau+2)2^{3.32\tau+8.6}$. After that, in real-time processing, for each frame i, we just compute $b^{i''} = b_0^{i''}, \ldots, b_{\tau-1}^{i''}$ with $b_n^{i''} = u_{t+n}^i \oplus s_{t+n}^i$ for $n = 0, \ldots, \tau-1$, deduce $y^i_{a_t-1}, \ldots, y^i_{a_t-\tau-2}$ from K', P^i and increment $G_{K'}$ by $h(y^i_{a_t-1}, \ldots, y^i_{a_t-\tau-2}; b^{i''})$. Thus, we get $T = m \cdot 2^{4(2+\tau)}$.

Moreover, when $2^{4(2+\tau)} \cdot 2^\tau \le m$, i.e. $2^{8+5\tau} \le m$, we can further reduce T down to $m + 2^{16+9\tau}$. Notice that it is the same subset of $4(2+\tau)$-bit Ω^i of P^i that is used to compute $y^i = (y^i_{a_t-1}, \ldots, y^i_{a_t-\tau-2})$ with K'. For convenience, let $g : GF(2)^L \to GF(2)^{4(2+\tau)}$ map P^i to Ω^i. We precompute a table $h'(\Omega, q)$ for every $4(2+\tau)$-bit Ω and τ-bit q defined by:

$$h'(\Omega, q) = \sum_{i=1}^{m} \mathbf{1}_{\Omega=g(P^i), q=u_t^i \oplus s_t^i}$$

with $u^i = (u_t^i, \ldots, u_{t+\tau-1}^i)$ and $s^i = (s_t^i, \ldots, s_{t+\tau-1}^i)$. This takes time $O(m)$ with memory $O(2^{8+5\tau})$. Recall that u^i is determined independent of K' by the distinguisher in Subsection 4.2, and u^i, s^i, y^i completely determine how to increment $G_{K'}$ for frame i, i.e. by $h(y^i; u^i \oplus s^i)$ from last paragraph. So, in real-time processing, for every K', instead of processing frame by frame to update $G_{K'}$, we simply go through every $(8+5\tau)$-bit pair (Ω, q), deduce $y = (y_{a_t-1}, \ldots, y_{a_t-\tau-2})$ from Ω and K', then increment $G_{K'}$ by $h'(\Omega, q)h(y; q)$. We reach the time complexity $T = m + 2^{4(2+\tau)} \cdot 2^{8+5\tau} = m + 2^{16+9\tau}$ for $2^{8+5\tau} \le m$.

To summarize, we have $T = m + 2^{4(2+\tau)} \cdot \min(m, 2^{8+5\tau})$. Table 4 lists the best complexities of our partial key-recovery attack corresponding to $\tau = 1, \ldots, 4$. Note that the success probability of the attack in the table is estimated according to the hypothesis test result of Eq.(11), i.e. the percentage of the $4(2+\tau)$-bit keys to generate a non-uniformly distributed sequence $\alpha'_{a_t-\tau-3}, \ldots, \alpha'_{a_t}$ with all the possible FSM state.

5.3 The Overall Key-Recovery Attack

Now we discuss how we proceed with the optimized Algorithm 1 to recover the full $\mathcal{K}$. With $\tau = 2$ and fixed k, we independently run Algorithm 1 three times with $t = 8k+1, \ldots, 8k+3$. And we expect at least two successes out of three

Table 4. Performance of our partial key-recovery attack against two-level E0

τ	Frames m	Time T	Prob. of Success	recovered key bits $4(2+\tau)$
1	2^{36}	2^{36}	50.8%	12
2	2^{35}	2^{35}	87.0%	16
3	$2^{34.5}$	2^{43}	99.0%	20
4	$2^{34.3}$	2^{52}	99.9%	24

runs. After checking consistency of all the overlapping bits for every possible pair of the algorithm outputs, we identify all the successful runs and obtain the minimum $16 + 4 = 20$ key bits.

We can easily adjust Algorithm 1 to target at any of the middle three biased bits on the right-hand side of Eq.(9) to recover 16 bits. With each modified partial key-recovery algorithm, we repeat previous procedure to recover minimum 20 bits. In total, we are sure to gather $4 \times 20 = 80$ bits. Since we already have 12 bits by the distinguisher, we finally exhaustively search the remaining $L-80-12 = 36$ bits within one frame. Algorithm 2 gives the abstract strategy of our complete attack. Therefore, to recover L-bit $\mathcal{K}$, our key-recovery attack works on $m = 2^{35}$ frames, in time $24m+4\times3\times2^{35} \approx 2^{40}$. The comparison of our attack with the best known attacks [7, 8, 9] against two-level E0 for $|\mathcal{K}| = L$ is available in Table 5.

Table 5. Comparison of our attack with the best attacks [7–9] against two-level E0 for $|\mathcal{K}| = L$

Attack	Precomputation	Time	Frames	Data	Memory
[7]	-	2^{73}	-	2^{43}	2^{51}
[8]	2^{80}	2^{65}	2	$2^{12.4}$	2^{80}
[9]	2^{80}	2^{70}	45	2^{17}	2^{80}
Our Attack	-	2^{40}	2^{35}	$2^{39.6}$	2^{35}

Algorithm 2 The abstract of the complete attack against two-level E0 for $|\mathcal{K}| = L$

Parameters:
1: m by Eq.(14)
Input:
2: m frames of 24-bit keystreams generated by the same $\mathcal{K}$ together with known nonces under two-level E0
Processing:
3: **for** each of the leftmost four biased bits on right-hand side of Eq.(9) **do**
4: choose $k \in \{0, 1, 2\}$ and set $\tau = 2$
5: **for** $t = 8k + 1, 8k + 2, 8k + 3$ **do**
6: use the optimized partial key-recovery attack to obtain 16 bits
7: **end for**
8: checking consistency among pairs of those 16 bits to obtain minimum 20-bit $\mathcal{K}$
9: **end for**
10: exhaustively search the remaining 36 key bits within one frame
11: output the L-bit $\mathcal{K}$

6 Conclusion

In this paper, based on one of the two largest biases inside the FSM within one-level E0, for the first time, we identify the bias at two-level E0 due to a

resynchronization flaw in Bluetooth E0. Unlike the traditional approach to find the bias, the characterized bias *does not* involve the precomputation of the multiple polynomial with low weight. Second, to utilize the identified bias, we develop a novel attack to directly recover the original encryption key for two-level E0 without reconstructing the initial state of E0 at the second level. Our key-recovery attack works with 2^{40} simple operations given the first 24 bits of 2^{35} frames. Compared with all existing attacks [15, 11, 7, 8, 9] against two-level Bluetooth E0, this is the best one so far, although the impossibly high amount of frames thwarts our attack to be practical. It remains an open challenge to decrease the data complexity with practical time and memory complexities. Finally, our attack illustrates the theory of statistical attacks in [2, 16] with an example which is not based on linear cryptanalysis.

Acknowledgments

We owe a lot grateful thanks to Anne Canteaut and Willi Meier. And we would also like to thank the anonymous reviewers for many helpful suggestions.

References

1. Frederik Armknecht, Matthias Krause, *Algebraic Attacks on Combiners with Memory*, Advances on Cryptography - CRYPTO 2003, Lecture Notes in Computer Science, vol.2729, D. Boneh Ed., Springer-Verlag, pp. 162-175, 2003
2. Thomas Baignères, Pascal Junod, Serge Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, in these proceedings
3. Bluetooth™, *Bluetooth Specification*, version 1.2, pp. 903-948, November, 2003, available at `http://www.bluetooth.org`
4. Philippe Chose, Antoine Joux, Michel Mitton, *Fast Correlation Attacks: An Algorithmic Point of View*, Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, vol.2332, L. R. Knudsen Ed., Springer-Verlag, pp. 209-221, 2002
5. Nicolas T. Courtois, *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Advances on Cryptography - CRYPTO 2003, Lecture Notes in Computer Science, vol.2729, D. Boneh Ed., Springer-Verlag, pp. 176-194, 2003
6. Patrik Ekdahl, Thomas Johansson, *Some Results on Correlations in the Bluetooth Stream Cipher*, Proceedings of the 10th Joint Conference on Communications and Coding, Austria, 2000
7. Scott Fluhrer, Stefan Lucks, *Analysis of the E0 Encryption System*, Selected Areas in Cryptography- SAC 2001, Lecture Notes in Computer Science, vol. 2259, S. Vaudenay and A. Youssef Eds., Springer-Verlag, pp. 38-48, 2001
8. Scott Fluhrer, *Improved Key Recovery of Level 1 of the Bluetooth Encryption System*, available at `http://eprint.iacr.org/2002/068`
9. Jovan Dj. Golić, Vittorio Bagini, Guglielmo Morgari, *Linear Cryptanalysis of Bluetooth Stream Cipher*, Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, vol. 2332, L. R. Knudsen Ed., Springer-Verlag, pp. 238-255, 2002

10. Miia Hermelin, Kaisa Nyberg, *Correlation Properties of the Bluetooth Combiner*, Information Security and Cryptology- ICISC'99, Lecture Notes in Computer Science, vol. 1787, JooSeok. Song Ed., Springer-Verlag, pp. 17-29, 2000
11. Matthias Krause, *BDD-Based Cryptanalysis of Keystream Generators*, Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, vol. 2332, L. R. Knudsen Ed., Springer-Verlag, pp. 222-237, 2002
12. Yi Lu, Serge Vaudenay, *Faster Correlation Attack on Bluetooth Keystream Generator E0*, Advances on Cryptography - CRYPTO 2004, Lecture Notes in Computer Science, vol.3152, M. Franklin Ed., Springer-Verlag, pp. 407-425, 2004
13. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology - EUROCRYPT'93, Lecture Notes in Computer Science, vol.765, Springer-Verlag, pp. 386-397, 1993
14. Alfred J. Menezes, Paul C. van. Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC, 1996
15. Markku Saarinen, *Re: Bluetooth and E0*, Posted at `sci.crypt.research`, 02/09/00
16. Serge Vaudenay, *An Experiment on DES - Statistical Cryptanalysis*, Proceedings of the 3rd ACM Conferences on Computer Security, pp. 139-147, 1996

Appendix

All our analysis here is similar with [16] and inspired by [2]. First, by Eq.(13), we have

$$\mathrm{Exp}(G_{K'}) = \frac{16m}{2^\tau} \sum_b f(b)$$

for a random wrong guess K', and by Eq.(11) we have

$$\mathrm{Exp}(G_K) = \frac{15m}{2^\tau} \sum_b f(b) + m \sum_b \mathcal{D}'(b) f(b),$$

for the right K. Hence,

$$\Delta\mathrm{Exp}(G_{K'}) = m \sum_b \left(\mathcal{D}'(b) - \frac{1}{2^\tau} \right) f(b). \tag{15}$$

Meanwhile, we compute the variance of $G_{K'}$ as

$$\mathrm{Var}(G_{K'}) = \frac{16m}{2^\tau} \sum_b \left(f(b)\right)^2 - \frac{16m}{2^{2\tau}} \left(\sum_b f(b) \right)^2. \tag{16}$$

We can estimate the rank of G_K over all possible $G_{K'}$ by

$$\mathrm{Exp}(\mathrm{Rank}_{G_K}) \approx 2^{4(2+\tau)} \Phi \left(- \frac{\Delta\mathrm{Exp}(G_{K'})}{\sqrt{2\mathrm{Var}(G_{K'})}} \right). \tag{17}$$

In order to achieve the top rank for G_K, we see that the fraction

$$\frac{\Delta\mathrm{Exp}(G_{K'})}{\sqrt{\mathrm{Var}(G_{K'})}} \tag{18}$$

must be large enough. This can be satisfied as long as the number m of available frames is sufficiently large. However, aiming at a practical attack, we are concerned with the question of how to choose f in order to minimize m under the constraint of the top rank G_K. In order to maximize the fraction (18), we first maximize the numerator with the constraint that the denominator is a constant and then try to maximize the fraction over all the solutions. Define the multivariate polynomial

$$g_f = m\sum_b \left(\mathcal{D}'(b) - \frac{1}{2^\tau}\right) f(b) + \frac{16m\gamma}{2^\tau}\sum_b (f(b))^2 - \frac{16m\gamma}{2^{2\tau}}\left(\sum_b f(b)\right)^2.$$

Using Lagrange's multiplier, we have

$$\frac{\partial g_f}{\partial f(b)} = m\left(\mathcal{D}'(b) - \frac{1}{2^\tau}\right) + \frac{32m\gamma}{2^\tau} f(b) - \frac{32m\gamma}{2^{2\tau}}\sum_{b'} f(b') = 0, \tag{19}$$

for all $b \in GF(2)^\tau$. From Eq.(19) we infer that

$$\frac{f(b) - f(b')}{\mathcal{D}'(b) - \mathcal{D}'(b')} = \text{const.}$$

for all $b \not/ b'$. Therefore we have a universal expression of f as

$$\frac{f(b) - \text{const.}}{\mathcal{D}'(b)} = \text{const}'.$$

for all $b \in GF(2)^\tau$, which yields the same quantity of (18) regardless of the constants in f. So the easiest way to define f could be $f(b) = \mathcal{D}'(b) - \frac{1}{2^\tau}$ for all $b \in GF(2)^\tau$. Then Eq.(15) reduces to

$$\begin{aligned}\Delta\mathrm{Exp}(G_{K'}) &= m\sum_b \left(\mathcal{D}'(b) - \frac{1}{2^\tau}\right)^2 \\ &= \frac{m}{2^\tau}\sum_{b\neq\mathbf{0}} \left(\hat{\mathcal{D}}(b)\right)^8 \\ &\approx \frac{m}{2^\tau}(2\tau - 1)\lambda^8.\end{aligned}$$

On the other hand Eq.(16) reduces to

$$\mathrm{Var}(G_{K'}) \approx \frac{16m}{2^{2\tau}}(2\tau - 1)\lambda^8.$$

So we deduce from Eq.(17) that

$$\begin{aligned}\mathrm{Exp}(\mathrm{Rank}_{G_K}) &\approx 2^{4(2+\tau)}\Phi\left(-\frac{\lambda^4}{4}\sqrt{\frac{m(2\tau-1)}{2}}\right)\\ &\approx \frac{2^{4(2+\tau)}}{\sqrt{2\pi}}e^{-\frac{m(2\tau-1)}{64}\lambda^8}.\end{aligned}$$

This means Rank_{G_K} is expected to top the chart with

$$m \approx \frac{256(2+\tau)\log 2}{\lambda^8(2\tau-1)} \approx \frac{\tau+2}{2\tau-1}\cdot 2^{34.5}.$$

On Provably Secure Time-Stamping Schemes

Ahto Buldas[1,2,3,⋆] and Märt Saarepera[4]

[1] University of Tartu, Liivi 2, 50409 Tartu, Estonia
Ahto.Buldas@ut.ee
[2] Cybernetica, Akadeemia tee 21, 12618 Tallinn, Estonia
[3] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia
[4] Independent researcher
marts@neoteny.com

Abstract. It is almost a folklore-knowledge that hash-based time-stamping schemes are secure if the underlying hash function is *collision-resistant* but still no rigorous proofs have been published. We try to establish such proof and conclude that the existing security conditions are improper because they ignore precomputations by adversaries. After analyzing a simplistic patent filing scenario, we suggest a new security condition for time-stamping schemes that leads to a new security property of hash functions – *chain-resistance*. We observe that if the variety of possible shapes of hash-chains is polynomial (and the verification procedure is suitably improved), then the time-stamping scheme becomes provably secure, assuming that the underlying hash function is collision-resistant. Finally, we show that in some sense, the restrictions in the security definition are necessary – conventional black-box techniques are unable to prove that chain-resistance follows from collision-resistance.

1 Introduction

The main goal of digital time-stamping is to prove that electronic data-items were created or registered at a certain time. A simple method is to use a trusted service (with a precise clock) that provides data items with current time value and digitally signs them. The assumption of unconditionally trusted service hides a risk of possible collusions that may not be acceptable in applications. The risks are especially high in centralized applications like electronic patent- or tax filing as well as in electronic voting, where the possible collusions are related to direct monetary (or even political) interests.

First attempts to eliminate trusted services from time-stamping schemes were made in [4], where cryptographic hash functions and publishing were used to replace electronic signatures. To date, several improvements of hash-based time-stamping schemes have been presented [1–3]. Such schemes have been used in business applications and are even included in international standards [9].

The combined monetary value of electronic content (insured, in particular, with time stamps) increases over time and so does the risk associated with it. A

⋆ Supported by Estonian SF grant no. 5113.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 500–514, 2004.

decision of a content manager to start using a certain time-stamping service for protecting electronic records must involve the assessment of long-term security risks. Desirably, such assessments should be based on analytical arguments. As an example of such argument, modern cryptography can prove that there are no *structural flaws* (or *principal design errors*) in security solutions, assuming that their basic building blocks (such as hash functions) are secure. The use of provably secure time-stamping schemes can avoid many practical risks.

Regardless of the growing importance of hash-based time-stamping schemes, their security is only superficially studied in scientific literature. In [5], a formal security condition for hash-based time-stamping schemes was presented and an informal sketch of a security proof was outlined. Though no rigorous proofs were presented it has become almost a public myth that the security of hash-based time-stamping schemes can be reduced to the *collision-resistance* of underlying hash functions. Thus far, no more related studies have been published.

In this paper, we revisit the security analysis of hash-based time-stamping schemes [5]. We observe that the formal security condition stated in [5] is unreachably strong because it overlooks pre-computations of the adversary.

Inspired by a simplistic patent filing scenario, we present a new security condition for time-stamping schemes that leads to a new security condition for hash functions – *chain-resistance* – necessary for the scheme [5] to be secure. We show that *additional checks* in the verifying procedure render the conventional time-stamping schemes provably secure in the new sense, based on *collision-resistance* of the hash function. The additions concern an examination of whether the shape of the hash-chain included into a time stamp belongs to a certain (polynomial) set of templates. This may seem a minor detail but as no currently used time-stamping schemes implement it, none of them are provably secure.

We further examine the necessity of said additional checks in the verification procedure and prove that without these checks *it is probably very hard (if not impossible) to prove the security of the schemes of type [5] based on collision-resistance alone*. We present an oracle relative to which there exist collision-resistant hash functions which are not chain-resistant. Almost all security proofs *relativize* – are valid relative to any oracle. Therefore, any security proof of the unmodified schemes should use either non-standard (non-relativizing) proof techniques or stronger/incomparable security assumptions on the underlying hash function. For example, it is easy to prove that entirely random hash functions (*random oracles*) are chain-resistant. In practice, it is often assumed that SHA-1 and other hash functions behave like random oracles which means that in such setting, their use in practical time-stamping schemes is justified. At the same time, it is still possible that a time-stamping scheme that uses SHA-1 is totally insecure while no collisions are found for SHA-1.

2 Notation and Definitions

By $x \leftarrow \mathcal{D}$ we mean that x is chosen randomly according to a distribution $\mathcal{D}$. If A is a probabilistic function or a Turing machine, then $x \leftarrow \mathsf{A}(y)$ means that

x is chosen according to the output distribution of A on an input y. By $\mathcal{U}_n$ we denote the uniform distribution on $\{0,1\}^n$. If $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are distributions and $F(x_1, \ldots, x_m)$ is a predicate, then $\mathsf{Pr}[x_1 \leftarrow \mathcal{D}_1, \ldots, x_m \leftarrow \mathcal{D}_m : F(x_1, \ldots, x_m)]$ denotes the probability that $F(x_1, \ldots, x_m)$ is true after the ordered assignment of $x_1, \ldots, x_m$. We write $f(k) = O(g(k))$ if there are $c, k_0 \in \mathbb{R}$, so that $f(k) \leq cg(k)$ $(\forall k > k_0)$. We write $f(k) = \omega(g(k))$ if $g(k) = O(f(k))$ but $f(k) \;/\; \Theta(g(k))$. A function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if $f(k) = k^{-\omega(1)}$. A Turing machine M is *polynomial-time* (*poly-time*) if it runs in time $k^{O(1)}$, where k denotes the input size. Let F^* be the class of all functions $f : \{0,1\}^* \to \{0,1\}^*$. Let FP be the class of all functions $f \in \mathsf{F}^*$ computable by poly-time Turing machines M. A distribution $\mathcal{D}$ on $\{0,1\}^*$ is *polynomially sampleable* if it is an output distribution of a poly-time Turing machine.

By an *oracle Turing machine* we mean an incompletely specified Turing machine S that comprises calls to *oracles*. The description can be completed by defining the oracle as a function $\mathcal{O} \in \mathsf{F}^*$. In this case, the machine is denoted by $S^{\mathcal{O}}$. An oracle $\mathcal{O}$ is not necessarily computable but may still have assigned a conditional (worst-case) running time $t(k)$, which may or may not reflect the actual amount of computations performed by $\mathcal{O}$ internally. Running time of $S^{\mathcal{O}}$ comprises the conditional worst-case running time $t(k)$ of oracle calls – each call takes $t(k)$ steps. An oracle $\mathcal{O}$ is *poly-time* if $t(k) = k^{O(1)}$. We say that S is a *poly-time oracle machine*, if the running time of $S^{\mathcal{O}}$ is polynomial, whenever $\mathcal{O}$ is poly-time. Let $\mathsf{FP}^{\bullet}$ denote the class of all poly-time oracle machines. Let $\mathsf{FP}^{\mathcal{O}}$ be the class of all functions computable by poly-time oracle machines $S^{\mathcal{O}}$.

A *primitive* $\mathfrak{P}$ is a class of (not necessarily computable by ordinary Turing machines) functions intended to perform a security related task (e.g. data confidentiality, integrity etc.). Each primitive $\mathfrak{P}$ is characterized by the success $\delta(k)$ of an adversary A. For example, a *collision-resistant hash function* is a function family $\{h_k\}_{k\in\mathbb{N}}$, where $h_k : \{0,1\}^{2k} \to \{0,1\}^k$ and

$$\delta(k) = \mathsf{Pr}[(x, x') \leftarrow \mathsf{A}(1^k) : x \;/\; \not\equiv x',\ h_k(x) = h_k(x')] \ .$$

In more rigorous definitions [12], h_k is randomly chosen from a set $\mathfrak{F} \subseteq \mathsf{F}^*$. Otherwise A may output a fixed collision. We write $h(x)$ instead of $h_k(x)$. Sometimes, we need hash functions $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ of type $\mathcal{H}_k : \{0,1\}^* \to \{0,1\}^k$. An adversary $\mathsf{A} \in \mathsf{F}^*$ *breaks* $f \in \mathfrak{P}$ (and write $\mathsf{A} \underset{\mathfrak{P}}{\mathsf{breaks}} f$) if A has non-negligible success. An instance $f \in \mathfrak{P}$ is *secure* if no $\mathsf{A} \in \mathsf{FP}$ breaks f. An instance $f \in \mathfrak{P}$ is *secure relative to an oracle* $\mathcal{O}$ if no $\mathsf{A} \in \mathsf{FP}^{\mathcal{O}}$ breaks f.

3 Time-Stamping Schemes and Their Security

3.1 The Scheme of Haber and Stornetta

A time-stamping scheme [5] involves three parties: a *Client* C, a *Server* S, and a *Repository* $\mathfrak{R}$; and two procedures for *time-stamping* a data item and for

verifying a time stamp (Fig. 1). It is assumed that $\mathfrak{R}$ is write-only and receives items from S in an authenticated manner.

Time-stamping procedure is divided into rounds of equal duration. During each round, S receives requests from Clients. For simplicity, all requests $x_1, \ldots, x_m$ are assumed to be bit-strings $x_i \in \{0,1\}^k$. If the t-th round is over, S computes a compound hash $r_t \in \{0,1\}^k$ by using a function $h\colon \{0,1\}^{2k} \to \{0,1\}^k$ and a tree-shaped hashing scheme $r_t = G_h(x_1, \ldots, x_m)$. For example, if the requests of the t-th round are x_1, x_2, x_3, x_4, then S may compute $r_t = h(x_1, h(h(x_2, x_3), x_4))$. Next, S sends r_t to $\mathfrak{R}$ (in a secure way), where it is stored as a pair (t, r_t).

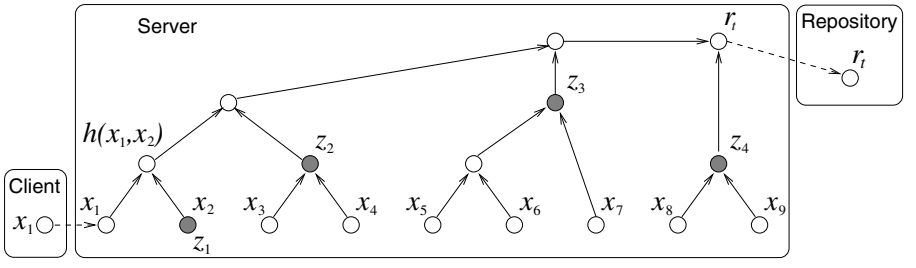


Fig. 1. The scheme of Haber and Stornetta (a simplified model)

After that, S issues for each request x a *time-certificate* $c = (x, t, n, z)$, where t is current time value, n is an identifier $n = n_1 n_2 \ldots n_\ell \in \{0,1\}^\ell$, and z is a sequence $z = (z_1, z_2, \ldots, z_\ell) \in \left(\{0,1\}^k\right)^\ell$. In the scheme of Fig. 1, the time-certificate for x_1 is $(x_1, t, \mathsf{0000}, (z_1, z_2, z_3, z_4))$, where $z_1 = x_2$, $z_2 = h(x_3, x_4)$, $z_3 = h(h(x_5, x_6), x_7)$, and $z_4 = h(x_8, x_9)$.

Verification procedure is performed by C as follows. To verify (x, t, n, z), C computes a hash value r'_t by using h and a $F_h(x; n; z)$, which computes a sequence $y = (y_0, y_1, \ldots, y_\ell) \in \left(\{0,1\}^k\right)^\ell$ inductively, so that $y_0 := x$, and

$$y_i := \begin{cases} h(z_i, y_{i-1}) \text{ if } n_i = 1 \\ h(y_{i-1}, z_i) \text{ if } n_i = 0 \end{cases} \tag{1}$$

for $i > 0$, and outputs $r'_t = F_h(x; n; z) := y_\ell$. Second, C sends a query t to $\mathfrak{R}$ and obtains r_t as a reply. Finally, C checks whether $r'_t = r_t$. Note that n and z can be equal to empty string $\lfloor\rfloor$, in which case $F_h(x; n; z) = x$.

Security condition [5] states that the time-stamping scheme above is secure against $\mathsf{A}_{\mathrm{HS}} \in \mathsf{FP}$ that sends requests $x_1, \ldots, x_q$ to S and queries to $\mathfrak{R}$. As a result, A_{HS} outputs a time-certificate (x, t, n, z), where $x \in \{0,1\}^k$, $n \in \{0,1\}^\ell$, and $z \in (\{0,1\}^k)^\ell$. The attack is considered successful, if $x \notin \{x_1, \ldots, x_q\}$ and $F_h(x; n; z) = r_t$, where r_t is assumed to be the correct response of $\mathfrak{R}$ to query t.

3.2 Analysis of the Security Condition

The scheme described above is insecure against the following behavior of A_{HS}:

- A_{HS} chooses x and z_0 uniformly at random.
- A_{HS} sends $x_0 = h(x, z_0)$ to S and obtains a time-certificate (x_0, t, n, z).
- A_{HS} computes a faked time-certificate $(x, t, \mathsf{0}\|n, z_0\|z)$,

where $\|$ denotes concatenation. By definition, $F_h(x; \mathsf{0}\|n; z_0\|z) = F_h(x_0; n; z) = r_t$. Hence, the attack is successful whenever $x \mathrel{/\!\!=} x_0$ because x_0 was the only request made by A_{HS}. If h has reasonable security properties then $\mathsf{Pr}[x \mathrel{/\!\!=} x_0]$ is non-negligible[1]. This "attack" shows that the formal security definition does not follow the intuition behind time-stamping, because it overlooks the possibility of precomputations. As a success criterion, the condition $x \mathrel{/\!\!\in} \{x_1, \ldots, x_q\}$ is improper because the notion of *already time-stamped items* is not sufficiently precise.

4 New Security Condition and Improved Schemes

4.1 New Security Condition

The new security condition is inspired by the following simplistic attack-scenario, where Bob, a criminal who steals inventions, co-operates with a server S:

- Bob precomputes (not necessarily with G_h) some hash values $r_1, \ldots, r_s$ that may help him to back-date documents in the future. His collaborator S sends the hash values to $\mathfrak{R}$, where they are stored as pairs $(t_1, r_1), \ldots, (t_s, r_s)$.
- Alice, an inventor, creates a description $X_A \in \{0,1\}^*$ of her invention and requests a time certificate for $x_A = \mathcal{H}(X_A)$, where $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^k$ is a (collision-resistant) hash function.
- Some time later, the invention is disclosed to the public and Bob tries to steal the rights to Alice's invention. He creates a slightly modified version X_B of X_A (at least the author's name should be replaced), and tries to back-date it relative to X_A. Bob is successful, if he finds n and z, so that $F_h(x_B; n; z) \in \{r_1, \ldots, r_s\}$. Bob can use (x_B, t, n, z) to claim his rights to the invention.

In order to formalize such attack scenario, a two-staged adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is needed. The first stage A_1 *precomputes* a set $\mathfrak{R} = \{r_1, \ldots, r_s\}$ after which the second stage A_2 obtains a *new document* $X \in \{0,1\}^*$ ("a document, unknown to mankind before") and tries to back-date it by finding n and z, so

[1] If h is collision-resistant, $\mathsf{Pr}[x = x_0] = \mathsf{Pr}[x, z_0 \leftarrow \mathcal{U}_k\colon h(x, z_0) = z] = \delta$, and $p_x = \mathsf{Pr}[x' \leftarrow \mathcal{U}_k\colon h(x, x') = x]$, then the collision-finding adversary $(xx', xx'') \leftarrow \mathsf{A}(1^k)$ (where $x, x', x'' \leftarrow \mathcal{U}_k$ are independent random bit-strings) has non-negligible success. Indeed, the probability δ' that A outputs a collision for h (possibly, with $x' = x''$) is $\delta' \geq \sum_x \mathsf{Pr}[x] \cdot p_x^2 \geq \left(\sum_x \mathsf{Pr}[x] \cdot p_x\right)^2 = \delta^2$, where $\mathsf{Pr}[x] = \mathsf{Pr}[X \leftarrow \mathcal{U}_k\colon X = x]$. Hence, the overall success of A is at least $\delta^2 - 2^{-k}$.

that $F_h(\mathcal{H}(X); n; z) \in \mathfrak{R}$. The term *new document* is hard to define formally. As we saw, the condition $\mathcal{H}(X) \,/\in x\{, \ldots, x_q\}$ does not guarantee that X is really *new*. We assume that X is chosen according to a distribution $\mathcal{D}$ on $\{0,1\}^*$ that is somewhat unpredictable to A. The success of A is defined as follows:

$$\delta(k) = \mathsf{Pr}[(\mathfrak{R}, a) \leftarrow \mathsf{A}_1(1^k), X \leftarrow \mathcal{D}, (n, z) \leftarrow \mathsf{A}_2(X, a) \colon F_h(\mathcal{H}(X); n; z) \in \mathfrak{R}] \ , \quad (2)$$

where a denotes a state information sent from A_1 to A_2. Note that (2) can be simplified by assuming $|\,\mathfrak{R}\,| = 1$, because this reduces $\delta(k)$ only polynomially.

Necessary Conditions for $\mathcal{D}$. Intuitively, the prediction of $\mathcal{D}$ must require super-polynomial time-success ratio, i.e. every A with running time $T(k)$ and success $\delta(k) = \mathsf{Pr}[\mathfrak{R} \leftarrow \mathsf{A}(1^k), x \leftarrow \mathcal{D} \colon x \in \mathfrak{R}]$ has time-success ratio $\frac{T(k)}{\delta(k)} = k^{\omega(1)}$. In case $\mathcal{D}$ is *polynomially sampleable*, an equivalent assumption is that

$$\mathsf{P}_{\mathrm{C}}(\mathcal{D}) = \mathsf{Pr}[y \leftarrow \mathcal{D}, x \leftarrow \mathcal{D} \colon x = y] = k^{-\omega(1)}, \quad (3)$$

where $\mathsf{P}_{\mathrm{C}}(\mathcal{D})$ is the *collision probability* of $\mathcal{D}$. Indeed, if for a poly-time A we have $\mathsf{Pr}[\mathfrak{R} \leftarrow \mathsf{A}(1^k), x \leftarrow \mathcal{D} \colon x \in \mathfrak{R}] = T(k) \cdot k^{-O(1)}$, then there is $\mathfrak{R}_{\mathsf{o}} \subseteq \{0,1\}^k$, so that $|\,\mathfrak{R}_{\mathsf{o}}\,| = k^{O(1)}$ and $\mathsf{Pr}[x \leftarrow \mathcal{D} \colon x \in \mathfrak{R}_{\mathsf{o}}] = k^{-O(1)}$. Thus,

$$\exists r \in \mathfrak{R}_{\mathsf{o}} \colon p = \mathsf{Pr}[x \leftarrow \mathcal{D} \colon x = r] = |\,\mathfrak{R}_{\mathsf{o}}\,|^{-1} \cdot k^{-O(1)} = k^{-O(1)}$$

and hence $\mathsf{P}_C(\mathcal{D}) \geq p^2 = k^{-O(1)}$. If, in turn, $\mathsf{P}_{\mathrm{C}}(\mathcal{D}) = k^{-O(1)}$, then every A with output distribution $\mathcal{D}$ has success $\delta(k) = k^{-O(1)}$.

The condition (3) is equivalent to the requirement that $\mathcal{D}$ has *Rényi entropy* $\mathsf{H}_2(\mathcal{D}) = -\log_2 \mathsf{P}_{\mathrm{C}}(\mathcal{D}) = \omega(\log k)$, and is in fact *necessary* for a time-stamping scheme to be secure relative to $\mathcal{D}$. Indeed, if A_1 is defined to output $y \leftarrow \mathcal{D}$ and $(\lfloor\rfloor, \lfloor\rfloor) \leftarrow \mathsf{A}_2(x, a)$ (for any x and a), then $(\mathsf{A}_1, \mathsf{A}_2)$ has success $\mathsf{P}_{\mathrm{C}}(\mathcal{D})$.

Chain-Resistant Hash Functions. The security definition (2) implies that h must satisfy the following new security condition for hash functions:

Definition 1. *A hash function h is* chain resistant (relative to a distribution $\mathcal{D}_k$ on $\{0,1\}^k$)*, if for every adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2) \in \mathsf{FP}$:*

$$\mathsf{Pr}[(\mathfrak{R}, a) \leftarrow \mathsf{A}_1(1^k), x \leftarrow \mathcal{D}_k, (n, z) \leftarrow \mathsf{A}_2(x, a) \colon F_h(x; n; z) \in \mathfrak{R}] = k^{-\omega(1)} \ . \quad (4)$$

It is easy to show that if a time-stamping scheme is secure relative to $\mathcal{D}$, then the hash function h is chain-resistant relative to $\mathcal{H}(\mathcal{D})$.

4.2 Improved Verification Procedure

We will prove later that the conventional black-box techniques are insufficient to imply chain-resistance from collision-resistance, and hence, also the security of time-stamping schemes in the sense of (2) cannot be proved under the collision-resistance condition alone. We modify the verification procedure in a way that prevents the adversary from finding chains for h without finding collisions as by-products. We restrict the set $\mathfrak{N} \subset \{0,1\}^*$ of identifiers (possible shapes of hash

chains) that are considered valid by the verification procedure and show that if $|\mathfrak{N}| = k^{O(1)}$, then the collision-resistance of h is sufficient for a time-stamping scheme to be secure. The modified verification procedure is defined as follows:

New Verification Procedure. To verify a time-certificate (x, t, n, z) for $X \in \{0,1\}^*$, C checks if $x = \mathcal{H}(X)$, computes a hash value r'_t using $F_h(x; n; z)$ defined by (1), sends a query t to $\mathfrak{R}$ to obtain r_t, and checks if $r'_t = r_t$ and $n \in \mathfrak{N}$.

To be usable in practical time-stamping, the condition $n \in \mathfrak{N}$ must be efficiently verifiable. One way to achieve this is to set $\mathfrak{N} = \{0,1\}^{k_0}$, where k_0 is constant, which means that $n \in \mathfrak{N}$ is equivalent to $\|n\| = k_0$ and is naturally efficiently computable. The set $\mathfrak{N}$ can be viewed as a template of a *hashing scheme* that is published by the service provider before the service starts. As we consider service providers as possible adversaries, $\mathfrak{N}$ is created by an adversary. The restrictions above lead us to a weaker condition with the following notion of success:

$$\mathsf{Pr}[(\mathfrak{R},\mathfrak{N},a) \leftarrow \mathsf{A}_1(1^k), X \leftarrow \mathcal{D}, (n,z) \leftarrow \mathsf{A}_2(X,a) \colon F_h(\mathcal{H}(X); n; z) \in \mathfrak{R}, n \in \mathfrak{N}] \ . \quad (5)$$

4.3 Proof of Security

We prove that the security of the modified hash-based time-stamping schemes follows from the collision-resistance of h and $\mathcal{H}$.

Definition 2. *Let $y = (y_0, y_1, \ldots, y_\ell)$ and $y' = (y'_0, y'_1, \ldots, y'_{\ell'})$ be two sequences produced by $F_h(x; n; z)$ and $F_h(x'; n'; z')$ respectively, by using (1). Let $z = (z_1, \ldots, z_\ell)$, $z' = (z'_1, \ldots, z'_{\ell'})$, $n = n_1 \ldots n_\ell$, and $n' = n'_1 \ldots n'_{\ell'}$. We say that sequences y and y' comprise a collision, if for some indices $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, \ell'\}$, $h(a,b) = y_i = y'_j = h(a',b')$, but $(a,b) \neq (a',b')$, where*

$$(a,b) = \begin{cases} (z_i, y_{i-1}) \text{ if } n_i = 1 \\ (y_{i-1}, z_i) \text{ if } n_i = 0 \end{cases} \quad \text{and} \quad (a',b') = \begin{cases} (z'_j, y'_{j-1}) \text{ if } n'_j = 1 \\ (y'_{j-1}, z'_j) \text{ if } n'_j = 0 \end{cases} .$$

Lemma 1. *If $x \neq x'$ and $F_h(x; n; z) = F_h(x'; n; z')$, then the sequences y and y' computed internally by $F_h(x; n; z)$ and $F_h(x'; n; z')$ comprise a collision.*

Proof. Let ℓ be the bit-length of n. As $x = y_0 \neq y'_0 = x'$ and $y_\ell = F_h(x; n; z) = F_h(x'; n; z') = y'_\ell$, there exists $i \in \{1, \ldots, \ell\}$, such that $y_i = y'_i$ and $y_{i-1} \neq y'_{i-1}$. Hence, either $h(z_i, y_{i-1}) = h(z'_i, y'_{i-1})$ or $h(y_{i-1}, z_i) = h(y'_{i-1}, z'_i)$. In both cases, we have a collision. □

Theorem 1. *If h and $\mathcal{H}$ are collision-resistant, then the time-stamping scheme is secure in the sense of (5) relative to every polynomially sampleable $\mathcal{D}$ with Rényi entropy $\mathsf{H}_2(\mathcal{D}) = \omega(\log k)$.*

Proof. Let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be an adversary with running time $T(k)$ that breaks the time-stamping scheme in the sense of (5) with success $\delta(k)$. Assuming the

collision-resistance of $\mathcal{H}$, we construct a collision-finding adversary A' for h with running time $T'(k) = k^{\mathcal{O}(1)}T(k)$ and with success $\delta'(k) \geq \frac{\delta^2(k)}{T^2(k)} - k^{-\omega(1)}$, and hence, $\frac{T'(k)}{\delta'(k)} = k^{O(1)}\left(\frac{T(k)}{\delta(k)}\right)^2$. The adversary A':

- calls A_1 and obtains $\mathfrak{R} = \{r_1, \ldots, r_m\}$, $\mathfrak{R} \subset \{0,1\}^*$, and $a \in \{0,1\}^*$;
- generates two independent random strings $X, X' \leftarrow \mathcal{D}$ and calls A_2 twice to obtain $(n, z) \leftarrow \mathsf{A}_2(X, a)$ and $(n', z') \leftarrow \mathsf{A}_2(X', a)$;
- simulates $F_h(\mathcal{H}(X); n; z)$ and $F_h(\mathcal{H}(X'); n'; z')$.

If $F_h(\mathcal{H}(X); n; z) = F_h(\mathcal{H}(X'); n'; z')$, $\mathcal{H}(X)$ / $\mathcal{H}(X')$, and $n = n'$, then by Lemma 1 above, A' is able to find a collision for h. By Lemma 2 below, the probability that all these conditions hold is at least $\frac{\delta^2(k)}{T^2(k)} - 2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))}$.

It remains to show that $2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))} = \mathsf{P}_{\mathrm{C}}(\mathcal{H}(\mathcal{D})) = k^{-\omega(1)}$. Let C be a collision-finding adversary that on input 1^k generates $X \leftarrow \mathcal{D}$ and $X' \leftarrow \mathcal{D}$ independently at random and outputs (X, X'). Let E_1 denote the event that $X = X'$. Hence, $\mathsf{Pr}[\mathsf{E}_1] = \mathsf{P}_{\mathrm{C}}(\mathcal{D}) = k^{-\omega(1)}$. Let E_2 be the event that $\mathcal{H}(X) = \mathcal{H}(X')$. As $\mathcal{H}$ is collision-resistant, the success of C is $\mathsf{Pr}[\mathsf{E}_2 \backslash \mathsf{E}_1] = k^{-\omega(1)}$ and due to $\mathsf{E}_1 \subseteq \mathsf{E}_2$, we have $\mathsf{P}_{\mathrm{C}}(\mathcal{H}(\mathcal{D})) = \mathsf{Pr}[\mathsf{E}_2] = \mathsf{Pr}[\mathsf{E}_1] + \mathsf{Pr}[\mathsf{E}_2 \backslash \mathsf{E}_1] = k^{-\omega(1)} + k^{-\omega(1)} = k^{-\omega(1)}$. □

Lemma 2. *The success of* A' *is at least* $\frac{\delta^2(k)}{T^2(k)} - 2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))}$. *(See Appendix)*

Remark. Using the improved verification procedure, it is possible to achieve the original security condition of Haber and Stornetta, assuming that the server S is honest and the set $\mathfrak{R}$ is a prefix-free code with a polynomial number of words.

5 Necessity of the Improved Verification

In this section, we prove that the conventional proof techniques used in theoretical cryptography – *black-box reductions* and *semi black-box reductions* – are unable to prove that collision-resistance implies chain-resistance. Hence, in some sense the modifications in time-stamping schemes are necessary for establishing their provable security. For the self-containedness of this paper, we introduce some basic results about *oracle separation*, which have been used to prove several "impossibilities" in theoretical cryptography [6–8, 13].

5.1 Cryptographic Reductions and Oracle Separation

Almost all known constructions of a new primitive $\mathfrak{P}_2$ from another $\mathfrak{P}_1$ belong to one of the following two types:

Definition 3. *A* semi black-box reduction *from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$ *is a machine* $P \in \mathsf{FP}^{\bullet}$, *so that (1)* $P^f \in \mathfrak{P}_2$ $(\forall f \in \mathfrak{P}_1)$; *and (2) for any* $\mathsf{A}_2 \in \mathsf{FP}^{\bullet}$ *there exists* $\mathsf{A}_1 \in \mathsf{FP}^{\bullet}$, *so that* $\mathsf{A}_2^f \underset{\mathfrak{P}_2}{\mathsf{breaks}} P^f$ *implies* $\mathsf{A}_1^f \underset{\mathfrak{P}_1}{\mathsf{breaks}} f$ $(\forall f \in \mathfrak{P}_1)$.

Definition 4. *A* (fully) black-box reduction *from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$ *is a pair of machines* $P, S \in \mathsf{FP}^\bullet$*, so that (1)* $P^f \in \mathfrak{P}_2$ $(\forall f \in \mathfrak{P}_1)$*; and (2)* $\mathsf{A} \underset{\mathfrak{P}_2}{\textsf{ breaks }} P^f$ *implies* $S^{\mathsf{A},f} \underset{\mathfrak{P}_1}{\textsf{ breaks }} f$ $(\forall f \in \mathfrak{P}_1, \forall \mathsf{A} \in \mathsf{F}^*)$.

Note that the universal quantifiers apply over F^* instead of FP. The reason is that uniform reductions stay valid if the quantifiers' range is extended from FP to F^* and this is exactly what expresses the *black-box nature* of f and A in these reductions. We will use the following folklore lemmas about oracle separation.

Lemma 3. *(A) If there is* $f \in \mathfrak{P}_1 \cap \mathsf{FP}^{\mathcal{O}}$ *secure relative to* $\mathcal{O}$ *but no* $g \in \mathfrak{P}_2 \cap \mathsf{FP}^{\mathcal{O}}$ *is secure relative to* $\mathcal{O}$*, then there exist no (fully) black-box reductions from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$*. (B) If in addition,* $\mathcal{O} = \pi^f$ *(equality of functions) for a* $\pi \in \mathsf{FP}^\bullet$*, then there exist no semi black-box reductions from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$*.*

Proof. (A) Suppose (S, P) is a black-box reduction from $\mathfrak{P}_1$ to $\mathfrak{P}_2$. According to the assumptions, $g = P^f \in \mathfrak{P}_2 \cap \mathsf{FP}^{\mathcal{O}}$ and g is insecure relative to $\mathcal{O}$. Hence, $\mathsf{A} \underset{\mathfrak{P}_2}{\textsf{ breaks }} g = P^f$ for some $\mathsf{A} \in \mathsf{FP}^{\mathcal{O}} \subset \mathsf{F}^*$. It follows that $S^{f,\mathsf{A}} \underset{\mathfrak{P}_1}{\textsf{ breaks }} f$, contradicting $S^{f,\mathsf{A}} \in \mathsf{FP}^{\mathcal{O}}$. (B) Suppose P is a semi black-box reduction from $\mathfrak{P}_1$ to $\mathfrak{P}_2$. Let $f \in \mathfrak{P}_1 \cap \mathsf{FP}^{\mathcal{O}}$ be a secure (relative to $\mathcal{O}$) instance of $\mathfrak{P}_1$. Let $\mathcal{O} = \pi^f$ for some $\pi \in \mathsf{FP}^\bullet$. According to the assumptions, $g = P^f \in \mathfrak{P}_2 \cap \mathsf{FP}^{\mathcal{O}}$ and g is insecure relative to $\mathcal{O}$. Hence, $\mathsf{A} \underset{\mathfrak{P}_2}{\textsf{ breaks }} g = P^f$ for some $\mathsf{A} \in \mathsf{FP}^{\mathcal{O}}$. Therefore, taking $\mathsf{A}_2 = \mathsf{A}^\pi \in \mathsf{FP}^\bullet$ we have that $\mathsf{A} = \mathsf{A}^{\mathcal{O}} = \mathsf{A}^{\pi^f} = \mathsf{A}_2^f \underset{\mathfrak{P}_2}{\textsf{ breaks }} P^f$. Hence, there exists $\mathsf{A}_1 \in \mathsf{FP}^\bullet$, so that $\mathsf{A}_1^f \underset{\mathfrak{P}_1}{\textsf{ breaks }} f$, which contradicts $\mathsf{A}_1^f \in \mathsf{FP}^{\mathcal{O}}$. □

Definition 5. *A (semi/fully) black-box reduction is said to be a* self reduction *if* P *is a trivial machine, i.e.* $P^f = f$ *(for every* f*).*

Lemma 4. *(A) If relative to* $\mathcal{O}$ *there is a secure instance of* $f \in \mathfrak{P}_1$*, which is also an insecure instance of* $\mathfrak{P}_2$*, then there exist no (fully) black-box self reductions from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$*. (B) If in addition,* $\mathcal{O} = \pi^f$ *(equality of functions) for a* $\pi \in \mathsf{FP}^\bullet$*, then there exist no semi black-box self reductions from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$*.*

The proof of Lemma 4 is completely analogous to the proof of Lemma 3.

5.2 Non-existence of Fully Black-Box Self Reductions

We define an oracle $\mathcal{O}$, relative to which there exist a collision-resistant hash function $H\colon \{0,1\}^{2k} \to \{0,1\}^k$ (chosen randomly from a set $\mathfrak{F}$ of functions) that is not chain-resistant. The oracle $\mathcal{O}$ responds to the following queries:

- H-queries that given as input $(x_1, x_2) \in \{0,1\}^{2k}$ return $H(x_1, x_2) \in \{0,1\}^k$.
- A_1-queries that given as input 1^k return the root r_k of a Merkle tree [11] M_k, the leaves of which are all k-bit strings in lexicographic order (Fig. 2).
- A_2-queries that given as input a bit string $x \in \{0,1\}^k$ find $z \in (\{0,1\}^k)^k$, based on M_k, so that $F_H(x; x; z) = r_k$ and output a pair (x, z).

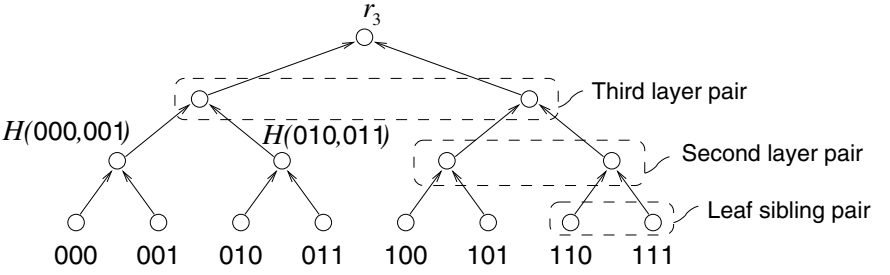


Fig. 2. Computations performed by $A_1(1^k)$ in case $k = 3$

We assume that $\mathcal{O}$-queries are of unitary cost and hence H is not chain-resistant relative to $\mathcal{O}$. We define $\mathfrak{F}$ so that $\mathcal{O}$ is insufficient for finding collisions for H.

Let $\mathfrak{F}$ be the set of all functions H, such that for all k: (1) all non-leaf vertices in M_k contain different elements of $\{0,1\}^k$ and (2) all sibling-pairs (including the leaves) are different. Hence, the argument-value pairs in M_k do not comprise collisions and A_1- and A_2-queries do not help in finding collisions for H.

Lemma 5. *Every collision finding adversary* $\mathsf{A}^{\mathcal{O}}$ *for* H *that makes* $p(k) = k^{O(1)}$ *oracle calls, has success probability* $k^{-\omega(1)}$.

Proof. Let $S \subseteq \{0,1\}^{2k}$ denote the set of all pairs in the tree M_k. There are exactly $2^k - 1$ of such pairs. Hence, there are $2^{2k} - 2^k + 1$ pairs in the complement $\overline{S} = \{0,1\}^{2k} \backslash S$. The restriction of H to $\overline{S}$ behaves like a uniformly random function while the restriction of H to S is injective. Hence, if $\mathsf{A}^{\mathcal{O}}$ finds a collision (p_1, p_2) for H, then one or both of the pairs p_1, p_2 belong to $\overline{S}$.

Let $K \subset \{0,1\}^{2k}$ be the set of all pairs for which the value of H is released. If $p_1, p_2 \in \overline{S}$, then the probability of finding collisions does not exceed $\frac{|K \cap \overline{S}|^2}{2^{k+1}} \leq \frac{p^2(k)}{2^{k+1}} = k^{-\omega(1)}$, because the values of $H|_S$ can only be obtained via H-queries.

If $p_1 \in S$ and $p_2 \in \overline{S}$, then the probability of finding a collision does not exceed $\frac{|K \cap S| \cdot |K \cap \overline{S}|}{2^k} \leq \frac{mk \cdot (p(k) - m)}{2^k}$, where m is the number of A_2-queries, each of which releases no more than k values of H. The maximum of the last function is achieved if $m \approx \frac{p(k)}{2}$, and hence the success is $\frac{k \cdot p(k)^2}{2^{k+2}} = k^{-\omega(1)}$. □

Corollary 1. *Fully black-box self reductions cannot prove that collision-resistance of* h *implies chain-resistance of* h.

5.3 Non-existence of Semi Black-Box Self Reductions

The oracle $\mathcal{O}$ defined above does not yet prove the non-existence of semi-black box self reductions because H does not provide full access to $\mathcal{O}$, i.e. $\mathcal{O} \not\equiv H$. Hence, we have to "embed" $\mathcal{O}$ into H. We define a new hash function (oracle) $\mathcal{O}$: $\{0,1\}^{2n} \to \{0,1\}^n$ recursively for all $n > 0$, assuming that the values of it are already defined for smaller indices.

Let M_n be a complete Merkle tree, the leaves of which are all n-bit strings in the lexicographic order. Each internal vertex v in M_n is computed as a hash $\mathcal{O}_n(v_L, v_R)$ of the child vertices v_L, v_R of v. Note that as we have not yet defined $\mathcal{O}_n: \{0,1\}^{2n} \to \{0,1\}^n$, the tree M_n is not yet defined either. We divide the domain $\{0,1\}^{2n} = \{(y_1, y_2): y_1, y_2 \in \{0,1\}^n\})$ into two non-intersecting parts:

- The set S of all sibling pairs in M_n that occur as inputs to $\mathcal{O}$ during the computation of M_n. It contains *leaf-sibling pairs* of the form $(y\mathsf{0}, y\mathsf{1})$, where $y \in \{0,1\}^{n-1}$, *second-layer pairs* of the form $(\mathcal{O}(t\mathsf{00}, t\mathsf{01}), \mathcal{O}(t\mathsf{10}, t\mathsf{11}))$, where $t \in \{0,1\}^{n-2}$ etc. (Fig. 2)
- The set P of all other pairs.

Hence, to define $\mathcal{O}_n$, we have to define two functions: $\mathcal{O}_n^S: S \to \{0,1\}^n$ and $\mathcal{O}_n^P: P \to \{0,1\}^n$. The function $\mathcal{O}_n^S$ is defined in a deterministic way and is injective (no collisions can be found inside S), while $\mathcal{O}_n^P$ is a random oracle (obviously collision-resistant!). In addition, if $n = 4k$, then we embed a chain-finding adversary for $\mathcal{O}_k$ into $\mathcal{O}_n^S$, which means that $\mathcal{O}$ can find chains for itself and is thereby not chain-resistant.

First of all, we define (for $n = 4k$) an oracle $\mathcal{A}_n: \{0,1\}^{2n} \to \{0,1\}^n$ that can be used to find chains for $\mathcal{O}_k$. The oracle $\mathcal{A}_n$ allows input pairs of the form $(\mathsf{0}^{2k} x \mathsf{0}^{k-m}\mathsf{1}^m, \mathsf{0}^k\mathsf{1}^k x \mathsf{0}^{k-m}\mathsf{1}^m)$, where $x \in \{0,1\}^k$ and $m \in \{0, \ldots, k\}$. The set D of all such pairs has exactly $(k+1)2^k$ elements. Let r_k be the root of M_k (which has been already defined). On input of such form, the oracle $\mathcal{A}_n$ finds (based on M_k) $z \in (\{0,1\}^k)^k$, such that $F_{\mathcal{O}}(x; x; z) = r_k$. We define $\mathcal{A}_n$ as follows:

$$\mathcal{A}_n(\mathsf{0}^{2k} x \mathsf{0}^{k-m}\mathsf{1}^m, \mathsf{0}^k\mathsf{1}^k x \mathsf{0}^{k-m}\mathsf{1}^m) = \begin{cases} \mathsf{1}^k x \mathsf{0}^{k-m}\mathsf{1}^m x & \text{if } m = 0 \ , \\ \mathsf{1}^k x \mathsf{0}^{k-m}\mathsf{1}^m z_m & \text{if } m \in \{1, \ldots, k\} \ . \end{cases}$$

Obviously, $\mathcal{A}_n$ is injective and its values never coincide with the allowed inputs.

Now we are ready to define $\mathcal{O}_n^S$. We begin with the case $n \neq 4k$ which is considerably easier, because there is no need to embed $\mathcal{A}_n$ into $\mathcal{O}_n^S$. To define $\mathcal{O}_n^S$ as an injection, it is sufficient to assign different n-bit strings to all $2^n - 1$ internal vertices of M_n. However, care must be taken that no sibling pairs (including the leaf sibling pairs) coincide with other pairs, because otherwise we may have a contradictory definition – different values are assigned to the same input pair. Such contraditions can be easily avoided if, as opposed to the leaf sibling pairs, the elements of internal sibling pairs are in the decreasing order.

If $n = 4k$, then we have to embed $\mathcal{A}_n$ as a function into $\mathcal{O}_n^S$. There are $2^{n-2} = 2^{4k-2}$ second layer pairs in M_n and $(k+1)2^k$ arguments of $\mathcal{A}_n$ (elements of D). As $(k+1)2^k \leq 2^{4k-2}$ for any $k > 0$, there is an injection $e: D \to \{0,1\}^{n-2}$ and we can embed D into the set of second layer pairs of M_n, so that for each $x \in \{0,1\}^k$ and $m \in \{0, \ldots, k\}$ there is $t = e(x, m) \in \{0,1\}^{n-2}$, such that $\mathcal{O}(t\mathsf{00}, t\mathsf{01}) = \mathsf{0}^{2k} x \mathsf{0}^{k-m}\mathsf{1}^m$ and $\mathcal{O}(t\mathsf{10}, t\mathsf{11}) = \mathsf{0}^k\mathsf{1}^k x \mathsf{0}^{k-m}\mathsf{1}^m$. Now we apply $\mathcal{A}_n$ to the second layer pairs in $e(D)$ and store the values into M_n as third layer vertices. Note that if $k > 1$, then there are still some second layer pairs for which the value of $\mathcal{O}$ has not yet been defined. Note also that all non-leaf

vertices defined thus far are different and hence to conclude the definition of $\mathcal{O}_n^S$, we define (in arbitrary way) the values of other vertices (not yet defined) so that all non-leaf vertices are different and hence $\mathcal{O}_n^S$ is injective.

As said above, for every n we choose $\mathcal{O}_n^P$ uniformly at random from the set of all functions $P \to \{0,1\}^n$. Now we can do it because P is fixed after the procedure above. Like in Lemma 5, we can show in a similar fashion that $\mathcal{O}$ is collision resistant but not chain-resistant, because $\mathcal{O}_{4k}$ can be used to find chains for $\mathcal{O}_k$ (for any $k > 0$) and therefore also a time-stamping scheme that uses $\mathcal{O}$ as a hash function (and (1) for verification) is insecure.

Corollary 2. *Semi black-box self reductions cannot prove that collision-resistance of h implies chain-resistance of h.*

6 Discussion and Open Problems

More Efficient Reductions. The reduction established in the proof of Theorem 1 does not give sufficient security guarantees for practical time-stamping schemes. To show this, assume that $k = 160$ (output size of SHA-1) and that there is an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ with running time $T(k) = 2^{16}$ and with success probability $\delta(k) = 2^{-16} \approx 1/65000$. Hence, the time-success ratio is $T(k)/\delta(k) = 2^{32}$. If the time unit denotes the time elapsed for one hash operation and a computer performs 10,000 hash operations per second, then $T(k)$ is about six seconds. For practical time-stamping schemes, an attack with such ratio is considered very serious. Now let us examine the consequences of Theorem 1. Assume that the collision-finding adversary A' is implemented very efficiently, so that $T'(k) = 2T(k)$. By Lemma 2, the time-success ratio of A' is $\frac{T'(k)}{\delta'(k)} \approx \frac{2 \times T(k)}{\frac{\delta^2(k)}{T^2(k)} - 2^{\mathsf{H}_2(\mathcal{D})}} \geq \frac{2T^3(k)}{\delta^2(k)} = 2^{81}$, which is close to the *birthday barrier* and says nothing essential about security – any 160-bit hash function can be broken with that amount (2^{81}) of computational resources. Hence, even the highest security of h does not exclude the attacks with ratio 2^{32}. The reduction gives practical security guarantees only in case $k > 400$, which is much larger than used in the existing schemes. Therefore, it would be very desirable to find more efficient reductions, say the *linear-preserving* ones [10], in which $\frac{T'(k)}{\delta'(k)} = k^{O(1)} \cdot \frac{T(k)}{\delta(k)}$.

Constructions of Chain-Resistant Hash Functions. We leave open the existence of efficient constructions of chain-resistant hash functions, possibly as atomic primitives. While we proved that collision-resistance does not imply chain-resistance, it is still unknown whether there exist more general black-box constructions ($g = P^h$) of chain-resistant hash functions (g) based on a collision-resistant one (h). In case such constructions exist, it would be sufficient to just replace the hash functions in the existing schemes.

Another interesting research topic is attempts at the opposite: to prove that there exist no general black-box constructions of chain-resistant hash-functions based on collision-resistant ones. It would be sufficient to find an oracle $\mathcal{O}$ relative

to which there exist collision-resistant hash functions while no function is chain-resistant. Inspired by the work of Simon [13] it may seem tempting to define an oracle $\mathcal{O}$ capable of finding chains to any computable $f\colon\{0,1\}^{2k} \to \{0,1\}^k$, the description of which is given to $\mathcal{O}$ as an argument. However, there seem to be no obvious ways of doing this. For example, if $\mathcal{O}$ is able to compute the root of the complete Merkle tree M_k^f for any (computable) hash function f, then one can show that such $\mathcal{O}$ can also be "abused" to find collisions for any hash function.

At the same time, it seems very likely that the oracle used by Simon [13] (to prove that collision-resistant hash functions are not black-box constructible from one-way functions) is also sufficient for showing that collision-resistant hash-functions cannot be constructed from the chain-resistant ones.

Stronger Security Conditions. The chain-resistance condition is still too simplistic, considering some scenarios that are very likely to happen in practical implementations of time-stamping schemes. Instead of having unconditional uncertainty about x, it is possible that A_1 has some partial knowledge $y = f(x)$ about x (e.g. ciphertexts or signatures). This suggests a stronger condition:

Definition 6. *A function h is* universally chain-resistant *if for any (probabilistic) function f and for any poly-time adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ with success $\delta = \mathsf{Pr}[x \leftarrow \mathcal{D}, (\mathfrak{R}, a) \leftarrow \mathsf{A}_1(f(x)), (n, z) \leftarrow \mathsf{A}_2(x, a)\colon F_h(x; n; z) \in \mathfrak{R}] = k^{-O(1)}$ there is a poly-time A' with success $\mathsf{Pr}[x \leftarrow \mathcal{D}, x' \leftarrow \mathsf{A}'(f(x))\colon x' = x] = k^{-O(1)}$.*

Loosely speaking, if x can be time-stamped based on $y = f(x)$, then x can be efficiently computed based on y, and hence the time stamp is "legitimate". *This condition implies chain-resistance* if we define $f(x) \equiv 1^k$.

Though the universal chain resistance condition seems natural, it is probably not achievable. To see this, assume that h is one-way even if one of the arguments is revealed to the adversary, i.e. every $\mathsf{A}' \in \mathsf{FP}$ has success

$$\delta'(k) = \mathsf{Pr}[(x, z) \leftarrow \mathcal{U}_{2k}, x' \leftarrow \mathsf{A}'(h(x, z), z)\colon x = x'] = k^{-\omega(1)} \quad . \tag{6}$$

This assumption is intuitively assumed to hold in the case of conventional hash functions. Let f be a probabilistic function such that $(h(x, z), z) \leftarrow f(x)$, where $z \leftarrow \mathcal{U}_k$; and let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be defined as follows: $(\{y\}, z) \leftarrow \mathsf{A}_1(y, z)$, and $(0, z) \leftarrow \mathsf{A}_2(x, z)$. Clearly, the success of A (in the sense of universal chain resistance) is $\delta = 1$, while no adversary A' can efficiently invert f. Therefore, no functions that are one-way in the sense of (6) are universally chain resistant, which means that this is a very strong security requirement. Even if h is defined as a *random oracle*, it is still insufficient for the universal chain-resistance. Nothing changes if the set of valid identifiers is polynomially restricted.

Acknowledgements

The authors are grateful to Matthew Franklin, to Peeter Laud, to Berry Schoenmakers, and to anonymous referees for their valuable remarks and suggestions that helped to improve the quality and readability of the paper, as well as to Estonian Science Foundation for supporting the study.

References

1. Dave Bayer, Stuart Haber, and W.-Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, pp.329-334, Springer-Verlag, New York 1993.
2. Josh Benaloh and Michael de Mare. Efficient broadcast time-stamping. Tech. report 1, Clarkson Univ. Dep. of Mathematics and Computer Science, August 1991.
3. Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-Stamping with Binary Linking Schemes. In *Advances in Cryptology – CRYPTO'98*, *LNCS* 1462, pp. 486-501, 1998.
4. Stuart Haber and W.-Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, Vol. 3, No. 2, pp. 99-111 (1991).
5. Stuart Haber and W.-Scott Stornetta. Secure Names for Bit-Strings. In *ACM Conference on Computer and Communications Security*, pp. 28–35, 1997.
6. Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS 2000, 41st IEEE Symposium on the Foundations of Computer Science*, pp. 325–335, 2000.
7. Susan Rae Hohenberger. The Cryptographic Impact of Groups with Infeasible Inversion. Master Thesis. Massachusetts Institute of Technology. May 2003.
8. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. *Proceedings of 21st Annual ACM Symposium on the Theory of Computing*, 1989, pp. 44 – 61.
9. ISO IEC 18014-3,Time-stamping services – Part 3: Mechanisms producing linked tokens.
10. Michael Luby. *Pseudorandomness and cryptographic applications*. Princeton University Press, 1996.
11. Ralph C. Merkle. Protocols for public-key cryptosystems. *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, pp.122-134, 1980.
12. Alexander Russell. Necessary and sufficient conditions for collision-free hashing. *Journal of Cryptology* (1995) 8: 87–99.
13. Daniel Simon. Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT'98*, *LNCS 1403*, pp.334–345. Springer-Verlag, 1998.

A Proof of Lemma 2

Let $\mathsf{Pr}[\mathfrak{R}, \mathfrak{N}, a] = \mathsf{Pr}[(\overline{\mathfrak{R}}, \overline{\mathfrak{N}}, \overline{a}) \leftarrow \mathsf{A}_1(1^k) : \overline{\mathfrak{R}} = \mathfrak{R},\ \overline{\mathfrak{N}} = \mathfrak{N},\ \overline{a} = a]$ and

$$\mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R}, \mathfrak{N}, a] = \mathsf{Pr}[X \leftarrow \mathcal{D}, (n, z) \leftarrow \mathsf{A}_2(X, a) : F_h(\mathcal{H}(X); n; z) \in \mathfrak{R},\ n \in \mathfrak{N}] \ .$$

By definition, $\mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R}, \mathfrak{N}, a]$ is the conditional success of $(\mathsf{A}_1, \mathsf{A}_2)$, assuming that A_1 outputs $(\mathfrak{R}, \mathfrak{N}, a)$. Thus, $\delta(k) = \sum_{\mathfrak{R},\mathfrak{N},a} \mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a]$, where the sum is computed over all possible outputs of $\mathsf{A}_1(1^k)$. Let

$$\mathsf{Pr}[\mathsf{Brk}(\rho,n) \mid \mathfrak{R},\mathfrak{N},a] = \mathsf{Pr}[X \leftarrow \mathcal{D}, (\overline{n}, z) \leftarrow \mathsf{A}_2(X,a) : F_h(\mathcal{H}(X); \overline{n}; z) = \overline{\rho} \in \mathfrak{R}, \overline{n} = n \in \mathfrak{N}]$$

be the conditional probability of success with additional condition that the identifier (output by A_2) is n and the result of the hash chain is $\rho \in \mathfrak{R}$. Now assume that A' has finished and hence the following computations have been performed:

$$(\mathfrak{R},\mathfrak{N},a) \leftarrow \mathsf{A}_1(1^k),\ \begin{matrix} X \leftarrow \mathcal{D} \\ X' \leftarrow \mathcal{D} \end{matrix},\ \begin{matrix} (\overline{n},z) \leftarrow \mathsf{A}_2(X,a) \\ (\overline{n'},z') \leftarrow \mathsf{A}_2(X',a) \end{matrix},\ \begin{matrix} \overline{\rho} = F_h(\mathcal{H}(X);\overline{n};z) \\ \overline{\rho'} = F_h(\mathcal{H}(X');\overline{n'};z') \end{matrix}\ .$$

Let Coll denote the event that A' finds a collision and let Coll' denote the event that A' finds a collision so that $\overline{\rho} = \overline{\rho'} \in \mathfrak{R}$, $\overline{n} = \overline{n'}$. By Lemma 1, the event Coll is a superset of $\mathsf{Coll}' \cap (\mathcal{H}(X)\ /\ \mathcal{H}(X'))$. Hence, the success $\delta'(k) = \mathsf{Pr}[\mathsf{Coll}']$ of A' satisfies

$$\begin{aligned}\delta'(k) &\geq \mathsf{Pr}[\mathsf{Coll}' \cap (\mathcal{H}(X)\ /\ \mathcal{H}(X'))] = 1 - \mathsf{Pr}[(\neg\mathsf{Coll}') \cup (\mathcal{H}(X) = \mathcal{H}(X'))] \\ &\geq 1 - (1 - \mathsf{Pr}[\mathsf{Coll}']) - \mathsf{Pr}[\mathcal{H}(X) = \mathcal{H}(X')] = \mathsf{Pr}[\mathsf{Coll}'] - \mathsf{P}_{\mathrm{C}}(\mathcal{D})\ .\end{aligned}$$

Therefore, it remains to estimate $\mathsf{Pr}[\mathsf{Coll}']$. Let $\mathsf{Coll}'(n,\rho)$ denote the product event $\mathsf{Coll}' \cap (\overline{n} = \overline{n'} = n) \cap (\overline{\rho} = \overline{\rho'} = \rho)$. From the independence of the two runs of A_2 it follows that $\mathsf{Pr}[\mathsf{Coll}'(n,\rho) \mid \mathfrak{R},\mathfrak{N},a] = \mathsf{Pr}^2[\mathsf{Brk}(n,\rho) \mid \mathfrak{R},\mathfrak{N},a]$ and hence,

$$\begin{aligned}&\mathsf{Pr}[\mathsf{Coll}' \mid \mathfrak{R},\mathfrak{N},a] = \sum_{n,\rho} \mathsf{Pr}[\mathsf{Coll}'(n,\rho) \mid \mathfrak{R},\mathfrak{N},a] = \sum_{n,\rho} \mathsf{Pr}^2[\mathsf{Brk}(n,\rho) \mid \mathfrak{R},\mathfrak{N},a] \\ &= \sum_{n,\rho} \mathsf{Pr}^2[\mathsf{Brk}|\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}^2[n,\rho|\mathfrak{R},\mathfrak{N},a,\mathsf{Brk}] = \mathsf{Pr}^2[\mathsf{Brk}|\mathfrak{R},\mathfrak{N},a] \cdot \sum_{n,\rho} \mathsf{Pr}^2[n,\rho|\mathfrak{R},\mathfrak{N},a,\mathsf{Brk}] \\ &\geq \mathsf{Pr}^2[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a] \cdot \frac{1}{|\mathfrak{R}| \cdot |\mathfrak{N}|} \geq \mathsf{Pr}^2[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a] \cdot \frac{1}{T^2(k)}\ ,\end{aligned}$$

where the first inequality holds because $\sum_{n,\rho} \mathsf{Pr}[n,\rho \mid \mathfrak{R},\mathfrak{N},\mathsf{Brk},a] = 1$ and for any probability space $\mathcal{X}$, we have $\sum_{x\in\mathcal{X}} \overset{2}{\mathsf{Pr}}[x] \geq \frac{1}{|\mathcal{X}|}$. The second inequality follows from the observation that $\mathfrak{R}$ and $\mathfrak{N}$ are produced by the adversary and hence their size cannot exceed the running time. Therefore,

$$\mathsf{Pr}[\mathsf{Coll}'] = \frac{\sum_{\mathfrak{R},\mathfrak{N},a} \mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}^2[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a]}{T^2(k)} \geq \frac{\left(\mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a]\right)^2}{T^2(k)} = \frac{\delta^2(k)}{T^2(k)},$$

which follows from the Jensen inequality. □

Strong Conditional Oblivious Transfer and Computing on Intervals

Ian F. Blake[1] and Vladimir Kolesnikov[2]

[1] Dept. Elec. and Comp. Eng, University of Toronto, Toronto, ON, M5S 3G4, Canada
ifblake@comm.utoronto.ca
[2] Dept. Comp. Sci., University of Toronto, Toronto, ON, M5S 3G4, Canada
vlad@cs.utoronto.ca

Abstract. We consider the problem of securely computing the Greater Than (GT) predicate and its generalization – securely determining membership in a union of intervals. We approach these problems from the point of view of Q-Conditional Oblivious Transfer (Q-COT), introduced by Di Crescenzo, Ostrovsky and Rajagopalan [4]. Q-COT is an oblivious transfer that occurs *iff* predicate Q evaluates to `true` on the parties' inputs. We are working in the semi-honest model with *computationally unbounded* receiver.

In this paper, we propose: (i) a stronger, simple and intuitive definition of COT, which we call *strong* COT, or Q-SCOT. (ii) A simpler and more efficient one-round protocol for securely computing GT and GT-SCOT. (iii) A simple and efficient modular construction reducing SCOT based on membership in a union of intervals (UI-SCOT) to GT-SCOT, producing an efficient one-round UI-SCOT.

1 Introduction

This work falls into the area of constructing *efficient* secure multi-party protocols for interesting functionalities. The more basic the functionality, the more common is its use, and the more significant is the value of any improvement of the corresponding protocol. We start with presenting the problems we investigate and their motivation.

The base functionality we consider - Greater Than (GT) - is one of the most basic and commonly used. Secure evaluation of GT is also one of the most famous and well-researched problems in cryptography. There exist a vast number of applications relying on it, such as auction systems or price negotiations.

Another typical example would be secure distributed database mining. The setting is as follows: several parties, each having a private database, wish to determine some properties of, or perform computations on, their *joint* database. Many interesting properties and computations, such as transaction classification or rule mining, involve evaluating a large number of instances of GT [12, 14]. Because of the large size of the databases, even a minor efficiency gain in computing GT results in significant performance improvements.

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 515–529, 2004.

Other functionalities – memberships in a set of intervals and their conjunctions and disjunctions – are less studied, but nevertheless are very useful. Their immediate uses lie in appointment scheduling, flexible timestamp verification, expression evaluation, in the areas of computational geometry, biometrics, and many others. Certain kinds of set membership problems, as studied by Freedman, Nissim and Pinkas [7], can be represented succinctly as instances of problems we consider. For example, the problem of membership in a set consisting of all even integers on a large interval (y, z) can be represented as a conjunction of two small instances of interval memberships ($S = \{x | x_0 < 1 \wedge x \in (y, z)\}$, where x_0 is the low bit of x). In such cases, using our solutions may have significant advantages over the general set intersection solution of [7].

The setting with computationally unbounded receiver (Alice) is very appealing, both for oblivious transfer and general computations. Numerous papers consider unconditional security against one or more parties, in particular, the receiver, e.g. [2, 3, 5, 11, 17]. Practical one-round computation with unbounded first party (Alice) currently seems to be hard to achieve. The best known general approach [21] offers only polynomial efficiency and only for computing NC^1 circuits. At the same time, if Alice is bounded, we could use very efficient Yao's garbled circuit approach ([15, 17, 20, 22]) at the cost linear with the size of the circuit. We solve the posed problems in the difficult setting (unbounded Alice), while achieving performance only slightly worse than the best known approach in the easier (bounded Alice) setting.

1.1 Our Contributions and Outline of the Work

After presenting preliminary definitions and constructions in Sect. 1.2, we start with a discussion of Conditional Oblivious Transfer (COT) (Sect. 2). We wish to strengthen the current definition of [4] in several respects. Firstly, we observe that the definition of [4] does not require the privacy of the sender's private input. Secondly, we propose and justify the "1-out-of-2" Q-COT, where the receiver obtains one of two possible secret messages depending on Q, but without learning the value of Q. This is opposed to the "all-or-nothing" approach of [4] where the receiver receives either a message or nothing, which necessarily reveals the value of Q. Our approach significantly adds to the flexibility of COT functionalities and allows for more powerful compositions of COT protocols. We propose a definition of *strong* conditional oblivious transfer (SCOT) that incorporates the above observations and some other (minor) points.

Then, in Sect. 3, we discuss previous work on the GT problem and present our main tool – an efficient protocol for computing GT-SCOT built from a homomorphic encryption scheme. We exploit the structure of the GT predicate in a novel way to arrive at a solution that is more efficient and flexible than the best previously known (of Fischlin [6]) for our model with unbounded Alice. Additionally, our construction is the first to offer transfer of c-bit secrets, with $c \approx 1000$ for practical applications, at no extra cost, with *one* invocation of the protocol, as opposed to the necessary c invocations of Fischlin's protocol. This results in additional significant efficiency gains.

Then, in Sect. 4, we show how to use the bandwidth of our GT-COT solution and present protocols for efficiently computing SCOT based on the interval membership (I-SCOT) and SCOT based on the membership in a union of k intervals (k-UI-SCOT). Because of their modularity, these protocols can also be constructed based on Fischlin's solution at the efficiency loss described in the previous paragraph. Because they leak the private inputs of the sender, we do not know of an efficient way to extend solutions of [4] to compute these functionalities. We remark on how to use UI-SCOT to compute the conjunction or disjunction of the memberships in unions of intervals. Finally, we compare and summarize resource requirements of schemes of Fischlin, Di Crescenzo et al., and ours in the Table in Sect. 4.2.

1.2 Definitions and Preliminaries

We start by introducing the necessary terminology and notation, and refer the reader to Goldreich [9] for in-depth discussion. We are working in a setting with two semi-honest participants, who use randomness in their computation. By a two-party *functionality* we mean a possibly random process that maps two inputs to two outputs. We denote the *view* (i.e. its randomness, input and messages received) of a party P executing a protocol Π with a party R on respective inputs x and y by $\mathrm{VIEW}_P^{\Pi}(x, y)$. We note that $\mathrm{VIEW}_P^{\Pi}(x, y)$ is a random variable over the random coins of P and R.

We stress that although our constructions and analysis are presented for a fixed security and correctness[1] parameters ν and λ, we have in mind their asymptotic notions. Therefore, for example, when talking about a view of a party $\mathrm{VIEW}_P^{\Pi}(x, y)$, we mean an ensemble $\{\mathrm{VIEW}_P^{\Pi}(x, y)\}_{\nu,\lambda}$ of views.

We denote statistical closeness of ensembles of random variables X and Y by $X \stackrel{s}{\equiv} Y$ and their computational indistinguishability by $X \stackrel{c}{\equiv} Y$. We say a function $\mu : N \to R$ is *negligible* if for every positive polynomial $p(\cdot)$ there exists an N, such that for all $n > N, \mu(n) < 1/p(n)$. We say a probability is *overwhelming* if it is negligibly different from 1.

Homomorphic Encryption. Our constructions use semantically secure public key probabilistic *additive homomorphic encryption*. Informally, a scheme is probabilistic (or *randomized*), if its encryption function uses randomness to encrypt a plaintext as one of many possible ciphertexts. It allows *re-randomization* if a random encryption of a plaintext can be computed from its ciphertext and the public key. In our work, we will rely on the unlinkability of encryptions of the same message. An encryption scheme (G, E, D) is *homomorphic*, if for some operations $\oplus$ and $\otimes$ (defined on possibly different domains), it holds that $D(E(x) \oplus y)) = D(E(x) \otimes E(y))$. A scheme is called additively (multiplicatively) homomorphic if it is homomorphic with respect to the corresponding operation (e.g. additive scheme allows to compute $E(x + y)$ from $E(x)$ and $E(y)$). Many of the commonly used schemes are homomorphic. For example, the ElGamal scheme is multiplicatively homomorphic, and Goldwasser-Micali [10] and Pail-

[1] Correctness parameter specifies the allowed probability of error in the protocols.

lier [18] schemes are additively homomorphic. Unfortunately, it is not known whether there exists a scheme that is algebraically (i.e. both additively and multiplicatively) homomorphic. We note that an additively homomorphic scheme allows multiplication by a known constant, i.e. computing $E(cx)$ from $E(x)$ and c, via repeated addition.

The Paillier Cryptosystem. Our protocols require an additional property of the encryption scheme: the large plaintext size, or *bandwidth*. The Paillier scheme [18] satisfies all our requirements, and we will instantiate all our protocols with it. We present it for completeness, but omit the number-theoretic justification.

Key generation: Let N be an RSA modulus $N = pq$, where p, q are large primes. Let g be an integer of order $N\alpha$ modulo N^2, for some integer α. The public key $pk = (N, g)$ and the secret key $sk = \lambda(N) = \text{lcm}((p-1),(q-1))$, where $\lambda(N)$ is the Carmichael's lambda function.

Encryption: to encrypt $m \in \mathbb{Z}_N$, compute $\text{Enc}(m) = g^m r^N \mod N^2$, where $r \in_R \mathbb{Z}_N^*$.

Decryption: to decrypt a ciphertext c, compute $m = \frac{L(c^{\lambda(N)} \mod N^2)}{L(g^{\lambda(N)} \mod N^2)} \mod N$, where $L(u) = \frac{u-1}{N}$ takes as input an element from the set $S_N = \{u < N^2 | u = 1 \mod N\}$.

Re-randomization: to re-randomize a ciphertext c, multiply it by a random encryption of 0, i.e. compute $cr^N \mod N^2$, for $r \in_R \mathbb{Z}_N^*$.

The underlying security assumption is that the so-called composite residuosity class problem is intractable (called the CCRA assumption). It it potentially stronger than the RSA assumption, as well as the quadratic residuosity assumption, used in [6]. We refer the interested reader to [18] for further details.

2 Strong Conditional Oblivious Transfer

The notion of COT was introduced by Di Crescenzo, Ostrovsky and Rajagopalan [4] in the context of timed-release encryption. It is a variant of Oblivious Transfer (OT) introduced by Rabin [19]. Intuitively, in COT, the two participants, a receiver R and a sender S, have private inputs x and y respectively, and share a public predicate $Q(\cdot,\cdot)$. S has a secret s he wishes (obliviously to himself) to transfer to R iff $Q(x,y) = 1$. If $Q(x,y) = 0$, no information about s is transferred to R. R's private input and the value of the predicate remain computationally hidden from S.

2.1 Our Definitions

We start by describing several ways of strengthening the existing definition with the goal of increasing modularity and widening the applicability of SCOT protocols. Our own construction for UI-SCOT, for example, requires its building blocks to have the proposed features.

First, while sufficient for the proposed timed-release encryption scheme, the definition of [4] lacks the requirement of secrecy of the sender's private input. We would like the new definition to include this requirement.

Secondly, we prefer the "1-out-of-2" approach. In our proposed setting, the sender possesses two secrets s_0 and s_1, and wishes (obliviously to himself) to send s_1 if $Q(x, y) = 1$, and to send s_0 otherwise. Unlike the COT "all-or-nothing" definition, this allows SCOT protocols to have the property of *not revealing* $Q(x, y)$ *to the receiver*. This proposal strengthens the definition since while a SCOT protocol can be trivially modified to satisfy COT definitions of [4], the opposite does not (efficiently) hold[2]. Further, note that it follows from our requirements that a Q-SCOT protocol can be trivially modified into a $(\neg Q)$-SCOT protocol. This also does not hold for COT. We will use this important property in our constructions later in the paper.

Finally, as a minor point, we only require statistical, as opposed to perfect, correctness and security against R, to allow for easier analysis of the protocols and wider applicability of the SCOT notion.

We now present our definition. Let sender S and receiver R be the participants of the protocol. Let ν be the security parameter and λ be the correctness parameter, upperbounding error probability by $O(2^{-\lambda})$. Let D_I and D_S be the respective domains of parties' private inputs and sender's secrets. Let $d_I = |D_I|$ and $d_S = |D_S|$. We assume that both domains are known to both parties. Let R have input $x \in D_I$, and S has input $(y \in D_I, s_0, s_1 \in D_S)$. Let $Q : D_I \times D_I \mapsto \{0,1\}$ be a predicate. Consider the SCOT functionality:

Functionality 1

$$f_{Q-\mathrm{SCOT}}(x, (y, s_0, s_1)) = \begin{cases} (s_1, \textit{empty string}) & \textit{if } Q(x, y) = 1, \\ (s_0, \textit{empty string}) & \textit{otherwise} \end{cases} \tag{1}$$

There are many models in which we can consider computing this functionality. Each of the two parties may be malicious or semi-honest and each party may or may not be computationally limited[3]. We wish to give one definition that refers to all possible models and rely on existing definitions of secure computations in these models. We refer the reader to Goldreich [9] for in-depth presentations of definitions of security in many interesting models.

Definition 1. *(Q-Strong Conditional Oblivious Transfer)*
We say that a protocol Π is a Q-strong conditional oblivious transfer *protocol with respect to a given model, if it securely implements functionality $f_{Q-\mathrm{SCOT}}$ (1) in the given model.*

We note that this general definition covers the case when Q is probabilistic.

[2] Clearly, because secure multi-party computation can be based on OT (Kilian [13]), COT implies SCOT. This solution, however, is inefficient.

[3] Of course, in some of the combinations it is not possible to have nontrivial secure SCOT protocols, such as when both parties are computationally unlimited.

One of the more practical and interesting settings is the model with the semi-honest unlimited receiver, semi-honest polytime sender and deterministic Q. We discuss our constructions in this model, and thus wish to explicate the definition for this setting.

Definition 2. *Let receiver R, sender S, their inputs x and y, secrets s_1 and s_0, unary parameters ν and λ, and predicate Q be as discussed above. We say that Π is a* strong conditional oblivious transfer *protocol for predicate Q in the semi-honest model with computationally unlimited receiver and polytime sender if*

- Transfer Validity. *With overwhelming probability in λ: If $Q(x, y) = 1$, R obtains s_1, otherwise R obtains s_0.*
- Security against R. *(R obtains essentially no information other than the transferred secret) There exists a simulator Sim_R, such that for any x, y, s, s' from appropriate domains:*

$$\text{if } Q(x,y) \text{ then } \{\mathrm{Sim}_R(x,s)\}_\nu \stackrel{s}{\equiv} \{\mathrm{VIEW}_R^\Pi(x,(y,s',s))\}_\nu$$
$$\text{if } \neg Q(x,y) \text{ then } \{\mathrm{Sim}_R(x,s)\}_\nu \stackrel{s}{\equiv} \{\mathrm{VIEW}_R^\Pi(x,(y,s,s'))\}_\nu$$

- Security against S. *(S gets no efficiently computable information about x) There exists an efficient simulator Sim_S, such that for any $x, (y, s_0, s_1)$ from appropriate domains:*

$$\{\mathrm{Sim}_S(y,s_0,s_1)\}_\nu \stackrel{c}{\equiv} \{\mathrm{VIEW}_S^\Pi(x,(y,s_0,s_1))\}_\nu.$$

As further justification, we wish to point out an interesting use of Q-SCOT protocols. When sufficiently long secrets are chosen randomly by S, upon completion of a Q-SCOT protocol, R does not know either the value of Q, or the non-transferred secret. Thus this can be viewed as a convenient way to share the value of Q among R and S. Further, the secret that R received may serve as a proof to S of the value of Q. This is not possible with COT, as R is only able to provide such proof if $Q(x, y) = 1$.

3 The GT-SCOT Protocol

Research specifically addressing the GT problem is quite extensive. It was considered (as a special case) in the context of general secure multi-party computation [1, 15, 17, 20, 23, 22], whose solution is now well-known and celebrated. This general approach is impractical. However, because the circuit for computing GT is quite small, it is the best currently known one-round solution in the model with the computationally bounded Alice. As people searched for efficient solutions to special classes of problems in different models, more efficient GT solutions implicitly appeared. Naor and Nissim [16] presented a general approach to securely computing functions with low communication overhead. While the application of their solution to GT is quite efficient in the message length, it needs at least $O(\log n + \log \frac{1}{\epsilon})$ 1-out-of-$O(n)$ oblivious transfers and the same

number of rounds, where ϵ is the tolerated probability of error. Sander, Young and Yung [21] showed that all functionalities in NC^1 (including GT) can be computed by a one-round polytime protocol. Their solution is secure against unbounded Alice. Unfortunately, when used with the natural shallow GT circuit[4] (which seems to be optimal for their approach), it requires at least n^4 modular multiplications and $n^4 \log N$ communication (where n is the input size, and N is the GM modulus used).

Finally, in 2001, Fischlin [6] proposed a solution that significantly reduced the number of modular multiplications, while also reducing the message size and maintaining the minimal one-round efficiency. This is the best previously known solution to the GT problem in the model with unbounded Alice. The number of modular multiplications required to complete his protocol is $8n\lambda$, where $2^{-\lambda}$ is the allowed error probability. The message complexity (in bits) is $n \log N(\lambda + 1)$. Fischlin also extends this protocol (at the cost of approximately doubling the communication and computation costs) to satisfy our definition of GT-SCOT, with the exception of leaking the value of the predicate. We remark that this extension can be further extended to fully satisfy our definitions at the expense of further approximately doubling the communication and computation costs.

3.1 Our Construction

Our constructions use semantically secure additively homomorphic encryption schemes with large message domains. For the ease and clarity of presentation and to enable resource analysis, we "instantiate" our protocols with the original Paillier scheme. We remark that the Paillier scheme has received much attention in the literature recently, and several variants, including an elliptic curve version [8], have appeared. Using more efficient implementations may further improve our results.

Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be the instance generation, encryption and decryption algorithms, respectively, of such a scheme. As in Definition 2, let R and S be the receiver and the sender with inputs x and y respectively and common parameters ν and λ. Let $x, y \in D_I$ and $s_0, s_1 \in D_S$. Let $d_S = |D_S|$ and, without loss of generality, $d_I = |D_I| = 2^n$.

Throughout this section, we will work with numbers which we will need to represent as binary vectors. For $x \in \mathbb{N}$, unless specified otherwise, x_i will denote the i^{th} most significant bit in the n-bit binary representation of x, including leading zeros, if applicable. Where it is clear from the context, by x we may mean the vector $< x_1, x_2, ..., x_n >$, and by $\mathrm{Enc}(x)$ we mean a vector $< \mathrm{Enc}(x_1), \mathrm{Enc}(x_2), ..., \mathrm{Enc}(x_n) >$. We will also write $\mathrm{Enc}(x)$ instead of $\mathrm{Enc}_{pk}(x)$, where pk is clear from the context.

For the clarity of presentation, we describe the setup phase outside of the protocol. We stress that it is run as part of R's first move, and in particular, *after* the parties' inputs x, and (y, s_0, s_1) have been fixed.

[4] The circuit based on the formula used by Fischlin's protocol [6].

Setup Phase. R sets up the Paillier encryption scheme with group size $N = pq$ by running Gen and generating secret and public keys (sk and pk). He chooses the number of bits in N to be $\max\{\nu, |d_S| + \lambda\}$.

We will view D_S as a subset of $\mathbb{Z}_N$, and will perform operations on elements of D_S modulo N.

Observation 1. *We envision the following practical parameter choices for our GT protocols. First, choose N and λ to satisfy the security and correctness requirements of the encryption scheme. In practice, $\log N (\approx 1000) \gg \lambda (\approx 40..80)$, so we set $|d_S| = \log N - \lambda > 900$ bits of the bandwidth of the encryption scheme to be used for sending secrets. If D_S needs to be much larger than that, it may be more practical to split it in blocks of size $|d_S|$ and run GT-SCOT several times. Choosing parameters in this manner also simplifies comparison of our results to others, and we follow this approach in Sect. 4.2.*

Observation 2. *There is a negligible (in λ) minority of elements of D_S in the group of size N.*

For our protocols, we are only interested in binary comparisons, i.e. one of $\{>, <, \leq, \geq\}$. We can trivially reduce $\{\geq, \leq\}$ to $\{>, <\}$. Furthermore, we assume that $x \not= y$. This can be enforced by mapping, for instance, $x \mapsto 2x, y \mapsto 2y + 1$. The mapping can be done entirely by S. Similarly, we assume that $s_0 \not= s_1$. The case when $s_0 = s_1$ can be reduced to the $s_0 \not= s_1$ case by, for example, S setting $y = \max\{D_I\}$ and $s_1 \in_R D_S \setminus \{s_0\}$, ensuring that $x < y$ and s_0 is always sent.

We now present the GT-SCOT construction. The intuition behind each step is presented immediately below, in the proof of the corresponding security theorem.

Construction 1. *(Computing Functionality GT-SCOT)*

1. *R runs the setup phase, then encrypts each bit x_i of x with the generated pk and sends $(pk, Enc(x_1), ..., Enc(x_n))$ to S.*
2. *S computes the following, for each $i = 1..n$:*
   (a) *an encryption of the difference vector d, where $d_i = x_i - y_i$.*
   (b) *an encryption of the flag vector f, where $f_i = x_i \ XOR \ y_i = (x_i - y_i)^2 = x_i - 2x_iy_i + y_i$.*
   (c) *an encryption of vector γ, where $\gamma_0 = 0$ and $\gamma_i = 2\gamma_{i-1} + f_i$.*
   (d) *an encryption of vector δ, where $\delta_i = d_i + r_i(\gamma_i - 1)$, where $r_i \in_R \mathbb{Z}_N$.*
   (e) *a random encryption of vector μ, where $\mu_i = \frac{s_1 - s_0}{2}\delta_i + \frac{s_1 + s_0}{2}$*
   and sends a random permutation $\pi(Enc(\mu))$ to R.
3. *R obtains $\pi(Enc(\mu))$, decrypts it, and determines the output as follows: if μ contains a single $v \in D_S$, output v, otherwise abort.*

Theorem 1. *The protocol of Construction 1 is a GT-SCOT protocol in the semi-honest model, assuming semantic security of the employed encryption scheme.*

Proof. (Sketch): We will now show that the protocol correctly computes the desired functionality. It is easy to see that the homomorphic properties of the encryption scheme allow S to perform all necessary operations. In particular, step 2b is possible because y_i are known to S.

Observe that the flag vector f is a $\{0,1\}$-vector, with the ones in positions where x and y differ. Furthermore, γ is a vector with the following structure: it starts with zero or more zeros, then a one, then a sequence of non-ones. Moreover, with overwhelming probability the non-zero elements $(\gamma_i - 1)$ are not multiples of either p or q, i.e. are in $\mathbb{Z}_N^*$. This is because the fraction of multiples of p or q in Z_N is negligible, and p and q are chosen randomly and independently of x and y.

Let ind_1 be the (only) position where $\gamma_{\text{ind}_1} = 1$. This position is where x and y first differ, and thus d_{ind_1} determines $\text{GT}(x, y)$. The transformation $(\gamma, d) \to \delta$ of step 2d randomizes all coordinates of δ, while setting δ_{ind_1} to the value of d_{ind_1}. Because, with overwhelming probability, $(\gamma_i - 1) \in \mathbb{Z}_N^*$, multiplying it by $r_i \in_R \mathbb{Z}_N$ randomizes δ perfectly in $\mathbb{Z}_N$.

With overwhelming probability, the transformation $(\delta, s_0, s_1) \to \mu$ of step 2e is a permutation on $\mathbb{Z}_N$ that maps $-1 \mapsto -s_0, 1 \mapsto -s_1$. Indeed, it is not such a permutation only when $(s_1 - s_0)$ is a multiple of p or q, the event that occurs with negligible probability, because p and q are are chosen randomly and independently of s_1 and s_0. This permutation preserves the randomness properties of all elements of the vector, and (as is easy to verify) performs the mapping we are looking for. The random re-encryption step hides the information that may be contained in the randomness of the encryption. Finally, the random permutation $\pi(\mu)$ of step 2 hides the index of the determining d_i.

It easily follows from Observation 2 that the probability that there is not exactly one element of size $|d_S|$ in the decrypted by R vector, is negligible. Thus, with overwhelming probability, R terminates and outputs the correct value.

Security of R (against the semi-honest S) trivially holds because of the semantic security properties of the employed encryption scheme.

We now prove security of S against an unlimited semi-honest R by constructing a protocol view simulator $\text{Sim}_R(x, s)$, where x is the input, and s is the output of the protocol. $\text{Sim}_R(x, s)$ has to generate a distribution statistically close to the view of R in a real execution - $\text{VIEW}_R(x, (y, s_0, s_1)) = \{x, r, \text{Enc}(\pi(\mu))\}$, where r is the randomness used by R to generate pk and sk (of the setup phase) and the random encryptions of the first message, and $\pi(\mu)$ is defined in the protocol construction. $\text{Sim}_R(x, s)$ proceeds as follows. It first generates a random string r' of appropriate length (to match r). It uses r' to compute the keys sk and pk (including N). It then computes a candidate μ': for $i = 1..n$, pick random $\mu'_i \in_R \mathbb{Z}_N$. It then replaces a random element of μ' with the received s, and outputs $\{x, r', \text{Enc}_{pk'}(\mu')\}$, where $\text{Enc}_{pk'}(\mu')$ is a vector of random encryptions of coordinates of μ' under the pk'. Because of the previously presented arguments of the randomness of all elements of $\pi(\mu)$ (other than the one that carries the secret) and the randomness of re-encryption, it is easy to see that Sim_R generates a distribution statistically close to the view of R. We note that the simulation is not perfect, since the transfer of the other secret is possible during the real execution, with negligible probability. □

We observe that a GT-SCOT protocol, such as presented above, immediately implies solution to GT, in the semi-honest model. Indeed, running GT-SCOT

with at least one of the secrets s_i known to R (say $s_1 = 1$), immediately yields the desired functionality. Moreover, for GT, the transformation of step 2e is unnecessary (while the re-randomization of the same step is still required).

3.2 Resource Analysis

We evaluate the message and modular multiplication efficiency of our construction based on the use of Paillier encryption scheme. We note that we do not include the relatively small computational cost of key generation, to be consistent with the compared results of [4] and [6]. Let n be the length of inputs x and y in binary, N-the size of the plaintext domain of the Paillier scheme. Then message complexity of Construction 1 is $l = 2n \log(N^2) = 4n \log N$ bits.

Let $w = w(y) \leq n$ be the weight (i.e. the number of ones) of the binary representation of y. To encrypt each bit, $\log N$ multiplications are required. Observe that it is not necessary to perform expensive randomized encryption in the intermediate steps of S. This allows us to make do with only w multiplications for each of the steps 2a, 2b, $2n$ - for step 2c, and $(\log N + 2)n$ - for step 2d, and $(|s_i| + \log N)n \leq 2n \log N$ - for step 2e of the protocol. We note that if we do not perform the transformation of step 2e (when, for example, computing GT), we only need $n \log N$ multiplications for the last step.

Decryption takes $2n \log N$ multiplications. Thus, in total, the protocol requires no more than $(5n+1) \log N + 6n$ modular multiplications ($(4n+1) \log N + 6n$ for GT). We stress that transferring up to $\log N - \lambda$ bit secrets requires the same resources. We observe that the encryption and re-encryption multiplications can be precomputed once the encryption scheme is initialized.

We now compare the efficiency of our approach to that of Fischlin [6], using appropriate parameters. We first note that in practice, no known attack on the Paillier system is better than factoring the modulus N. Clearly, factoring based attacks would also be effective against the GM scheme with the same modulus size. Thus, having already assumed CCRA (see Sect. 1), we also assume that the security of Paillier and GM schemes with the modulus of the same size are approximately the same.

Compared with [6], our scheme offers a factor of $\lambda/4$ improvement in message complexity: $((4n \log N)$ vs $(n \log N(\lambda + 1))$ bits). We pay higher cost in the number of modular multiplications: $((4n+1) \log N + 6n)$ vs $(6n\lambda)$. Additionally, our multiplications are four times slower, since we are working with modulus length twice that of the Goldwasser-Micali encryption scheme employed in [6]. These comparisons are summarized in the Table in Sect. 4.2.

4 SCOT for Unions of Intervals

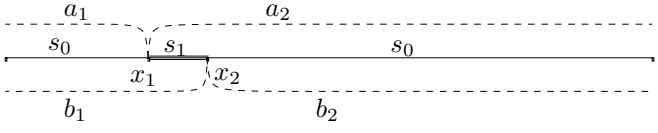
In this section we present new efficient protocols for I-SCOT (SCOT based on the membership in an interval) and UI-SCOT (SCOT based on the membership in a union of intervals), both of which are generalizations of GT-SCOT. We build these protocols on our GT-SCOT solution. While other GT-SCOT approaches (such as based on Fischlin's protocol) are also suitable for these constructions,

our solution is simpler and produces more efficient protocols in terms of both multiplication and communication complexity. In our constructions, we denote the instance of the Q-SCOT functionality with the secrets s_0, s_1 on parties' inputs x, y by Q-SCOT $(s_1|s_0?Q(x,y))$.

In Sect. 4.1 we show how to reduce UI-SCOT to I-SCOT and I-SCOT to GT-SCOT. In our model, secure reductions provide us with secure protocols when the underlying oracles are replaced by their secure implementations (see Goldreich [9] for the composition theorem.) Furthermore, in our model the oracles' implementations may be run in parallel, which, with our implementations, provides secure one-round protocols for I-SCOT to UI-SCOT.

4.1 The UI-SCOT Protocol

Without loss of generality, we assume that the domain of secrets D_S is an additive group[5] $\mathbb{Z}\!\!\mathbb{Z}^+_{d_S}$. All additions of secrets will be done in D_S, unless specified otherwise. In the I-SCOT setting, S's input $x_1, x_2 \in D_I$ represents an interval I, and s_1 (resp. s_0) are to be obliviously transferred if $x \in I$ (resp. $x \notin I$), for R's input $x \in D_I$. The following diagram illustrates the idea of the reduction of I-SCOT to GT-SCOT:



Interval I splits D_I in three parts, and S wishes to transfer s_1 "on the central part" (I) and s_0 "on the side parts" $(D_I \backslash I)$. The idea is to represent these secrets as sums of independently random (i.e. random if taken separately) elements $(a_1, a_2, b_1, b_2 \in D_S)$ which are to be transferred using GT-SCOT.

Construction 2. *(Reducing I-SCOT to GT-SCOT)*

1. *S randomly chooses $a_1 \in D_S$ and sets $b_1, a_2, b_2 \in D_S$ to satisfy $s_0 = a_1 + b_1 = a_2 + b_2$ and $s_1 = a_2 + b_1$*
2. *-Reduction: R and S (in parallel) invoke oracles for GT-SCOT$(a_1|a_2?x < x_1)$ and GT-SCOT$(b_1|b_2?x < x_2)$.*
3. *R obtains $a', b' \in D_S$ from GT-SCOT oracle executions and outputs $a' + b'$.*

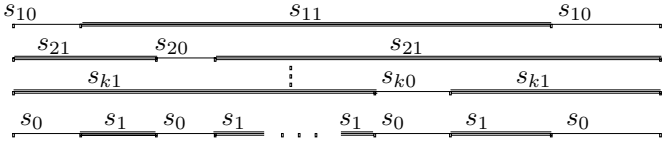
Theorem 2. *The protocol of Construction 2 securely reduces functionality I-SCOT to GT-SCOT in the semi-honest model.*

Proof. (sketch): The transfer validity property of this reduction trivially holds. Since S does not receive any messages from R or oracle executions, the reduction is secure against semi-honest S. We show how to construct Sim_R, simulating the

[5] We stress that we use GT-SCOT as black box, and, in particular, addition in D_S is unrelated to the corresponding operation in the GT-SCOT implementation.

following ensemble (view of R): $\text{VIEW}_R(x,(x_1,x_2,s_0,s_1)) = \{x, r_1, r_2\}$, where r_1, r_2 are the sent (via the GT-SCOT oracles) a_i, b_j. Let s be the transferred secret. Then $\text{Sim}_R(x,s) = \{x, r'_1, r'_2\}$, where r'_i are independently random elements of D_S that sum up to s. Because, by construction, r_1, r_2 are also independently random with the same sum, Sim_R perfectly simulates view of R. □

We now wish to reduce UI-SCOT of polynomially many intervals to I-SCOT. Here, S's input represents a set of *disjoint* intervals $\{I_i = (x_{i1}, x_{i2} \in D_I)\}$, and the secrets $s_0, s_1 \in D_S$. S wishes to transfer s_1 if $x \in \bigcup I_i$, and transfer s_0 otherwise. Let k be the number of intervals in the set (to avoid leaking k to R, S can pad it to a known upper bound by adding empty intervals). We represent $\bigcup I_i$ as the intersection of one "regular" and $k-1$ "cutout" intervals as illustrated on the following diagram.



The bottom line represents the input set of intervals on the domain, and all other lines represent the constructed (by S) intervals that together correspond to this set. The s_i are the secrets to be transferred by the UI-SCOT construction, and the s_{ij} are the intermediate secrets to be created by UI-SCOT and transferred by the existing I-SCOT protocol. Because the input intervals are disjoint, the cut out (thin, on the diagram) parts of the constructed intervals do not intersect, and thus any x either belongs to all or to all but one constructed intervals.

To reduce UI-SCOT to I-SCOT, we need to choose $s_{ij} \in D_S$ based on the given s_i. Because of the above observation we only need to satisfy the following: $s_1 = \sum_i s_{i1}$ and $s_0 = (\sum_{i \neq j} s_{i1}) + s_{j0}, \forall j = 1..k$ Observe that the second condition is equivalent to requiring $s_1 - s_0 = s_{j1} - s_{j0}, \forall j = 1..k$.

Construction 3. *(Reducing UI-SCOT to I-SCOT)*

1. *S chooses $s_{11}, ..., s_{(k-1)1} \in_R D_S$ and sets $s_{k1} = s_1 - \sum_{i=1..k-1} s_{i1}$ and $s_{i0} = s_{i1} - (s_1 - s_0), i = 1..k$.*
2. *-Reduction: S and R (in parallel) invoke oracles for I-SCOT$(s_{i1}|s_{i0}?x \in I_i)$, for each $i = 1..k$.*
3. *R obtains $a_1, ..., a_k \in D_S$ from k oracle executions and outputs $\sum_i a_i$.*

Theorem 3. *The protocol of Construction 3 securely reduces functionality UI-SCOT to I-SCOT in the semi-honest model.*

Proof. (Sketch): The transfer validity property of this reduction trivially holds. Since S does not receive any messages from R or oracle executions, the reduction is secure against semi-honest S. We show how to construct Sim_R simulating the view of R $\text{VIEW}_R(x,y) = \{x, r_1, ..., r_k\}$, where $r_1, ..., r_k$ are the oracle sent

elements of D_S defined by step 1 of the construction. Let s be the transferred secret. Then $\mathrm{Sim}_R(x, s) = \{x, r'_1, ..., r'_k\}$, where $r'_i \in_R D_S$ with the restriction $s = \sum_i r_i$. Sim_R perfectly simulates view of R because both ensembles are $(k-1)$-wise independent random numbers that sum up to the same value s. □

The $(\bigwedge_i Q_i(x_i, y_i))$-COT Protocol. We now build $\bigwedge_i Q_i(x_i, y_i))$-COT (in the sense of [4]) using oracles for corresponding Q_i-SCOT. R now has input $x_1, ..., x_n$, and S has $y_1, ..., y_n$. S wishes to send a secret s to R iff $\bigwedge_i (Q_i(x_i, y_i)) = 1$. The idea is to introduce "specialness" of s like we did for GT-SCOT, by, for example, extending the domain of secrets D_S to group $D'_S = \mathbb{Z}^+_{d'_S}$, where $d'_S = |D'_S| \gg |D_S|$, Then S represents $s \in D_S$ as a sum of random secrets $s_i \in_R D'_S$, and runs Q_i-SCOT$(s_i | r_i ? Q_i(x_i, y_i))$, where $r_i \in_R D'_S$. Indeed, if the conjunction holds, then only the s_i's will be transferred, and they will sum up to $s \in D_S$. If any (or any number of) predicates do not hold, one (or more) r_i will be transferred, which will randomize (in D'_S) the sum obtained by R.

Construction 4. *(Reducing $(\bigwedge_i Q_i(x_i, y_i))$-COT to Q_i-SCOT)*

1. *S chooses $r_1, ..., r_n, s_1, ..., s_{n-1} \in_R D'_S$ and sets (in D'_S) $s_n = s - \sum_{i=1..n-1} s_i$.*
2. *R and S in parallel invoke oracles for Q_i-SCOT$(s_i | r_i ? Q_i(x_i, y_i))$, $\forall i = 1..n$.*
3. *R obtains $a_1, ..., a_n \in D'_S$ from the Q_i-SCOT oracle executions and sets $v = \sum_i a_i$. R outputs v, if $v \in D_S$, and outputs $\perp$ otherwise.*

Theorem 4. *The protocol of Construction 4 securely reduces functionality $(\bigwedge_i Q_i(x_i, y_i))$-COT to Q_i-SCOT in the semi-honest model.*

Proof: The simple proof is very similar to the previous ones and is omitted. □

Corollary 1. *There exists (via construction 4 and DeMorgan laws) efficient one-round protocols for computing conjunction and disjunction of memberships in sets of intervals, secure against computationally unlimited R.*

4.2 Resource Analysis

We continue and expand the resource analysis of Sect. 3.2. Recall that λ and ν are the correctness and security parameters. As discussed in Observation 1, we choose $\nu = \log N$ and λ as in [6]. This determines the secrets domain D_S to be of size $2^{\nu-\lambda}$. As noted in Sect. 3.2, we do not include the cost of key generation in any of the compared solutions.

It is easy to see that Construction 3 makes $2k$ calls to the underlying λ-bit GT-COT oracle. Thus, when using our implementation of GT-SCOT, UI-SCOT requires sending $8kn \log N$ bits and performing about $40kn \log N$ multiplications in group of size N. Using λ-bit GT-SCOT oracle implementation based on Fischlin's GT results in almost full factor of $2k$ blowup in communication since server sends most of the traffic. The $2k$ factor blowup in the computation also seems necessary when using this scheme.

The following table summarizes the cost of comparable modular multiplications and communication of our protocol in relation to others.

Protocol	GT predicate		c-bit GT-SCOT, $c<\nu$-λ		k-UI-SCOT	
	mod. mult.	comm.	mod. mult.	comm.	mod. mult.	comm.
of [6]	$8n\lambda$	$\lambda n \log N$	$32nc\lambda$	$4nc\lambda \log N$	$64kn\lambda^2$	$8kn\lambda^2 \log N$
of [4]	$8n$	$4n \log N$	N/A	N/A	N/A	N/A
our work	$16n \log N$	$4n \log N$	$20n \log N$	$4n \log N$	$40kn \log N$	$8kn \log N$

We see no obvious way to transform the schemes of [4] to GT-SCOT, and thus do not include the corresponding resource calculations.

5 Conclusions and Future Work

We presented simple, intuitive and stronger definitions for Q-SCOT. We presented a flexible and efficient scheme for securely computing the GT predicate and GT-SCOT, in the semi-honest setting with unbounded receiver. We then showed simple modular reductions from UI-SCOT to GT-SCOT. In addition to the presented results, we noticed that natural efficient variants of our protocols are resilient to several natural attacks by malicious receivers. Devising versions of our protocols secure in the malicious model is an interesting aspect of further consideration.

Acknowledgments. The second author would like to thank Travis Gagie, Steven Myers, and especially Charles Rackoff for many insightful discussions. He also thanks Marc Fischlin and Pascal Paillier for useful comments relating to their schemes used in this paper.

References

1. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 503–513, 1990.
2. Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Muller. One-round secure computation and secure autonomous mobile agents. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, 2000.
3. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. CRYPTO 87*, pages 462–462. Springer-Verlag, 1988. Lecture Notes in Computer Science, vol. 293.
4. G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and time-released encryption. In *Proc. CRYPTO 99*, pages 74–89. Springer-Verlag, 1999. Lecture Notes in Computer Science, vol. 1592.
5. Yvo Desmedt. Unconditionally secure authentication schemes and practical and theoretical consequences. In *Proc. CRYPTO 85*, pages 42–55. Springer, 1986. Lecture Notes in Computer Science, vol. 218.
6. Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *RSA Security 2001 Cryptographer's Track*, pages 457–471. Springer-Verlag, 2001. Lecture Notes in Computer Science, vol. 2020.

7. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Proc. EUROCRYPT 2004*, pages 1–19. Springer-Verlag, 2004. Lecture Notes in Computer Science, vol. 3027.
8. Steven D. Galbraith. Elliptic curve paillier schemes. *Journal of Cryptology*, 15(2):129–138, 2002.
9. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.
10. S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 365–377, San Francisco, 1982. ACM.
11. Shai Halevi. Efficient commitment schemes with bounded sender and unbounded receiver. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 12(2):77–89, 1999.
12. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, 2002.
13. J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 20–31, Chicago, 1988. ACM.
14. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Proc. CRYPTO 00*, pages 20–24. Springer-Verlag, 2000. Lecture Notes in Computer Science, vol. 1880.
15. Yehuda Lindell and Benny Pinkas. A proof of yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175, 2004. `http://eprint.iacr.org/`.
16. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 590–599. ACM Press, 2001.
17. Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *1st ACM Conf. on Electronic Commerce*, pages 129–139, 1999.
18. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. EUROCRYPT 99*, pages 223–238. Springer-Verlag, 1999. Lecture Notes in Computer Science, vol. 1592.
19. M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
20. Phillip Rogaway. *The round complexity of secure protocols*. PhD thesis, MIT, 1991.
21. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *Proceedings 40th IEEE Symposium on Foundations of Computer Science*, pages 554–566, New York, 1999. IEEE.
22. A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 160–164, Chicago, 1982. IEEE.
23. A. C. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symp. on Foundations of Comp. Science*, pages 162–167, Toronto, 1986. IEEE.

Improved Setup Assumptions for 3-Round Resettable Zero Knowledge

Giovanni Di Crescenzo[1], Giuseppe Persiano[2], and Ivan Visconti[3]

[1] Telcordia Technologies, Piscataway, NJ, USA
giovanni@research.telcordia.com
[2] Dip. di Inf. ed Appl., Univ. di Salerno, Baronissi, Italy
giuper@dia.unisa.it
[3] Dép. d'Inf., École Normale Supérieure, Paris, France
ivan.visconti@ens.fr

Abstract. In the *bare public-key* model, introduced by Canetti *et al.* [STOC 2000], it is only assumed that each verifier deposits during a set-up phase a public key in a file accessible by all users at all times. As pointed out by Micali and Reyzin [Crypto 2001], the notion of soundness in this model is more subtle and complex than in the classical model. Indeed Micali and Reyzin have introduced four different notions which are called (from weaker to stronger): one-time, sequential, concurrent and resettable soundness. In this paper we introduce the *counter public-key model* (the cPK model for short), an augmentation of the bare public-key model in which each verifier is equipped with a counter and, like in the original bare public-key model, the key of the verifier can be used for any polynomial number of interactions with provers. In the cPK model, we give a *three-round concurrently-sound resettable zero-knowledge* argument of membership for NP. Previously similar results were obtained by Micali and Reyzin [EuroCrypt 2001] and then improved by Zhao *et al.* [EuroCrypt 2003] in models in which, roughly speaking, each verifier is still equipped with a counter, but the key of the verifier could only be used for a fixed number of interactions.

1 Introduction

The notion of Zero Knowledge, put forth by Goldwasser, Micali and Rackoff [1], has proved to be a fundamental concept in the area of complexity-based cryptography. The original notions of security with respect to malicious provers (formalized by the *soundness* requirement) and the security with respect to malicious verifiers (captured by the *zero-knowledge* requirement) only considered a prover and a verifier acting in isolation. Recently, the case in which provers and verifiers are part of a large system (and thus prover-verifier interactions may be interleaved) has been considered and stronger notions of soundness and zero knowledge have been proposed. In a sequence of papers the notions of concurrent zero knowledge [2] and resettable zero knowledge [3] were introduced and protocols in the standard model were provided [4, 5, 6, 3].

P.J. Lee (Ed.): ASIACRYPT 2004, LNCS 3329, pp. 530–544, 2004.

An important measure of an efficiency of a system is the number of rounds needed. Lower bounds for the number of rounds for concurrent and resettable zero knowledge have shown that these strong notions of security cannot be implemented, in the standard model, using a constant number of rounds if black-box zero knowledge is sought [7]. Canetti *et al.* [3] were thus motivated to introduce the *bare public-key* model (the BPK model for short) in which, during a set-up stage, each verifier stores in a public file his public key to be used in all subsequent interactions and keeps secret the associated private key. In this model constant-round concurrent and resettable zero-knowledge arguments for all $\mathcal{NP}$ were shown to exist in [3].

Other models have been proposed in order to achieve similar results by focusing on black-box concurrent zero knowledge, in particular the results of [2, 8, 9] in the timing model, those of [10, 11] in the common reference string model, those of [13] in the preprocessing model, and those of [12] in some partially synchronous model.

Among the different proposed models, the BPK model has the following advantages: 1) it is not based on any trusted third party; 2) no timing assumption is made; 3) the set-up stage is non-interactively performed only by the verifiers. Consequently, the public-key model is, from among the currently proposed models, the one that makes the least set-up assumptions and, in particular, it is weaker than the widely accepted *Public Key Infrastructure* model.

Subsequently to the introduction of the BPK model, Micali and Reyzin [14] noticed that, unlike in the standard model for interactive zero knowledge, distinct notions of soundness arise depending on whether the verifier's public key is used for once (one-time soundness), for polynomially many sequential arguments (sequential soundness), for polynomially many concurrently interleaved arguments (concurrent soundness), or whether the prover is allowed to reset the verifier to a given state during the interaction (resettable soundness). However, they showed that resettably sound zero knowledge cannot be achieved in the black-box model for non-trivial languages. Consequently, for black-box zero knowledge, the strongest possible notion is that of a *concurrently sound resettable zero-knowledge* argument. In [14], Micali and Reyzin showed that in the BPK model, concurrent soundness cannot be achieved in less than four rounds. Moreover they showed that the argument system of Canetti *et al.* presented in [3] is only sequentially sound and the same holds for the four-round resettable zero-knowledge argument presented in [14]. Recently, the existence of a constant-round concurrently-sound resettable zero-knowledge argument in the BPK model has been proved by [15] where a 4-round concurrently-sound resettable zero-knowledge argument in the BPK model has been given for all $\mathcal{NP}$ languages.

Prior to the work of [15], augmented variations of the BPK model had been presented in which constant-round concurrently-sound resettable zero knowledge could be achieved. These proposals are interesting, even in light of the result of [15], since they achieve three-round concurrently-sound resettable zero knowledge which is remarkable as no non-trivial (black-box) zero knowledge can be

achieved in less than three rounds in any variation of the BPK model [14, 16]. In particular, three-round concurrently sound resettable zero-knowledge arguments for all $\mathcal{NP}$ are possible in the *upper-bounded* public-key (UPK, for short) model (see [17]) and in the *weak* public-key (WPK, for short) model (see [18]). In the UPK model, each public key can be used for a fixed number of arguments to be determined during set-up. The verifier is equipped with a counter to keep track of the number of arguments he has been involved in. In the WPK model, instead, there is a fixed upper bound on the number of times the verifier can be involved in sessions regarding the same statement. Thus, also in this model, the verifier must have a counter (actually, one for each possible statement).

Our Contribution. In this paper we introduce the *counter* public-key (cPK for short) model, that is a model weaker than the WPK (and thus of the UPK) model and only slightly stronger than the BPK model. The cPK model, like the UPK of Micali and Reyzin and the WPK of Zhao *et al.*, is an extension of the BPK model in which the verifier is equipped with a counter that, roughly speaking, keeps track of the number of sessions that she has been involved with. However, unlike the UPK model and the WPK model, each public key of the verifier can be used any polynomial number of times exactly like in the original BPK model. Therefore, our proposed model, although slightly stronger than the original BPK model, can be considered much weaker than the UPK model and the WPK model. Indeed, in the cPK model the verifier has no bound on the number of proofs he can engage with the provers while in both UPK and WPK models once the bound is reached, soundness is not guaranteed to hold.

We first present a *three-round concurrently sound resettable zero-knowledge argument of membership* for $\mathcal{NP}$ in the cPK model. This construction improves the previous works of [17, 18] that achieved the same result but in stronger models. We notice that, in the BPK model, concurrent soundness requires 4 rounds. Our protocol is based on the existence of sub-exponentially hard primitives, as in all previous works for obtaining a constant-round resettable zero-knowledge argument in any public-key model.

Our second construction is a *three-round concurrently sound concurrent zero-knowledge argument of knowledge* for all $\mathcal{NP}$ relations in the cPK model under standard intractability assumptions. We notice that, in the black-box model, resettable zero-knowledge arguments of knowledge exist only for trivial languages and thus concurrent zero knowledge is the strongest notion of zero knowledge that can be achieved when arguments of knowledge are sought.

2 The cPK Model

The cPK model assumes that:

1. there exists a public file F that is a collection of records, each containing a public key;
2. an (honest) prover is an interactive deterministic polynomial-time algorithm that takes as input a security parameter 1^n, F, an n-bit string x, such that

$x \in L$ and L is an $\mathcal{NP}$-language, an auxiliary input y, a reference to an entry of F and a random tape;

3. an (honest) verifier V is an interactive deterministic polynomial-time algorithm that works in the following two stages: 1) in a first stage on input a security parameter 1^n and a random tape, V generates a key pair $(\mathtt{pk}, \mathtt{sk})$ and stores $\mathtt{pk}$ in one entry of the file F; 2) in the second stage, V takes as input the private key $\mathtt{sk}$, a counter value c, a statement "$x \in L$" and a random string, then V performs an interactive protocol with a prover , and outputs "accept" or "reject";
4. interactions between prover and verifier start after all verifiers have completed their first stage.

Definition 1. *Given an $\mathcal{NP}$-language L and its corresponding relation R_L, we say that a pair $\langle P, V\rangle$ is* complete *for L, if for all n-bit strings $x \in L$ and any witness y such that $(x, y) \in R_L$, the probability that V interacting with P on input y, outputs "reject"is negligible in n.*

Malicious Provers in the cPK *Model.* We will give argument systems that are sound with respect to s-concurrent malicious provers, for any positive polynomial s. An s-concurrent malicious prover $P^\star$ for the complete pair $\langle P^\star, V\rangle$ is a probabilistic polynomial-time algorithm that takes as input V's public key $\mathtt{pk}$, and, if $P^\star$ is concurrently running i sessions, for $0 < i \leq s(n)$, $P^\star$ can pick a new statement x_{i+1} and a value c_{i+1} of the counter and start a new session with V on input x_{i+1} and c_{i+1}. The only restriction is that, for each value c of the counter, $P^\star$ can only start one session with value c. Also, we allow the malicious prover to schedule his messages in the concurrent sessions in any way he wants and, for each message of $P^\star$, V's reply is immediately received.

We stress here that our definition of malicious prover is the same used by L. Reyzin (see [16]) for the UPK model. Instead, in [18], the value of the counter is assumed to be private to the verifier and the malicious prover has no way of manipulating it. Moreover, we stress that in the cPK model there is no bound on the number of sessions in which the verifier can be involved, thus the model is weaker than the WPK and UPK models and very close to the standard BPK model.

Given an s-concurrent malicious prover $P^\star$ and an honest verifier V, a *concurrent attack* is performed in the following way: 1) the first stage of V is run on input 1^n and a random string so that a pair $(\mathtt{pk}, \mathtt{sk})$ is obtained; 2) $P^\star$ is run on input 1^n and $\mathtt{pk}$; 3) whenever $P^\star$ starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and $\mathtt{sk}$.

Definition 2. *Given a complete pair $\langle P, V\rangle$ for an $\mathcal{NP}$-language L in the* cPK *model, then $\langle P, V\rangle$ is a* concurrently sound *interactive argument system in the* cPK *model for language L if, for all positive polynomial s, for all s-concurrent malicious prover $P^\star$, for any false statement "$x \in L$" the probability that in an execution of a concurrent attack V outputs "accept" for such a statement is negligible in n.*

The strongest notion of zero-knowledge, referred to as resettable zero knowledge, gives to a verifier the ability to reset the prover to a previous state. This is significantly different from a scenario of multiple interactions between prover and verifier since after a reset, the prover uses the same random bits. We now give the formal definition of a black-box resettable zero-knowledge argument system with concurrent soundness for $\mathcal{NP}$ in the counter public-key model.

Definition 3. *An interactive argument system $\langle P, V\rangle$ in the* cPK *model is black-box* resettable zero-knowledge *if for any polynomial $t(\cdot)$, and for any probabilistic adversary $V^\star$ running in time $t(\cdot)$, there exists a probabilistic polynomial-time algorithm S such that for any polynomial $s(\cdot)$, for any $x_1, \cdots, x_{s(n)} \in L$ of length n, the following two distributions are indistinguishable:*

1. *the output of $V^\star$ consisting of a public file F with $s(n)$ entries and the transcript of a polynomial number of (even concurrent) interactions with each $P(x_l, y_l, r_g, F, i)$ where y_l is a witness for "$x_l \in L$", $|x_l| = n$, r_g is a random tape and i specifies an entry of the public file, for $1 \leq i, l, g \leq s(n)$;*
2. *the output of S that has black-box access to $V^\star$ on input $x_1, \ldots, x_{s(n)}$.*

Moreover we define such an adversarial verifier $V^\star$ as an (s,t)-resetting malicious verifier.

3 Cryptographic Tools

We review the cryptographic tools that we will use in our constructions. We start from the notions of an η-secure digital signature scheme and of a γ-secure commitment scheme.

Definition 4. *An* η-secure digital signature scheme $\mathtt{SS}$ *is a triple of algorithms* $\mathtt{SS} = (\mathtt{G}, \mathtt{Sig}, \mathtt{Ver})$ *such that*

1. **Correctness:** *for all messages $m \in \{0,1\}^k$,*

$$Pr[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{G}(1^k); \hat{m} \leftarrow \mathtt{Sig}(m, \mathtt{sk}) : \mathtt{Ver}(m, \hat{m}, \mathtt{pk}) = 1] = 1.$$

2. **Unforgeability:** *for all algorithms A running in time $o(2^{k^\eta})$ it holds that*

$$Pr[(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathtt{G}(1^k); (m, \hat{m}) \leftarrow A^{\mathcal{O}(\mathtt{sk})}(pk) : m \notin \mathtt{Query} \text{ and } \mathtt{Ver}(m, \hat{m}, \mathtt{pk}) = 1]$$

is negligible in k where $\mathcal{O}(\mathtt{sk})$ is a signature oracle *that on input a message returns as output a signature of the message and* $\mathtt{Query}$ *is the set of signature requests submitted by A to $\mathcal{O}$.*

We assume that signatures of k-bit messages produced by using keys with security parameter k have length k. This is not generally true as for each signature scheme we have a constant a such that signatures of k-bit messages have length k^a but this has the advantage of not overburdening the notation. It is understood that all our proofs continue to hold if this assumption is removed.

Standard secure signature schemes exist under the assumption of the existence of one-way functions [19]. In our case we need the existence of functions that are one-way with respect to subexponential-time adversaries.

Definition 5. *An* γ-secure bit commitment scheme *is a pair of algorithms* (Com, Dec) *such that*

1. **Correctness:** *for all* $b \in \{0,1\}$ *and for all* k,
$$Pr[(\mathtt{com}, \mathtt{dec}) \leftarrow \mathtt{Com}(b, 1^k) : \mathtt{Dec}(\mathtt{com}, \mathtt{dec}, b) = 1] = 1;$$
2. **Perfect Binding:** *for all* k, *the set of strings* com *of length* k *for which there exist strings* $\mathtt{dec}_0, \mathtt{dec}_1$ *such that* $\mathtt{Dec}(\mathtt{com}, \mathtt{dec}_0, 0) = 1$ *and* $\mathtt{Dec}(\mathtt{com}, \mathtt{dec}_1, 1) = 1$ *is empty;*
3. **Computationally Hiding:** *the ensembles of random variables*
$$\{[(\mathtt{com}, \mathtt{dec}) \leftarrow \mathtt{Com}(0, 1^k) : \mathtt{com}]\}_{k>0} \text{ and } \{[(\mathtt{com}, \mathtt{dec}) \leftarrow \mathtt{Com}(1, 1^k) : \mathtt{com}]\}_{k>0}$$
are indistinguishable with respect to algorithms running in time $o(2^{k^\gamma})$*;*
4. **Extractability:** *there exists an extractor algorithm* E *running in time* 2^{k^γ} *such that, for all commitments* com *computed by a probabilistic polynomial-time committer adversary* A, *if* A *succeeds in decommitting* com *as* b *with non-negligible probability, then* $E(\mathtt{com}) = b$ *with overwhelming probability.*

The above definitions can be easily extended to the case in which we wish to commit to a string (instead of committing to a bit). Such commitment scheme exists, for instance under the assumption that there exist permutations that are one-way with respect to polynomial-time adversaries but such that they can be inverted in subexponential time. In [20], these type of commitment schemes are used in order to achieve straight-line extractability in superpolynomial time.

Finally, we review the notion of a ZAP[21].

Definition 6. *A triple of polynomial-time algorithms* (ZG, ZP, ZV) *is a* ZAP *for the NP-language* L *with polynomial-time relation* R_L *iff:*

1. **Completeness:** *given a witness* y *for "*$x \in L$*" and* $z = ZG(1^k)$ *then* $ZV(x, z, ZP(x, y, z)) = 1$ *with probability* 1.
2. **Soundness:** *for all* $x \not\in L$, *with overwhelming probability over* $z = ZG(1^k)$, *there exists no* z' *such that* $ZV(x, z, z') = 1$.
3. **Witness-Indistinguishability:** *let* y_1, y_2 *such that* $(x, y_1) \in R_L$ *and* $(x, y_2) \in R_L$. *Then* $\forall z$, *the distributions on* $ZP(x, y_1, z)$ *and on* $ZP(x, y_2, z)$ *are computationally indistinguishable.*

In [21] a ZAP is presented under the assumption that non-interactive zero-knowledge proofs exist, thus the existence of ZAPs is implied by the existence of one-way trapdoor permutations.

Since we will need ZAPs to be secure with respect to subexponentially strong adversaries, we need subexponentially strong versions of these assumptions.

4 Three-Round Arguments in the cPK Model

In the cPK model we show that there exist three-round arguments of *membership* for all $\mathcal{NP}$ languages that are concurrently sound and black-box resettable

zero knowledge. We stress that a concurrently sound resettable zero-knowledge argument in the BPK model requires at least four rounds (see [14]) while the previously presented three-round protocols require stronger models than the cPK model (see [17, 18]).

Our construction assumes the existence of η-secure signature schemes, γ-secure commitment schemes, pseudo-random family of functions (which can be constructed assuming the existence of one-way functions) and the existence of ZAPs secure with respect to subexponential-time adversaries. We use subexponentially strong cryptographic primitives since we crucially use complexity leveraging for our result.

We then show how to obtain three-round arguments of knowledge for all polynomial-time relations that are concurrently sound and black-box concurrent zero-knowledge under standard complexity assumptions. We stress that black-box resettable zero-knowledge arguments of knowledge are not possible [22, 14], that concurrent soundness needs four rounds in the BPK model, and that three rounds is optimal for zero-knowledge in any public-key model.

4.1 Three-Round RZK Argument of Membership in the cPK Model

In this section we present our construction for the three-round argument of membership for all $\mathcal{NP}$ in the cPK that is concurrently sound and resettable zero-knowledge. Throughout the section, L will be a fixed $\mathcal{NP}$ language.

Our proof system will follow the FLS paradigm for zero knowledge [23] and we next define the auxiliary language $\Lambda = \Lambda(L)$ that we are going to use.

Definition 7. *The 8-tuple* $\tau = (x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$ *belongs to the language* Λ *if*

- $x \in L$ *or*
- *there exist* $a_1, a_2, \alpha_1, \alpha_2, b_1, b_2, \beta_1, \beta_2$ *such that*
  1. $\mathtt{pk}$ *is a public key in the output space of* $\mathtt{G}(1^k)$*;*
  2. $a_1 \neq a_2$*;*
  3. $(\tilde{a_1}, \alpha_1) = \mathtt{Com}(a_1, 1^k)$ *and* $(\tilde{a_2}, \alpha_2) = \mathtt{Com}(a_2, 1^k)$*;*
  4. $(\tilde{b_1}, \beta_1) = \mathtt{Com}(b_1, 1^k)$ *and* $(\tilde{b_2}, \beta_2) = \mathtt{Com}(b_2, 1^k)$*;*
  5. $\mathtt{Ver}(a_1 \circ c, b_1, \mathtt{pk}) = 1$ *and* $\mathtt{Ver}(a_2 \circ c, b_2, \mathtt{pk}) = 1$.

Informally speaking, $\tau = (x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$ belongs to Λ if x belongs to L or if, for $i = 1, 2$, $\tilde{b}_i$ is the commitment of a valid signature b_i (with respect to $\mathtt{pk}$) of the concatenation of message a_i committed to by $\tilde{a}_i$ concatenated with c.

Assumptions. In our construction we assume the existence of the following cryptographic tools.

1. an η-secure digital signature scheme $\mathtt{SS} = (\mathtt{G}, \mathtt{Sig}, \mathtt{Ver})$;
2. a γ-secure commitment scheme $(\mathtt{Com}, \mathtt{Dec})$;
3. a pseudo-random family of functions $\mathcal{F}$;
4. a ZAP (ZG, ZV, ZP) for the language Λ.

High-Level Overview. Let k be the security parameter. The public entry of a verifier contains a public key $\mathtt{pk}$ for a secure signature scheme and the first message z of ZAP for Λ. The actual proof that $x \in L$ consists of a first round where the prover sends a random message m to the verifier. The verifier replies with the current value c (in unary) of the counter and a signature $\hat{m}$ of $m \circ c$. The prover first verifies that $\hat{m}$ is a valid signature and then constructs 4 commitments $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ of 0^k. Finally the prover computes the second message of the ZAP in which he proves that $x \in L$ or that $\tilde{b}_1$ and $\tilde{b}_2$ are the commitments of valid signatures b_1, b_2 of messages $a_1 \circ c$ and $a_2 \circ c$ such that $a_1 \not= a_2$ and a_1 and a_2 are the messages committed in $\tilde{a}_1$ and $\tilde{a}_2$.

Let us now informally argue the properties of our construction. For the concurrent soundness we observe that even if the prover opens polynomially many concurrent sessions with the verifier, he will receive signatures of messages relative to different values of the counter. In particular, the prover will never see the signature of two messages with the same value of the counter as suffix. We will use a γ-secure commitment scheme along with a ZAP in order to show that a prover that proves false statements can be used by a superpolynomial-time algorithm in order to break a subexponentially strong assumption. Such a telescopic use of the hardness of different cryptographic assumptions is referred to as *complexity leveraging* and its power is gaining interest. The use of an *extractable* commitment along with a ZAP is also discussed and used in [20].

For the resettable zero knowledge property, the simulator while interacting with the verifier V^* will try, by rewinding $V^\star$, to get signatures for two messages with the same value of the counter as suffix. More precisely, to simulate the proof that $x \in L$, the simulator will first ask for the signature of message $m \circ c$ (where c is the current value of the counter), then he starts a look-ahead (by rewinding $V^\star$) in order to obtain the signature of a new message $m' \circ c$. Once the signature is obtained, the look-ahead ends and the simulator goes back to the previous original execution since it is now able to successfully run the third round.

The crucial observation to show that the simulation ends in expected polynomial time is that the values of the counters cannot be greater than the running time of the adversarial verifier (since $V^\star$ sends the value of the counter in unary). More precisely, each look-ahead starts after the first signature corresponding to a given counter value and to a given public key has been received by S. Since the number of public keys is polynomially bounded (the size of the public file does not change after the preprocessing stage) and the running-time of an (s,t)-resetting verifier is bounded by the polynomial t, we have that the number of look-aheads is polynomial. Moreover, each look-ahead starts because a given counter value has been sent by $V^\star$ on input a given transcript. Therefore, the expected number of rewinds that will be needed in order to obtain again the same counter (in the look-ahead the corresponding signature is asked for a different message) is the inverse of the probability that $V^\star$ plays such a value. Finally, by observing that the simulation between two rewinds can be run by S in polynomial time, we have that the simulator runs in expected polynomial-time.

Formal Description. Let k be the security parameter. The verifier runs the key generator $\mathtt{G}$ for the η-secure signature scheme on input 1^k obtaining the pair $(\mathtt{pk}, \mathtt{sk})$. Moreover, the verifier runs ZG on input 1^k and obtains z that will be used as the first round of the ZAP for the language Λ. The entry of the verifier in the public file consists of the pair $(\mathtt{pk}, z)$. The private key $\mathtt{sk}$ associated with $\mathtt{pk}$ is kept secret by the verifier. Moreover, the verifier initializes her counter c by setting $c = 0$. The protocol is found in Figure 1.

Common input: security parameter k, public file $F\{(\mathtt{pk}, z)\}$ and instance x.
P's private input: a witness y for $x \in L$.
V's private input: private key $\mathtt{sk}$ and a counter c.

P-round-1:

1. randomly pick seed s;
2. compute $w = \mathcal{F}(s, x \circ y \circ \mathtt{pk})$ and use w as randomness;
3. randomly pick $m \leftarrow \{0,1\}^k$ and sends it to V;

V-round-2:

1. increment c;
2. compute $\hat{m_c} = \mathtt{Sig}(m \circ c, \mathtt{sk})$;
3. send $(1^c, \hat{m_c})$ to P;

P-round-3:

1. verify that $\mathtt{Ver}(m \circ c, \hat{m_c}, \mathtt{pk}) = 1$;
2. randomly pick seed s';
3. compute $w' = \mathcal{F}(s', x \circ y \circ \mathtt{pk} \circ \hat{m_c} \circ c)$ and use w' as randomness;
4. use a γ-secure commitment function $\mathtt{Com}$ to compute commitments $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ where $\tilde{a}_j = \mathtt{Com}(0^k, 1^{k_1})$ and $\tilde{b}_j = \mathtt{Com}(0^k, 1^{k_1})$ for $j = 1, 2$;
5. compute $Z = ZP((x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk}), y, z)$;
6. send $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ and Z to V;

V-decision: verify that the ZAP is valid by running ZV on input instance $\tau = (x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$, initial ZAP-message z and ZAP-reply Z.

Fig. 1. The 3-round concurrently sound rZK argument for $\mathcal{NP}$ in the $\mathtt{cPK}$ model. The value k_1 is chosen based on η, γ and k (see the proof of concurrent soundness)

Theorem 1. *If, for some positive η and γ, there exist an η-secure digital signature scheme, a γ-secure commitment scheme and sub-exponentially strong ZAPs for all $\mathcal{NP}$ languages then there exists a* 3-round concurrently sound resettable zero-knowledge *argument system for any language $L \in \mathcal{NP}$ in the* $\mathtt{cPK}$ *model.*

Proof. Consider the protocol in Figure 1. Completeness follows by inspection.

Concurrent Soundness. Assume by contradiction that the protocol is not concurrently sound; then there exists a malicious prover $P^\star$ that in a concurrent attack has a non-negligible probability of making verifier V accept x for some $x \not\in L$.

We assume we know the session $j^\star$ of the verifier in which the prover will succeed in cheating. This assumption will be later removed. In order to obtain a contradiction we show an algorithm $\mathcal{A}$ running in time $o(2^{k^\eta})$ that breaks the η-secure digital signature scheme $\mathtt{SS}$ used in the construction. Algorithm $\mathcal{A}$ receives a signature public key $\mathtt{pk}$, obtained by running $\mathtt{G}$ on input 1^k, has access to $P^\star$ and to a signing oracle $\mathcal{O}$ for a public key $\mathtt{pk}$ and outputs a signature of a message for which the oracle had not been queried.

We now describe algorithm $\mathcal{A}$. On input challenge public key $\mathtt{pk}$, $\mathcal{A}$ performs the set-up procedure and builds the public entry of the verifier as $(\mathtt{pk}, z)$ where z is the output of algorithm ZG on input 1^k. Algorithm $\mathcal{A}$ starts the interaction with the prover $P^\star$ and, for all sessions j constructs the message to be sent in the second round of the protocol by following the verifier's algorithm and by resorting to the oracle $\mathcal{O}$ to compute signatures of messages $m \circ c$, for messages m received from $P^\star$. At the end of session $j^\star$, since $x \not\in L$ then, by the soundness of the ZAP (ZG, ZP, ZV), it must be the case that $P^\star$ has exhibited commitments $\tilde{a}_1, \tilde{a}_2$ of two different messages a_1, a_2 and commitments $\tilde{b}_1, \tilde{b}_2$ of two signatures b_1, b_2 such that, b_1 is a signature of a_1 and b_2 is a signature of a_2. Moreover, messages a_1 and a_2 have the value of the counter chosen by $P^\star$ for the $j^\star$-th session as suffix. Then $\mathcal{A}$ in time $O(\mathtt{POLY}(k_1) \cdot 2^{{k_1}^\gamma})$ breaks the secrecy of the commitments and obtains the two messages along with their corresponding signatures. Now, $\mathcal{A}$ has queried the oracle for public key $\mathtt{pk}$ once for each value of the counter (we remind the reader that the adversary $P^\star$, for each value of the counter, is allowed to run the verifier at most once) and thus $\mathcal{A}$ has not queried the oracle for at least one of $a_1 \circ c$ or $a_2 \circ c$. By picking k_1 such that $k_1^\gamma < k^\eta$, we have that $\mathcal{A}$ runs in time $o(2^{k^\eta})$ and we have reached a contradiction.

In our proof we assumed that $\mathcal{A}$ knows the value $j^\star$. If this is not the case that $\mathcal{A}$ can simply guess the value and the same analysis applies since this decreases only by a polynomial factor the probability of breaking the digital signature scheme. Moreover $\mathcal{A}$ can also try to break all the commitments of all sessions, since the running time will still be $o(2^{k^\eta})$.

Resettable Zero Knowledge. Let $V^\star$ be an (s,t)-resetting verifier. We next describe a probabilistic polynomial-time algorithm $S \equiv S^{V^\star}$ that has black-box access to $V^\star$ and whose output is computationally indistinguishable from the view of the interactions between P and $V^\star$.

The simulator S receives from $V^\star$ requests that can be described wlog by a quadruple (x, i, r, v), where x denotes the input instance for language L, i denotes the index of the public key with respect to which the interaction has to be simulated, r is the index of the random tape that must be used in the simulation, v is the index of the message that S must send (and, for our specific protocol, $v = 1$ or $v = 3$). We remark that the resetting adversary $V^\star$ is allowed to reset the prover to any previous state and even request that a different random

tape has to be used (however, $V^\star$ is not allowed to feed the prover with a random tape of its choice).

Simulator S maintains several data structures which will be implicitly used in its description below and performs all the consistency checks requested by the protocol (for example, that the signatures received are valid). In addition, S also builds a table $S(i, c)$ of signatures with entry (i, c) holding signatures of messages with suffix c computed with respect to the i-th public key.

The interaction between $V^\star$ and S consists of essentially four types of requests. Indeed, two rounds of the argument system are played by the prover and we distinguish two cases depending on whether or not the same round has been requested since the last rewind by the verifier.

1. The request is $(x, i, r, 1)$ and it is the first such request since last rewind.
   In this case, S follows exactly the prover's algorithm using the r-th random tape as source of randomness.
2. The request is $(x, i, r, 1)$ and such request has already been presented to S by $V^\star$ since last rewind.
   In this case, S re-plays the same message used in the previous round.
3. The request is $(x, i, r, 3)$ and S has already received such a request since the last rewind.
   In this case, S re-plays the same message used in the previous round.
4. The request is $(x, i, r, 3)$ and S has not received such a request since the last rewind. Let c be the value of the counter declared by $V^\star$ in the message just preceding the request. We have two sub-cases:
   (a) $S(i, c)$ contains two (or more) signatures.
   In this case S uses two signatures from $S(i, c)$ as witness to compute the last message of the ZAP.
   (b) $S(i, c)$ contains one signature.
   S has obtained the signature in the second round played by $V^\star$ (the case in which $|S(i, c)| = \emptyset$ and S has to play the third round cannot happen). In this case S needs to obtain a second signature with suffix c in order to be able to compute the last message of the ZAP. Thus, S starts a look-ahead for (i, c). More precisely, S rewinds $V^\star$ to the state just after he has sent the first request $(x, i, r, 1)$ (notice that since $V^\star$ is a resetting adversary, there could be several such requests) and uses a new random string r' instead of r. S will repeat such a rewind strategy until a rewind ends by appending a second entry to (i, c).
   As we shall argue, the simulation will halt in expected polynomial-time as the number of pairs (i, c) (and thus the number of look-aheads) is bounded by $s(n)t(n)$ which is a polynomial (we are considering by definition an (s, t)-resetting verifier).

The Views Are Indistinguishable. The first message played by S in each session has exactly the same distribution of the one played by the prover since S simply runs the prover's algorithm. We stress that even though after each rewind S changes one randomness r_g for a given $g \in \{1, \ldots, s(n)\}$, $V^\star$ is not aware of such an update since its view does not go back with respect to the last rewind.

The third round played by S has the following two differences with respect to the one played by the prover.

1. The prover commits to junk bits (the 0^k strings) while the simulator commits to a pair of different messages with the same suffix, along with their signatures with respect to a given public key. By the computational hiding of the commitment scheme, $V^\star$ does not distinguish the commitments of S from the commitments of the prover. More formally, if $V^\star$ distinguishes with non-negligible probability the commitments of S from the commitment of P, then $V^\star$ can be used to contradict the hiding of the commitment scheme.
2. The prover uses y such that $(x, y) \in R_L$ in order to compute the auxiliary witness for running ZP on input the auxiliary statement "$\tau \in \Lambda$". Instead, the simulator uses his knowledge of the different messages with c as suffix along with their signatures to compute the auxiliary witness for "$\tau \in \Lambda$". Both the prover and S follows the honest prover algorithm for the ZAP by running ZP on a good witness for the auxiliary statement. Therefore, an adversarial verifier $V^\star$ that distinguishes the witness used by the simulator from the one used by an honest prover can be used to contradict the witness-indistinguishability of the ZAP. Note that in our implementation of the ZAP the prover uses as randomness a pseudorandom string of both z and the message sent by the verifier. Therefore, as remarked in [21, 22], this implementation of ZAP preserves witness-indistinguishability even in case of reset attacks, i.e., the implemented ZAP is a resettable witness-indistinguishable proof system.

The Simulation Ends in Polynomial Time. S has to compute two messages for each session. Note that for the first message, S always performs a straight-line simulation since the first round of a session is played by running the prover algorithm, and since no witness is needed, it can be computed by S without rewinding $V^\star$.

The analysis is more complicated for the second message. First of all, observe that the simulator starts a new look-ahead procedure only after receiving a request $(x, i, r, 3)$. Such a request is immediately preceded by a message from $V^\star$ containing one valid signature for a pair (i, c) for which $S(i, c)$ was empty before the request (for otherwise, no look-ahead procedure would be started since S has at least 2 valid signatures). In other words, the simulator starts a look-ahead procedure only after receiving a useful signature. However, observe that both the number of entries in the public file (and thus the number of possible values of i) and the number of possible values of the counter are bounded by the running time of the adversary $V^\star$ that is assumed to be polynomially bounded. Next, we argue that the contribution of each entry to the expected work of the simulator is also polynomially bounded. Roughly speaking, the contribution of each pair (i, c) is equal to the probability that counter c appears in a session with public key $\mathtt{pk}_i$ times the number of rewinds needed to have a new session with the same public key and the same value of the counter in which we ask for the signature of a different message. It is easy to see that this quantity is polynomially bounded.

4.2 Three-Round CZK Argument of Knowledge in the cPK Model

In this section we present a three-round concurrently-sound concurrent zero-knowledge argument of *knowledge* in cPK for any language $L \in \mathcal{NP}$ under standard intractability assumptions.

The argument of knowledge that we present is derived from the argument of membership of the previous section by replacing the ZAP with a 3-round witness-indistinguishable proof system to prove a statement of the form "$\alpha \vee \beta$" where α is known at the beginning of the protocol while β is known only in the last round. Additionally, we need witness extraction with respect to α. An implementation of this primitive can be given by using a variation of the protocol presented in [25] (this is also used in [26]).

To prove the properties of our construction, we assume the existence of signature and commitment schemes secure with respect to polynomial-time adversaries, and the existence of the mentioned 3-round witness-indistinguishable proof system of membership π for any NP language (which can, in turn, be based on the existence of one-way permutations). We use this proof system as a black box and denote the messages computed in it as (wi_1, wi_2, wi_3), where wi_1 is sent by the prover, wi_2 is sent be the verifier and wi_3 is sent by the prover again. Note that in order to prove that $x \in L$, for some NP language L, the message wi_1 can be computed in polynomial time given only the length value $|x|$ (that is, neither x nor a witness for it is necessary).

The Public File. Let k be the security parameter. The i-th entry of the public file F consists of a randomly generated public key pk with security parameter k for the secure signature scheme SS and of the first round of a two-round computationally-binding perfectly-hiding commitment scheme.

Private Inputs. For the statements "$x \in L$", the private input of the prover consists of a witness y for $x \in L$. The private input of the verifier consists of the private key sk corresponding to the public key pk and a counter c.

The Protocol. Suppose that the prover wants to prove that $x \in L$ and that the verifier knows the private key sk of the i-th public key pk of the public file F.

In the first round P randomly picks string m of length k, computes wi_1 according to the proof system π, and sends the pair (m, π) to V. Then V increments c and uses the private key sk to compute a digital signature $\hat{m_c}$ of $m \circ c$, computes message wi_2 and sends the triplet $(1^c, \hat{m_c}, wi_2)$ to P. In the last round P verifies that $\hat{m_c}$ is a valid signature of $m \circ c$ with pk and computes the commitments $\tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ of 0^k. Then P computes message wi_3 according to proof system π by using instance $(x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$ and string y as the input and witness for π, respectively. P sends $wi_3, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2$ to V. Finally, V verifies that (wi_1, wi_2, wi_3) is valid by running the verifier's accepting predicate in proof system π, using as input the instance $(x, c, \tilde{a}_1, \tilde{a}_2, \tilde{b}_1, \tilde{b}_2, k, \mathtt{pk})$.

Theorem 2. *If there exist one-way permutations, then there exists a three-round concurrently-sound concurrent zero-knowledge argument of knowledge for $\mathcal{NP}$ in the* cPK *model.*

The proof of Theorem 2 is omitted from this extended abstract.

5 Conclusion

In this paper we have presented a 3-round concurrently-sound resettable zero-knowledge argument system in the cPK model that improves the previous works of Micali and Reyzin [17] and Zhao *et al.* [18]. The cPK model is only a slight variation of the BPK model, and we have shown that it can be used to go beyond the barrier of four rounds needed for concurrent soundness in the BPK model. Our result makes a big step for closing the gap between a public-key model that admits three-round concurrently-sound resettable zero-knowledge arguments and the BPK model. Moreover, we have shown a 3-round concurrently-sound concurrent zero-knowledge argument of knowledge in the cPK model under standard intractability assumptions.

References

1. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof-Systems. SIAM J. on Computing **18** (1989) 186–208
2. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: Proceedings of the 30th ACM Symposium on Theory of Computing (STOC '98), ACM (1998) 409–418
3. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero-Knowledge. In: Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC '00), ACM (2000) 235–244
4. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Advances in Cryptology – Eurocrypt '99. Volume 1592 of LNCS, (1999) 415–431
5. Kilian, J., Petrank, E.: Concurrent and Resettable Zero-Knowledge in Poly-Logarithmic Rounds. In: Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC '01), ACM (2001) 560–569
6. Kilian, J., Petrank, E., Rackoff, C.: Lower Bounds for Zero Knowledge on the Internet. In: Proceedings of the 39th Symposium on Foundations of Computer Science, (FOCS '98). (1998) 484–492
7. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-Box Concurrent Zero-Knowledge Requires $\omega(\log n)$ Rounds. In: Proceedings of the 33st ACM Symposium on Theory of Computing (STOC '01), ACM (2001) 570–579
8. Dwork, C., Sahai, A.: Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In Krawczyk, H., ed.: Advances in Cryptology – Crypto '98. Volume 1462 of LNCS, (1998) 442–457
9. Goldreich, O.: Concurrent Zero-Knowledge with Timing, Revisited. In: Proceedings of the 34th ACM Symposium on Theory of Computing (STOC '02), ACM (2002) 332–340

10. Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-Interactive Zero-Knowledge. SIAM J. on Computing **20** (1991) 1084–1118
11. Damgard, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Advances in Cryptology – Eurocrypt '00. Volume 1807 of LNCS, (2000) 418–430
12. Di Crescenzo, G., Removing Complexity Assumptions from Concurrent Zero-Knowledge Proofs, In: Proc. of Cocoon 2000. LNCS (2000).
13. Di Crescenzo, G., Ostrovsky, R., On Concurrent Zero-Knowledge with Preprocessing, In: Advances in Cryptology – Crypto '99. Volume 1666 of LNCS, (1999)
14. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Advances in Cryptology – Crypto '01. Volume 2139 of LNCS, (2001) 542–565
15. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant-Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In: Advances in Cryptology – Crypto '04. Volume 3152 of LNCS, (2004) 237–253
16. Reyzin, L.: Zero-Knowledge with Public Keys, Ph.D. Thesis. MIT (2001)
17. Micali, S., Reyzin, L.: Min-Round Resettable Zero-Knowledge in the Public-key Model. In: Advances in Cryptology – Eurocrypt '01. Volume 2045 of LNCS, (2001) 373–393
18. Zhao, Y., Deng, X., Lee, C., Zhu, H.: ResettableZero-Knowledge in the Weak Public-Key Model. In: Advances in Cryptology – Eurocrypt '03. Volume 2045 of LNCS, (2003) 123–139
19. Rompel, J.: One-Way Functions are Necessary and Sufficient for Digital Signatures. In: Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC '90). (1990) 12–19
20. Pass, R.: Simulation in Quasi-Polynomial Time and Its Applications to Protocol Composition. In: Advances in Cryptology – Eurocrypt '03. Volume 2045 of LNCS, (2003) 160–176
21. Dwork, C., Naor, M.: Zaps and their applications. In: IEEE Symposium on Foundations of Computer Science. (2000) 283–293
22. Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-Sound Zero-Znowledge and its Applications. In: Proceeding of the 42nd Symposium on Foundations of Computer Science, (FOCS '01), (2001) 116–125
23. Feige, U., Lapidot, D., Shamir, A.: Multiple Non-Interactive Zero Knowledge Proofs Under General Assumptions. SIAM J. on Computing **29** (1999) 1–28
24. De Santis, A., Persiano, G.: Zero-Knowledge Proofs of Knowledge Without Interaction. In: Proceedings of the 33rd Symposium on Foundations of Computer Science, (FOCS '92). (1992) 427–436
25. Lapidot, D., Shamir, A.: Publicly Verifiable Non-Interactive Zero-Knowledge Proofs. In: Advances in Cryptology – Crypto '90. Volume 537 of LNCS, (1991) 353–365
26. Katz, J., Ostrovsky, R.: Round-Optimal Secure Two-Party Computation. In: Advances in Cryptology – Crypto '04. Volume 3152 of LNCS, (2004).

Author Index