Bruce Christianson   Bruno Crispo
James A. Malcolm   Michael Roe (Eds.)

Security Protocols

12th International Workshop
Cambridge, UK, April 26-28, 2004
Revised Selected Papers

Springer

Volume Editors

Bruce Christianson
University of Hertfordshire
Computer Science Department
Hatfield AL10 9AB, UK
E-mail: b.christianson@herts.ac.uk

Bruno Crispo
Vrije Universiteit
Department of Computer Science
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
E-mail: crispo@cs.vu.nl

James A. Malcolm
University of Hertfordshire
Computer Science Department
Hatfield AL10 9AB, UK
E-mail: j.a.malcolm@herts.ac.uk

Michael Roe
Microsoft Research Ltd.
7 J.J. Thomson Avenue
Cambridge CB3 0FB, UK
E-mail: mroe@microsoft.com

Library of Congress Control Number: 2006932036

CR Subject Classification (1998): E.3, F.2.1-2, C.2, K.6.5, J.1, K.4.1, D.4.6

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-40925-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-40925-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11861386 06/3142 5 4 3 2 1 0

Preface

Here are the proceedings of the 12th International Workshop on Security Protocols. We hope that you will enjoy them, and that they will cause you to think at least one heretical thought. Please write or e-mail and share it with us.

Our theme this workshop was “Authentic Privacy.” Traditionally we have based authentication upon a rather strong notion of identity, and have then built other security services on top of authentication. Perhaps if we want a more nuanced notion of privacy, then we need to re-examine some of our assumptions, particularly when attackers and defenders share the same resources and infrastructure.

The position papers published here have been revised by the participants in the workshop, and are followed by edited (heavily in some cases) transcripts of parts of the discussions which they led.

Our thanks to Sidney Sussex College Cambridge for the use of their facilities, to Johanna Hunt at the University of Hertfordshire for organizing the logistics of the workshop and orchestrating the production of these proceedings, to Lori Klimaszewska of the University of Cambridge Computing Service for transcribing the audio tapes (in which “viruses without halos” could have caused havoc but didn’t), and to Donald Hunt for impeccable copyediting.

Finally, it is both a sadness and a pleasure to pay our tribute to David Wheeler, one of the original forty-niners at the Cambridge Computer Laboratory and author of the initial orders for EDSAC. The second version of initial orders *is* the Platonic bootstrap.

These workshops grew out of a series of informal meetings, which migrated between David’s office in the old Computer Laboratory tower and the front room of the Eagle across the road, and where so many of us were touched forever by our encounters with his decades of fearless thought. Time was finally called while these proceedings were being prepared.

February 2006

Bruce Christianson
Bruno Crispo
James Malcolm
Michael Roe

Previous Proceedings in This Series

The proceedings of previous International Workshops on Security Protocols have also been published by Springer as *Lecture Notes in Computer Science*, and are occasionally referred to in the text:

11th Workshop (2003), LNCS 3364, ISBN 3-540-28389-7
10th Workshop (2002), LNCS 2845, ISBN 3-540-20830-5
9th Workshop (2001), LNCS 2467, ISBN 3-540-44263-4
8th Workshop (2000), LNCS 2133, ISBN 3-540-42566-7
7th Workshop (1999), LNCS 1796, ISBN 3-540-67381-4
6th Workshop (1998), LNCS 1550, ISBN 3-540-65663-4
5th Workshop (1997), LNCS 1361, ISBN 3-540-64040-1
4th Workshop (1996), LNCS 1189, ISBN 3-540-63494-5

Table of Contents

Introduction: Authentic Privacy

Bruce Christianson

University of Hertfordshire, UK

Well hello, and welcome to the twelfth Security Protocols Workshop. When this all started we had no idea what a juggernaut we were creating, and it's particularly nice to see so many young people here. [Laughter]

There's a tradition that we start by spending five minutes introducing the theme and then we don't mention it again for the next 48 hours. This year being no exception, I shall now explain this year's theme.

There was a time when on the Internet you would see something called privacy being talked about as if it were an absolute good like, say, free education or health care. I think those days are now pretty thoroughly gone. It's clear that beyond a certain point amplifying privacy actually gives the attackers of the system more of an advantage than it gives the defenders, and there's also a point beyond which increased privacy doesn't really seem to be particularly useful to legitimate users of the system. These remarks are intended to be controversial by the way. [Laughter]

But what is privacy? Everyone is using different definitions of the word. When they say privacy, some people mean anonymity, some mean uncorrelatability, *i.e.* that the different things that you do can't be correlated. Some mean that you can tell who someone is but not what they're doing, some mean that you can tell exactly what is being done but not who is doing it, some mean that you can tell who's doing it, and what they're doing, but you can't tell why they're doing it. Quite often people slide between different definitions in the same paper, and sometimes they are smuggling: the sliding definition is a false bottom in the privacy suitcase, and they're using this to smuggle some different concept in.

Matt Blaze: There's another definition that the privacy people who are not technologists are more concerned with. For example, the OECD[1] privacy guidelines. This is more concerned with personally identifiable information, the ability to control the propagation of this, and perhaps how the data is used.

Reply: Yes indeed, and this whole side of the business is about how you control information once the cat is out of the bag: how do you track where it goes, what it does, and what the result is used for. And what sanction do you have against people who are misusing it when they themselves may have some form of anonymity or privacy?

Well, having almost gotten away with saying that unlimited privacy is bad — whatever privacy is taken to mean — the contrary alternative is to say that there's no privacy at all. This extreme alternative point of view says that com-

[1] www.oecd.org

puter systems should be a panopticon[2], so that every single thing that everyone does should be visible to all.[3] This approach has a certain twisted intellectual appeal.

Now we've got to be a little careful here. Some operations in the underware[4] like using a cryptographic key, or entering a password, have to be in some sense private because that's the nature of them. But we're computer scientists, we're used to working at lots of different levels of abstraction, and the question is, is there a nice level of abstraction at which privacy is not an issue, or at least is not something anyone particularly desires[5].

Some of you may remember the Cambridge active badge system that used to operate in the old Computer Laboratory. One of the nice features of that system was that, yes indeed, you could ping anybody and find out where they were and then speculate about what they were doing there, but the fact that you had pinged them was instantly visible to the person whom you pinged. They knew who was pinging them and where you were, the loss of privacy was symmetric.

It's quite nice to believe that something like this could be done on a larger scale, but it really only seems to be viable in a relatively closed community, which the Computer Lab at that time perhaps was. [Laughter] As soon as you start to have a more open environment, it becomes hard to see how to implement the kind of counter-mechanisms that would be needed to ensure that loss of privacy was symmetric.

But it's a nice idea that somebody who abuses the protocols that are supposed to protect the security of the system is punished by losing some element of their privacy (in whatever way we interpret privacy). We might perhaps see information of a personal nature about them having something done with it that the system would not be able to do had they not breached the protocol.

This is is a bit like what happens if you double-spend some forms of digital cash, where breaking the protocol is what releases the information that allows the privacy loss to take place. But we don't, I think, really have a strong enough cross-domain infrastructure to be able to do very much along that line yet.

There's also the issue of by whom personal information, or private information, is (or should be) held. It's clear that traditional answers along the lines of the system, or the administrator, or the government, are no longer candidates for being the right answer.

I think I'd like to argue that getting any further with the development of protocols for authentication, or of other protocols which need to be audited by third parties who are not necessarily trusted by the first two parties, might require rather gentler notions of personal identity than we've been in the habit

[2] See Jeremy Bentham, "Panopticon, or The Inspection-House", 1787, Crecheff (published 1791, London + Dublin) see http://cartome.org/panopticon2.htm

[3] Michael Roe points out that this analogy is not quite as straightforward as it seems. The inmates of Bentham's panopticon do retain some privacy, because they cannot see one another (and so are prevented from acting in combination). It is only the overlookers who have no privacy at all.

[4] Or whatever we call it now.

[5] After all, this is just about the case with other properties such as determinism.

of tying authentication to. Traditionally we've tied authentication to a rather strong notion of personal identity, and then we've pinned everything else on the back of that strong form of authentication.

Maybe if we want a more nuanced notion of privacy, and a more graceful degradation in the event of misuse of information, then we have to look at different ways of separating, or trying to orthogonalize, the issues of who's doing it, what are they doing, why are they doing it, who knows about it, and who else is affected.

At the moment we have a single authentication mechanism living in the basement that we try to make (or just hope will) do service for everything. We encapsulate this strong authentication as a service, and then encourage everyone else to build their secure service on top of it. It's not clear that this is the correct way forward in a more open environment. Perhaps strong authentication now belongs in the attic, along with the first Mrs Rochester.

This is a workshop not a conference, and the intention is that the presenters should be leading a discussion, or in some cases, trying to keep up with a discussion that has gone in a different direction to the direction in which they intended their talk to go. Please don't allow yourself to become constrained by the presentation which you prepared before you came.

Likewise, within the normal limits of academic debate (no personal attacks, no hitting with the closed fist) you're free to make whatever points from the floor you wish. The only proviso is that if you break somebody else's idea, then you are under an obligation to help them sort out the pieces at teatime. A correctly broken idea is often more interesting than a flawlessly polished one.

Limits to Anonymity When Using Credentials

Andreas Pashalidis[⋆] and Chris J. Mitchell

Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom,
{A.Pashalidis, C.Mitchell}@rhul.ac.uk
http://www.isg.rhul.ac.uk

Abstract. This paper identifies certain privacy threats that apply to anonymous credential systems. The focus is on timing attacks that apply even if the system is cryptographically secure. The paper provides some simple heuristics that aim to mitigate the exposure to the threats and identifies directions for further research.

Keywords: anonymous credential systems, pseudonym systems, unlinkability, privacy, timing attacks.

1 Introduction

Credential systems allow subjects to prove possession of attributes to interested parties. In a sound credential system subjects first need to obtain a structure termed a *credential* from an entity termed the credential *issuer*. The issuer encodes some well-defined set of attributes together with their values into the credential which is then passed on, or 'granted', to the subject. Only after having gone through this process can the subject prove possession of those (and only those) attributes that are encoded in the credential. During this latter process, the interested party is said to 'verify the credential' and is therefore called a *verifier*. Subjects are typically human users, issuers are typically well-known organisations with authority over the attributes they encode into the credentials they issue, and verifiers typically are service providers that perform attribute-based access control.

An example of a credential system is a Public Key Infrastructure (PKI). In a PKI, credentials are public key certificates that bind together subject attributes such as subject name, public key, its issue and expiry dates, and so on. The credential issuer is the Certification Authority (CA); it grants public key certificates according to some subject registration procedure. Finally, credential verifiers are the entities within the PKI that accept the certificates issued by the CA.

In conventional credential systems (e.g. a PKI), issuers and verifiers identify any given subject by a system-wide identifier. This has a potentially severe impact on the subject's privacy, as it enables issuers and verifiers to combine their knowledge about the subject. Indeed, they can construct individual transaction

[⋆] The author is sponsored by the State Scholarship Foundation of Greece.

histories for all the subjects in the system, simply by correlating credential-related events using these identifiers.

Over the last 20 years, a significant amount of research has been performed on credential systems that try to address the above privacy issue (see, for example, [2,3,4,6,7,8,10,11]). These systems are known as *anonymous* credential systems. In an anonymous credential system, subjects establish a different identifier with each issuer and verifier they wish to interact with, where we assume throughout that these pseudonyms cannot be connected to the subject's true identity. These identifiers, termed the subject's *pseudonyms*, are unlinkable, i.e. they do not possess any connection with one another. This means that it is infeasible, for colluding issuers and verifiers, to decide with certainty whether or not any given pair of pseudonyms belongs to the same subject[1]. While a subject obtains a credential under the pseudonym that was established with the issuer, proof of its possession[2] takes place under the pseudonym established with the verifier. Of course, in order for the system to remain sound, subjects should only be able to successfully prove possession of credentials that they were indeed issued by some legitimate issuer.

In this paper, we consider practical limits to the level of pseudonym unlinkability (and, thus, subject privacy) offered by anonymous credential systems. In particular, assuming the soundness and security of such a system, we consider how timing attacks, launched by colluding issuers and verifiers, may affect pseudonym unlinkability. Finally, we outline possible pragmatic approaches to minimising exposure to such attacks.

The paper is structured as follows. The next section outlines the assumptions we make about anonymous credential systems, section 3 discusses the issue of encoding freshness into credentials and section 4 presents the timing attacks. Section 5 provides some simple heuristics to counter the attacks and section 6 concludes, giving directions for further research.

2 A General Model for Anonymous Credential Systems

A number of anonymous credential systems have been proposed in the literature, each with its own particular set of entities, underlying problems, assumptions and properties. This section presents the model of anonymous credential systems on which the rest of the paper is based. It is intended to be as general as possible, in order to be consistent with the majority of existing schemes.

We consider an anonymous credential system to involve three types of player: subjects, issuers and verifiers. We refer to issuers and verifiers, collectively, as 'organisations'. It is assumed that subjects establish at least one pseudonym with each organisation with which they wish to interact. These pseudonyms are assumed to be indistinguishable, meaning that they do not bear any connection

[1] Assuming that at least two subjects exist within the system.

[2] Proving possession of a credential amounts to proving possession of the attributes that are encoded within the credential. We refer to this process also as the *showing* of a credential.

to the identity of the subject they belong to. We further assume that pseudonyms are unlinkable, i.e. two pseudonyms for the same subject cannot be linked to each other. Subjects may obtain credentials, i.e. structures that encode a well-defined, finite set of attributes together with their values, from issuers. They may subsequently show those credentials to verifiers, i.e. convince them that they possess (possibly a subset of) the encoded attributes. A credential is issued under a pseudonym that the subject has established with its issuer, and it is shown under the pseudonym that the subject has established with the relevant verifier.

It is assumed that the anonymous credential system is sound. This means that it offers *pseudonym owner protection*, i.e. that only the subject that established a given pseudonym can show credentials under it. Soundness also implies credential *unforgeability*; the only way that subjects may prove possession of a credential is by having obtained it previously from a legitimate issuer. In some applications, it is required that the system offers the stronger property of credential *non-transferability*. This property guarantees that no subject can prove possession of a credential that it has not been issued, even if the subject colludes with other subject(s) that may have (legitimately) obtained such a credential. In other words, a system that offers non-transferability prohibits credential sharing, whereas a system that offers only unforgeability, does not. (Of course, the degree of protection against credential sharing is always limited, since if one subject gives all its secrets to another subject then the latter subject will always be able to impersonate the former and use its credentials.) We require that credentials are bound to the subject to which they have been issued. We therefore assume that either the system offers non-transferability or that in practice subjects do not share their credentials.

It is assumed further that the system properly protects privacy in that a subject's transactions with organisations do not compromise the unlinkability of its pseudonyms. We note, however, that this unlinkability can only be guaranteed up to a certain point, as credential *types* potentially reveal links between pseudonyms. The type of a credential is defined as the collection of attribute values that are encoded into the credential. An organisation, for example, that issues demographic credentials containing the fields `sex` and `age group`, with possible values of {`male`,`female`} and {`18-`,`18-30`,`30-50`,`50+`} respectively, may actually issue up to 8 different types of credential (one for each combination of values). To see how credential types can be exploited to link subject's pseudonyms, consider the following trivial scenario. At time τ, a credential of type t is shown under the pseudonym p. However, suppose that up to time τ, only one credential of type t has been issued, and this was done under pseudonym p'. It follows, under the assumption that credentials are bound to subjects, that the two pseudonyms p, p' belong to the same subject; the colluding organisations can successfully link those two pseudonyms.

We note that, as part of credential showing, some anonymous credential systems allow subjects to reveal only a subset of the encoded attributes; in the above example it may be possible for the subject to reveal only the value of `sex`. For these systems, it is tempting to define the type of a credential as the collection of attributes that is revealed to the verifier during showing. However, we restrict our attention to the scenario where the verifier, rather than the subject, selects the attributes to be revealed during credential showing. This is, as far as our analysis is concerned, equivalent to the case where only the required set of attributes is encoded into a credential in the first place. This scenario is also likely to be valid for the case where verifiers perform attribute-based access control.

3 Encoding Freshness into Credentials

In certain applications, typically those involving short-lived credentials, verifiers need to validate the freshness of credentials. Thus, some indication of freshness has to be encoded into the credentials by their issuers. However, this indication constitutes an additional attribute, and its value helps determine the type of the credential. If the indication of freshness is unique for each credential (such as a serial number, a counter value, or a nonce), then it becomes trivial for organisations to link pseudonyms, as every credential will have its own, unique, type. It is thus desirable, from a privacy perspective, that the indication of freshness is shared among as many credentials as possible. One possible freshness indication is a timestamp generated using a universal clock, with a sufficiently coarse accuracy. We thus henceforth assume that one of the attributes that is present in all credentials is such a timestamp.

A question that arises in this context is who decides whether or not a credential has expired. If it is the issuer, it seems more appropriate for the timestamp to indicate the time of expiry. If, on the other hand, it is the verifier, then it makes more sense for the timestamp to indicate the time of issue. Since the latter alternative enables verifiers to have individual policies with respect to expiry of credentials, in the sequel we assume that the timestamp indicates the time of credential issue[3]. With τ_i denoting the time interval at which the timestamp is being refreshed by the issuer (and without loss of generality), we assume that all credentials issued between time τ and $\tau + \tau_i$ will carry the timestamp τ. More generally, credentials issued between $n\tau_i$ and $(n+1)\tau_i$ are assumed to carry the timestamp $n\tau_i$ (with $n \geq 0$ being some non-negative integer indicating the current period).

4 Timing Attacks

We consider some attacks that may be launched by colluding organisations who wish to link pseudonyms that belong to the same subject. It is sufficient for the organisations to link the events of credential issuing and showing as corresponding to the same subject; this amounts to linking the pseudonyms that correspond

[3] We do not consider the case where two timestamps are encoded into the credential, indicating both the beginning and the end of its validity period.

to those events. We distinguish between two attack strategies. As both of them exploit temporal information in the system, they both fall into the category of timing attacks.

The first strategy does not exploit the timestamps that are encoded into the credentials. Instead, it exploits the behaviour of subjects in certain scenarios, i.e. the high likelihood that a subject will show a credential to a verifier soon after it was issued to them by an issuer. That is, if a credential is issued at time τ and subsequently shown at time $\tau + \tau_\delta$, where τ_δ is small, then the issuer and verifier could collude to learn (with high probability) that the two pseudonyms involved belong to the same subject. Of course, the meaning of 'small' here will depend on the application, in particular on the rate of credential issuing. As a result, this timing attack is not equally serious in all scenarios. While in some applications (e.g. driving licences) it may not be a concern at all, in others (e.g. tickets, electronic cash, authentication tokens) the threat may be much more significant.

The second strategy is essentially the one already mentioned in section 3 above, namely the correlation of issuing and showing events based on the type of the credentials involved, and thereby linking the associated pseudonyms. However, the fact that we are now assuming that credentials encode timestamps (whose freshness is checked by verifiers), guarantees that credentials issued in different periods are of different types. Exploiting this particular fact may render the generic correlating-by-type attack much more dangerous.

5 Countermeasures

An obvious countermeasure to the first attack strategy is to require subjects to wait between obtaining and showing a credential. However, this needs to be managed carefully, since simply imposing some fixed waiting time, say τ_w, does not really reduce the exposure to the attack. This is because correlation between issuing and showing of credentials can still be performed by pairing these events if they are separated by a time difference of a little over τ_w. Thus, the delay τ_w should be randomised in some way. This, however, raises new questions: what are the minimum and maximum acceptable values for τ_w, given that it is likely (if not certain) to affect the system's usability? How does the choice of these limits affect unlinkability? How does the probability distribution according to which τ_w is drawn affect unlinkability?

The second attack scenario requires slightly different countermeasures. The objective here is to require subjects *not* to show credentials of some type until sufficiently many credentials of that type have been issued (to other subjects). Again, an obvious countermeasure is to require some (randomised) delay between the issuing and showing of credentials. The requirement by verifiers to validate the timestamps of credentials, however, introduces an additional constraint, namely that, at the time of showing, credentials should not have expired.

We now describe a simple heuristic that, using random delays, tries to approach a reasonable compromise between usability, security and privacy in the face of the above timing attacks. It requires issuers to generate credentials in batches, containing a range of consecutive issue timestamps. More precisely, suppose a subject requests an issuer to provide a credential encoding a certain set of attributes α. Instead of issuing a single credential with type defined by the concatenation of α with the current timestamp $n\tau_i$, the issuer generates a set of k credentials with types defined by the concatenation of α with the timestamps $(n-j)\tau_i$, where $0 \leq j \leq k-1$, where k is a policy-based parameter (which may be chosen by the subject or by the issuer, depending on the system context).

This means, of course, that timestamps no longer precisely encode the time of issue; it is assumed, however, that this does not affect security since those credentials with 'old' timestamps are bound to expire sooner than those with current ones. It is also required that subjects maintain a loosely synchronised clock, such that they can distinguish time periods. Further, it is assumed that verifiers accept credentials that were issued during the k most recent periods (including the current one).

With respect credential showing, the subject must follow a two stage process. Note that we suppose that the set of k credentials were actually issued at time τ_{is}, where $n\tau_i \leq \tau_{is} < (n+1)\tau_i$.

1. The subject is not permitted to show any of the issued credentials until a randomised waiting time τ_w has elapsed, where $\tau_{\mathtt{min}} \leq \tau_w \leq \tau_{\mathtt{max}}$, and $\tau_{\mathtt{min}}$ and $\tau_{\mathtt{max}}$ are domain specific parameters. Clearly $\tau_{\mathtt{min}}$ will depend on the rate of credential issue and showing, and should be chosen to prevent simple correlation between credential issue and showing. Further, $\tau_{\mathtt{max}}$ must satisfy $\tau_{\mathtt{max}} < (n+k)\tau_i - \tau_{is}$, since otherwise all the credentials may have expired before they can be used. In addition, $\tau_{\mathtt{max}}$ should be large enough to sufficiently randomise the delay between issue and showing, but not so large as to damage usability. The precise choices for both parameters will be a sensitive issue, and these parameters can be subject-specific. Similarly, the probability distribution to be used to randomly select the waiting time could be varied, but it seems reasonable to use a uniform distribution.
2. The subject should then show the credential with the oldest timestamp which is still valid (according to the policy of the verifier). That is, if we assume that the credential is to be shown at time τ_{sh} (where $\tau_{sh} \approx \tau_{is} + \tau_w$), then the subject should show the credential with timestamp $(\lfloor \tau_{sh}/\tau_i \rfloor - k + 1)\tau_i$. Observing that $n\tau_i \leq \tau_{is}$ and that $0 \leq \tau_w < (n+k)\tau_i - \tau_{is}$, we have $n \leq \tau_{sh}/\tau_i < (n+k)$ and hence $n \leq \lfloor \tau_{sh}/\tau_i \rfloor \leq n+k-1$. Thus $(n-k+1)\tau_i \leq (\lfloor \tau_{sh}/\tau_i \rfloor - k + 1)\tau_i \leq n\tau_i$, and hence such a credential exists within the set of credentials that was issued to the subject.

Figure 1 illustrates the operation of the heuristic for the case $k = 2$, i.e. in the case where the verifier accepts credentials that were issued during the last two periods (this amount of time is denoted τ_{acc} in the figure). At time τ_{is} the subject obtains two credentials: c_1 and c_2, where c_1 contains timestamp $n\tau_i$ and c_2 contains timestamp $(n-1)\tau_i$. A random waiting time t_w is then selected. In case (a) the waiting time is such that τ_{sh} falls within the next period, so

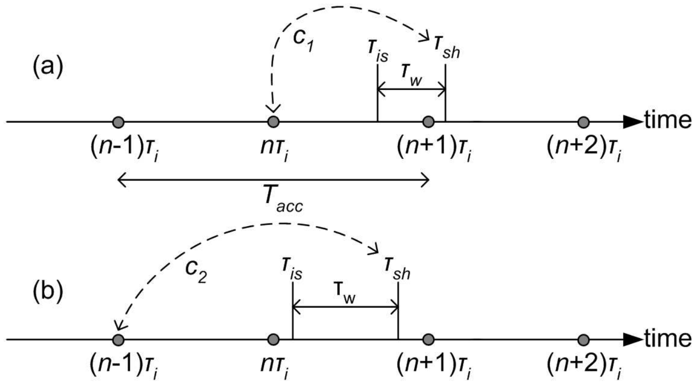

Fig. 1. Example for $k = 2$

credential c_1 is shown (c_2 has expired). In case (b) τ_{sh} falls within the same period, so the 'older' credential c_2 is shown.

The limit $\tau_{\mathtt{min}}$ should be selected such that, in every time period of that length, the expected number of subjects obtaining credentials of the type in question is sufficiently large. Imposing this lower limit is necessary because otherwise organisations may unambiguously link of a pair of showing and issuing events whenever some subject selects a waiting time no more than $\tau_{\mathtt{min}}$.

In some applications it may be unacceptable for a subject to obtain k credentials instead of one. A simple variation of the above heuristic does not require the subject to obtain k credentials. This involves simply choosing the waiting time τ_w first, and then only obtaining the credential that is appropriate to be shown at time $\tau_{is} + \tau_w$. This variation, however, offers significantly less protection against attacks based on correlation of credential types, since the number of issued credentials drops by a factor of k.

There are trade-offs between the choices for k and τ_i. If τ_i is relatively large then k can be made smaller, saving work in credential issue and storage. However, choosing a large value for τ_i also has disadvantages; in particular it decreases the precision available for specifying lifetimes of credentials, bearing in mind that choices for k may vary across the populations of users and organisations (depending on their privacy requirements).

We note that some cryptographic constructions allow for zero-knowledge proofs of the fact that some variable lies within a certain range, without revealing its value. Anonymous credential systems that make use of such constructions (see, for example, [1]) may overcome the timing attacks introduced by timestamps, since subjects can convince verifiers that timestamps lie within some acceptable range, without revealing any information beyond this fact. In order to maximise unlinkability in this scenario, however, this range has to be selected carefully. Assuming that timestamps encode the time of issue, the lower limit of the range should indicate 'just before expiry', i.e. the subject should prove that the issue timestamp is at least $\tau_{sh} - \tau_{acc}$, where τ_{acc} denotes the time after which credentials expire (according to the policy of the verifier).

6 Concluding Remarks

Timing issues often arise in the context of privacy preserving schemes. One proposed solution to the problem of anonymous communications is mix networks [5,9]. A mix network enables senders to send messages to recipients in a way that preserves the anonymity of the sender. A cascade of trusted parties, called 'mixes', provides the anonymising service, i.e. it hides the identities of the senders from the recipients. Each mix is susceptible to timing attacks that try to correlate incoming and outgoing messages. A typical countermeasure requires the mix to wait for a number of incoming messages before forwarding them (in shuffled order) to the next mix in the cascade (or, in the case of the final mix, to the recipient of the message).

Mix networks rely on trusted parties. Unfortunately, such entities do not exist in anonymous credential systems. In fact, in many scenarios, relying on third parties may be undesirable. In this paper we outlined some heuristics that offer protection against timing attacks, in the context of anonymous credentials. Although they do not rely on a trusted third party, they come with a cost in terms of communication and computational overhead and a potential impact on usability.

Clearly, more work is needed to devise optimal solutions which address the threat of timing attacks on anonymous credentials. These solutions should minimise three, probably contradicting, requirements: the probability of pseudonym linking, the inconvenience introduced to subjects, and the impact on system security. In other words, they should offer reasonable tradeoffs between usability, security and privacy protection – a non-trivial task indeed.

References

1. Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates — Building in Privacy.* The MIT Press, Cambridge, Massachusetts, 2000.
2. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceedings*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, Berlin, 2001.
3. D. Chaum. Showing credentials without identification: Transferring signatures between unconditionally unlinkable pseudonyms. In J. Seberry and J. Pieprczyk, editors, *Advances in Cryptology — AUSCRYPT '90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990, Proceedings*, number 453 in Lecture Notes in Computer Science, pages 246–264. Springer Verlag, Berlin, 1990.
4. D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, number 263 in Lecture Notes in Computer Science, pages 118–167. Springer Verlag, Berlin, 1987.

5. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
6. L. Chen. Access with pseudonyms. In E. Dawson and J. D. Golic, editors, *Cryptography: Policy and Algorithms, International Conference, Brisbane, Queensland, Australia, July 3-5, 1995, Proceedings*, number 1029 in Lecture Notes in in Computer Science, pages 232–243. Springer Verlag, Berlin, 1995.
7. A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. M. Heys and C. M. Adams, editors, *Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 9-10, 1999, Proceedings*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer Verlag, Berlin, 2000.
8. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity — a proposal for terminology. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, number 2009 in Lecture Notes in in Computer Science, pages 1–9. Springer-Verlag, Berlin, 2001.
9. J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, volume 2009 of *Lecture Notes in Computer Science*, pages 10–29. Springer-Verlag, Berlin, 2001.
10. A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, Berlin, 2002.
11. S. Steinbrecher and S. Köpsell. Modelling unlinkability. In R. Dingledine, editor, *Privacy Enhancing Technologies, Third International Workshop, PET 2003, Dresden, Germany, March 26-28, 2003, Revised Papers*, volume 2760 of *Lecture Notes in Computer Science*, pages 32–47. Springer-Verlag, Berlin, 2003.

Limits to Anonymity When Using Credentials (Transcript of Discussion)

Chris J. Mitchell

Royal Holloway, University of London

Rather sadly, I'm afraid, I don't think my talk will be terribly controversial.

I want to talk about anonymity, a special aspect of privacy, and anonymous credentials. Whilst one might have a scheme which is theoretically secure in some sense, of course there are limits to what that might guarantee you about anonymity, just through the practicalities of using this scheme. Having observed that, what might one do in order to try and maximise the level of anonymity that one achieves in the practical use of a credential scheme? I'm not going to look at scheme-specific issues, I'm not going to look at proofs of security, or details of how particular credential schemes work, but just assume some general properties of credential schemes, and look at what practical implications there are from the use of such schemes. This is based on joint work with a student of mine, Andreas Pashalidis, and as always, this is 90% Andreas and 10% me, and that accounts for the 10% of errors that will naturally creep into the talk.

The notion of a credential is something we may not explicitly use all the time, but which I guess is inherent in a lot of the transactions we do in electronic worlds. We use credentials to substantiate our identity, or to show what privileges we might have. So things like public key certificates, attribute certificates, even a credit card number, all kinds of things we use to engage in transactions that are somehow bound to us, might be regarded as a credential in some sense.

Bruce Christianson: There is a fine line between identity and privileges; a credit card number is a lovely example of something that is in the grey area.

Reply: Indeed, it's kind of an identifier as well. . . A public key certificate, or attribute certificate, is a very nice kind of idea because it just binds some privileges to some identifier. Of course, you still have to prove that you own that identifier in some sense, and I guess you can do that with a public key certificate perhaps, or with some other means.

I'm not sure anybody's actually used them, but there are lots of nice systems in the literature which allow credentials to be issued and used in a way that preserves anonymity of users. So you could go to a credential issuer and get a credential saying you're authorised to do certain things, and the credential issuer may only know you under a pseudonym, which may be specific to your relationship with that credential issuer. Some kinds of anonymous e-cash might fit that model. Credentials are issued and used using pseudonyms to protect the real identity of the user, and we'll refer to the *issue* of a credential, and the *showing* of a credential, so when you're given the credential that's the issue, and when you use that credential to prove that you own certain attributes, that's the showing of the credential.

Now obviously we want the system to be sound, we don't want people to be able to create credentials that they're not entitled to, or steal credentials, that is claim to own a credential which isn't theirs. And we also want some privacy or anonymity properties. There's the issue of pseudonymity, that you can't match the pseudonym under which the credential was issued to the pseudonym under which it was shown. So even if there is collusion between the issuer who issued the credential, and the service provider to whom the credential was shown, they still can't work out which particular credential that was issued is the one that's being shown.

We may also want this property of *unlinkability* where if two service providers collude they still can't link two showings of the same credential. So I guess this corresponds roughly to the kind of things that Bruce was saying in his introductory remarks, that users don't want to be able to be tracked, the behaviour by the same user should not be able to be linked. I would claim that unlinkability is a stronger property than pseudonymity. Under at least one definition, the one that my student came up with, one can prove that unlinkability implies pseudonymity. If you haven't got pseudonymity, that means you can link the pseudonym under which something was shown to the one under which it was issued, and if you can do that twice then you can link the two pseudonyms used to show a credential back to a common point. But if you can break linkability, it's not clear that you can then break pseudonymity.

Bruce Christianson: You could be able to link one pseudonym to another?

Reply: The reasonable definitions I can conceive of will all be probabilistic in nature because we're not dealing with absolute security, we're typically dealing with complexity-based security, so we make some statement that you can't link things, or you can't break pseudonymity with probability greater than some small probability.

Now credentials typically contain information such as the privileges of the owner, and/or the date and time at which it expires, maybe the date and time it was issued, all kinds of other stuff, and I want to refer to the collection of all this information as the *type* of the credential. If you haven't got any information in the credential it's not very useful because it doesn't tell you anything, and then why you are bothering to show it in the first place.

So typically, when you show a credential the type is revealed, it's got your privileges in it, so the person you're showing it to will obviously need to be told what those privileges are. The type limits the degree to which the unlinkability and pseudonymity can hold, for example, if you have only ever one credential of type T that's been issued, then whenever that credential is shown, you can obviously link it to its issue because there was only ever one of that type.

Rebecca Wright: You're making an assumption that a credential must be shown in order to be used.

Reply: Yes, that's my model of what credentials are for, I appreciate there may be other ways.

Rebecca Wright: How about a zero-knowledge proof that the credential has a particular property?

Reply: You'd still have to know what privileges held, but you could then generalise this to say, if there's only one credential that's been issued that matches these, maybe we'd have set theoretic containment or something instead of equality. I guess that's something we ought to build into the paper.

Often credentials are time dependent, so there's going to be one issued at the beginning of the time period. Suppose the type changes every day, then if you get a credential at one second after midnight, that's likely to be the only one that's been issued of this type at this time. And typically credentials are shown very shortly after they're issued, so two seconds past midnight you show this credential, and you may very well be the only entity to which a credential of this type has been issued, so you can be linked. Of course, by lunchtime things should be a lot better, but it's nasty that we've got such a tight restriction on anonymity for a short period of time.

Now these kinds of problems have been faced elsewhere, in mix nets the solution is to just accumulate enough messages before you relay them, if you haven't got enough messages to mix up to anonymise, you just wait till you have, and you rely on trusted third parties to enforce this. But here we haven't got any trusted third parties, we can put some mix nets in place, but inherently in the credential system there are no obvious trusted third parties, so what can we do.

George Danezis: Isn't the issuer really the trusted third party that provides the anonymity guarantee? Maybe it could also give other people credentials of the same type, so you're not the only one to stick out.

Reply: Of course, that's true. If the credential issuer gives everything a different type then it can quickly track, so we'd hope that it would play by the rules. The point I'm trying to make is that even if it does play the rules there are still nasty things. If the only thing you can trust is that it will play by the rules, then that's not necessarily enough.

Bruce Christianson: But obviously a credential issuer could say, I'm never going to just issue one credential of a type, if you come to me asking for this type of credential I shall issue a credential of a similar type to somebody else also.

Reply: So if we're looking at the case where the type codes are time intervalled, then we've got to do something about what happens near the beginning of the interval. I want to quickly whiz through a few strategies, and then come to a conclusion.

Well there's a naive strategy, that you wait for a delay period before you use it, in the hope that some more credentials of that type have been issued by the time you do come to use it, but the problem with this is that if all users wait the same time d then you don't get anywhere. If the first credential of type T is issued at time t, and the second one of this type is issued at time t+delta, then the first issued credential can be shown at any time after t+d, because they have to wait d. So suppose they show it sometime between t+d and t+d+delta, well then we can still link the showing unambiguously to the issue, because the second issue credential cannot be shown before t+d+delta.

We can add a random wait time, but it doesn't really help that much. If we suppose the delay is always between d1 and d2, and if delay for the first issued credential of type T happens to be d1, then the same problems apply as previously.

Tuomas Aura: Does a random delay ever help at all?

Reply: If you randomly wait a longer time then there may be some other people who have been issued credentials who might have been eligible to show by then. So if you wait time d2, which is significantly larger than d1, then you hope that somebody else will have been issued a credential shortly after you and they might have chosen d1.

Tuomas Aura: The nub of this is that there are multiple people with the same type of credential at any given time. And the only place to enforce this is with the issuer. If you do it a second and third time randomly at the credential holder, then you are just randomly giving up your identity.

Reply: That's the point I'm making.

George Danezis: I think that any delay has to make an assumption about how many people will have been issued with a credential within that delay, but the delay has to be random because of Chris' earlier reasoning.

Tuomas Aura: This leads to the next question. The validated times are printed on the credential aren't they? So then every credential is unique.

Reply: Well, let's assume that we only print the day of issue on the credential because otherwise everybody has a different type. But unless you're using a protocol in which you can show whether or not your credential satisfies some kind of general criterion, then you've got problems.

Bruce Christianson: But Chris' point is actually a little stronger because even if the credentials didn't have time stamps on them at all, if you know that a particular person is in the habit of getting credentials just before they want to use them, there's still a problem.

Reply: Yes, I wanted to put up some rather weak solutions and knock them down again quickly, just to show my train of thought.

Another naive strategy requires users to know when the first credential of a type might have been issued. If we assume that credentials are stamped with the day of issue, then you know that the first time at which a credential with Monday on it could be issued is one second past midnight. Then you wait till 3 o'clock in the morning before you show a Monday credential. But there's a problem now, what do you do between midnight and 3 o'clock in the morning when you want to show a credential? Well basically you could show a Sunday credential, but that's kind of awkward. So actually you start issuing Monday credentials 9 o'clock on Sunday evening.

So if you ask for a credential after 9 o'clock on Sunday evening, you get a Sunday credential and a Monday credential, and you choose which one you show depending on what the time is. This is hardly rocket science, but there still remain a lot of questions about how you choose the delay d. You have to base any calculation on some assumptions about the numbers of credentials that are being issued, so maybe you assume some probability distribution for the issuing and showing of credentials, and then you have to choose your d optimally, because you don't want to be issuing more credentials than you really need to. It may be that you always issue seven credentials at a time, one for today, and one for every day for the whole week, I guess it would depend on the number of users there are, the number of credentials that are issued, the number of credentials that are shown of a particular type, and one could do sort of cost benefit analyses of these numbers, not that I have.

Matt Blaze: There seems to be an underlying assumption that credentials are short-lived, but not too short-lived, at least in the examples you've given. There aren't that many applications that are using credentials of this form yet, can you give a motivating example?

Reply: I suppose the example we keep talking about is parking tickets, that kind of thing, short-lived rights that you buy to use pretty much at that time, and then will expire a relatively short period afterwards. Rights to access somewhere, I mean.

Matt Blaze: One of the reasons we tend to use short-lived credentials is specifically to avoid the revocation problems, we just wait a while and then it will stop being valid, and you avoid needing to build a public key infrastructure with a revocation system. But in your examples you're introducing some skew that prevents you in some sense from issuing too short-lived credentials.

Reply: Yes, actually I think you then need to issue lots of short-lived credentials, but there's certainly a trade-off.

Richard Clayton: Isn't there a difficulty for the issuer issuing a ticket on Monday evening, one which is valid three hours on Monday, and another valid all day on Tuesday, being used twice?

Reply: Maybe, but I guess, a season ticket which lasts 27 hours instead of 24 is not such a big deal.

Frank Stajano: If you have a situation where the issuer issues maybe seven credentials instead of one, and these credentials are anonymous, because that's the point of it, then couldn't the recipient start a second market trading those credentials, and nobody knows it's him. That would be ripping off the issuer wouldn't it?

Reply: I think the answer to that would depend on the details of the credential system, because some credential systems would require you to divulge all your secrets in order to enable somebody else to use your credential. There's this non-transferability requirement, and different schemes operate in different ways. But generally speaking I could always give somebody my credentials, and provided I'm prepared to give them all my secrets then they can masquerade as me. I'm not sure what you can do about that.

Birgit Pfitzmann: What of the threat model?

Reply: Yes I guess the threat model may sound rather unrealistic, we're assuming that on the one hand the issuer of the credentials is trusted to obey the rules, and on the other hand that they may collude with certain other parties to try and correlate behaviour. I'm not sure that's completely unrealistic because there may be somebody monitoring the behaviour of the issuer to check that they are behaving correctly. I suppose the other thing is, since this is the threat model that everybody else uses, it may be crazy, but there's lots of other crazy people out there too, who are worried about the same things.

Tuomas Aura: You mentioned the kind of credentials that you can show multiple times and then cannot be linked together. It seems, the user will already lose anonymity because each credential is a different bit string so you can always tell them apart.

Reply: Yes, but when you show a credential that doesn't mean I actually give you the bit string, it means that I proved possession of a credential of these characteristics.

Tuomas Aura: So it's some kind of zero knowledge?

Reply: Well maybe not zero knowledge, but that kind of property. When you show a credential there's a protocol whereby the showee has assurance that the shower possesses a credential of the particular type without actually seeing the bit string. There are lots of schemes and they work differently, so I'm kind of abstracting away, but I guess the point is we have all these wonderful theoretical schemes, but you have to be jolly careful to think about how you're actually going to use them, otherwise you may lose some of the properties that you're trying to achieve.

You may be able to link showings, people may show a credential twice in a short space of time, or in a particular place, and you may be able to break unlinkability in that way, and you may be able to link issuing and showing, if people always show a credential shortly after it's been issued. So you may want to impose a randomised lower limit on the delay between showings of a credential, and also between issuing and showing. And, in some cases, you might actually want to require people to only use credentials once, even if they're capable of being used more than once.

Bruce Christianson: That's similar to the sanction on double-spending that's sometimes used.

Reply: So how can we best decide how to choose the delays of the various types that I suggested, and what other strategies are there? Well of course we can go outside the model to more sophisticated credential systems where instead of showing the type of the credential, I just show that the type satisfies some statement like, yes, I do own a credential which was issued within the last 12 hours, and which gives me at least these privileges. Of course, I suppose a cunning service provider can ask a question which is likely to reveal your identity, but maybe you're very careful about which questions you answer.

George Danezis: I think your comparison with the literature on mix nets is really potent because the same problems exactly are occurring for a number of communications systems. When we send a message through a mix net, do we rely on the fact that others will also be sending messages, and if others are also real people and the attack is to give the impression of activities, then the same attack applies here immediately. You observe the issuer and see other people also being issued at the same time as you, but the attacker may pretend to be many people at once. Now an advantage that we might have in the case of credentials is that the act of issuing is not meant to be private, whereas in the context of anonymous communications it is always considered extremely bad PR to require people to first register with their real identity before they can send a message, so that you can be assured that they are different people. But in the process of issuing credentials, which is here the thing which you are trying to mix, actually many protocols assume that there is an authentication process. So maybe there could be a protocol by which third parties are assured that for a certain amount of time some real people who are the users of the system have been issued credentials, and that would get you out of this problem.

Reply: Yes, good point. I guess I've overrun my time, but I think the instructions are we're not allowed to have a complete paper at the time of the talk.

Bruce Christianson: Yes, that's excellent. Thank you very much.

A Cryptographic Framework for the Controlled Release of Certified Data

Endre Bangerter[1], Jan Camenisch[1], and Anna Lysyanskaya[2]

[1] IBM Zurich Research Laboratory, Säumerstrasse 4, 8803 Rüschlikon, Switzerland
{eba, jca}@zurich.ibm.com

[2] Computer Science Department, Brown University, Providence, RI 02912, USA
anna@cs.brown.edu

Abstract. It is usually the case that before a transaction can take place, some mutual trust must be established between the participants. On-line, doing so requires the exchange of some certified information about the participants. The easy solution is to disclose one's identity and reveal all of one's certificates to establish such a trust relationship. However, it is clear that such an approach is unsatisfactory from a privacy point of view. In fact, often revealing *any* information that uniquely corresponds to a given individual is a bad idea from the privacy point of view.

In this survey paper we describe a framework where for each transaction there is a precise specification of what pieces of certified data is revealed to each participant. We show how to specify transactions in this framework, give examples of transactions that use it, and describe the cryptographic building blocks that this framework is built upon. We conclude with bibliographic notes on the state-of-the-art in this area.

1 Introduction

The problem of privacy protection is to control the dissemination of personal data. There exist various privacy principles that describe at a conceptual level what measures have to be taken to protect privacy. Examples of these principles are: an individuals's right to access and to request correction of data about oneself and the requirement for an individual to consent to the disclosure of her personal data. Another principle is that of data minimization: It states that an individual should only disclose the minimal necessary data for a given purpose. Determining these data is often a difficult task and one usually needs to balance an individual's privacy interests and the legitimate interest of other parties in the individual's data. An example of this trade-off is an individual's wish to be anonymous conflicting with he requirements imposed by law enforcement to be able to identify and get hold of criminals. Such trade-offs impose limits on privacy that cannot be overcome by any technology.

When data is stored in digital form, the privacy problem is more severe than with paper based processes. Once data is disclosed in digital form, it can be easily stored, distributed, and linked with various other data. While the use of digital media aggravates the privacy problem, it also offers new opportunities and technologies to implement principles of privacy protection. Today's electronic transaction systems essentially reproduce the non-privacy protecting paper based business processes and thereby most often fail to take advantage of the digital media as a privacy enabler.

Incorporating privacy principles in digital media calls for a *privacy architecture.* While certain aspects of privacy protection are well understood and corresponding technologies are known, no comprehensive and stable privacy architecture exists yet. Various efforts to enhance privacy in digital media are underway. Examples are the NSF-funded PORTIA project [1] or the European project PRIME [2]. The latter aims, among other things, to develop a comprehensive framework and architecture to enable privacy in digital media. One can expect that a standard privacy architecture will materialize in the near future. Such a privacy architecture will consist of a combination of various technologies ranging from software technologies such as access control, auditing, and policy management to more theoretical cryptographic techniques.

In this paper we take a step towards enabling privacy in digital media and present a cryptographic framework that enables data minimization. In this framework, for each transaction, there is a precise specification of what data gets revealed to each participant. This is called "controlled release of data". In our framework the key feature is that the data in question is certified. That is to say, its validity can be verified by the recipient.

Besides the framework we also describe cryptographic building blocks that allow one to efficiently implement electronic transactions with our framework. That is we describe particular encryption schemes, [21,31], commitment schemes [47,33], and signature schemes [17,18] These schemes are all discrete logarithm which allows for their combination with various efficient zero-knowledge proof techniques. For an overview of such techniques we refer to [24].

The framework turns out to be usable for the realization of a large number of privacy enabling applications. For instance, it can be used to construct anonymous credential systems [27,15,50,45], group signature schemes [30,22,4], and electronic cash [29,9].

2 A Cryptographic Framework for the Controlled Release of Certified Data

A (digital) certificate consists of *data items*, provided by a *user*, and a digital signature by a *(certificate) issuer* on the *data items.* By signing the user's data items the issuer certifies for instance the user's authorization to perform some given task and that it has verified the validity of (some of) the user's data items. To demonstrate its authorization and the validity of the data items the user can for instance show the certificate to a *verifier* who checks the certificate's validity by verifying the correctness of its signature. The verifier will accept the claims associated to a certificate as far as he trusts the issuer w.r.t. these claims.

In the following we describe desirable properties of (non-traditional) certificates allowing the user to control what data items are disclosed to the issuer and verifier of certificates respectively.

Required Properties When Showing a Certificate. By showing a certificate we mean the process whereby a user using a certificate she possesses to convince a verifier of the contents of the certificate. We stress that during this process the user does not necessarily send the actual certificate to the verifier.

We require a process that allows the user to show certificate such that the following properties are met.

Multi-show unlinkability: Conventional (public-key) certificates are represented (encoded) by unique strings. Thus when the user would just send the certificate obtained from the issuer to the verifier, the issuer and the verifier can link the transactions. Furthermore, multiple showings of the same certificate to the same or different verifiers are linkable. We would like to emphasize that linkability is an inherent property of traditional certificates, which is independent of the data items contained in a certificate. In particular, even so-called pseudonymous certificates, i.e., certificates that do not contain personally identifying data items, are linkable. Linkability is known to be a serious threat to the privacy of individuals. We require that the showing of a certificate cannot be linked to the issuing of the certificate as well as to other showings of the same certificate, unless of course the data items being disclosed allow for such linking.

Selective show of data items: Given a certificate, we require that the user in each showing of the certificate can select which data items she wants to disclose (and which data items she does not want to disclose) to the verifier. For numerical data items, we require that it be possible to show that a data item lies in some interval without revealing the exact value of the data item. As an example, consider a driver's license certificate consisting of the user's name, address, and date of birth. When stopped on the road at a police checkpoint, the user shows that the certificate is valid, i.e., that she is authorized to drive, without disclosing her name, address, and date of birth. Using the same certificate, in a supermarket when purchasing alcohol, the user shows the certificate such that she only discloses that she is not underage.

Conditional showing of data items: We require that the user be able to conditionally disclose certified data, when showing a certificate. More precisely, let us assume that there is a third party, and that prior to certificate showing the user picks the data items she wishes to show conditionally to the issuer; also the user and the verifier agree on the conditions under which the verifier may learn the selected data items. In a conditional showing, the user discloses to the verifier information (on the conditionally shown data elements) such that the verifier cannot recover the conditionally shown data items from the information. Yet, the verifier can be assured that, when asked to do so, the third party is able to recover the data items.

Hence, if the third party recovers the data items only if the mentioned condition is fulfilled (where we assume that it knows the condition), then the above mechanism implements showing of (certified) data under the agreed condition.

As an example, consider a user accessing a university library's reading room with valuable books and the third party being the university administration. The user's identity, e.g., contained in her student identity certificate, will be disclosed to the librarian only under the condition that books are stolen from the reading room. To find out the user's identity, the librarian will need to involve the university administration.

Proving relations between data items: When showing multiple certificates by different issuers, the user should be able to demonstrate that data items in the certificates are related without disclosing the data items. For instance, when showing her drivers certificate and her credit card certificate to a car rental company, the user should not need to disclose her name contained in the certificates, but only to demonstrate that both certificates are issued to the same name.

Desirable Properties of Certificate Issuing. We now describe the properties we require of the process where the user gets issued a certificate by an issuer. Let $\{m_1, \ldots, m_l\}$ denote a set of data items and H a subset of these data items. It should be possible for the user to obtain certificate on $\{m_1, \ldots, m_l\}$ such that the issuer does not learn any information on the data items H, while it learns the other data items, i.e., $\{m_1, \ldots, m_l\} \setminus H$. We refer to such an issuing as *blind certification.*

Obviously, the data items in H are chosen by the user, however the other data items could be chosen by the issuer or by the user. For the data item that remain hidden from the issuers, we require that the user is able to assert that some of them were previously certified by another issuer. An example where this property is useful is e-cash with online double spending tests. Here the user chooses a random and unique number that is certified by the bank (issuer) such that the bank does not learn the number (c.f. §3.2).

2.1 A Framework of Cryptographic Primitives

In this section we illustrate how a framework of encryptions, commitments, signatures, and zero-knowledge proofs can be used to implement certificates having properties as described above. The presentation is (quite) informal and intended to be accessible for non-specialists in cryptography. We first recall the abstract properties of encryptions, commitments, signatures, and zero-knowledge proofs of knowledge.

By $\omega = A(\alpha)$ we denote that ω is output by the (probabilistic polynomial-time) algorithm A on input α.

An *(asymmetric) encryption scheme* consists of the algorithms SetupEnc, Enc, and Dec with properties as follows. The *key-generation algorithm* SetupEnc outputs an encryption and decryption key pair (EK, DK). The *encryption algorithm* Enc takes as input a message m, a label L, and the encryption key EK and outputs an encryption E of m, i.e., $E = \mathrm{Enc}(m, L; EK)$. The *decryption algorithm* Dec takes as input an encryption E, a label L and the decryption key DK and outputs the message m, i.e., $m = \mathrm{Dec}(E; DK)$. An encryption scheme is secure, if an encryption $E = \mathrm{Enc}(m\,EK)$ does not contain any computational information about m to an adversary who is given E and EK, even if the adversary is allowed to interact with the decryptor. (For more on definitions of security for cryptosystems, see, for example, Goldreich [39].) The notion of encryptions with labels was introduced in [31]. Labels allow to bind some public data to the ciphertext at both encryption and decryption time. In our applications, user would attach a label to an encryption E that indicates the conditions under which should be decrypted.

A *commitment scheme* consists of the algorithms Commit and VerifyCommit with properties as follows. The *commitment algorithm* Commit takes as input a message m, a random string r and outputs a commitment C, i.e., $C = \mathrm{Commit}(m, r)$. The *(commitment) verification algorithm* VerifyCommit takes as input a C, m and r and outputs 1 (accept) if C is equal to $commit(m, r)$ and 0 (reject) otherwise. The security properties of a commitment scheme are as follows. The *hiding property* is that a commitment $C = \mathrm{Commit}(m, r)$ contains no (computational) information on m. The *binding property* is that given C, m, and r, where $1 = \mathrm{VerifyCommit}(C, m, r)$, it is (computationally) impossible to find a message m' and a string r' such that $1 = \mathrm{VerifyCommit}(C, m', r')$.

A *signature scheme* consists of algorithms: SetupSign, Sign, VerifySign as follows. The *key-generation algorithm* SetupSign outputs a verification and signing and pair (VK, SK). The *signing algorithm* Sign takes as input a message m and a signing key SK and outputs an signature S on m, i.e., $S = \mathrm{Sign}(m; SK)$. The *(signature) verification algorithm* VerifySign takes as input an alleged signature S, the message m, and the verification key VK; it decides whether to accept or reject the signature. A signature scheme is secure [41] if, on input VK, no adversary can produce a valid signature on *any* message m even after a series of adaptive queries to the signing algorithm (provided that the adversary did not explicitly ask for a signature on m). In a variant we use, Sign takes as input a *list* of messages $m_1, \ldots, m_l$ and a signing key SK and outputs an signature S on $m_1, \ldots, m_l$, i.e., $S = \mathrm{Sign}(m_1, \ldots, m_l; SK)$. The verification algorithm also looks at a list of messages and a purported signature. For our purposes we also require an extended signature scheme which additionally features a two party protocol HiddenSign between a signer and a (signature) requestor. Let be given messages $m_1, \ldots, m_l$ and commitments $C_1 = \mathrm{Commit}(m_1), \ldots, C_l = \mathrm{Commit}(m_{l'})$ with $l' \leq l$. The common input to the protocol are $C_1, \ldots, C_{l'}$ and $m_{l'+1}, \ldots, m_l$ and the signer's input is a signing key SK. At the end of the protocol the requestor's output is a signature S on $m_1, \ldots, m_l$. We denote such a protocol execution by $S = \mathrm{HiddenSign}(C_1, \ldots, C_{l'}, m_{l'+1}, \ldots, m_l; SK)$. We see that by the hiding property of commitments the signer does not learn any information on the messages $m_1, \ldots, m_{l'}$ in the protocol HiddenSign.

Finally we consider *zero-knowledge proofs of knowledge*. Let W denote an arbitrary boolean predicate, i.e., a function that on input some string α either outputs 1 (true) or 0 (false). A proof of knowledge is a two party protocol between a prover and a verifier, where the common input is a predicate W, and the prover's input is a string w for which W is true, i.e., $1 = W(w)$. At the end of the protocol the verifier either outputs 1 (accept) or 0 (reject). The protocol has the property that if the verifier accepts, then it can be assured that the prover knows a string w' such that $W(w') = 1$. The protocol is zero-knowledge if the verifier does not learn any (computational) information about the provers input w. We denote such a zero-knowledge proof of knowledge by $\mathrm{PK}\{(w) : W(w) = 1\}$. Often we use proofs of knowledge where W is a composite predicate in multiple variables. Our notational convention is that the elements listed in the round brackets denote quantities the knowledge of which is being proved. These are (in general) not known to the verifier, and the protocol is zero-knowledge with respect to these parameters. Other parameters mentioned in a proof of knowledge expression are known to the verifier. (In particular, the description of the predicate W is known to the verifier.) For instance, $\mathrm{PK}\{(x, y) : W_1(x, y) = 1 \wedge W_2(x, z) = 1\}$ denotes a protocol where the parameters mentioned are (x, y, z); the value z is known to both parties (since it is not listed in the round brackets); the protocol is zero-knowledge with respect to (x, y). Upon completion of this protocol, the verifier will be convinced that the prover knows some x' and y' such that $W_1(x', y')$ and $W_2(x', z)$ are satisfied.

2.2 Cryptography for the Controlled Release of Certified Data

In this section we discuss how the cryptographic building blocks discussed in the previous paragraph can be used to implement the controlled release of certified data.

By I_1 and I_2 we denote certificate issuers with verification and signing key pairs (VK_1, SK_1) and (VK_2, SK_2), respectively. The verification keys VK_1 and VK_2 shall be publicly known and authenticated. Also, we assume that the user holds a certificate $Cert_1 = \mathrm{Sign}(m_1, \ldots, m_{l_1}; SK_1)$ from I_1 and a certificate $Cert_2 = \mathrm{Sign}(\tilde{m}_1, \ldots, \tilde{m}_{l_2}; SK_2) = 1$ from I_2.

Multi-Show Unlinkability and Selective Show of Data Items. The key idea that underlies the controlled release of certified data is to prove knowledge (in zero-knowledge) of a certificate instead of disclosing a certificate to the verifier. To show the certificate $Cert_1$ to the verifier without disclosing, e.g., the data items $m_1, \ldots, m_{l_1'}$ (where $l_1' \leq l_1$), the user (as the prover) and the (certificate) verifier (as the verifier in the proof of knowledge) compute a protocol such at the following

$$\begin{aligned}&\mathrm{PK}\{(Cert_1, m_1, \ldots, m_{l_1'}) :\\ &\qquad \mathrm{VerifySign}(Cert_1, m_1, \ldots, m_{l_1'}, m_{l_1'+1}, \ldots, m_{l_1}; VK_1) = 1\} \ . \qquad (1)\end{aligned}$$

Protocol (1) proves that the user has (knows) a valid certificate with respect to the verification key VK_1. By the zero-knowledge property of the protocol, the verifier does not learn any information on $Cert_1$ and the data items $m_1, \ldots, m_{l_1'}$. ¿From this observation it follows that multiple showings of the certificate $Cert_1$ using Protocol (1) are unlinkable, unless the data items $m_{l_1'+1}, \ldots, m_{l_1}$ disclosed to the verifier are linkable. The ability to selectively show data items follows trivially, as the user can choose, in each execution of Protocol (1), which data items to disclose to the verifier and of which data items to proof knowledge.

Proving Relations Between Data Items. This property is straightforward to achieve by using protocols such as the following one

$$\begin{aligned}\mathrm{PK}\{(&Cert_1, m_1, \ldots, m_{l_1'}, Cert_2, \tilde{m}_2, \ldots, \tilde{m}_{l_2'}) : \\ &\mathrm{VerifySign}(Cert_1, m_1, \ldots, m_{l_1'}, m_{l_1'+1}, \ldots, m_{l_1}; VK_1) = 1 \\ &\wedge \mathrm{VerifySign}(Cert_2, m_1, \tilde{m}_2, \ldots, \tilde{m}_{l_2'}, m_{l_2'+1}, \ldots, \tilde{m}_{l_2}; VK_2) = 1\}\ . \quad (2)\end{aligned}$$

Using protocol (2) the user can prove that she possesses a certificate $Cert_1$ from I_1 and a certificate $Cert_2$ from I_2. Additionally, she proves that the first data items m_1 and $\tilde{m}_1$ of the certificates are equal. Yet, by the zero-knowledge property the verifier does not learn the respective data items. Thus we see that demonstrating relations between certified attributes is achieved using techniques to prove knowledge of relations, such as equality.

Conditional Showing of Data Items. Let us assume that there is a third party which, using the algorithm SetupEnc, has created the encryption and decryption key pair (EK, DK). The encryption key EK shall be publicly known and authenticated. To show, e.g., the data item m_1 contained in $Cert_1$ conditionally, the user encrypts m_1 under the encryption key EK of the third party, i.e., $E = \mathrm{Enc}(m_1, Cond; EK)$. Here, $Cond$ denotes a label that describes the condition under which the user agrees m_1 to be released to the verifier. Then the user and the verifier execute the following protocol

$$\begin{aligned}\mathrm{PK}\{(&Cert_1, m_1, \ldots, m_{l_1'}) : \\ &\mathrm{VerifySign}(Cert_1, m_1, \ldots, m_{l_1'}, m_{l_1'+1}, \ldots, m_{l_1}; VK_1) = 1 \\ &\qquad \wedge E = \mathrm{Enc}(m_1, Cond; EK)\}\ . \quad (3)\end{aligned}$$

Besides of showing the certificate $Cert_1$, the user demonstrates in the protocol (3) that E is an encryption of the first data item contained in the certificate under the encryption key EK (such proofs are referred to as verifiable encryption). ¿From the zero-knowledge property of the protocol and security property of the encryption scheme, it follows that the verifier does not get any (computational) information on the value encrypted in E.

To obtain the data item m_1, the verifier sends E and $Cond$ to the third party. The third party verifies if the condition $Cond$ is fulfilled, and if so, he returns the decryption $m_1 = \mathrm{Dec}(E; DK)$ of E. We note that by the security property of the encryption scheme, the third party can not be fooled to decrypt under a condition other than the one described by $Cond$.

Blind Certification. Let us see how the user can get a certificate $Cert_3$ on data items m_1 and m' from issuer I_2 without disclosing m_1 to I_2, whereas the issuer I_2 can be asserted that m_1 is a data item certified by I_1; the data item m' is disclosed to the issuer. We recall that $Cert_1 = \mathrm{Sign}(m_1, \ldots, m_{l_1}; SK_1)$. To this end, the user commits to m_1, i.e., $C = \mathrm{Commit}(m_1, r)$. Then the user (as prover) and issuer (as verifier) execute the following protocol

$$\begin{aligned}\mathrm{PK}\{(Cert_1, r, m_1, \ldots, m_{l'_1}) : \quad & C = \mathrm{Commit}(m_1, r) \ \wedge \\ & \mathrm{VerifySign}(Cert_1, m_1, \ldots, m_{l'_1}, m_{l'_1+1}, \ldots, m_{l_1}; VK_1) = 1\} \ . \end{aligned} \tag{4}$$

With this protocol the user demonstrates the issuer that C is a commitment to the first data item contained in the certificate $Cert_1$ issued by I_1. From the zero-knowledge property of the protocol and the hiding property of the commitment scheme, it follows that the issuer does not get any information on the data item m_1. If protocol (4) is accepted by the issuer, then he issues the certificate $Cert_3$ on m' and hidden m_1 using the protocol

$$Cert_3 = \mathrm{HiddenSign}(C, m'; SK_2) \ , \tag{5}$$

where it is important to note that C is the same commitment as used in (4). ¿From the properties of HiddenSign, it follows that in protocol (5) the issuer learns m' but does not learn any information on m_1.

Finally, the user checks the correctness of $Cert_3$ by evaluation if

$$\mathrm{VerifySign}(m_1, m'; SK_2) = 1 \ .$$

3 Example Applications of the Framework

The controlled disclosure techniques described above have a large number of applications to privacy protection, such as anonymous credential systems [27, 15, 50, 45], group signature schemes [30,22,4], and electronic cash [29,9].

In this section we sketch how one can use these techniques to implement an anonymous credential system with identity revocation and e-cash with offline double-spending tests.

3.1 An Anonymous Credential System with Anonymity Revocation

The key idea underlying the implementation of anonymous credentials is that every user is represented by a unique identifier ID, which remains the user's secret throughout the lifetime of a credential system.

Now, a credential from an organization simply is a certificate on the identifier ID (issued by the organization). Credentials are shown by using protocols of the form (1), such that the user's identifier ID is not disclosed to the verifier. Then the *unlinkability* of credentials follows from the (multi-show) unlinkability property of certificates discussed above. Credentials are issued using blind certification such that the user's ID is not disclosed to the issuing organization. The *unforgeability* of credentials trivially follows from the unforgeability property of the signature scheme being used for blind certification.

A credential system is called *consistent*, if it is impossible for different users to team up and to show some of their credentials to an organization and obtain a credential for one of them that a user alone would not have gotten [45,43,16]. We achieve consistency as follows. When the user shows multiple credentials from different organizations she proves that the same identifer ID underlies all credentials being shown, i.e., that the credentials belong to the same user. To this end we use combined showing techniques as in protocol (2). When issuing credentials, the issuer asserts that the identifier ID it is blindly signing is the same as in existing credentials of the user. This can be achieved using the blind certification protocols described (4) and (5).

Optionally, credentials can have attributes. Examples of credential attributes are an expiration date, the users age, a credential subtype. When showing a credential, the user can choose which attribute(s) to prove something about, and what to prove about them. E.g., when showing a credential that has attributes $(\mathit{expdate} = 2002/05/19, \mathit{age} = 55)$, the user can decide to prove only that $\mathit{age} > 18$. Credential attributes are implemented by adding data items (additional to the user's identifier ID) to certificates. When showing credentials, the user can decide what information on attributes she discloses using the selective showing techniques described above.

Finally, in many applications of credentials it is desirable that under certain conditions the user's anonymity is revoked. Anonymity revocation can be implemented using our conditional showing techniques by conditionally disclosing the user's identity.

3.2 Anonymous e-Cash

Let us sketch an implementation of an anonymous e-cash system with offline double-spending tests. Such a system consists of banks issuing e-coins, users spending e-coins at shops, which in turn deposit spent coins at the bank.

An e-coin is a certificate issued by the bank. To retrieve an e-coin, the user identifies herself at the bank. The bank assigns a unique number ID to the user. The user secretly chooses a random serial number s and a random blinding number b. The bank issues a certificate Cert_{ecoin} on the data items ID, s, and b using blind certification such that it does not learn s and b.

At a shop the user spends the e-coin Cert_{ecoin} as follows. The shop chooses a random integer challenge c. The user computes $u = ID \cdot c + b$ and uses the following variant of a selective showing protocol

$$\begin{aligned}\mathrm{PK}\{(\mathit{Cert}_{ecoin}, ID, b, ID', b') :&\\ &\mathrm{VerifySign}(\mathit{Cert}_{ecoin}, ID, s, b;\, VK) = 1\\ &\wedge u = (ID' \cdot c + b') \wedge ID = ID' \wedge b = b'\}, \quad (6)\end{aligned}$$

where VK is the bank's signature verification key. We note that the shop learns the value of s in the proof (6). Here we additionally assume that the proof (6) can be carried out non-interactively, i.e., it can be represented in terms of string Π which is sent from the user to the shop. Such a non-interactive proof can be validated by the shop by applying an appropriate verification algorithm on

Π. Also, in analogy to the zero-knowledge property of interactive proofs, a non-interactive proof shall not reveal any (computational) information on $Cert_{ecoin}$, ID, and b.

To deposit the e-coin the shop sends the tuple (c, s, u, Π) to the bank. The bank first verifies the non-interactive proof Π to see if the tuple (c, s, u, Π) corresponds to a valid spending of an e-coin. In case of double spending the bank can recover the cheating user's ID as follows. The bank verifies if there already exists an e-coin with serial number s in its database of deposited e-coins. If so, it retrieves the corresponding tuple (c', s, u', Π'). We may safely assume that $c \neq c'$, and also we recall that by (6) the validity of Π asserts that $u = ID \cdot c + b$ and $u' = ID \cdot c' + b$. Therefore from u, u', c, and c' the bank can compute the user's identity $ID = (u - u')/(c - c')$. Thus we see why non-interactive proofs are needed: it is because the bank itself needs to be able to verify the correctness of the proof (6) to ensure it correctly reveals a cheating user's ID.

Other desirable properties of e-cash, such as unforgeability and anonymity immediately follow from the properties of our certificates and the associated controlled disclosure techniques discussed above.

4 Concrete Framework

In theory one could use any secure signature and encryption scheme for our framework, their combination by zero-knowledge proofs as described in the previous sections would in general not be efficient at all. Therefore we describe in this section concrete implementations of these scheme can to be efficiently combined. That is, they are all amendable to efficient proofs of knowledge of discrete logarithms.

4.1 Preliminaries

Notation. In the sequel, we will sometimes use the notation introduced by Camenisch and Stadler [22] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$\mathrm{PK}\{(\alpha, \beta, \gamma) : y = g^{\alpha} h^{\beta} \ \wedge\ \tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\gamma} \ \wedge\ (u < \alpha < v)\}$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers α, β, and γ such that $y = g^{\alpha} h^{\beta}$ and $\tilde{y} = \tilde{g}^{\alpha} \tilde{h}^{\gamma}$ holds, where $u < \alpha < v$*," where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The convention is that Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof-protocol can be described by just pointing out its aim while hiding all details. ¿From this protocol notation, it is easy to derive the actual protocol as the reader can see from the example we give below.

In the random oracle model, such protocols can be turned into signature schemes using the Fiat-Shamir heuristic [36,48]. We use the notation $\mathrm{SPK}\{(\alpha) : y = g^{\alpha}\}(m)$ to denote a signature obtained in this way and call it proof signature.

Throughout, we use ℓ_s as a parameter controlling the statistical indistinguishability between to distributions, ℓ_n as the length for RSA moduli that are hard to factor, and ℓ_q as a parameter such that discrete logarithms in a subgroup of order $q > 2^{\ell_q - 1}$ are hard to compute. Finally, we use ℓ_c as a parameter to denote the length of the challenges in the PK protocols.

Let a be a real number. We denote by $\lfloor a \rfloor$ the largest integer $b \leq a$, by $\lceil a \rceil$ the smallest integer $b \geq a$, and by $\lceil a \rfloor$ the largest integer $b \leq a + 1/2$. For positive real numbers a and b, let $[a]$ denote the set $\{0, \ldots, \lfloor a \rfloor - 1\}$ and $[a, b]$ denote the set $\{\lfloor a \rfloor, \ldots, \lfloor b \rfloor\}$ and $[-a, b]$ denote the set $\{-\lfloor a \rfloor, \ldots, \lfloor b \rfloor\}$.

Bi-linear Maps. Suppose that we have a setup algorithm BiLinMapSetup that, on input the security parameter ℓ_q, outputs the setup for $G = \langle g \rangle$ and $\mathsf{G} = \langle \mathsf{g} \rangle$, two groups of prime order $q = \Theta(2^{\ell_q})$ that have a non-degenerate efficiently computable bilinear map e. More precisely: We assume that associated with each group element, there is a unique binary string that represents it. (For example, if $G = \mathbb{Z}_p^*$, then an element of G can be represented as an integer between 1 and $p - 1$.) Following prior work (for example, Boneh and Franklin [7]), e is a function, $e : G \times G \rightarrow \mathsf{G}$, such that

- (Bilinear) For all $P, Q \in G$, for all $a, b \in \mathbb{Z}$, $e(P^a, Q^b) = e(P, Q)^{ab}$.
- (Non-degenerate) There exists some $P, Q \in G$ such that $e(P, Q) \neq 1$, where 1 is the identity of G.
- (Efficient) There exists an efficient algorithm for computing e.

We write: $(q, G, \mathsf{G}, g, \mathsf{g}, e) \in_R$ BiLinMapSetup(ℓ_q). It is easy to see, from the first two properties, and from the fact that G and G are both of the same prime order q, that whenever g is a generator of G, $\mathsf{g} = e(g, g)$ is a generator of G.

Such groups, based on the Weil and Tate pairings over elliptic curves (see Silverman [49]), have been extensively relied upon in cryptographic literature over the past few years (cf. [42,7,8,38] to name a few results).

4.2 Commitment Scheme

Pedersen's Commitment Scheme. There are several commitment schemes that are suitable for our purposes. The first one is due to Pedersen [47]. It uses elements g and h of prime order q such that $g \in \langle h \rangle$, where q is an ℓ_q-bit number.

To commit to a message $m \in_R \mathbb{Z}_q$ one chooses a random $r \in_R \mathbb{Z}_q$ and computes the commitment $C := g^m h^r$. The commitment can be opened by revealing m and r. To prove knowledge of the value contained in a commitment C, one can use the protocol denoted PK$\{(\mu, \rho) : C = g^\mu h^\rho\}$.

The Pedersen commitment scheme is information theoretically hiding and computationally binding. That is, a commitment does no leak *any* information about the committed message but someone who is able to compute the discrete logarithm $\log_h g$ can open the commitment to different messages. However, the commitment scheme can be turned into one that is computationally hiding and unconditionally binding: Let $C = (C_1, C_2) = (g^m h^r, g^r)$. Such a commitment can be opened by revealing m and r and PK$\{(\mu, \rho) : C_1 = g^\mu h^\rho \ \wedge \ C_2 = g^\rho\}$ can be used to prove knowledge of the message committed by it.

An Integer Commitment Scheme. The Pedersen commitment scheme can be used only to commit to elements of $\mathbb{Z}_q$. However, we sometimes need to commit to elements from $\mathbb{Z}$. Therefore, we describe the integer commitment scheme due Damgård and Fujisaki [33].

Let $\mathfrak{n}$ be the product of two safe $(\ell_n/2)$-bit primes $\mathfrak{p} = 2\mathfrak{p}' + 1$ and $\mathfrak{q} = 2\mathfrak{q}' + 1$, and $\mathfrak{g}$ and $\mathfrak{h}$ be two generators of $\mathfrak{G}_{\mathfrak{n}'} \subset \mathbb{Z}_{\mathfrak{n}}^*$, where $\mathfrak{n}' = \mathfrak{p}'\mathfrak{q}'$. Note that $\mathfrak{G}_{\mathfrak{n}'}$ is the subgroup of $\mathbb{Z}_{\mathfrak{n}}^*$ of order $\mathfrak{n}'$.

Assume that one is given $\mathfrak{n}$, $\mathfrak{g}$, and $\mathfrak{h}$ such that the factorization of $\mathfrak{n}$ as well as the value $\log_{\mathfrak{h}} \mathfrak{g}$ are unknown to at least the party computing the commitment. The parameters $\mathfrak{n}$, $\mathfrak{g}$, and $\mathfrak{h}$ could be for instance provided by a trusted third party. Then, one can commit to an integer $m \in \{0,1\}^{\ell_m}$, where ℓ_m is some public parameter, as follows: Choose a random $r \in_R [\mathfrak{n}/4]$ and compute the commitment $\mathfrak{C} := \mathfrak{g}^m \mathfrak{h}^r$. The commitment can be opened by revealing m and r. To prove knowledge of the value contained in a commitment C, one can use the protocol $\mathrm{PK}\{(\mu,\rho) : \mathfrak{C} \equiv \mathfrak{g}^\mu \mathfrak{h}^\rho \pmod{\mathfrak{n}}\}$.

Proving the Length of a Discrete Logarithm. Assume the availability of $\mathfrak{n}$, $\mathfrak{g}$, $\mathfrak{h}$ as above. Let $G = \langle g \rangle$ be a group of prime order q and let $y = g^m$ such that $-2^{\ell_m} < m < 2^{\ell_m}$, where $2^{\ell_m} < q2^{-\ell_s-\ell_c-1}$. To convince a verifier that $-2^{\ell_s+\ell_c+\ell_m} < m < 2^{\ell_s+\ell_c+\ell_m}$, the prover commits to x using the above integer commitment scheme, i.e., chooses a random $r \in_R [\mathfrak{n}/4]$,d computes $\mathfrak{C} := \mathfrak{g}^m \mathfrak{h}^r$, and then runs the protocol

$$\mathrm{PK}\{(\mu,\rho) : y = g^\mu \ \wedge\ \mathfrak{C} \equiv \mathfrak{g}^\mu \mathfrak{h}^\rho \pmod{\mathfrak{n}} \ \wedge\ -2^{\ell_m+\ell_s+\ell_c} < \mu < 2^{\ell_m+\ell_s+\ell_c}\}$$

with the verifier. As an example of how such a protocol can be derived from its notations, we spell this one out below.

The input to the both parities is g, y, $\mathfrak{n}$ $\mathfrak{g}$, $\mathfrak{h}$, $\mathfrak{C}$, ℓ_m, ℓ_s, and ℓ_c, where ℓ_s and ℓ_c are two security parameters. The prover, in addition get m and r in its input.

1. The prover chooses a random $r_\mu \in \{0,1\}^{\ell_s+\ell_c+\ell_m}$ and $r_\rho \in \{0,1\}^{\ell_s+\ell_c+\ell_n}$, where ℓ_n is the length of $\mathfrak{n}/4$, and computes $\tilde{y} := g^{r_\mu}$ and $\tilde{\mathfrak{C}} := \mathfrak{g}^{r_\mu}\mathfrak{h}^{r_\rho} \bmod \mathfrak{n}$ and sends $\tilde{y}$ and $\tilde{\mathfrak{C}}$ to the verifier.
2. The verifier replies with a randomly chosen $c \in \{0,1\}^{\ell_c}$.
3. The prover computes $s_\mu := r_\mu + cm$ and $s_\rho := r_\rho + cr$ and sends these values to the verifier.
4. The verifier accepts if the equations
$$\tilde{y} = y^{-c} g^{s_\mu}, \quad \tilde{\mathfrak{C}} \equiv \mathfrak{C}^{-c}\mathfrak{g}^{s_\mu}\mathfrak{h}^{s_\rho} \pmod{\mathfrak{n}}, \quad \text{and} \quad s_\mu \in \{0,1\}^{\ell_s+\ell_c+\ell_m}$$
hold. Otherwise the verifier rejects.

For the analysis of why the protocol indeed proves that $-2^{\ell_m+\ell_s+\ell_c} < \log_g y < 2^{\ell_m+\ell_s+\ell_c}$, we refer the reader to [21].

The above protocol can be extended to one that proves equality of discrete logarithms in two groups $\langle g_1 \rangle$ and $\langle g_2 \rangle$ of different order, say q_1 and q_2. That is, for $y_1 = g_1^m$ and $y_2 = g_2^m$ with $m0 \in \{0,1\}^{\ell_m}$ and ℓ_m such that $2^{\ell_m+\ell_s+\ell_c+1} < \min\{q_1, q_2\}$, it is not hard to see that the protocol

$$\mathrm{PK}\{(\mu,\rho) : y_1 = g_1^{\mu} \ \wedge \ y_2 = g_2^{\mu} \ \wedge \\ \mathfrak{C} \equiv \mathfrak{g}^{\mu}\mathfrak{h}^{\rho} \pmod{\mathfrak{n}} \ \wedge \ -2^{\ell_m+\ell_s+\ell_c} < \mu < 2^{\ell_m+\ell_s+\ell_c}\}$$

achieves this goal, where $\mathfrak{C}$ is a commitment to m as above.

4.3 The SRSA-CL Signature Scheme and Its Protocols

We now present the first signature scheme that is suited for our framework. The signature scheme has been proposed by Camenisch and Lysyanskaya and proven secure under the strong RSA assumption [17]. The strong RSA assumption was put forth by Baric and Pfitzmann [5] as well as be Fujisaki and Okamoto [37] and has be proven to be hard in the generic algorithms model [34]. Together with the signature scheme, Camenisch and Lysyanskaya have also put forth protocols to obtain a signature on committed messages and to prove knowledge of a signature on committed messages. In the following, however, we present more efficient protocols that use some research results that have appeared since.

The SRSA-CL Signature Scheme. Let ℓ_n, ℓ_m, and $\ell_e = \ell_m + 3$ be parameters. The message space of the signature scheme is the set $\{(m_1, \ldots, m_L) : m_i \in \pm\{0,1\}^{\ell_m}\}$.

Key generation. On input 1^{ℓ_n}, choose a ℓ_n-bit RSA modulus $n = pq$, where $p = 2p'+1$, $q = 2q'+1$, q', and p' are primes of similar size. Choose, uniformly at random $S \in_R \mathrm{QR}_n$ and $R_1, \ldots, R_L, Z \in_R \langle S \rangle$. Provide non-interactive proofs that $R_1, \ldots, R_L, Z \in_R \langle S \rangle$, e.g., run

$$\mathrm{SPK}\{(\rho_1, \ldots, \rho_L, \zeta) : \ R_1 \equiv S^{\rho_1} \pmod{n} \ \wedge \ \ldots \\ \ldots \ \wedge \ R_L \equiv S^{\rho_L} \pmod{n} \ \wedge \ Z \equiv S^{\zeta} \pmod{n}\}$$

using $\ell_c = 1$. Output the public key $(n, R_1, \ldots, R_L, S, Z, \ell_m)$ and the secret key p.

Signing algorithm. On input $m_1, \ldots, m_L$, choose a random prime number e of length $\ell_e + \ell_s + \ell_c + 1 > \ell_m + \ell_s + \ell_c + 3$, and a random number v of length $\ell_v = \ell_n + \ell_m + \ell_r$, where ℓ_r is a security parameter [17]. Compute the value A such that $Z \equiv R_1^{m_1} \ldots R_L^{m_L} S^v A^e \pmod{n}$. The signature on the message $(m_1, \ldots, m_L)$ consists of (e, A, v).

Verification algorithm. To verify that the tuple (e, A, v) is a signature on messages $(m_1, \ldots, m_L)$, check that $Z \equiv A^e R_1^{m_1} \ldots R_L^{m_L} S^v \pmod{n}$, and check that $2^{\ell_e+\ell_s+\ell_c+2} > e > 2^{\ell_e+\ell_s+\ell_c+1}$.

Theorem 1 ([17]). *The signature scheme is secure against adaptive chosen message attacks [41] under the strong RSA assumption.*

The original scheme considered messages in the interval $[0, 2^{\ell_m} - 1]$. Here, however, we allow messages from $[-2^{\ell_m} + 1, 2^{\ell_m} - 1]$. The only consequence of this is that we need to require that $\ell_e > \ell_m + 2$ holds instead of $\ell_e > \ell_m + 1$. Also, in the above scheme we require that $e > 2^{\ell_e+\ell_s+\ell_c+1}$, whereas in the original scheme $e > 2^{\ell_e-1}$ was sufficient.

Furthermore, an analysis of the security proofs shows that it is in fact sufficient if to chose the parameter v from $\mathbb{Z}_e$ [14]. However, if one uses this scheme to sign committed messages, then v should be chosen from a larger interval such that these messages are statistically hidden (cf. next paragraph).

Finally, to allow for a protocol to prove knowledge of a signature that is zero-knowledge, Camenisch and Lysyanskaya [17] required the signer to prove that n is the product of two safe primes, whereas due to the improved protocols presented below we require the signer only to prove that $R_i, Z \in \langle S \rangle$, which is considerably more efficient.

Obtaining of a Signature on Committed Messages. Let $c_1 = g^{m_1}h^{r_1}$, …, $c_{L'} = g^{m_{L'}}h^{r_{L'}}$ be commitments to messages and let $m_{L'+1}, \ldots, m_L$ be messages known to (and possibly chosen by) the signer. To get a signature on these messages, the signer and the recipient of the signature can execute the following protocol (cf. [17]):

The parties' common inputs are c_1, …, $c_{L'}$, $m_{L'+1}$, …, m_L, (n, R_1, …, R_L, S, Z, ℓ_m). The signer's secret input is p and q and the recipient secret input is m_1, …, $m_{L'}$, r_1, …, $r_{L'}$. The parties execute the following steps.

1. The recipient chooses a random integer $v' \in_R \{0,1\}^{\ell_n+\ell_s}$, computes $C := R_1^{m_1} \ldots R_{L'}^{m_{L'}} S^{v'} \bmod n$, and sends C to the signer.
2. The recipient runs the following proof protocol with the signer:

$$\begin{aligned} \mathrm{PK}\{(\varepsilon, \mu_1, \ldots, \mu_{L'}, \rho_1, \ldots, \rho_{L'}, \nu) :\ & c_1 = g^{\mu_1}h^{\rho_1} \ \wedge\ \ldots\ \wedge \\ & c_{L'} = g^{\mu_{L'}}h^{\rho_{L'}},\ \wedge\ C \equiv R_1^{\mu_1} \ldots R_{L'}^{\mu_{L'}} S^{\nu} \pmod{n} \wedge \\ & \mu_1, \ldots, \mu_{L'} \in \{0,1\}^{\ell_m+\ell_c+\ell_s}\} \end{aligned}$$

3. The signer chooses a random ℓ_e-bit integer e' such that $e := 2^{\ell_e+\ell_c+\ell_s+1} + e'$ is a prime. The signer also chooses a random $v'' \in \mathbb{Z}_e$, computes

$$A := \left(\frac{Z}{CR_{L'+1}^{m_{L'+1}} \ldots R_L^{m_L} S^{v''}}\right)^{1/e} \bmod n$$

and sends (A, e, v') to the recipient.

4. To convince the signer that $A \in \langle S \rangle$, she runs following proof protocol with the recipient.

$$\mathrm{PK}\{(\delta) : \quad A \equiv \pm\left(\frac{Z}{CR_{L'+1}^{m_{L'+1}} \ldots R_L^{m_L} S^{v''}}\right)^{\delta} \pmod{n}\}$$

5. The recipient verifies that $e > 2^{\ell_e+\ell_c+\ell_s+1}$ is prime and stores $(A, e, v := v' + v'')$ as signature on the message tuple $(m_1, \ldots, m_L)$.

Compared to the protocol presented in [17], the signer proves to the recipient that $A \in \langle S \rangle$. This is necessary to assure that the recipient can prove knowledge of a signature on committed messages such that the proof does not reveal any information about the signature or messages. The method we apply to prove $A \in \langle S \rangle$ was put forth in [12] to which we refer for details on why this proof actually works.

Prove Knowledge of a Signature on Committed Messages. Let $c_1 = g^{m_1} h^{r_1}, \ldots c_{L'} = g^{m_{L'}} h^{r_{L'}}$, be commitments to the messages $m_1, \ldots, m_{L'}$ that are not revealed to the verifier; and let $m_{L'+1}, \ldots, m_L$ be the messages that are revealed to the verifier. Let (e, A, v) is a signature on the messages $(m_1, \ldots, m_L)$, $L \geq L'$. To prove knowledge of this signature, keeping the messages $m_1, \ldots, m_{L'}$ secret, the prover and the verifier can use the protocol below which uses ideas put forth in [14].

The parties' common inputs are c_1, …, $c_{L'}$, $m_{L'+1}$, …, m_L, (n, R_1, …, R_L, S, Z, ℓ_m). The prover's secret input is m_1, …, $m_{L'}$, r_1, …, $r_{L'}$, and (e, A, v). The parties execute the following steps.

1. The prover chooses a random $r_A \in_R \{0,1\}^{\ell_n + \ell_s}$, computes $\tilde{A} := AS^{r_A}$, and sends $\tilde{A}$ to the verifier.
2. The prover executes the proof protocol

$$\begin{aligned} \mathrm{PK}\{(\varepsilon, \mu_1, \ldots, \mu_{L'}, \rho_1, \ldots, \rho_{L'}, \nu) :\ & c_1 = g^{\mu_1} h^{\rho_1} \ \wedge \ \ldots \ \wedge \ c_{L'} = g^{\mu_{L'}} h^{\rho_{L'}} \ \wedge \\ \frac{Z}{\tilde{A}^{2^{\ell_e + \ell_c + \ell_s + 1}} R_{L'+1}^{m_{L'+1}} \cdots R_L^{m_L}} & \equiv \tilde{A}^{\varepsilon} R_1^{\mu_1} \cdots R_{L'}^{\mu_{L'}} S^{\nu} \pmod{n} \ \wedge \\ & \varepsilon \in \{0,1\}^{\ell_e + \ell_c + \ell_s} \ \wedge \ \mu_1, \ldots, \mu_{L'} \in \{0,1\}^{\ell_m + \ell_c + \ell_s}\} \end{aligned}$$

   with the verifier.

4.4 The BM-CL Signature Schemes and Its Protocols

We now describe the second signature scheme that is suited for our purpose. The scheme was put forth by Camenisch and Lysyanskaya [18] and is based on the LSRW assumption introduced by Lysyanskaya et al. [45]: Let $G = \langle g \rangle$ be a group of prime order q, and let $X, Y \in G$, $X = g^x$, and $Y = g^y$. Now the assumption states that given triples $(a_i, a_i^y, a_i^{x + m_i xy})$ with randomly chosen a_i but for adaptively chosen messages $m_i \in \mathbb{Z}_q$ it is hard to computes a (a, a^y, a^{x+mxy}) with $m \neq m_i$ for all i. The assumption was proved to hold in the generic algorithms model [45]. The BM-CL signature scheme uses this assumption in the setting of bi-linear maps.

The Signature Scheme. The message space of the signature scheme is the set $\{(m_1, \ldots, m_L) : m_i \in \mathbb{Z}_q\}$. Its algorithms are as follows.

Key generation. Run the BiLinMapSetup algorithm to generate $(q, G, \mathsf{G}, g, \mathsf{g}, e)$. Choose $x \in_R \mathbb{Z}_q$, $y \in_R \mathbb{Z}_q$, and for $1 \leq i \leq L$, $z_i \in_R \mathbb{Z}_q$. Let $X = g^x$, $Y = g^y$ and, for $1 \leq i \leq L$, $Z_i = g^{z_i}$ and $W_i = Y^{z_i}$. Set $SK = (x, y, z_1, \ldots, z_L)$, $VK = (q, G, \mathsf{G}, g, \mathsf{g}, e, X, Y, \{Z_i\}, \{W_i\})$.

Signature. On input $(m_1, \ldots, m_L) \in \mathbb{Z}_q^L$, secret key $SK = (x, y, z_1, \ldots, z_L)$, and public key $VK = (q, G, \mathsf{G}, g, \mathsf{g}, e, X, Y, \{Z_i\}, \{W_i\})$ do:

1. Choose a random $v \in_R \mathbb{Z}_q$.
2. Choose a random $a \in_R G$.
3. Let $A_i = a^{z_i}$ for $1 \leq i \leq L$.
4. Let $b = a^y$, $B_i = (A_i)^y$.
5. Let $c = a^{x+xyv} \prod_{i=1}^{L} A_i^{xym_i}$.

Output $\sigma = (a, \{A_i\}, b, \{B_i\}, c, v)$.

Verification. On input $VK = (q, G, \mathsf{G}, g, \mathsf{g}, e, X, Y, \{Z_i\}, \{W_i\})$, a message tuple $(m_1, \ldots, m_L) \in \mathbb{Z}_q^L$, and purported signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c, v)$, check the following:

1. $\{A_i\}$ were formed correctly: $e(a, Z_i) = e(g, A_i)$.
2. b and $\{B_i\}$ were formed correctly: $e(a, Y) = e(g, b)$ and $e(A_i, Y) = e(g, B_i)$.
3. c was formed correctly: $e(X, a) \cdot e(X, b)^v \cdot \prod_{i=1}^{L} e(X, B_i)^{m_i} = e(g, c)$.

Theorem 2 ([18]). *The above signature scheme is correct and secure under the LRSW assumption.*

Obtaining of a Signature on Committed Messages. Let $c_1 = g^{m_1} h^{r_1}$, …, $c_{L'} = g^{m_{L'}} h^{r_{L'}}$ be commitments to messages $m_1, \ldots, m_{L'}$ that are chosen b the recipient and are not known the signer; and let $m_{L'+1}, \ldots, m_L$ be messages known to (or chosen by) the signer. To get a signature on these messages, the signer and the recipient of the signature can execute the following protocol (cf. [17]):

The parties' common inputs are c_1, …, $c_{L'}$, $m_{L'+1}$, …, m_L, ℓ_m and $(q$, G, G, g, g, e, X, Y, $\{Z_i\}$, $\{W_i\})$. The signer's secret input is $(x, y, z_1, \ldots, z_L)$ and the recipient secret input is m_1, …, $m_{L'}$, r_1, …, $r_{L'}$. The parties execute the following steps.

1. The recipient chooses a random $v \in_R \mathbb{Z}_q$ and computes $M := g^v \prod_{i=1}^{L} Z_i^{m_i}$. Next, the user gives a zero-knowledge proof of knowledge that M contains the same messages as the commitments $c_1, \ldots, c_{L'}$:

$$\mathrm{PK}\{(\nu, \rho_1, \ldots, \rho_{L'} \mu_1, \ldots, \mu_{L'}) : c_1 = g^{\mu_1} h^{\rho_1} \ \wedge \ \ldots \ \wedge \ c_{L'} = g^{\mu_{L'}} h^{\rho_{L'}} \ \wedge \ M = g^{\nu} \prod_{i=1}^{L'} Z_i^{\mu_i}\} \ .$$

2. The signer
   (a) chooses $\alpha \in_R \mathbb{Z}_q$, $a = g^{\alpha}$,
   (b) for $1 \leq i \leq L$, lets $A_i = a^{z_i}$, sets $b = a^y$, and for $1 \leq i \leq L$, lets $B_i = A_i^y$,
   (c) sets $c = a^x M^{\alpha x y}$, and
   (d) sends the recipient the values $(a, \{A_i\}, b, \{B_i\}, c)$.
3. The recipient stores the signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c, v)$.

Prove Knowledge of a Signature on Committed Messages. Let $c_1 = g^{m_1}h^{r_1}, \dots c_{L'} = g^{m_{L'}}h^{r_{L'}}$, be commitments to the messages $m_1, \dots, m_{L'}$ that are not revealed to the verifier; and let $m_{L'+1}, \dots, m_L$ be the messages that are revealed to the verifier. Let $(a, \{A_i\}, b, \{B_i\}, c, v)$ be a signature on messages $(m_1, \dots, m_L)$, $L \geq L'$. To prove knowledge of this signature, keeping the messages $m_1, \dots, m_{L'}$ secret, the prover and the verifier can use the protocol below.

The parties' common inputs are c_1, …, $c_{L'}$, $m_{L'+1}$, …, m_L, (q, G, G, g, g, e, X, Y, $\{Z_i\}$, $\{W_i\}$). The prover's secret input is m_1, …, $m_{L'}$, r_1, …, $r_{L'}$, and $(a, \{A_i\}, b, \{B_i\}, c, v)$. The parties execute the following steps.

1. The prover computes a blinded version of his signature σ: She chooses random $r, r' \in_R \mathbb{Z}_q$ and forms $\tilde{\sigma} = (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \tilde{c})$ as follows:
$$\tilde{a} = a^r, \quad \tilde{b} = b^r \text{ and } \tilde{c} = c^r$$
$$\tilde{A}_i = A_i^r \text{ and } \tilde{B}_i = B_i^r \text{ for } 1 \leq i \leq L$$
Further, she blinds $\tilde{c}$ to obtain a value $\hat{c}$ that it is distributed independently of everything else: $\hat{c} = \tilde{c}^{r'}$.
She then send $(\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \hat{c})$ to the verifier.
2. Let v_x, v_{xy}, $\mathsf{V}_{(xy,i)}$, $i = 1, \dots, L$, and v_s be as follows:
$$\mathsf{v}_x = e(X, \tilde{a}) \ , \ \mathsf{v}_{xy} = e(X, \tilde{b}) \ , \ \mathsf{V}_{(xy,i)} = e(X, \tilde{B}_i) \ , \ \mathsf{v}_s = e(g, \hat{c}) \ .$$
The prover and verifier compute these values (locally) and then carry out the following zero-knowledge proof protocol:
$$\begin{aligned} \mathrm{PK}\{(\varepsilon, \mu_1, \dots, \mu_{L'}, \rho_1, \dots, \rho_{L'}, \nu, \rho) : \\ c_1 = g^{\mu_1}h^{\rho_1} \ \wedge \ \dots \ \wedge \ c_{L'} = g^{\mu_{L'}}h^{\rho_{L'}} \ \wedge \\ \mathsf{v}_x^{-1} \prod_{i=L'+1}^{L} (\mathsf{V}_{(xy,i)})^{-m_i} = (\mathsf{v}_s)^{-\rho}(\mathsf{v}_{xy})^{\nu} \prod_{i=1}^{L'} (\mathsf{V}_{(xy,i)})^{\mu_i}\} \ . \end{aligned}$$
The Verifier accepts if it accepts the proof above and (a) $\{\tilde{A}_i\}$ were formed correctly: $e(\tilde{a}, Z_i) = e(g, \tilde{A}_i)$; and (b) $\tilde{b}$ and $\{\tilde{B}_i\}$ were formed correctly: $e(\tilde{a}, Y) = e(g, \tilde{b})$ and $e(\tilde{A}_i, Y) = e(g, \tilde{B}_i)$.

4.5 The CS Encryption and Verifiable Encryption

We finally describe an encryption scheme that fits our framework. The scheme was proposed by Camenisch and Shoup [21], who also provided a protocol that allows an encrypter to efficiently prove that a ciphertext contains a discrete logarithm (or an element of a representation). Camenisch and Shoup further provided the analogue, i.e., a protocol that allows a decryptor to prove that the decryption of a given ciphertext revealed a discrete logarithm. Such a protocol could for instance be used to ensure that a trusted third party behaves correctly. However, we do not present the latter protocol here.

The Camenisch-Shoup encryption scheme is based on Paillier's Decision Composite Residuosity (DCR) assumption [46] is that given only n, it is hard to distinguish random elements of $\mathbb{Z}^*_{n^2}$ from random elements of the subgroup consisting of all n-th powers of elements in $\mathbb{Z}^*_{n^2}$.

The Encryption Scheme. Let ℓ be a further security parameter. The scheme makes use a hash function $\mathcal{H}(\cdot)$ that maps a triple $(u, \{e_i\}, L)$ to a number in the set $[2^\ell]$. It is assumed that $\mathcal{H}$ is collision resistant, i.e., that it is computationally infeasible to find two triples $(u, \{e_i\}, L) \neq (u', \{e'_i\}, L')$ such that $\mathcal{H}(u, \{e_i\}, L) = \mathcal{H}(u', \{e'_i\}, L')$. Let $\text{abs} : \mathbb{Z}^*_{n^2} \to \mathbb{Z}^*_{n^2}$ map $(a \bmod n^2)$, where $0 < a < n^2$, to $(n^2 - a \bmod n^2)$ if $a > n^2/2$, and to $(a \bmod n^2)$, otherwise. Note that $v^2 = (\text{abs}(v))^2$ holds for all $v \in \mathbb{Z}^*_{n^2}$.

We now describe the key generation, encryption, and decryption algorithms of the encryption scheme, as they behave for a given value of the security parameter ℓ.

Key Generation. Select two random ℓ-bit Sophie Germain primes p' and q', with $p' \neq q'$, and compute $p := (2p' + 1)$, $q := (2q' + 1)$, $n := pq$, and $n' := p'q'$. Choose random $x_{(1,1)}, \ldots, x_{(1,L')}$, x_2, $x_3 \in_R [n^2/4]$, choose a random $g' \in_R \mathbb{Z}^*_{n^2}$, and compute $g := (g')^{2n}$, $y_{(1,i)} := g^{x_{(1,i)}}$ for $i = 1, \ldots, L'$, $y_2 := g^{x_2}$, and $y_3 := g^{x_3}$. The public key is $(n, g, \{y_{(1,i)}\}, y_2, y_3)$. The secret key is $(n, \{x_{(1,i)}\}, x_2, x_3)$.

Below, let $h = (1 + n \bmod n^2) \in \mathbb{Z}^*_{n^2}$ which is an element of order n.

Encryption. To encrypt a message tuple $(m_1, \ldots, m_{L'})$, $m_i \in [n]$, with label $L \in \{0,1\}^*$ under a public key as above, choose a random $r \in_R [n/4]$ and compute

$$u := g^r \ , \quad e_i := y^r_{(1,i)} h^{m_i} \quad (i = 1, \ldots, L'), \text{ and} \quad v := \text{abs}\left((y_2 y_3^{\mathcal{H}(u,\{e_i\},L)})^r\right) .$$

The ciphertext is $(u, \{e_i\}, v)$.

Decryption. To decrypt a ciphertext $(u, \{e_i\}, v) \in \mathbb{Z}^*_{n^2} \times (\mathbb{Z}^*_{n^2})^{L'} \times \mathbb{Z}^*_{n^2}$ with label L under a secret key as above, first check that $\text{abs}(v) = v$ and $u^{2(x_2 + \mathcal{H}(u,\{e_i\},L)x_3)} = v^2$. If this does not hold, then output reject and halt. Next, let $t = 2^{-1} \bmod n$, and compute $\hat{m}_i := (e_i/u^{x_{(1,i)}})^{2t}$. If all the $\hat{m}_i$ are of the form h^{m_i} for some $m_i \in [n]$, then output $m_1, \ldots, m_{L'}$; otherwise, output reject.

Theorem 3. *[[21]] The above scheme is secure against adaptive chosen ciphertext attack provided the DCR assumption holds, and provided $\mathcal{H}$ is collision resistant.*

Verifiable Encryption of Discrete Logarithms. Let $c_1 = g^{m_1} h^{r_1}, \ldots c_{L'} = g^{m_{L'}} h^{r_{L'}}$, be commitments to the messages $m_1, \ldots, m_{L'} \in \mathbb{Z}_q$. We now present a protocol that allows a prover to encrypt $m_1, \ldots, m_{L'} \in \mathbb{Z}_q$ and then to convince a verifier that the resulting ciphertext indeed encrypts the values contained in these commitment.

The protocol requires an integer commitment scheme, i.e., the auxiliary parameters $\mathfrak{n}$, and $\mathfrak{g}$ and $\mathfrak{h}$ such that the prover is not be privy to the factorization of $\mathfrak{n}$.

Recall that ℓ_c is a security parameter controlling the size of the challenge space in the PK protocols. Finally, we require that $q < n2^{-\ell_s-\ell_c-3}$ holds, i.e., that $m_i \in \mathbb{Z}_q$ "fits into an encryption". (If this condition is not meet, the m_i's could be split into smaller pieces, each of which would then be verifiable encrypted. However, we do not address this here.)

The common input of the prover and verifier is: the public key $(n, g, \{y_{(1,i)}\}, y_2, y_3)$ of the encryption scheme, the additional parameters $(\mathfrak{n}, \mathfrak{g}, \mathfrak{h})$, a group element (δ), a ciphertext $(u, \{e_i\}, v) \in \mathbb{Z}_{n^2}^* \times (\mathbb{Z}_{n^2}^*)^{L'} \times \mathbb{Z}_{n^2}^*$, and label L. The prover has additional inputs $m_1, \ldots, m_{L'} \in \mathbb{Z}_q$ and $r \in_R [n/4]$ such that

$$u = g^r, \qquad e = y_{(1,i)}^r h^{m_i}, \quad \text{and} \qquad v = \operatorname{abs}\left((y_2 y_3^{\mathcal{H}(u,e,L)})^r\right) .$$

The protocol consists of the following steps.

1. The prover chooses a random $s \in_R [\mathfrak{n}/4]$ and computes $\mathfrak{k} := \mathfrak{g}^m \mathfrak{h}^s$. The prover sends $\mathfrak{k}$ to the verifier.
2. Then the prover and verifier engage in the following protocol.

$$\begin{aligned} \mathrm{PK}\{(r, m, s) : (\nu, \rho_1, \ldots, \rho_{L'} \mu_1, \ldots, \mu_{L'}) : \\ c_1 = g^{\mu_1} h^{\rho_1} \ \wedge \ \ldots \ \wedge \ c_{L'} = g^{\mu_{L'}} h^{\rho_{L'}} \ \wedge \\ u^2 = g^{2r} \ \wedge \ v^2 = (y_2 y_3^{\mathcal{H}(u,e,L)})^{2r} \ \wedge \\ e^2 = y_{(1,1)}^{2r} h^{2\mu_1} \ \wedge \ \ldots \ \wedge \ e^2 = y_{(1,L')}^{2r} h^{2\mu_{L'}} \ \wedge \\ \mathfrak{k}_1 = \mathfrak{g}^{\mu_1} \mathfrak{h}^s \ \wedge \ \ldots \ \wedge \ \mathfrak{k}_{L'} = \mathfrak{g}^{\mu_{L'}} \mathfrak{h}^s \ \wedge \ -n/2 < \mu_i < n/2\} \ . \end{aligned}$$

5 Bibliographic Notes

Chaum pioneered privacy-preserving protocols that minimize the amount of personal data disclosed. His work put forth the principles of anonymous credentials [25,27,28], group signatures [30], and electronic cash [26]. The inventions of zero-knowledge proofs [40,11] and zero-knowledge proofs of knowledge [6] have made a tremendous impact in this area. In particular, Damgård gave the first proof of concept [35] of an anonymous credential where a credential was represented by a signature on an individual's name, obtained in a way that kept the name hidden from the issuer; while showing a credential was carried out via a zero-knowledge proof of knowledge. These first developments were of great theoretical interest, but did not suggest solutions that were usable in practice.

Brands [10] invented efficient techniques for proving relations among committed values. The applications of these include electronic cash and a system where individuals do not have to disclose all attributes of their public keys in all contexts, but can choose which subset of attributes to disclose, or can choose to prove more complex relations among attributes. His techniques fell short of a full-fledged privacy framework as we describe above, because they did not have multi-show unlinkability. That is, if a user used his public key in several transactions, even though in each transaction he had a choice of which attributes to disclose, still in every transaction he had to disclose some information uniquely linkable to his public key.

Research on group signatures further contributed to the development of the key techniques for our privacy framework. In a group signature scheme, each member of the group can sign on behalf of the entire group such that one cannot determine which signer produced a given signature, or even whether two given signatures were produced by the same signer. What makes it non-trivial is that, in case of an emergency, a group signature can be traced to its signer with the help of a trusted third party. Camenisch and Stadler [22] came up with the first group signature scheme where the size of the public key was independent of the size of the group. Subsequent work in this area [24,20] put forth a more general framework for group signatures. Finally, Ateniese et al. [4] invented the first provably secure group signature scheme (see also Camenisch and Michels [19] and Cramer and Shoup [32] that paved the way for the Ateniese et al. scheme).

Anonymous credential systems as described above were introduced by Lysyanskaya et al. [45]. The first efficient and provably secure scheme was put forth by Camenisch and Lysyanskaya [16], whose construction was largely inspired by the Ateniese et al. group signature scheme construction.

The foundation of our framework and the first key building block — a signature scheme with efficient protocols — was identified as such in a further study on anonymous credentials. Generalizing prior work [16], Lysyanskaya [44] showed how to obtain an anonymous credential system using this building block. The SRSA-CL signature scheme [17,44] described in this paper emerged as a generalization of the techniques needed for anonymous credentials. Camenisch and Groth [14] made further improvements to the parameters of this signature and to the associated protocols; the parameters and protocols described in the present paper reflect these improvements. This signature scheme and associated protocols have since been implemented as part of the Idemix project at IBM [23], and incorporated into the TCG standard as part of the direct anonymous attestation protocol [12]. The BM-CL signature scheme described above was invented very recently [18]. It has not been implemented yet, but it is also quite practical.

The other key building block — verifiable encryption — was introduced by Camenisch and Damgård [13] and independently by Asokan, Shoup, and Waidner [3], as part of efforts in group signature scheme design and fair exchange of digital signatures, respectively. The verifiable encryption scheme described here is due to Camenisch and Shoup [21] and is the state-of-the-art.

Acknowledgement

The information in this document 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. Part of the work reported in this paper is supported by the IST PRIME project; however, it represents the view of the authors only. The PRIME project receives research funding from the Community's Sixth Framework Programme and the Swiss Federal Office for Education and Science. Anna Lysyanskaya is supported by NSF Career grant CNS-0347661.

References

1. Portia project, website. `crypto.stanford.edu/portia`.
2. PRIME project, website. `www.prime-project.eu.org`.
3. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, April 2000.
4. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270. Springer Verlag, 2000.
5. Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *LNCS*, pages 480–494. Springer Verlag, 1997.
6. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *LNCS*, pages 390–420. Springer-Verlag, 1992.
7. Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer Verlag, 2001.
8. Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. In *Topics in Algebraic and Noncommutative Geometry, Contemporary Mathematics*, volume 324, pages 71–90. American Mathematical Society, 2003.
9. Stefan Brands. Untraceable off-line cash in wallets with observers. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, volume 773 of *LNCS*, pages 302–318, 1993.
10. Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.
11. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, October 1988.
12. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. Technical Report Research Report RZ 3450, IBM Research Division, March 2004.
13. Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer Verlag, 2000.
14. Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In submission.
15. Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. Technical Report Research Report RZ 3295, IBM Research Division, November 2000.
16. Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2001.

17. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002*, volume 2576 of *LNCS*, pages 268–289. Springer Verlag, 2003.
18. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO 2004 (to appear)*, LNCS. Springer Verlag, 2004.
19. Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In Kazuo Ohta and Dinqyi Pei, editors, *Advances in Cryptology — ASIACRYPT '98*, volume 1514 of *LNCS*, pages 160–174. Springer Verlag, 1998.
20. Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pages 413–430. Springer Verlag, 1999.
21. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144, 2003.
22. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer Verlag, 1997.
23. Jan Camenisch and Els Van Herreweghen. Technical report.
24. Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem.* PhD thesis, ETH Zürich, 1998. Diss. ETH No. 12520, Hartung Gorre Verlag, Konstanz.
25. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
26. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology — Proceedings of CRYPTO '82*, pages 199–203. Plenum Press, 1983.
27. David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
28. David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pages 118–167. Springer-Verlag, 1987.
29. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *LNCS*, pages 319–327. Springer Verlag, 1990.
30. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *LNCS*, pages 257–265. Springer-Verlag, 1991.
31. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *LNCS*, pages 13–25, Berlin, 1998. Springer Verlag.
32. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *Proc. 6th ACM Conference on Computer and Communications Security*, pages 46–52. ACM press, November 1999.
33. Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *LNCS*. Springer, 2002.

34. Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In Lars Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 256–271. Springer, 2002.
35. Ivan Bjerre Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *LNCS*, pages 328–335. Springer Verlag, 1990.
36. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer Verlag, 1987.
37. Eiichiro Fujisaki and Tatsuaki Okamoto. Witness hiding protocols to confirm modular polynomial relations. In *The 1997 Symposium on Cryptograpy and Information Security*, Fukuoka, Japan, January 1997. The Institute of Electronics, Information and Communcation Engineers. SCSI97-33D.
38. Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer Verlag, 2002.
39. Oded Goldreich. *Foundations of Cryptography II: Basic Applications.* Cambridge University Press, 2004.
40. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. In *Proc. 27th Annual Symposium on Foundations of Computer Science*, pages 291–304, 1985.
41. Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
42. Antoine Joux. A one-round protocol for tripartite Diffie-Hellman. In *Proceedings of the ANTS-IV conference*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.
43. Anna Lysyanskaya. *Signature schemes and applications to cryptographic protocol design.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.
44. Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.
45. Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *LNCS*. Springer Verlag, 1999.
46. Pascal Paillier. Public-key cryptosystems based on composite residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–239. Springer Verlag, 1999.
47. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *LNCS*, pages 129–140. Springer Verlag, 1992.
48. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *LNCS*, pages 387–398. Springer Verlag, 1996.
49. Joseph Silverman. *The Arithmetic of Elliptic Curves.* Springer-Verlag, 1986.
50. Eric Verheul. Self-blindable credential certificates from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 533–551. Springer Verlag, 2001.

A Cryptographic Framework for the Controlled Release of Certified Data (Transcript of Discussion)

Endre Bangerter

IBM Zurich Research Laboratory, Switzerland

Welcome to this talk on linked credential systems and controlled release of certified data. This talk is actually about what cryptographics can do for mass protection; more precisely it's about real world cryptographic techniques to control the release of certified data.

The structure of the talk is as follows: I will start with an outline of the problem to be solved. I then want to discuss a crypto framework for controlled release of data, and as an application of this framework I am going to show how one can construct a truly anonymous and practical credential system. I conclude with some open problems.

Let me recall some well-known facts. One is that once personal data is released, it is out of control, it can be fully distributed. If there is data on the same person from more than one source, then it can be linked. All this is particularly bad in the digital world where these things can be done quickly at a low cost. A key principle to avoid such problems is to actually not release data, or only to release minimal data. Here minimal means release only data which is really needed for the purpose for which the data is given. And while the digital world is a threat to privacy, I think it can also be privacy-enabling in the sense that in the digital world you can implement technologies and solutions which are not doable in the paperbased world. So, I'm actually going to show you techniques that won't work in the physical world but can be done in the digital world.

I start with an extended example which underlies the entire talk. It is about privacy authentication. The scenario is that we have a customer here who wants to rent a car. She will have to have a driver's licence from some administration; the licence will contain name, address, age, etc. She will also need a credit card containing her name as well. Then she shows these documents to the car rental company, and provides them with all this data. This is not necessarily bad because for instance if the customer decides to steal a car I will need some data about the customer to call the police, so some data might really be required. On the other hand the car rental company really also learns private data about the customer like her address. Moreover, as for instance on the driver's licence and the credit card, you have the same name of the person, so you have this problem of data linking. The car rental company, the licence administration and the bank can work together to compare the data on the customer.

I think it's important to put this scenario into contrast with the optimal solution where data is really minimised. So what would this solution be like? If you look at the car rental company, actually all the company needs to know is that the customer does have a driver's licence, and that she has a valid credit card. Only when some conflict occurs, when she doesn't bring back the car for instance, will the car rental company need to have some identity, but not otherwise. So that's the optimal solution. Let's see how this scenario is typically implemented in the digital world, and then I will come back to the optimal solution.

I think it is well known the standard implementation of this is using digital signatures. Each organisation will choose some signing keys, and verification keys etc. The driver's licence is replaced by a certificate which is a signature, a digital identity, and has attributes like name, age, etc. The same is true for the credit card, and showing documents is equivalent to verifying digital signatures. So this is a one-to-one translation of the paper-based scenario to the digital world. The problems are the same. What one can do to improve the customer's privacy is, for instance, to introduce pseudonyms. That would mean that the customer has a different name with the administration, and another name with the bank, so these two cannot link the customer data. But that doesn't really solve the problem. It's not the optimal solution because when these certificates are shown to the car rental company, the company will learn these pseudonyms and will still be able to link the data with their organisations.

So of course the question is, can we reach the optimal solution, which I have outlined before, using practical technology? The answer is yes, and really, in the sense you can do that practically and efficiently, not just theoretically. The idea is, instead of giving certificates, or sending certificates, to the car rental company, to prove that one has a certificate without giving it away.

So let me briefly recall the role of crypto in this whole setting. It is more or less well known that our so-called generic efficient cryptographic techniques are very powerful, and they allow you to potentially implement any two-party, or multi- party, protocol. You can do private things, you could also implement such an optimal system, which by the way I call an anonymous credential system. However the problem with these generic techniques is that efficient means probabilistic polynomial time algorithms, but not the efficiency which is needed in the real world solution where you need algorithms that run maybe in seconds or fractions of seconds so that you can build interactive systems.

So there is also lots of practical crypto, which is much more efficient, but less generic, and such techniques have been used to implement many cryptographic solutions, such as e-cash, group signatures, etc.

So the focus of this talk is on efficient techniques.

Matt Blaze: I'm seeing a little bit of a disconnect between your example and here. Did you intend the car rental example to be a practical example for the use of anonymous credentials? Or are you using it just as a simplifying, motivating example. In particular, it takes ten or fifteen minutes to rent a car. A lot of that time is spent verifying your credentials. They look at your driver's licence, your credit card, they do online database look-ups. In fact we tolerate quite a lot of inefficiency there, and we don't get any anonymity at all. So I'm not sure your assertion about efficiency matches the example you gave.

Reply: If you don't like renting a car, you can come up with some system where you want to have instant access. For instance, if you want to read the New York Times online anonymously, you won't wait 20 minutes until you get an article to display on the machine.

There's no reason not to rent cars efficiently by the way; it would be much faster if you use efficient systems.

But you can think for yourself of some examples where efficiency is required for authentication. In a hospital you have certain emergency situations where you want to access some resource which has to be done quickly. And, these theoretical techniques I've been talking about can run for hours or days.

There are efficient constructions for particular solutions like e-cash etc. I'm going to focus not on a particular solution, but rather on a framework of cryptographic abilities which are efficient, and which are designed to control the use of data. By using this framework one can build things like group signatures and credential signatures etc, so this framework is more generic than these particular solutions, because it was designed to solve privacy problems.

So let me show you this framework. I apologise for my machine. It doesn't want to show that slide.

Matt Blaze: It happened as soon as I asked my question. [laughter]

Reply: The framework actually is a bunch of cryptographic primitives like commitments, encryptions, signatures, and so-called zero knowledge proofs. These primatives were designed to work well together informally. Before I explain to you how these primitives work together, I will briefly recall the notion of a zero knowledge proof of knowledge, which is important in this context. A zero knowledge proof of knowledge is a protocol between a prover and a verifier, where the prover knows some secret. In this example the secret would be the secret recipe of Coca Cola. The goal of the protocol is to convince the verifier that the prover knows the secret. The trivial solution of course would be to send the recipe to the verifier, but we want something more sophisticated than that; what we want is that the prover can convince the verifier, without giving away the secret. That's actually what a zero knowledge proof of knowledge does. Zero knowledge means that you can convince the verifier without the verifier learning the secret. These are really key protocols in cryptographic privacy applications. Now what you have for instance is an encryption scheme, a signature scheme designed such that I can encrypt a signature, send that signature to someone, and then use zero knowledge proof of knowledge that the encryption really contains a valid signature. And you will see in a moment, such combinations are quite powerful, such that they can be used for designing a functionality to really have controlled release of data. I'm going to give two examples of this.

For instance, in this framework we can combine encryption with zero knowledge proof of knowledge, to get so called verifiable encryption. Verifiable encryption can be used to conditionally release data, and I come back to the example of the car rental scenario. Let us assume that the customer has anonymously shown her driver's licence and her credit card, so the car rental company has no information on who the customer is except that she has the documents. We're facing the problem of what happens when the lady doesn't return the car. Verifiable encryption solves this problem, as follows; we will have the police or the law enforcement agency set up an encryption scheme; they choose an encryption key, and the decryption key, and make the encryption key public. What the customer now does when she rents the car, she will encrypt her identity under the encryption key of the police, she can do that in the public key encryption scheme. Then she sends the encryption to the car rental company, and additionally proves in zero knowledge that the encryption really contains her identity, so the car rental company can be certain that they have an encryption which contains the customer's identity, but because it's an encryption they have no information on the customer, so far. However when the customer decides not to show up with the car, the car rental company can send the encryption to the police, who can then decrypt her identity using the decryption key, so they can identify the person.

So we clearly have a scenario of conditional release of data. The identity of the customer is conditionally released to the car rental company. What is also important here is that the car rental company can get her identity without the cooperation of the customer. The trust model underlying this application is that the customer and the car rental company don't trust each other at all, they don't need to trust each other at all, but both trust the law enforcement to decrypt, and to reveal the identity of the customer when necessary. So this is an optimal solution to one of the problems we were facing at the beginning, that it is a problem to reveal one's identity.

Bruno Crispo: But to solve the problem, is it not enough to issue an anonymous driver's license? Then just if something goes wrong, once you have got a problem to solve, then law enforcement can just ask the people that issued the anonymous driver's licence, what is the link between the real user and the anonymous licence. They will even issue the driving licence to a pseudonym. Why do we need to put the identity in the driver's licence to solve the problem?

Reply: The point is that under this system the car rental company has no information on the customer, all they know is that she has a licence, and she has a credit card. That's the optimal solution, we have no information, they could be anyone.

Tuomas Aura: So the police can issue a driver's licence to a customer that doesn't have the customer's name on it?

Reply: Yes.

Bruce Christianson: The question that's being asked is why we have to use zero knowledge at all, why not simply put the unique identity information in the credential?

Reply: But of course we can do that; that's trivial, but then the car rental company will know the identity of the customer, and that's exactly what you want to avoid.

Any information that is on certificate can be used for linking later on, and that's the problem we want to avoid, that's not the optimal solution.

Bruno Crispo: Yes, but the certificate doesn't need to be an identity certificate, they can be just pseudonym certificates.

Reply: Yes, but that doesn't solve the problem either.

Tuomas Aura: It can be used only once, you know.

Bogdan Popescu: It can be issued multiple times but have different public key which can be issued every day and access your driver's licence.

Reply: Sure, it will do that, but these are throw-away credentials. OK, these solutions are not really practicable because each time you show something you have to get a new credential set up.

Bogdan Popescu: Set up multiple credentials at the beginning, and then you show them one by one.

Reply: You still have the problem that you have to get new credentials all the time. This solution is in that sense better, because we only get one credential once. Otherwise you would have to contact the motor vehicle association each time you show your driver's licence to get a new credential. This will take lots of online transactions, our solution has no such online transactions.

Fabio Massacci: But you don't rent a car very often.

Reply: It's just an example, you can think of anything else.

Matt Blaze: In your example the police are acting essentially as an online oracle for revealing identities, or maybe a semi-online oracle for revealing identities.

Reply: Semi-online would be close I think, yes.

Matt Blaze: I'm sorry to keep harking on your car rental example, but in fact there are many different reasons in the legal system for revealing the identity of the driver, that are much less serious than car theft. For example, there might be an accident, when you're returning the car it might not have had the full tank of gas that you were required to return it with, it might have been left dirty. There are many reasons that one might exercise this oracle, and get the identity. I'm sorry your computer broke again, just as soon as I asked a question. I must try to use this power for good, not for evil. But if this oracle can be exercised for a wide variety of reasons, then what real privacy service that the user can depend on is really being provided here? If there is a semi-online service that will reveal your identity, and it can be used essentially at will, can the user really depend on their identity being kept secret?

Reply: That's part of trust. You trust law enforcement, or whatever agencies, only to reveal the identity when really necessary. That's certainly a problem but it is outside the system. Of course the system doesn't resolve all privacy problems.

Matt Blaze: But in the particular application you showed us, there may be more instances where the identity is revealed than where it isn't revealed.

Reply: I don't mind if you don't like my example but you can easily translate this example to any example you like.

Matt Blaze: I'm not convinced it's so easy. In many of these examples where there exists an oracle we'll end up finding out that the oracle is used more than we originally expected it to be used.

Reply: I put that in the matter of access control. For instance, our encryption scheme also supports so-called labels, but these labels are typed in encryption. These labels will contain for instance the condition on which the encryption might be opened. So if you rent a car and you could activate an encryption, the customer only agrees to her identity being revealed if she doesn't bring back the car, not if the car breaks down or something else. So actually we have a solution for that within the encryption scheme, and so maybe it's time to link in some access control systems.

Bruno Crispo: The problem there is that you are really upholding the conditions in the messages, which means that if you just forget a condition, or if a new condition comes up, then it's very difficult to...

Reply: Look, I think you don't want that. The customer wants the conditions to be fixed when she gives her identity, she doesn't want conditions to be changed. That would open the door to abuse.

Bruno Crispo: But a new condition may arise, even if we don't want.

Reply: I'm not so sure. They may arise, but I give away my data under a policy. That's another privacy principle, that you give away your data under certain policies which make completely clear on what conditions your data may be given away.

George Danezis: Is it a cryptographic condition or is it a plaintext label?

Reply: No, this label is a string, a string you can import whenever you like.

George Danezis: It's a string that says, do not open me unless there was car theft.

Reply: It could be free text; it could be something else.

Pasi Eronen: All these things assume that the law enforcement can actually determine if the condition holds. If the car breaks down, law enforcement doesn't want to spend its time verifying that; well we got this request from the car rental company:- did this condition actually hold?
Reply: Sure, that's right.

Pasi Eronen: It would be simpler if the law enforcement simply gave the identity back and notified the customer that now we gave your identity.

Reply: You're completely right but that again is outside of the technical problems. That is a question if the law enforcement agency is careful or not.

Tuomas Aura: No, it is outside the cryptographic problems here. It's a research problem, but it is certainly not outside the design of a sensible system for car rental companies; it's actually these kind of conditions on the boundary that usually break down, So it is very important to consider it.

Reply: I don't say it's not important, I think they are very important. But this is a layer of technology for doing anonymous authentication, on top of this layer you will have many other technical layers; access control infrastructures, policy problems, etc. These technologies don't solve these problems, I completely agree, but they solve the authentication problems. They're a good solution to that problem. On top of that you can provide other good solutions, that's the idea.

Bruce Christianson: Yes, but I think the argument is that the rabbit is now hiding under the trust relationship between the customer and law enforcement.

George Danezis: And I'm afraid I do not believe that this escrow system, let's call it what it is, does not have other conditions that actually you can do without. I don't think that just assuming that the customer and law enforcement are on the best of terms (when clearly there is a conflict there) is a realistic kind of working premise.

Because in fact most of these systems are going to trust the car rental people more than the police. I don't basically have a problem dealing with the car rental company, I would actually mind if it automatically went to the police.

Reply: That problem doesn't arise if you are someone who trusts the car rental company. Then you can trust that car rental company will never give away your encrypted identity to the law enforcement. You don't have a problem if you trust them.

George Danezis: But can you also include some accountability mechanisms for this decryption process that demonstrate compliance with the policy?

Reply: That is right, it is difficult to comply. There are some mechanisms for verified decryption which allow us to renew the trust into the law enforcement, because law enforcement could, for instance, decrypt data and reveal completely different identities. By using verified decryption you could avoid that problem.

So back to the framework. We also have what we call Sent Signatures. This is a combination of signatures, commitments, and zero knowledge proof of knowledge. What they allow you to do is to selectively hide messages from the signer of a message, and also to selectively hide messages from the verifier of the signature. So back to the car rental example again. Here instead of using an ordinary signing algorithm the customer gets a regular signature of exact use. What is particular about this system is that the customer can control which information the signer learns, so in this example here, the administration will learn only whether she wears eyeglasses or not.

A similar thing happens when showing the certificate. The customer can exactly control what information the car rental company learns about the certificate. For instance, the company certainly learns whether the certificate is correct or not, but the customer can hide their address, their age, whatever, and can also selectively choose only to reveal that her age is bigger than 25. It is a very focused way to release information containing certificates.

Fabio Massacci: Does the customer need to have a very powerful PDA with her that does all these things?

Reply: Yes, it's an extra requirement.

Fabio Massacci: So if you don't have computing power as a customer, then you are dead in the water from the very beginning.

Reply: Well these are all protocols, which are implemented in computers.

Bob Mayo: All she has to do is have the ability to call up the computing power somewhere else, maybe on the Internet, but in a trusted way.

Reply: Right, that would be a solution. The efficiency of this technology is another question, and maybe these things will run easily on PDAs.

Tuomas Aura: In respect to the earlier discussion, it seems that if we think of anonymity as something which is an absolute requirement, that you must not reveal this name, and the cost of losing that anonymity is infinite, then none of this kind of system can be designed in practice to work. But if we put some fixed price on giving out your identity, like $100, then we probably can design it sensibly so that if your identity is revealed without your own fault then you could get back the $100.

Matt Blaze: No I disagree. I'm sure that's very difficult to price reasonably. You are invulnerable perhaps to a broad attack but extremely vulnerable to targeted attacks in which it is worth $100 to the attacker. So your solution works well against data mining, but it doesn't work well against targeted threats.

Reply: OK, what might be more interesting, or at least less controversial, is that we have implemented this system in a research prototype. For laptop users using JAVA to issue these credentials we have a running time of one or two seconds, and showing credentials, and verifying credentials, one to three seconds, so I think it's quite reasonable for a prototype, especially without optimisations, there's lots of computational algebra inside the system which could improve.

Audience: How about on a PDA?

Reply: Sure. I mean, you have to think ahead maybe five years, but it's not completely out of sight. Then there could also be dedicated PDAs, and you could also look at the scenario where we have access to the Internet, etc.

I think most of the questions that have been raised at this point of the outlook are about how to integrate this technology in the future infrastructure. I know that's an important problem, I don't want to deny that, but there are lots of pressures to make precise what would be the access control infrastructure, the service infrastructure, how could I satisfy credentials, that's tackling the PDA question. These problems are not yet solved. The same holds true for what would be a nice interface for the user to handle these credentials sensibly, you need to have some standard, there need to be warnings when users give away too much information etc, etc. These are certainly open questions and not solved by using technology.

One User, Many Hats; and, Sometimes, No Hat: Towards a Secure Yet Usable PDA

Frank Stajano

University of Cambridge

Abstract. How can we design a PDA that is at the same time secure and usable? In current implementations the two properties are mutually exclusive. Because normal users find password entry inconvenient, the balance usually shifts away from security, leaving the PDA vulnerable if lost or stolen.

We begin by envisaging what an ideal PDA authentication mechanism might look like and by carefully examining alternatives to passwords such as tokens and biometrics.

We then expose another aspect of the security vs. usability problem. In many cases, when we turn on our PDA, we only access functionality (dictionary, calculator, web browser...) that requires no access to private data stored in the machine; why, then, should we pay the usability penalty of authentication in such cases? Moreover, we may want to grant another person temporary access to such "harmless" functionality, but without being forced to grant them unrestricted access to the whole machine.

To solve this problem we describe a system in which we may assign more than one "hat" to the owner of this single-user device, with each hat having specific privileges. The machine supports concurrent graphical logins for several hats and a convenient mechanism to switch between them. There is also provision for a userid associated with "no hat", to which one can switch without the need for authentication, and which can access all the harmless functionality. This scheme turns out to be applicable and useful well beyond the limited realm of PDAs.

1 Introduction

Nowadays, a usable and secure PDA[1] is an oxymoron. The two properties are more or less opposites: the usable PDA, like a paper diary, is accessible as soon as you open the cover; but anyone who finds it can read (and alter) all the data it contains. The secure PDA, instead, requests a password every time it wakes up, which makes it quite inconvenient for the legitimate owner to look up an appointment or address.

[1] Personal Digital Assistant, a pocket-sized computer with the primary function of an organizer. It usually works as a diary, address book and notepad. More advanced models may also act as an email client and web browser. The best modern PDAs have an open architecture and allow the user to rewrite the ROM with the operating system; some even run Linux natively.

The amount of sensitive information entrusted to our digital butlers keeps growing [1]. The desire for a PDA that would stay secure even if lost or stolen is therefore justified. In practice, though, it is only hardened crypto-geeks who heroically withstand the password-induced inconvenience with quasi-religious fervour; standard human beings, instead, of which few use PDAs in the first place, immediately disable the password facility. Whenever security and usability fight, usability wins.

The first problem to be addressed is therefore that of finding alternatives to passwords that may offer greater usability for a comparable level of security. This issue will be examined in section 2.

Even if we devised a perfect authentication mechanism, though, we would still face another security problem. Consider the usage pattern of granting someone else temporary access to the non-private features of our machine, for example to allow them to play Tetris. Normally we have no way of doing so without also giving up control of the whole PDA. This scenario will be examined in section 3. Solving this problem will also, as a side effect, improve usability for the case in which the authentication method is still a cause of user frustration.

Finally, on the significance of the PDA platform: notwithstanding our own personal enthusiasm for it, we must reluctantly concede that the PDA is still a niche gadget of little commercial significance. Despite that, the quest for a secure and usable PDA is still a commercially relevant research topic in so far as most of the functionality of the PDA is now resurfacing in the computing product that, more than any other, can truly be described as ubiquitous—the mobile phone. Besides, much of what we shall say in section 3 also applies to laptops and even desktop computers.

2 Authentication

Imagine that, using telepathy, we had perfect authentication: imagine that the PDA could always tell, with absolute reliability and without the user having to do anything, who is currently holding it[2]. Then the usable and secure PDA could be implemented as follows: while my PDA is held by me, unlock it. As soon as it isn't, lock it. As you would expect, the correct way to establish this association in the first place is through the Resurrecting Duckling security policy model [2]: when I buy the PDA, I imprint it to me. Then nobody other than me could use my PDA (making it secure), whereas the PDA would always automatically unlock itself for me when in my hands (making it usable). In theory we would still have a problem with coercion[3] but the telepathic method may be assumed

[2] To be more precise, this hypothetical "perfect authentication mechanism" only needs to be able to tell whether the PDA is being held by its designated owner or not. It does not have to be able to recognize the identity of any *other* potential users, so long as it can reliably distinguish them from the owner.

[3] Cartoon image: the beaten-up banker, tied to a chair by the mafia thugs, has his PDA placed in his hand so that it unlocks and they can read his secret note about the combination of the safe.

to detect the duress condition and refuse access. The only problem is with implementation, because we don't know how to authenticate the owner to the PDA using telepathy.

Having drawn this asymptote of ideal behaviour, let us state our protection goals more explicitly. We want the PDA to be secure and usable for its owner. **Secure** here means that, if the device is lost or stolen, the effect is the same as if the device disintegrates: nobody can use the hardware any more, the owner loses any data not yet backed up, and nobody can retrieve any of the data that was on the machine, even if they have physical control of it (or of its remains). **Usable**, conversely, means that the owner finds the device unlocked every time she attempts to use it. She doesn't have to type a password into the device at every access. Ideally, just as for a paper diary, she doesn't have to type a password *ever*.

It is of course easy to provide each of these two properties on its own if the other can be ignored.

According to a traditional taxonomy of authentication methods, alternatives to passwords (*something you know*) include tokens (*something you have*) and biometrics (*something you are*). Either of these would be a step forward in the usability direction. They will be examined in sections 2.1 and 2.2 respectively.

We note from the requirements discussed above that the PDA is a machine with one very privileged user. In fact, even more than the so-called "personal" computer, the PDA is the archetypal *single-user* machine—so much so that the traditional login sequence for a PDA only asks for a password, without even *mentioning* a userid.

We also note that, to comply with the above security requirements, access control at the operating system level is by itself insufficient, since the attacker might open up the machine and access the secondary storage (flash memory or miniature hard disk) directly [3]. A secure PDA will have to encrypt its file system, although no current model does so in its manufacturer's configuration.

When should the authentication check be performed? In the ideal "perfect authentication" case it is a continuous ongoing process: as soon as the PDA is no longer held by the user, it notices and locks itself. In most practical implementations, instead, the check is only performed when certain (relatively rare) events occur: usually when the machine is initially turned on and when it wakes up from its sleep state. There is therefore a window of vulnerability between the moment in which the owner stops using the device and the moment in which the device goes to sleep. To reduce the size of this window one may force sleep soon after detecting inactivity; but, since the machine can't distinguish between thinking time and the end of the current burst of activity, setting an excessively short timeout will greatly inconvenience the user, as the machine will switch off in mid-operation. This is yet another trade-off between security and usability.

Assuming an authentication mechanism that allows the verification to be performed without disturbing the user, the dependency between sleep and authentication should be reversed: instead of triggering authentication when the device comes out of sleep, the device should continuously verify the presence of the user[4] and lock itself up, as well as going to sleep[5], when the user is not present.

2.1 Token

The PDA could verify that it is being held by its owner by checking for the proximity of a token held by the owner, for example a special microchip embedded in a ring or in the strap of a wristwatch. The quality of this solution can be measured along several dimensions, including at least the following.

Communication Technology. The communication between the token and the PDA may require precise contact between two physical devices (electrical connection, e.g. iButton; mechanical coupling, as with lock and key); loose contact between two designated areas (inductive coupling); it may be contactless but still require physical alignment (infrared pulses, barcode and other optical methods); and it may be completely unconstrained within a certain range (radio, e.g. Bluetooth or RFID). The latter solutions in this spectrum are almost as convenient as telepathic interaction but may be subject to eavesdropping.

Protocol. The most trivial authentication procedure is for the token to present a previously agreed secret to the verifying PDA. This solution (RFID, barcode) is cheap because it only requires a static, read-only token and unidirectional communication from token to PDA. If the communication channel is subject to eavesdropping, though, or if the attacker can once obtain access to the token, then replay attacks are possible. To prevent these, the responses from the token must not be predictable. Common solutions include one-time passwords, subject to synchronization problems, and challenge-response, requiring the additional device costs of bidirectional communication between token and PDA and active cryptographic capabilities in the token (iButton, wireless dongle). Furthermore, if an active adversary can insert itself in the channel between token and PDA and masquerade as one to the other and vice versa, then a man-in-the-middle attack may be possible. If all the man-in-the-middle does is tunnel data (to pretend that the token is near the PDA even while it isn't, for example to unlock a stolen PDA) then the only countermeasure may be a distance-bounding protocol [4].

Binding. All the authentication procedure can do is give the verifier some confidence that the *token* is nearby. To infer from this that the *owner* of the token is nearby requires a reason to believe that the token is still physically with the owner. If the token itself may be lost or stolen, the problem

[4] For example by polling several times per minute—although the ideal would be to have an "interrupt" generated (but by what?) as soon as the owner is no longer there.

[5] Sending the machine to sleep, as well as locking it, has however the disadvantage that it prevents the user from running unattended background computations. For some applications, e.g. backup, it may be more convenient to lock user I/O but only force sleep after the currently running applications have completed their task.

only goes up one level. While it is true that the user is less likely to lose a wristwatch-bound token than a pocket-heavy PDA, there are scenarios (e.g. sports practice) in which the user will have to take off both the watch and the PDA, and even have to store them together in a clothes bag or in a flimsy locker.

More complicated countermeasures could be envisaged to cover this threat, such as a token that deactivates itself when the watch strap is opened and which requires a PIN for reactivation; but the complication makes the implementation more expensive and goes against the original goal of usability. In particular: the armoured chip must sense the open-closed status of the strap; there must be a way to enter a PIN in the chip—and this data entry peripheral, being outside the tamper-proof enclosure, could be abused to brute-force the PIN; finally, the user has to re-enter the PIN whenever the watch strap is undone for any reason—this, although probably less frequent, is not an enormous improvement over having to type the password in the PDA in the first place.

An original idea on the theme of binding, although definitely not the solution we would hope to see deployed, is suggested in an excellent and thought-provoking "ubicomp envisionment" video co-produced by Keio University and Tokyo University[6], which features a teenager from the near future whose wireless e-wallet is embedded in a hard-to-misplace tongue piercing.

It is interesting to observe how, in a spectrum of "tightness of binding of token to owner" that might span such sample points as key fob, watch strap, ring, tongue piercing and subcutaneous RFID implant [5], the "something you have" token blurs into a "something you are" quasi-biometric, with the corresponding consequences on availability (the more tightly bound tokens are harder to forget at home or lose), transferability (the ability to sign a document in one's name, for example, is easier to delegate if the signature is affixed with a stamp, as commonly done in Japan, rather than as a handwritten scribble), deniability (the tighter the binding, the harder to pretend that the owner was not with the token) and privacy (particularly when recognition happens without the consent or knowledge of the owner).

The nicest existing implementation of token-based locking of a personal device that we know of is that of Corner and Noble [6]. The device being locked is a laptop, while the prototype token is a PDA, which the authors consider representative of what might be embedded in a watch in the near future. The two devices communicate over an 802.11 radio link in ad-hoc networking mode. The laptop, whose file system is encrypted, constantly polls for the presence of the token. When the token goes out of radio range, all the data in the file cache is re-encrypted within five seconds. When the token returns, decryption back to the original state occurs in just over six seconds.

[6] The short video, *A Love Triangle*, is part of a set of three "Small stories in 2008". At the time of writing, a streaming feed for the videos was available at `http://stoneroom.mlab.t.u-tokyo.ac.jp/vrep/archives/2003_12_08.html`.

We note in passing that, while in principle losing the token leads to inaccessibility of the PDA, if the binding between the two items is Duckling-compliant, then the PDA will remain accessible, because the second of the four principles of the Duckling policy [7, section 4.2] prescribes that the imprinting key be backed up. Alternative tokenless solutions might also be adopted, such as unlocking the PDA by typing a password into it. It would however be prudent to verify with care whether the chosen alternative is completely equivalent to the solution described by the Duckling policy and to look in detail at any differences.

2.2 Biometric

The biometric authenticator could be a fingerprint. There is indeed a model of cellular phone, first offered for sale in Japan in 2003, that incorporates a fingerprint reader and uses it to lock the address book and call history.

It is of course impossible to forget one's fingerprint at home, which is an advantage compared to the token, but it is not entirely impossible to have it stolen. Apart from the gory but thankfully infrequent scenario in which the actual finger is chopped off, it has been shown that most fingerprint readers can be fooled by a rubber-like mould [8]; all that remains is for the adversary who found or stole the PDA to get hold of a fingerprint from which to fashion the rubber finger. The James Bond approach might be to offer the victim a Martini and keep the glass; a more practical solution, which does not require interaction with the former owner, is to look for fingerprints on the PDA itself—it is, after all, a "handheld" device.

The biometric could also be a voice print. From the usability viewpoint this may be less convenient than silent forms of authentication: unlocking the PDA would embarrassingly attract the attention of anyone nearby. Compared to many other biometrics, though, it would offer the advantage of being better suited to a challenge-response protocol ("please read out this randomly-generated sentence") that might thwart replay attacks. Handwriting recognition, though probably still immature for authentication, is another biometric method that would share this property.

Another well known class of problems opened up by biometrics as opposed to other forms of authentication is that of the inevitability of false positives (accepting an impostor) and false negatives (wrongly rejecting the owner). The biometric scheme with the best performance along this dimension is currently iris recognition [9]. This scheme could be readily adopted for our application, particularly with the current trend towards camera-equipped phones. But here too, especially for a known victim, it wouldn't be impossible for the illegitimate holder of a lost PDA to find a photo of good enough quality for a replay attack, as suggested by the story of the Afghan woman featured on the cover of a 1984 National Geographic, identified 18 years later by comparing her iris codes to those of the photo [10]. To counter this threat one should resort to a verification procedure that ascertains the presence of a live iris, for example by measuring the pupil's dilation in response to varying illumination. The general consensus is that it is hard to eliminate this kind of false positives if the attacker can perform the authentication without supervision.

Yet another problem of using a biometric measurement instead of a password or token is the impossibility to revoke it. You can't get new irises, except by following the grisly procedure demonstrated in Spielberg's *Minority Report.*

Perhaps the strongest threat for biometrics, therefore, is the current worldwide trend, under strong pressure from the US government, towards embedding biometric information such as fingerprints or iris codes in machine-readable state-issued identity documents. Ignoring a number of implementation details, this is more or less equivalent to having your non-changeable password printed in cleartext in your passport.

Much of the justified stigma associated with biometrics, though, comes from the Orwellian overtones of a Big Brother entity menacingly collecting privacy-threatening information about our activities and whereabouts. The case of PDA authentication, however, is different: here the verifier is under our control, works for our benefit and won't be reporting us to a global aggregating observer.

After pointing out the problems of tokens and biometrics it is only fair to emphasize that passwords, too, are far from perfect. Apart from the fundamental usability problem of having to enter the password at every interaction, which motivated the search for alternatives in the first place, passwords are often poorly chosen—easy to guess and, worse, recycled across systems, so that the discovery of one (which can be trivial for the provider of a password-protected web service) means the compromise of several accounts. Except for long and entirely random ones, passwords are subject to brute force and smart dictionary attacks (including "variations on a theme")[7]. When they are hard to guess they are also hard to type and remember, sometimes denying access to the device until the user can get hold of a backup (if one exists). The backup itself, when it exists, is usually better protected from the viewpoint of availability than from that of confidentiality and is therefore a vulnerability of its own. Finally, if fingers can be chopped off, so passwords can be tortured out of victims.

Just as most of us still confidently rely on passwords on a daily basis despite all of the above, the problems previously pointed out for tokens and biometrics should not be taken as implying that those solutions are entirely hopeless. Let us therefore suspend disbelief on the assumption that a non-password method can be sufficiently secure for protecting the confidentiality and integrity of the data in the PDA against the envisaged threats, and that the additional hardware cost is justified by the improvement in usability.

2.3 But Do We Need Authentication?

The main thesis so far is that passwords are a nuisance from the usability viewpoint and that a user-friendly PDA should adopt some different mechanism for

[7] On a PDA, though, where the attacker cannot crack a password file offline, brute-force attacks are hard to mount unless one interfaces an automaton to the keyboard, either mechanically or electrically. Even then, a throttling mechanism can easily reduce the frequency of attempts after only a small number of consecutive failures, making brute force impossible.

user authentication. This thesis is easy to defend but not particularly novel intellectually.

Another take on this problem is to step back and ask whether authentication is really needed. Some of the PDA functions we use most frequently, such as the multilingual dictionary, gain no benefit whatsoever from password protection. Yet we are forced to keep the password feature active in order to protect our diary, our personal notes and our contact list.

In previous work on authentication and authorization for ubiquitous computing we introduced the Big Stick principle [7, section 4.2.8], one of the most robust security policy models you can adopt: "whoever currently has physical control of the device is allowed to take it over". This is the security policy governing access to your lawnmower, your fountain pen and your English dictionary. In the world of electronic devices, it's the policy of your watch and pocket calculator, for which it is highly appropriate. It is also the policy of your digital camera, for which it is perhaps less so. The strength of this policy is that it is a very close model of what usually happens anyway; it is therefore less likely to be violated than others that attempt to impose unnatural behaviours on the real world: when the real and the ideal world are at odds, the real world usually wins.

The significance of Big Stick in the current context is that there are parts of a PDA that operate as devices for which that policy model is appropriate: the calculator application, the Japanese-English dictionary application, the multimedia encyclopedia, the World Time application, as well as most games and puzzles, carry no state and have no confidentiality concerns. If the PDA only offered those, Big Stick would be a perfectly suitable way to manage it and there would be no need to look for alternatives to passwords because one could dispense with authentication altogether[8]. The PDA would be ready for use all the time by the bearer with no access control restrictions. This is indeed what happens for separate appliances offering these functions—the stand-alone pocket calculators, electronic dictionaries and world clocks never ask you to log in, and are therefore much more convenient to use than the corresponding function of your PDA.

Especially with passwords, that emphasize the cost of every authentication operation, it is quite frustrating to have to pay the penalty of unlocking the PDA just in order to access a function, such as looking up a word in the dictionary, for which the access control restriction is meaningless in the first place.

2.4 Lending Your Machine—or part of it

There is more: if, seeing that you have just used an interesting-looking dictionary on your PDA, your dinner party companions ask you whether they can borrow the device and look up some words of their own, under the current state of

[8] The only viewpoint from which Big Stick might be inappropriate is if one wanted to make the device inoperable for a thief—in other words, if one wanted to protect the hardware. In this paper we assume that the hardware will continue to drop in cost and that its value is insignificant compared to that of the data. If you lose your pocket calculator, even a scientific pocket calculator, you just buy a new one without great regrets.

affairs you can only hand them a usable device if you give it to them unlocked, i.e. in the state in which you have already logged in. At this point you must blindly trust your friends to access only the dictionary and not, while the device is in their hands and too far for you to actually see[9], the content of your diary, personal finance spreadsheet or secret business plan.

Things would be much nicer if we could draw a kind of security perimeter ("private area") around the data and applications of which we wanted to protect confidentiality and integrity, while leaving the rest of the functionality of the PDA in the "public area" (or perhaps "DMZ" in firewall terms) outside the security perimeter. Then anything in the public area could be accessed and run without authentication. Authentication would only be required for entering the private area. With this arrangement we could hand over the PDA without having logged in, thereby giving access to all applications in the public area but preventing access to anything in the private area.

Assigning applications to the private or the public area involves some subtlety. While the calculator can be made public with no second thoughts, the web browser requires some attention to detail: despite being essentially stateless, and therefore a good candidate for the public area, it might still contain sensitive information in the cache, history, bookmarks, cookies, "remembered fields" and so on. This kind of application should therefore be allowed to run in both areas, but it should access a different set of preferences in each. There should also be a way to erase in a global way any preferences left by any applications in the public area. Finally a third type of applications, such as backup and restore, should only be allowed to run in the private area.

The most obvious way to implement this split using standard operating system facilities is by making use of two user accounts, one for the private area and one for the public area, the latter equipped with an empty password so as to require no credentials for authentication.

As we noted earlier, the PDA is the archetypal *single-user* machine. Many PDAs don't even have a concept of user accounts in their operating system—my first one, an HP, just ran DOS. There are, however, PDAs based on a multi-user OS (my current one, a Sharp Zaurus, runs Linux), even though the supplied system software pretends and assumes that there is only ever going to be one user.

Is it then sufficient to pick a PDA with a multi-user OS, create a user called "public" (with empty password) and one called "private", and assign the right permissions to the relevant directories and applications?

Not quite. Because, if "private" is logged in and the PDA goes to sleep, the machine will be unusable until "private" types in his password to wake it up, so you can't suddenly open the PDA as "public" if you just want to use the calculator. Besides, if you are working as "private" with half a dozen applications open in useful places, you don't want to have to log out in order to be able to lend the machine in "public" mode to a friend who wishes to check the web briefly.

Would this be something solved by the unix "su" facility, or its Win32 equivalent "run as"? Not quite, because both of these are nested invocations: the

[9] And you are too polite to watch over their shoulder like a prison guard anyway.

person running the application under the new userid can always close the application and therefore switch back to the previous userid, the one that invoked the "su" or "run as" command. So this would perhaps be suitable for going from "public" to "private", but never vice versa.

Interestingly, any good solution to the sharing problem highlighted here would also fix the problem of the previous section (2.3): if you wanted to access the dictionary or any other application in the public area, you now would be able to do so without password.

3 One User, Many Hats

To solve the problem just presented in section 2.4, a different facility is needed: a multi-user OS that can keep several sessions (several "desktops") open at the same time and allow random-access, non-nested switching between them. This is similar to the CTRL-ALT-F1 console switching facility of Linux, with the two important differences that it should work for graphical sessions and that switching to a different user should require that user's credentials again. The closest approximation to this facility in a current desktop OS is Windows XP's Fast User Switching.

In order not to invalidate too many of the assumptions on which existing applications may be based we accept the restriction that, at any given time, each userid can have at most one active desktop. But several desktops may simultaneously be active, so long as they belong to different userids. One of these will be the one and only "public" userid—the one requiring no credentials. There may be several other accounts, all private in the sense of requiring credentials, each with a different set of privileges: this one can access the whole home directory but cannot access the network, this one can access the network but can only access a chrooted jail of the file system, this one can view Word files but can't do anything else, and so on. The underlying idea is sandboxing. This is clearly an arrangement that could be useful also outside the realm of PDAs. In fact, most laptops are just as single-user as PDAs.

The main point to note is that all of the "private" accounts actually belong to the same real-world user—the owner of the machine. These accounts form a group of different userids (we could say "personalities") for the same user. This is still a machine with a single real-world user, and yet there are several userids that are logged in simultaneously. One user, many hats. The hat can be viewed as a personality and also as a credential: the machine sees you wear the fireman's hat and recognizes you as a fireman; it sees you wear the Robin Hood hat and recognizes you as Robin Hood, giving you access to Robin Hood's files and privileges. And sometimes it sees you come with no hat, as an anonymous user, and grants you the (non-null) privileges of an anonymous guest, which include looking up words in the dictionary, playing stateless videogames and surfing the web. Hence the title of this paper: *One user, many hats; but, sometimes, no hat.*

When the machine comes out of sleep, all active "private" sessions are locked—meaning that you need to exhibit the corresponding credentials to access them. The "public" session requires no credentials and therefore is never locked. When the machine comes out of sleep, you don't have to log back into the same session that was active when the machine went to sleep; you can instead switch to another session and log into that, leaving the original one locked. You can also start an entirely new session under an unused userid, assuming there are sufficient resources left to allow that. You can also atomically close all the sessions and reboot the machine without having to show any credentials. This is in accordance with the Big Stick principle, since you could always do that by removing the battery anyway.

This "switcher" accessory of the operating system can be operated without any credentials and always allows you to get back to the "public" session. It is invoked automatically when the machine is turned on or woken up, but it can also be recalled explicitly at any other time. In a well-designed PDA, the facility to switch to the "public" session would even be given its own physical button, which for the user would have the semantics of "turn on the machine *right now*[10], without any of that password hassle".

The "public" session, usable by anyone to whom the owner may lend the machine, does not retain any permanent state (browser history etc). When closed, it forgets everything. It is also always active: whenever closed, it reopens itself automatically, so that there is always an open "public" session to switch to.

With fingerprints as authenticators, one could even have a different userid for every fingerprint, with appropriate mnemonics: index to search Google, middle finger to read the dreaded Word attachments and so forth.

The baseline requirement, however, and probably the most useful configuration, is simply one private area and the common public area.

To get maximum protection from the sandboxing idea one might have to combine the OS facility described above with a carefully thought out mandatory access control policy. The goal might be to ensure that, for example, a virus coming through the mail program can never read the appointments in your diary. This, incidentally, will result in restrictions that prevent you from cutting and pasting data between the diary and the mail application. The need for smooth inter-application communication, not just through the file system but especially via higher level mechanisms such as the clipboard, is probably going to be the main constraint limiting excessive subdivision of the private area into a fine-grained multitude of separate userids.

We fear however that, while the sandboxing idea will stop most of the malware attacks targeted at the pre-hats versions of the underlying OS, a determined attacker working specifically against the multi-hat configuration will in most cases be able to bypass the protection, possibly using techniques inspired by the API attacks community [11]. There are just too many possible interactions if the machine must still be usable and user-friendly.

We envisage a few ways of building a prototype of this functionality into a Linux-based PDA. The first is based on the X Window system. X already has the facility to host several independent graphical sessions on the same machine: some work is required to provide a suitable switcher applet, but thereafter each

[10] The "right now" may require some RAM preallocation, otherwise swapping may induce a substantial delay.

session will run on its own X display. A rather different approach would instead use the Xen hypervisor [12], which allows a machine to be split into several virtual machines each of which runs its own operating system. This would provide greater separation between the hats, allowing for totally separate file systems and peripherals (e.g. network). It would also require a port of Xen to the PDA's architecture and a port of the PDA's version of Linux to Xen.

4 Research Questions

Once such a prototype is built, there will be further research questions to explore. It may already be possible to investigate some of them on a laptop by implementing the multi-hat strategy on top of Windows XP's Fast User Switching; however, with a closed proprietary OS, one has no freedom to adapt and possibly optimize the underlying mechanism towards the intended usage pattern and one cannot subsequently port the system from a laptop to a PDA. A solution based on free software would clearly be preferable.

Perhaps one of the most interesting research questions is how to reconcile asynchronous notifications with the security requirement of keeping everything encrypted. Imagine a PDA in which the whole file system is encrypted and data is only decrypted on the fly with a key that is forgotten as soon as the owner is no longer around, as in the cited work by Corner and Noble [6]. How will the PDA be able to wake up and beep at the next scheduled alarm event from the diary, if the diary file itself is encrypted and inaccessible while the machine is asleep? Should there be a special queue of wake-up events, kept around in unencrypted format, to which the diary application writes before going to sleep? This seems ad-hoc and inelegant.

If one could invoke the silver bullet of tamper resistance, and rely on a safeguard capable of erasing the RAM at the first attempt of tampering, the solution would be easy: the file system on secondary storage would always stay encrypted but the RAM and file cache would be unencrypted and would allow programs to run in the background (assuming they had preloaded all the disk pages they needed) even with the machine (meaning actually the user interface) locked. This may be the best compromise yet, but we have learnt to look at tamper resistance claims with some suspicion [13].

Furthermore, how would an encrypted file system interact with the multiple hats? Clearly, anything in the "public" area, including OS and applications, would have to be unencrypted. Would this then threaten the integrity of the "private" area? Should there be several complete replicas of the file system (an approach that the Xen-based implementation would support better than any other)? Is there any scope for a less drastic `chroot`-style sandboxing?

We may also have to reconcile the need for secure isolation between the hats with the potential usability requirement of transferring information between them in some controlled but convenient way. If, for example, the web browser runs only in the public area, it may be desirable to transfer a file downloaded with it into one of the "private" areas. Clearly this raises a number of issues from the field of multilevel secure systems—not trivial to begin with, but made much harder by the additional requirement of usability. Once in that realm, it will then be interesting to discover and counter the creative ways in which a no-hat user might escalate privileges.

5 Conclusions

Today, security and usability still tend to be antithetic. For a PDA, they roughly translate to "password" and "no password" respectively. Neither property is satisfactory without the other.

Performing the authentication with means other than passwords is one way to increase usability. Tokens and biometrics both have some problems, but ought to be taken more seriously. In this paper we examined a number of alternatives in some detail, exposing their security vs. usability trade-offs.

Whatever the authentication method, though, it would be nice if the machine were smarter and less pedantic. Why ask for authentication when accessing a facility that does not require protection? Our "hats" approach addresses this problem and, as a side benefit, also supports the otherwise dangerous usage pattern of temporarily lending the machine to a friend.

Even a strongly single-user machine such as the PDA will benefit from our new arrangement in which the single real-world user can have one hat, or several, for accessing protected material, and can switch hats—or take off the hat altogether—without logging out, and without having to provide any credentials when switching to "no hat".

When one day our machines feature a correctly implemented one-touch "no hat" button, we will have won a significant usability battle without having compromised security.

Acknowledgements

As usual, the original draft was improved thanks to the many comments received at the workshop. I am grateful to, among others, Matt Blaze, Tuomas Aura and particularly James Malcolm. After the workshop I also benefited from discussions with Matt Jackson, who has since become my PhD student.

References

1. Frank Stajano. "Will Your Digital Butlers Betray You?" In Paul Syverson and Sabrina De Capitani di Vimercati (eds.), "Proceedings of the 2004 Workshop on Privacy in the Electronic Society", pp. 37–38. ACM, Washington, DC, USA, 28 Oct 2004. ISBN 1-58113-968-3.
2. Frank Stajano and Ross Anderson. "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks". In Bruce Christianson, Bruno Crispo, James A. Malcolm and Michael Roe (eds.), "Security Protocols, 7th International Workshop, Proceedings", vol. 1796 of *Lecture Notes in Computer Science*, pp. 172–182. Springer, 2000. ISBN 3-540-67381-4. ISSN 0302-9743. `http://www.cl.cam.ac.uk/~fms27/duckling/`.

3. Tony Sammes and Brian Jenkinson. *Forensic Computing: A Practitioner's Guide.* Springer, 2000. ISBN 1-85233-299-9.
4. Stefan Brands and David Chaum. "Distance-Bounding Protocols (Extended Abstract)". In Tor Helleseth (ed.), "Advances in Cryptology—EUROCRYPT 93", vol. 765 of *Lecture Notes in Computer Science*, pp. 344–359. Springer-Verlag, 1994, 23–27 May 1993.
5. Julia Scheeres. "Implantable Chip, On Sale Now", 25 Oct 2002. `http://www.wired.com/news/privacy/0,1848,55999,00.html`.
6. Mark D. Corner and Brian D. Noble. "Zero-interaction authentication". In "Proceedings of the eighth Annual International Conference on Mobile Computing and Networking (MOBICOM-02)", pp. 1–11. ACM Press, New York, Sep 23–28 2002.
7. Frank Stajano. *Security for Ubiquitous Computing.* John Wiley and Sons, Feb 2002. ISBN 0-470-84493-0. `http://www.cl.cam.ac.uk/~fms27/secubicomp/`.
8. Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada and Satoshi Hoshino. "Impact of Artificial Gummy Fingers on Fingerprint Systems". In "Proceedings of SPIE", vol. 4677, Optical Security and Counterfeit Deterrence Techniques IV. 2002. `http://cryptome.org/gummy.htm`.
9. John Daugman. "How Iris Recognition Works". *IEEE Transactions on Circuits and Systems for Video Technology*, **14**(1), Jan 2004. `http://www.cl.cam.ac.uk/users/jgd1000/csvt.pdf`.
10. John Daugman. "How the Afghan Girl was Identified by Her Iris Patterns", 2002. `http://www.cl.cam.ac.uk/users/jgd1000/afghan.html`.
11. Mike Bond and Ross J. Anderson. "API-Level Attacks on Embedded Systems". *IEEE Computer*, **34**(10):67–75, 2001. `http://www.cl.cam.ac.uk/users/mkb23/research/API-Attacks.pdf`.
12. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho and Rolf Neugebauer. "Xen and the art of virtualization". In "Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)", pp. 164–177. ACM, Bolton Landing, NY, USA, Oct 2003.
13. Ross Anderson and Markus Kuhn. "Tamper Resistance—A Cautionary Note". In "Proc. 2^{nd} USENIX Workshop on Electronic Commerce", 1996. ISBN 1-880446-83-9. `http://www.cl.cam.ac.uk/~mgk25/tamper.pdf`.

One User, Many Hats; and Sometimes, No Hat (Transcript of Discussion)

Frank Stajano

University of Cambridge

Matt Blaze: One of the problems with biometrics is that the incentives are all wrong. In the case of most biometric identity systems the benefit is for the operator of the system, who usually wants to prevent transfer of usability of credentials, not for the user of the system herself. When I give my biometric identity to the passport agency this is for the benefit of the passport system, not for my system. In fact it degrades my privacy, because now the passport authorities have enough information to fake my biometrics in other applications. This might be an instance in which the semantics of the trust properties of the application are the problem.

Reply: Yes, I agree entirely with what you say and I can draw a parallel between this and the DRM situation, where most of the DRM is put in place not for the owner of the machine, but for the benefit of someone else.

Matt Blaze: But you shifted gears, because when you were talking about the PDAs that was for the benefit of the user.

Reply: Yes, this is one of the rare cases where the owner of the machine is the one who gains the benefit from doing the biometric check. In cases of DRM, in contrast, the owner of the machine has no interest in doing the checking for someone else who's not there, so that interacts badly with the rest of the infrastructure.

But after having rubbished tokens and biometrics perhaps I should go back and for fairness rubbish passwords as well.

Tuomas Aura: There are some advantages of using passwords in this particular application. It's atypical, but some of the attacks that usually exist on a password system don't exist here. For example, you don't have off-line brute force attacks where someone observes the protocol or you have a false verifier. Another thing is that if you enter the wrong password you *can* actually lock the device for a while before you are allowed another try, because in this case that doesn't lead to denial of service attacks whereas it usually does.

Reply: Yes, that's a good point. To some extent it depends on how tamper-resistant the actual construction of the PDA is.

Bogdan Popescu: I think the biggest problem with biometrics is that they are good as long as there's an officer that supervises the verification process. But if there isn't then you can mount any sort of attack in private and get a surprise.

Reply: When biometric verification is completely unattended of course you can cheat.

Bogdan Popescu: But I don't think this is the same for passwords, because passwords are not spoofable at all. After 300 unsuccessful tries, we still don't know the password.

Reply: Yes, I agree with you. But, perhaps it's time to say it, secure passwords are problematic because they inconvenience the user.

In reality, there are parts of my PDA for which the big stick policy would be more appropriate. For example, my PDA contains a calculator application, my PDA also contains a world time application, and these things are things that I would like to be able to access instantly. There is no security advantage to me in having them access protected.

Matt Blaze: I disagree that there's no security advantage to having those things secured. One advantage is denial of benefit to someone who steals it from you. One reason mobile phones don't get stolen more often than they do is the fact that once it does get stolen you can have it turned off. So to the extent that this is an expensive device, being able to render it useless if it's stolen has some value.

Reply: Yes, that's a very good point, I hadn't thought of that, thank you.

Turning next to consider implementation of the multi-hat PDA, I envisage really an open-source approach. But I don't mind if people take the idea further with other operating systems, whether they're open source or not, I will just use them and be happy, and this is why I am disclosing as much of this as I can right now in a public forum so it cannot be patented.

Matt Blaze: Another example of applications, the US DoD has expressed interest in smart weapons that will only work with the authorised user. There all of the requirements are exactly what you stated but much more extreme. You want to be able to very, very quickly authenticate the user, but not use one that's been taken from the user even a few moments ago. So, it might be interesting to look at some of the work that's been done in that area to see how it can be applied to PDAs.

Reply: Yes, you are pointing out something that I also thought about, how to automatically close a session. There's a sleep mechanism here which is based on a timeout. So of course there's the usual trade off, the shorter you make the timeout the less convenient it is, but also there are other things that it would be nice to experiment with such as, for example, adding extra inputs to this login system such as a motion sensor.

Fabio Massacci: What happens if the device is connected to the network? Because your model only works if you cannot remotely access the PDA, what happens if access can come through the network to the device?

Reply: Oh, I don't trust networks so I didn't have network connectivity in mind, but the theory of this vision is that networks are also a resource that you would be able to sandbox. So you would say, this user has privilege for that but this other user doesn't.

I think that use of the network would be limited to particular sandboxes. I certainly don't advocate having a system that has open servers to offer outside activity or even worse having the switcher open to network control. I certainly wouldn't touch that with a barge pole.

I do have several users local to this PDA, several computer user IDs, but they're all the same user, they're all me, and so you could call them hats if you want to. This is still definitely very much a single user machine, but I have different hats for different roles. I have another one here, this is a special hat[1], there's a feature that as soon as you put it on your IQ is reduced by 30%, if you put it on backwards like this it's reduced by 60%.

[1] Puts on baseball cap.

Authentication Components: Engineering Experiences and Guidelines

Pasi Eronen[1] and Jari Arkko[2]

[1] Nokia Research Center
pasi.eronen@nokia.com
[2] Ericsson Research NomadicLab
jari.arkko@nomadiclab.com

Abstract. Security protocols typically employ an authentication phase followed by a protected data exchange. In some cases, such TLS, these two phases are tightly integrated, while in other cases, such as EAP (Extensible Authentication Protocol) and Kerberos, they are separate and often implemented in different endpoints. However, careless application of this separation has lead to several vulnerabilities. In this paper we discuss reasons why this separation is often useful, what mistakes have been made, and what these mistakes have in common. We then describe some approaches how these problems could be avoided, especially focusing on EAP in wireless LANs. We also present some engineering observations that should be taken into account when designing reusable authentication components in the future.

1 Introduction

When designing a new system, a common engineering practice is to use existing components as building blocks. Typical examples of such components include not only things such as IP transport, TCP, XML and HTTP, but also components providing security services.

One well-known example is running HTTP over TLS. TLS uses certificates to authenticate the web server (usually client authentication is implemented by some other means), and provides encryption and integrity protection for the traffic. This case is rather simple, and its properties are easily understood even by those who are not intimately familiar with TLS. [1]

Other examples of reusable components include SASL and GSS-API [17,18]. Both provide an extensible framework for authentication mechanisms, and may optionally provide also protection (encryption, integrity protection, replay protection, etc.) for application traffic. A common feature of all these approaches is that traffic protection (when present) is handled together with authentication and key exchange.

[1] Though occasionally some developers have forgotten that it is not enough to check that the server has a valid certificate—you must also check that the DNS name in the certificate matches the URL you are trying to access!

However, Kerberos separates authentication/key exchange from traffic protection [15], as does Extensible Authentication Protocol (EAP) which is used for Wireless LAN authentication, among other applications [7].

In this paper we explore issues related to this separation. We begin by exploring in Section 2 why this separation is often useful. In Section 3, we continue by presenting some examples of systems having this separation, and in Section 4 we analyze pitfalls found in some of them. In Section 5, we present how some of the pitfalls can be solved in EAP, and finally, Section 6 contains our conclusions from this work.

2 Reasons for Separating Authentication and Traffic Protection

It seems the main reason for separating authentication and traffic protection is that they can be distributed to separate endpoints. The following sections discuss various reasons why this may be desirable.

2.1 Centralized Storage of Secrets

When authentication is based on shared secrets (or some other non-public-key based credentials), it is convenient to store the secrets in a centralized place, instead of having copies of them around in all nodes that need to do authentication.

In Kerberos, this would be called a Key Distribution Center (KDC); in EAP, it is called AAA server (authentication, authorization and accounting); in cellular networks, it is Authentication Center (AuC).

In cellular networks, a somewhat similar separation is also present on the client side: authentication is done by a smartcard (Subscriber Identity Module or SIM), which hands the traffic encryption keys over to the mobile phone. However, this situation is quite simple since a single smartcard is always inserted in a single phone.

2.2 Centralized Authorization

PKI could be used for decentralized authentication, without requiring an online authentication server. However, current PKIs are not that well suited to authorization; either each service has to handle its own authorization information, or a centralized server for authorization is needed. Often authentication and authorization information are combined in a single AAA server.

2.3 Centralized Audit Trail

When authentication uses a centralized KDC, this leaves an audit trail at the KDC. This makes e.g. detection of compromised credentials or other “fraud detection” easier.

This evidence of authentication is also useful for charging-related purposes. In typical roaming environments, the visited operator reports how much services the user has used, and is paid according to pre-agreed rates. Having evidence of authentication in the home operator AAA server prevents a visited operator from fabricating charges for users who have never been on the network (of course, the visited operator may still report inflated usage figures).

2.4 Extensibility of Authentication

Experience has shown that there is no single authentication method suitable for all environments. Centralized authentication server simplifies implementations since the nodes providing the various services do not need to implement all possible authentication methods.

For instance, IEEE 802.11i wireless LAN [11] access points typically implement only simple shared-secret authentication themselves, and everything else (such as passwords, token cards, certificates, or Kerberos) is delegated to an AAA server that actually implements the authentication protocol.

Another aspect of extensibility is the service-to-KDC authentication. While e.g. Kerberos supports public-key based authentication of users, services and KDC must still share a symmetric key. EAP, on the other hand, allows this authentication to be chosen independently of user authentication method.

For instance, in many environments a “transitive” authentication and authorization via AAA proxies is enough (and provides scalability when e.g. the number of WLAN access points around the world can be huge). This works especially in environments where e.g. a home (KDC) operator just needs to know that that the request came from network X, but not which from WLAN access point in X.

2.5 Different Protocol Layers

Performance issues can sometimes dictate that the endpoints for authentication and data protection are different, or at least that they are implemented on a different layer. For instance, in NFSv4, it would be desirable to delegate the traffic protection to the IPsec layer with widely available hardware acceleration, even if for flexibility and access control reasons the authentication has to reside at the application.

3 Case Example

Figure 1 shows how EAP is used in typical wireless LAN environment. The EAP conversation itself (Step 1) is between the client and the AAA server. EAP payloads are transported using EAP-over-LAN from the laptop to the access point (AP), and over EAP-over-RADIUS or Diameter from the AP to the AAA server [2,10]. The client and AAA server authenticate each other using some EAP method; for instance, EAP-TLS uses public-key certificates.

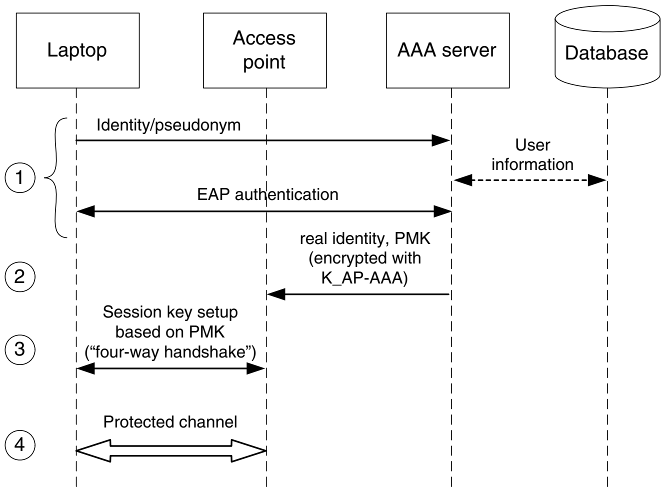

Fig. 1. EAP authentication in typical wireless LAN situation. AAA server and database are often combined in a single element.

The AAA server then determines whether the user should be granted access or not, and if access is granted, sends a Diameter-EAP-Answer message (Step 2) to the AP, containing a successful result code, the client's identity, and the Pairwise Master Key (a session key that was established during the EAP conversation). There is also a second authentication taking place: the Diameter conversation between the AP and the AAA server is protected using, e.g., TLS. Thus, the PMK is encrypted using a key shared between the AP and AAA server.

After the AP has received the key and permission to grant access, it initiates the 802.11i "four-way handshake" (Step 3); basically a nonce exchange that also protects the ciphersuite negotiation that took place earlier. This creates fresh keys that are then used to encrypt the traffic and protect its integrity (Step 4).

Figure 2 shows similar scenario for UMTS. The protocols used are completely different (and the figure omits many details), but the same steps can be identified. Kerberos, shown in Figure 3 handles issues such as replay protection differently, but again, there are many similarities.

4 Analysis and Problems

4.1 Service Identities and Lack of Them

In Kerberos, the client requests a ticket for a particular server (identified by a principal identifier, usually comprising of service type and host name) from the KDC, and when the session key is used to protect communication, the client can be assured that it is talking to the intended server (assuming, of course, that the KDC is trustworthy and the server in question has not been compromised).

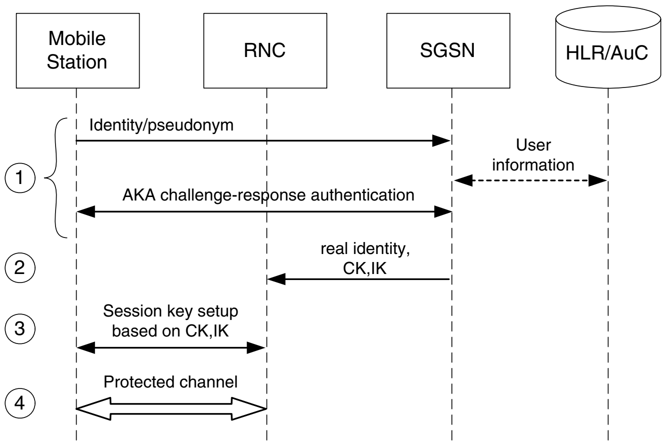

Fig. 2. UMTS authentication for packet switched access (simplified)

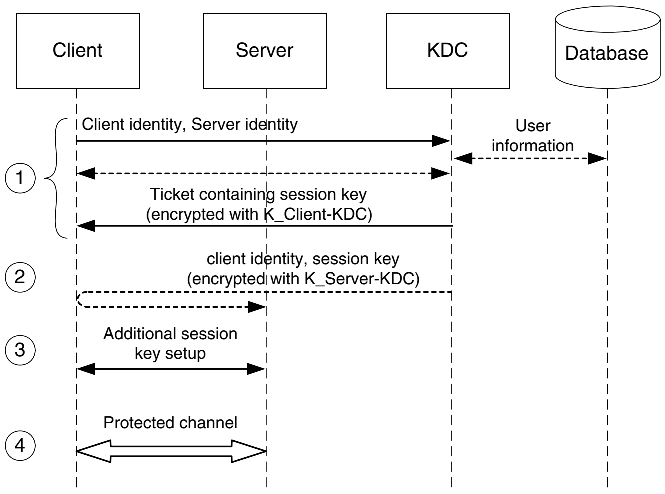
Client
Server
KDC
Database
Client identity, Server identity
User information
1
Ticket containing session key (encrypted with K_Client-KDC)
2
client identity, session key (encrypted with K_Server-KDC)
3
Additional session key setup
4
Protected channel

Fig. 3. Kerberos authentication (simplified). Note the Server identity sent by the client.

Currently EAP has no concept of "service identity", an identifier that would be authenticated to the client: the AAA server might have an identity, but not the node providing the service. That is, after a successful EAP authentication, the client knows it is talking to *some* network element trusted by the AAA server, but not which one [3]. The network element can, of course, tell the client its identity, but this information is not verified by the AAA server; or in other words, a malicious or compromised server could lie about its identity.

Cellular network authentication mechanisms such as UMTS AKA have a similar limitation. This "feature" probably has historical reasons behind it. For instance, in UMTS the cell is uniquely identified by a digit sequence called Cell Global Identification (CGI) [22]. However, since the mobile terminal does not really know which CGI it should be talking to (simplifying things a bit, it is usually the one with the best signal strength), CGI is not authenticated. (The CGI is not something that the user would enter or ever see, unlike in Kerberos where the host name is often selected by the user).

Probably it was also assumed that a compromise of a base station would be a rare event, and even if that happened, the intruder would not be interested in impersonating some other CGI towards the client (because doing so would not have any real advantage over continuing to use the real CGI value).

Somewhat similar thing can be found in EAP. In WLANs, the access points are uniquely identified by BSSID, a 48-bit random-looking string that is not meaningful to the user, since BSSID is chosen based on signal strength. Successful EAP/802.11i authentication ensures that the user is talking to some WLAN AP part of the e.g. corporate WLAN, but not which one—again, authenticating this identifier is not that important if the client cannot anyway tell the difference. Similarly, it might have been assumed that an attacker who compromises an access point would be interested in hacking into the corporate intranet rather than masquerading as some other AP towards the client.

The common feature here is that UMTS AKA and EAP consider authentication as having two different parties: the client and "the network". Internally, the network may have different nodes with different roles (e.g. UMTS authentication information is stored in AuC), but this structure is not directly visible to the client.

This situation is quite different from e.g. Kerberos, where service identities (such as host names) are usually very visible to the client, and different nodes are offering radically different types of service.

4.2 Attacks Across Protocols

While in many cases the client is not interested in the exact identifier of a node providing some particular service, it can at least know the type of service it is expecting. If this service type is not authenticated, the effects of security problems can "spread" from one service type to another. For instance, in EAP a compromised IKEv2 gateway can impersonate a WLAN access point, if same user and AAA server credentials are used for both.

A different example is the GSM A5/2 attack [6]. An attacker can use this vulnerability in GSM air interface encryption to masquerade as a WLAN client with EAP-SIM [12,9]. Similarly, if it were later found that, for instance, IKEv2 has serious weaknesses[2], this could be used to attack WLANs using the same credentials.

There have been proposals to add a "context field" to EAP methods, communicating the type of service to be provided [21,16]. Even without the concept of

[2] We do not believe this is likely, and this example is for illustrative purposes only.

service identities, this would prevent attacks from one type of service to another. For some EAP methods, it may also be feasible to include context information in a certificate presented by the AAA server [13].

The "context field" is quite similar to "Special RANDs" that have been proposed for GSM. In this case, it is not possible to add a new field, so a couple of bits from the random challenge (RAND) field are used for that [20,19].

4.3 Problems in Protocol Composition

Problems can also occur when several components providing various functions are composed together in a system. Asokan et al. [5] presented several examples where having one component responsible for server authentication and another one for client authentication could result in man-in-the-middle vulnerabilities.

5 Adding Service Identities to EAP

In addition to adding a "context field" to EAP, there have been some discussion about adding proper service identities as well.

While in some cases, authenticating the service as being part of the legitimate "network" is enough, there are cases where the client could be interested in parts of the network structure, and could, in some cases at least, tell the difference between nodes that it is "supposed to be" talking to and nodes that it is not interested in talking to. This is emphasized if some parts of the network are in fact less trustworthy than others.

For instance, in wireless LANs, the client could also know the Service Set Identifier (SSID) of the WLAN. This is a usually human-readable string identifying some set of access points, and in many cases the WLAN client software shows it to the user, or asks the user to select the right SSID from a list.

We are currently working on a mechanism to add this functionality (sometimes called "channel bindings" and "connection bindings") to existing EAP methods in an extensible way [4]. The basic idea is to embed a "context block" to an EAP messages that are protected by the method-specific integrity check. This context block would contain not only the service type (e.g. "IEEE 802.11i wireless LAN") but also service identities each party could be interested in. For wireless LANs these could be BSSID and SSID, and possibly also a human-readable network name (e.g. "Joe's Coffee Shop, Memphis, Tennessee").

Actually implementing this requires the ability to connect various types of identities. For instance, if Diameter protocol over TLS is used between the WLAN access point and AAA server, the identities that are authenticated are usually the DNS names of the parties. However, the WLAN client is unlikely to be interested in the DNS name of the WLAN access point, which could be something quite cryptic such as "ap1733.mph.tn.example.net". Therefore, the AAA server has to be able to map this securely to more meaningful identities such as "Joe's Coffee Shop, Memphis, Tennessee". This more meaningful identifier could then be sent to the client in the "context block".

This approach extends what is possible in Kerberos: there it is usually assumed that the service identifier is a host name (i.e., a DNS name).

6 Conclusions

In this paper we have analyzed the reasons why separating authentication from traffic protection is often desirable, and how this has been implemented in various systems. Many of these implementations have gotten things wrong, often due to historical reasons. However, it may not be fair to say that people have been reinventing the wheel badly—often they have tried to solve problems that better-designed systems, such as Kerberos, did not solve.

One of these real-world requirements is the choice of authentication mechanisms. While from an abstract viewpoint, PKIs might be more "secure" (whatever that means) than passwords, in reality security is mostly about other things than cryptography—and then issues such as comparing required investments with reduction in risks are appropriate criteria.

One could also say that politics has had a lot to do with many of these problems: sometimes there has been reluctance to modify protocols to better answer real-world requirements, so to avoid spoiling "clean" protocols with messy real-world details.

Consider the case of Internet Key Exchange: Original IKE did not support authentication based on token cards or one-time passwords. Since then, there has been various efforts (XAUTH, CRACK, PIC, IKEv2), but after more than five years of discussion, there is still no standardized way to e.g. authenticate users with token cards (vendors certainly do have their own proprietary solutions). All along, the argument seems to have been "all password-based authentication is insecure; IPsec is designed to be secure; therefore, you have to deploy a PKI for it".

We do not agree with this. While it does not make sense to engineer bad security solutions, it does make sense to engineer security solutions that solve real-world problems, rather than require the real-world problem to adapt to some abstract notion of "perfect security". We believe that authentication mechanisms are an example of a "tussle space" [8]; something that should not be hardcoded into architectures or protocols, but instead left open to choice.

Related to the choice of authentication methods is also the choice of identifier types. We argued earlier that e.g. in wireless LANs, the DNS name of the WLAN access point is not a meaningful identifier for the WLAN client. Since ultimately these identities should be somehow connected to the *intent* of some person, different applications should be able to use different identities, and a single service may require multiple identities of different types.

These considerations suggest that we will see more systems separating authentication and traffic protection, some of them probably repeating the same mistakes already made. The examples we have used (EAP, wireless LANs, and cellular networks) certainly did not get things right the first time. In this paper we have highlighted some similarities between these mistakes in hope that they could be avoided in the future.

Acknowledgments

We would like to thank N. Asokan and Valtteri Niemi for their valuable comments and suggestions.

References

1. Martin Abadi and Roger Needham, “Prudent Engineering Practice for Cryptographic Protocols”, *IEEE Transactions on Software Engineering* 22(1):6–15, 1996.
2. Bernard Aboba and Pat Calhoun, “RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)”, RFC 3579, 2003.
3. Bernard Aboba, Dan Simon, Jari Arkko, and Henrik Levkowetz, “EAP Key Management Framework”, work in progress (IETF Internet-Draft draft-ietf-eap-keying-01.txt), 2003.
4. Jari Arkko and Pasi Eronen, “Authenticated Service Identities for the Extensible Authentication Protocol (EAP)”, work in progress, 2004.
5. N. Asokan, Valtteri Niemi, and Kaisa Nyberg: “Man-in-the-Middle in Tunnelled Authentication Protocols”, *Proc. International Workshop on Security Protocols 2003*.
6. Elad Barkan, Eli Biham, and Nathan Keller, “Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication”, *Proc. CRYPTO 2003*.
7. Larry Blunk, John Vollbrecht, Bernard Aboba, James Carlson, and Henrik Levkowetz, “Extensible Authentication Protocol (EAP)”, work in progress (IETF Internet-Draft draft-ietf-eap-rfc2284bis-09.txt), 2004.
8. David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden, “Tussle in Cyberspace: Defining Tomorrow’s Internet”, *Proc. ACM SIGCOMM 2002*.
9. Ericsson and TeliaSonera, “Implications of the A5/2 Attack for 3GPP WLAN Access”, 3GPP TSG SA3 working document S3-030733, 2003.
10. Pasi Eronen, Tom Hiller, and Glen Zorn, “Diameter Extensible Authentication Protocol (EAP) Application”, work in progress (IETF Internet-Draft draft-ietf-aaa-eap-05.txt), 2004.
11. Institute of Electrical and Electronics Engineers, “IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements”, work in progress (IEEE Draft P802.11i/10.0), 2004.
12. Henry Haverinen and Joseph Salowey, “EAP SIM Authentication”, work in progress (IETF Internet-Draft draft-haverinen-pppext-eap-sim-13.txt), 2004.
13. Russ Housley and Tim Moore, “Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)”, RFC 3770, 2004.
14. Charlie Kaufman (ed.), “Internet Key Exchange (IKEv2) Protocol”, work in progress (IETF Internet-Draft draft-ietf-ipsec-ikev2-13.txt), 2004.
15. John Kohl and Clifford Neuman, “The Kerberos Network Authentication service (V5)”, RFC 1510, 1993.
16. Hugo Krawzcyk, “Changes to EAP methods”, message on eap@frascone.com mailing list 2003-09-03, http://mail.frascone.com/pipermail/public/eap/2003-September/001638.html.

17. John Linn, "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, 2000.
18. John Myers, "Simple Authentication and Security Layer (SASL)", RFC 2222, 1997.
19. Nokia, "Using Special RANDs to separate WLAN and GSM/GPRS", 3GPP TSG SA3 working document S3-040100, 2004.
20. Orange and Vodafone, "Introducing the special RAND mechanism", 3GPP TSG SA3 working document S3-030698, 2003.
21. Jose Puthenkulam, Victor Lortz, Ashwin Palekar, and Dan Simon, "The Compound Authentication Binding Problem", work in progress (IETF Internet-Draft draft-puthenkulam-eap-binding-04.txt), 2003.
22. 3rd Generation Partnership Project, "Technical Specification Group Core Network; Numbering, addressing and identification (Release 5)", 3GPP Technical Specification 23.003 V5.8.0, 2003.
23. 3rd Generation Partnership Project, "Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (Release 5)", 3GPP Technical Specification 33.102 V5.1.0, 2002.

Authentication Components: Engineering Experiences and Guidelines (Transcript of Discussion)

Pasi Eronen

Nokia Research Center

My talk is on authentication components. What I mean by an authentication component is basically a reusable building block. I'm talking about building blocks in a strictly engineering sense (there is very little novel cryptograph use involved), building blocks that are useful to system designers when they're designing a system and need a protocol for doing something, and they don't want to reinvent all the cryptographic stuff themselves. Usually they are not experts in that either, so it's a good thing that they don't always reinvent things from scratch.

But then again, using these components seems to have some difficulties. There are some components that are reasonably easy for system designers to understand how they work, and what properties they have, especially those that essentially have only two parties, or where the third party is not very actively involved. Many people nowadays, if they have to secure something, just run it over TLS, and these days they sometimes get it right, although there are several mistakes you can make using TLS, that's well known. It's not too complicated to understand even for people who are not security professionals, or who are professionals in designing something other than crypto. But when we get to these essentially three party protocols where three active parties are involved during the protocol run, then things start getting tricky.

There are basically three things happening here usually. One common example is that, when two principals use some sort of Key Distribution Centre, they authenticate each other, and set up a session key – the KDC somehow directly or indirectly distributes this session key to the service, and then there's some sort of communication between this client and the service. And the obvious example of this is of course Kerberos which is pretty well designed and reasonably well understood. But in many situations you can't really reuse Kerberos as such. There have been several attempts essentially to make this kind of three part protocol that have gotten several things wrong. We thought about this issue and we came to the conclusion that the reason that they got things wrong was not simply because they were not very competent in crypto, but because there were some tricky issues concerning, for instance, the identities of the principals.

There are several reasons why you might want to have a three point protocol. The obvious one is centralised storage of secrets, you need only start with $n + m$ keys instead of $n * n$, and that's the obvious reason for, let's say, Kerberos. But, if you have a public key based system where you don't necessarily[1] need this

[1] See LNCS 2133, pp 182-193.

centralised place where all the secrets are stored, there are still some pretty good reasons why you might want to involve this third party acting in the protocol later. One is centralised authorisation; normal PKI certificates give you only authentication, and that's often not very meaningful when determining whether to grant access or not, and while you can use attribute certificates, sometimes it's difficult to do that in real time, meaning that you could not revoke the certificates very quickly.

Another purpose is centralised audit trail, tracking which user accessed which service. Sometimes it's OK to have that within each service (which can of course keep logs), but sometimes it would be more useful to have that in a centralised place. Consider especially a case where the centralised place sessions will have an account for you; you're spending some money in services on this bank, or some other third party based in service providers, this bank would like to have this information in its servers to do some sort of fraud detection and see how much you're spending. The other obvious example would be a telephone operator, or a mobile operator, who wants to have this information.

Yet another quite different type of reason is extensibility of authentication. In some systems you can decide beforehand that you will use, say X.509 certificates to authenticate clients, but in some systems you really would like to have an option that you decide what kind of credentials you use (whether it's passwords, secure ID cards, PKI, or something else). You decide this only when you are deploying the system, and even different users in the same system could have different ways of authenticating.

And finally sometimes the reason for having these three party protocols is that you have end-points at different protocol layers, for performance reasons. This gets a bit tricky especially as we haven't really fully considered this last reason.

One illustrative example of this kind of three party protocol is the extensible authentication protocol (EAP) for wireless LANS. This has nothing to do with WEP by the way, this is a new 802.11i stop that doesn't have the obvious crypto flaws, there are still some problems that are known but it's not like they are using a stream cipher. So, we have a laptop on the left, and some sort of AAA server managing these accounts on the right, that authenticate each other. Communication goes through the access point so there's no direct link between laptop and AAA, but the access point doesn't look at the packets, it just forwards them. Finally this AAA server sends its identity and Pairwise Master Key to the access point. This communication has to be protected somehow, so we run a TLS between the access point and AAA (it is RADIUS used here). Finally you'll have an access point set up some sort of session key based on this master key, and then encrypt everything with AES. And this works, it is used, but there's something flawed involved.

In the paper we have two other diagrams, one showing how UMTS authenticates key exchange when we're using 3G mobile networks, and the other one showing a very simplified version of Kerberos. One crucial difference between these first two and Kerberos is that here there's no identity for the access point involved. Kerberos will send a ticket request to the KDC; give me a ticket for this service, and then you send some kind of string identifying the service. But there's nothing like that present here, and there's nothing like that present in UMTS authenticated key exchange either. And the reason that this identifier is not sent is really because it's not known, especially in UMTS, but also in many wireless LANS. There isn't any really good concept of identifier, or name, or that kind of thing, that would identify this right service that we want to talk to.

Basically this leads to the situation that if one of these services, let's say Wireless LANS, is compromised, it can still do some sort of attack against the others until this compromise is discovered. In both UMTS and Wireless LANS there are of course unique identifiers that could have been used by these access points, unique MAC address, but the reason that it was not included in the protocols is probably because it doesn't really care about the MAC address. The system will work just fine if each access point chooses a random MAC address each time it reboots, so that MAC addresses can be used for a meaningful identifier of the client. It's not authenticated, and perhaps it should not be.

I don't have a good word for it so I call this 2.5 party protocols. It's a three party protocol meaning that there are three different parties involved, but only two of those parties have any names, or identifiers, that are exchanged in the protocol. The client has a user name, the AAA server has some sort of name, but the node providing the actual service doesn't have a useful name. The result is that if one place gets compromised, or one place uses a protocol which is later found out to have a serious flaw, then this flaw can be exploited in other ways as well. This is discussed in 3G standardisation because there was a serious attack against the GSM voice encryption protocols. That's quite boring, but that attack can be used to attack other parts of the system which use SIM authentication.

So there have been some attempts to do something about this. Part of the reason why it's difficult to fix the wireless LAN system, is because it's standardised in several different places, with several standardisation organisations involved, and things like that.

But the solutions that have been proposed all seem to have some similarities. The main idea seems to be that if we don't really care about these exact MAC addresses (or some base station identifier in GSM) we still include at least some context words. So we say that EAP is now used within the ap115 wireless LAN context.

I was recently asked by a couple of people who were involved in standardisation about lessons that have been learned, so as not to make the same mistakes again. What components should they use in their systems so they get it right? But I have to admit that I don't really know who does. I think from some sort of theoretical point of view authentication is a solved problem, at least if you don't want anonymous credentials. But from a practical engineering point of view, I couldn't really tell what they should do. If they choose Kerberos they get a well designed system that doesn't match the credentials that are being used in the real world, so they have not just deploy the system, but also give everyone basically new credentials. That's one possibility.

But usually they don't like that, they want to use existing credentials. If you are a corporate IP administration and you are running a 50,000 PC network, with several thousand different services, could we really manage to have single passwords? Although I think you have to deploy some sort of PKI, and you could use that, or you could have some kind of security policy for us to do that. Or if you are a mobile operator and you have already deployed a hundred million SIM cards then you might want to use them and not deploy passwords because passwords are a pain to use on PCs, and would prove very painful on your mobile phone.

So these are the questions I would like to ask you. What should I tell these persons? Is there a need to get leverage off some kind of regular type Kerberos or are there some systems that just don't know how to use that work?

Tuomas Aura: There are two more problems. One is that there was this kind of dream in the early 90s of having one security layer in the protocol stack, like IPsec, or well, TLS because IPsec just didn't turn up. Once you have this one layer you can use it for all your security needs, and then of course you would use whatever kind of identity or credential this one layer supports. But that doesn't necessarily work because if the security layer is low in your protocol stack it will use something like the MAC address as your identifier, some kind of hardware network address; if it's for LAN communication at the network layer then it might be an IP address that it uses as an identifier; if it's for TLS then it uses maybe some kind of user name; if it's for an application then you might need an application specific identifier. And authentication in one layer, using the identifier in one layer, doesn't translate to the other layers.

Reply: Another lesson I think we should learn from IPsec was that its original use was for PKI shared secrets, and even then people wanted to use something else like one time passwords, or use their ordinary passwords, and delegate something else. This has being going on now for more years than I have been involved.

Tuomas Aura: The second point is that usually we think of this from various application orientated ways. The right solution to this problem of authenticating different layers is always to bring up the authentication, or at least use the higher layers for identifiers, and build the system so that it also supports the security of your topmost layer, whatever it is in there that you want to secure. But that's not really the kind of security problem that we currently want to solve in the Internet, in mobile phone networks, or any other communication networks. Actually the real open problems that we are developing protocols for are in some other layer; for example, mobility protocols, which could be anywhere in the link layer, network layer, transport layer, session layer... and if we want a secure mobility protocol then we actually have to use the identifiers and the concepts of the particular layer where the mobility is implemented. And that means it isn't always the right solution to use the highest layer identifier. For most mobility protocols the name doesn't mean anything; like for a mobile phone, the name of the person whose hand it is in, has very little to do with it. If he's phoning his bank the name matters, but if it's a question of securing the mobile phone network infrastructure the user name doesn't matter at all. So I would say that you actually *have* to have security at several layers of the protocol stack where they all do different authentication; you can't necessarily use the same mechanisms, the same identifiers, at all layers.

Bruce Christianson: You make a point which I think is a very good one, that existing authentication components don't take into account the different types of identity needed for different contexts. The question now is to what extent the authentication for different parts of the system which rely on communication need to know about correlation between these different identities. This happens where you have different identities above the API as well sometimes. The rather nuanced view of identifier solution might be, from the point of view of someone who wants to write a nice reusable mechanism - you tell me what your policy is and my mechanisms will support it. How flexible does the mechanism writer need to be?

Reply: It's a good question. We're working also on this Wireless LAN stuff and I think there isn't any single sort of identifier that suits all possible purposes. In some cases it might be there is this idea of service identifier, what the network names as corporate identity, and you actually want to authenticate some human readable version of this. Presumably there could be some trusted party that would coordinate these names, that might be feasible, but then how does this fit into the protocol?

Accountable Privacy

Mike Burmester[1], Yvo Desmedt[1], Rebecca N. Wright[2], and Alec Yasinsac[1]

[1] Department of Computer Science,
Florida State University, Tallahassee, FL, 32306–4530, USA
[2] Department of Computer Science,
Stevens Institute of Technology, Hoboken, NJ, 07030, USA

Abstract. As the Internet has gained widespread use, and advanced technologies such as high-speed multi-media technologies and automated digital monitoring have become a reality, privacy is at the greatest risk of all time. At the same time, sophisticated threats from hackers, terrorists, thieves, and others that would abuse privacy highlight the need to find technologies that provide some accountability. However, the goals of accountability and of privacy appear to be in contradiction: accountability tends to be about determining which entities committed which actions, while privacy seeks to hide this information.

In this paper, we discuss the apparent conflict that exists between privacy and accountability. We survey some of the issues in privacy and in accountability and highlight research directions for balancing the needs of both.

1 Introduction

As more of our daily activities are conducted on networked computers, privacy has become harder to maintain. Advances in networking, data storage, and data processing make it easier for information to be retained and accessed. Privacy is an important issue to many people, businesses, and governments, though different entities may not even agree on what privacy is. Loosely speaking, privacy is the ability to control private information, including identity and identifiers, sensitive information such as medical or financial records, and information about certain kinds of personal, corporate, or government activity. The kind of privacy usually desired in the real world is not hiding all information from all parties, but rather having the ability to disclose selected information to selected parties under certain circumstances, while preventing other disclosure.

At the same time as technological advances have made it easier to store and access data, the value of such data has often increased. For example, widespread use of personal data for authentication purposes makes personal data an increasingly attractive target for potential criminals. Governments and businesses wish to carry out data mining for security or marketing purposes, often to an extent that many individuals find uncomfortably privacy-invasive. Governments and businesses themselves also have privacy concerns regarding their own sensitive or proprietary information, as well as wanting to avoid poor public relations associated with mishandling of their citizens' or customers' private information.

Although many think privacy is important, few would advocate a society in which no data is collected and no collected data is ever used. There are legitimate needs and desires for using collected data, including uses as varied as providing customers with their desired services, detection of terrorist or criminal plots or attacks underway, identification of terrorists, criminals, and hackers, and medical research. The goal, then, is to develop and deploy technologies and policies that satisfy both legitimate privacy needs and legitimate accountability needs, and that balance the sometimes conflicting desires of individuals and society. We start by discussing privacy and accountability separately, in Sections 2 and 3, and then discuss them in relation to each other in Section 4.

2 Privacy Issues and Abuses

Users often desire to hide their activities on the network. For example, a user may desire to access Internet services without allowing anyone to know (or preventing specific parties from knowing) that she accessed the service. Legitimate instances where users may desire anonymity include:

- a potential customer would like to "window shop" electronically without the shop knowing, e.g., the information he has gathered about competing products,
- an employee may not want her boss to know that she has been accessing employment services web sites (see e.g. [41]), and
- decision-making processes where ideas are intended to be judged based on their merits rather than on their originators.

On the other hand, undesirable uses of anonymity include sending threatening e-mail, money laundering, denial of service attacks, and spam, see e.g. [40,14].

As discussed above, being able to identify oneself is a matter of authentication, while preventing others from accessing one's data is a matter of privacy. However, because knowledge of personal data is often used as a means of authentication, and because some means of authentication is often required in order to enforce limits on information access, the two are inextricably linked.

Some of the basic privacy issues are given in Figure 1. We note that some of the described "benefits" of privacy are very similar to some of the described "abuses." The difference is in the content and intention of the action, according to value judgments imposed by the owners of the network, the relevant laws, and/or social conventions. This makes a general purpose solution elusive, and suggests that legal and public policy issues must always remain part of any solution.

3 Accountability

Accountability hinges on the ability to attribute actions to the entity that caused those actions. It seems only possible if entities can be accurately identified, either individually or as part of a group [3]. Entity authentication [26] has been the

Controlling Privacy

1. Personal information
   - Identity and anonymity: I may or may not be X
   - Speech: I may or may not have said X
   - Activity: I may or may not have done X
   - Location: I may or may not have been at X
   - Knowledge: I may or may not have known X
2. Privacy with partial disclosure
   - Identity: I am not saying who I am, but I am not X or one in a list Y
   - Speech: I am not divulging what I said, but I never said X
   - Activity: I am not divulging what I did, but I never did X
   - Location: I am not saying where I am or was, but I was not at place X during time Y
   - Knowledge: I am not saying what I know, but I did not know X before time Y

   Partial disclosure can be useful, for example, for showing that you are not a member of some undesirable set, such as persons on a terrorist watchlist, without disclosing your identity.
3. Privacy benefits
   - Prevents fear of retribution and enables:
     - freedom through anonymity
     - freedom of speech, activity, movement and thought
   - Protects property rights for:
     - identity, activities, location and thoughts
4. Privacy abuses
   - Total anonymity precludes verification and lawful retribution
     - Freedom from personal retribution of malicious acts
     - Freedom from legal deterrence
     - Freedom from legal attribution
   - Emboldens malicious parties
   - Facilitates cover-up

Fig. 1.

topic of extensive research, mostly focused on cryptography as the enabling technology. Passwords and personal identification numbers (PINs) are the de facto authentication standards, though they provide only weak guarantees of authentication. While cryptography has broad power and flexibility in solving multiple security problems, its application to the accountability problem requires a trust infrastructure such as a public key infrastructure.

Beyond identification of entities, accountability also requires the ability to enact particular consequences for particular actions. Methods of accountability without authentication are also possible, if consequences can be enforced anonymously. For example, one might use anonymous digital cash to ensure that appropriate payments are made if and only if certain events occur, without requiring explicit authentication of those involved.

Accountability is presently accomplished by a combination of: entity and message authentication, action/event binding, monitoring, and trust infrastructures, each of which we elaborate on.

Entity and Message Authentication Methods. Identity may be traced [38] from an IP address, to a pseudonym [9], to an account name, to the registered name at the ISP, and often through other identifying stages, eventually, to some fundamental identifying information such as a social security number, a finger print, or possibly a DNA match. An important characteristic of authentication is the degree of confidence that can be achieved. Traditional methods depend on both rigor and heuristics to show that authenticated mappings are accurate. Another research direction addresses authentication methods based on interactive zero-knowledge proofs [20] .

Binding Actions to Events. Few mechanisms for binding actions to events have been proposed, and indeed, when such mechanisms have been engineered into the infrastructure, public outrage has purged them from the marketplace, e.g. Carnivore and Clipper [34]. On the other hand, many commercial and public domain mechanisms were developed using operational information (for example information included in a message that allows a response) to trace events to the action that caused the event, and to bind the root actions to an end user. Tools for cyberforensics have come a long way in the past five years. However, there is still more to be done, particularly to find tools that properly balance accountability with privacy.

Monitoring. To ensure accountability, monitoring must be feasible and effective. Tracing all packets on the Internet leads to massive data volumes that challenge even the most sophisticated data mining techniques to analyze. On the other hand, tracing too little activity leads to insufficient information to identify and prove the guilt of attackers, since large efforts are sometimes required to know the identity of a message originator. Despite the fact that legal actions have sometimes been taken, hackers have not been sufficiently deterred. The amount of effort required to trace perpetrators, plus the effect on the efficiency of the network, must be weighed against the effectiveness of doing so.

Trust Infrastructures. Public Key Infrastructures (PKIs) have been proposed and partially deployed as a method for managing trust in security protocols. However, there have also been advisories issued about the weaknesses of such infrastructures, particularly when they are improperly used [18]. Nonetheless, because of their architectural position between the end user and the actions taken by, and on the behalf of, the users, PKIs and related security protocol infrastructures offer an excellent environment that can balance accountability and privacy concerns when properly implemented. An important focus is to develop cryptographic infrastructure models, techniques, principles, and tools to facilitate accurate, efficient, privacy-balanced accountability.

To complement authentication methods, efforts must be made to investigate policies, tools, and techniques that facilitate accountability during system operation. Authentication in itself does not ensure accountability. Results must be traceable to action originators when necessary as a response to particular events. Traceability can either be triggered through policies that allow access under certain circumstances (along with necessary auditing or other mechanisms to ensure that these privileges are not abused), or more elegantly, through the existence of such events themselves. The classical example of the latter is Chaum's e-cash [8], in which it is the information contained in the double-spending of a coin that allows the traceability of the double spender. More recently, Jarecki and Shmatikov [28] extend this idea to releasing escrow after a pre-specified threshold of activity is reached.

In Figure 2, we highlight several issues related to accountability.

Managing Accountability

1. Binding identities to entities
   – Certificates, Tokens, Registration authorities
2. Associating actions with entities
   – Bind actions to identities to the following extent(s) and no more:
     • They can be successfully prosecuted
     • Victims can recoup losses
     • The extent of damage can be verified
   – Malicious parties can be found, but only when triggering events occur and/or when approved by proper authority
     • Federal Communication Commission E-911 Technology
   – Non-repudiation without unreasonable search
3. Associating (Intellectual) Property to Entities

Fig. 2.

4 Balancing Privacy and Accountability (Responsible Privacy)

Two canonical examples of situations in which demands of accountability must be balanced with demands of privacy are electronic voting and digital cash. Keeping a balance between the conflicting rights of different parties seems almost impossible. However, early work on electronic voting (e.g. [24,23,27,37]) and digital cash showed that some balance is possible. Additionally, research on some forms of key escrow (e.g. [2,29,34,16,15]) has suggested that it is possible to deal with conflicting requirements in a "fair" or "equitable" way using cryptographic techniques.

We note that it is not the case that privacy is necessarily the realm of the individual, nor accountability of society. As discussed before, business and governments also have privacy needs. Conversely, individuals are often well-served by a system of accountability so that they know there will be consequences for others if individuals' data is abused, as well as a deterrence effect of such consequences. In particular, a necessary component for preserving privacy in a system that allows some traceability is accountability of those that are sometimes allowed to violate privacy so that they do not routinely abuse this ability.

To properly analyze the requirements of accountability and privacy, one must understand and balance a number of potentially conflicting desires. In particular, it is important to consider the conflicts between:

– Anonymity and identification
– Confidentiality and required information disclosure
– Freedom of action balanced with attribution of action
– Cyber-privacy and cyber-forensics
– Free speech and liability/copyright

We address these tradeoff issues in more detail in the following subsections, as well as discussing some potential directions for solutions.

4.1 Anonymity Balanced with Identification

Identity protection is commonly accomplished through anonymity. Unfortunately, anonymity may be abused, and used to undermine the protection goal of accountability [14]. In particular, malice accomplished anonymously cannot be deterred by retribution. On the other hand, as we discussed before, there are scenarios where anonymity is desirable or even necessary. Clearly, anonymity and identification both have an important role to play in security and balancing their conflicting characteristics will require significant effort.

We recognize a natural conflict between allowing free speech and enforcing liability and copyright laws. For example, the owner of a Web page wants freedom to post arbitrary information, but should be liable for the content. In some countries, such as Germany and Britain, Internet Service Providers have decided unilaterally to censor controversial Web pages, an overly simplistic solution to solving this accountability problem (by removing content that might need its originators identified) that unnecessarily restricts freedom of speech.

Findings of a National Security Council study reflect similar tradeoffs between confidentiality and authentication [39], (p. 214):

> The committee emphasized the importance of authentication (over confidentiality) for both technical and policy reasons. The technical reason is that authentication is the first line of defense against the most serious threats to NISs [Network Information Systems] for critical infrastructures—intruders attempting to deny the benefits of using NISs to authorized users.

As we discussed before, often the only difference between desirable and undesirable uses of anonymity may be in the content and intention of the action, according to some kind of social assessment. For example, the U.S. Constitution provides for freedom of speech, but there are some exceptions, such as malicious,

A Toy Example: Balancing Free Speech with Liability

A speaker, Alice, distributes shares of a message to Bob and Carol. The message is a binary string m. Bob's share is a random binary string m_1 which has the same length as m, and Carol's share is m_2, where $m_2 = m_1 \oplus m$ (so $m = m_1 \oplus m_2$). Bob and Carol together can compute the message m, but separately they get no information about m. This is a 2-out-of-2 secret sharing scheme, but the shares are not linked to Alice, so Alice is not liable (accountable) for the content of m.

Next, suppose that Alice appends to each share m_i her digital signature. This enables accountability, but results in loss of anonymity. For anonymity, Alice may encrypt her signatures with the public key of a trusted friend, Justice, who will only reveal Alice's signatures when compelled by a legitimate authority.

However, this toy solution does not really solve the problem unless everyone trusts Justice and agrees on the conditions under which the signatures should be revealed. Clearly Alice's privacy is dependent on Justice. If Justice is untrustworthy or may be subjected to too much pressure from others, or if Justice's computer systems are not sufficiently well-protected from outside attack, then Alice's privacy is at risk. Privacy also requires trust on the part of Bob and Carol who may not know Justice. Similarly, accountability requires that Justice will in fact reveal the signatures under the appropriate conditions. If Justice is really working for Alice and will not reveal the information, or if Justice accidentally misplaces the relevant files, then accountability will not be properly served.

Since such mutually trusted parties do not usually exist, it become necessary to have a complex system of checks and balances to prevent and/or detect misuse. One component of such a solution is to share the signature among many justices using robust threshold techniques, so multiple justices must collaborate to reveal the signatures, and a few malicious justices cannot prevent such revelation.

Fig. 3.

slanderous, or threatening speech. On the other hand, political speech has special protections: Supreme court ruling precludes laws requiring identity on political flyers.

Several cryptographic models will support the security tradeoff issue of anonymity and identification. For example, pseudonym models, blind cryptography [8,11,10], e-commerce, e-auctions, and, of course, e-voting models.

4.2 Confidentiality Balanced with Required Information Disclosure

Another easily recognizable conflict is between confidentiality and authentication. Once the identity of the origin is divulged for authentication, the ability of entities to control all aspects of their privacy is significantly reduced. One option to protect confidentiality is to refuse to be authenticated or prove your identity. Unfortunately, authentication is a fundamental element of many security and network activities. Access control typically requires some form of identification. This opens the problem of authenticated all-or-nothing-disclosure-of-secrets (ANDOS) [6]. In ANDOS, the client privacy is protected, but without access control. We envision authenticated ANDOS will protect access control, while the server does not know which files the client has access to or those the client has accessed.

Similarly, if an entity wants to store information on a network and to protect that information from general access or manipulation, he must be able to distinguish himself as the information owner, or at the very least present some credential indicating authority to access the information. A similar conflict occurs with voting systems and auction systems.

Several cryptographic models address this security tradeoff. These employ fair [33] or equitable [7] public key encryption schemes in which the secret key is escrowed.

4.3 Freedom of Action Balanced with Attribution of Action

Privacy includes prevention of knowledge being gained by others of one's actions. This may be accomplished by either conducting activities in an undetectable or untraceable way. Unfortunately, such actions may be used to undermine accountability. In particular, bad acts accomplished anonymously cannot be deterred by fear of punishment.

To balance freedom of action and accountability, mechanisms may be devised that will attribute the action. Several distinct cryptographic tracing models can support this security tradeoff issue. For example, the broadcast encryption model can provide evidence of service theft by using tracing mechanisms [13,30,5,31]. Additionally, digital fingerprinting models can be used to deter disclosure of secrets by trusted parties, through traitor tracing mechanisms. Watermarking is another cryptographic model. Another model involves privacy-preserving database queries, which we discuss in Section 4.7.

4.4 Privacy Balanced with Cyber-Forensics

Cyber-forensics techniques are frequently at odds with privacy considerations. At the core of cyber-forensics techniques is finding the identities of those responsible for actions. Moreover, it is also important to identify exactly when an action occurred, why it was taken, what the goal of the action was, and so on. A cyber-forensics expert seeks to know every detail about every aspect of every principal under investigation. Thus, the goals of privacy are apparently in direct conflict with the goals of cyber-forensics.

Current research on cyber-forensics focuses on how to use current and emerging technologies to track, examine, and correlate information, e.g. by wiretapping, confiscating computers, and using hysteresis properties of magnetism to recover erased data. One should explore before-the-fact policies and procedures that will facilitate after-the-fact system analysis. Techniques perfected for forensics applications must be combined with management policies and practices to give a level of accountability and a manner of situational awareness to privacy issues that are not presently possible by addressing operational policies and their impact on cyber-forensic efforts.

Cryptographic methods can also be used to extend cyber-forensic capabilities by assisting investigators at reconstructing elements of an intrusion, while protecting such information in the typical, non-intrusion case. This technology is based on research on copyright techniques that focus on using special encoding (as encryption) to enhance tracing of copyright violators.

Several cryptographic models address this security tradeoff. These include targeted/limited/controlled forensics tools. Of particular interest are privacy-preserving database queries (see Section 4.7).

4.5 Freedom of Movement Balanced with Location Tracking

One approach to balancing freedom of movement with location tracking is to reveal one's location only if unauthorized movement occurs. Closed circuit television, global positioning systems [42], and RFID tags can reveal location. Information release should be controlled by appropriate policy implemented by escrow or other security mechanisms. An alternative approach is to employ group identification (group entity authentication) where one identifies oneself as belonging anonymously to a group where anonymity is escrowed [17].

4.6 Bonded Privacy

In this approach, abuse of privacy forfeits the e-bond. The cryptographic model that addresses this tradeoff issue involves an anonymous entity, a Service Provider (SP), a Trusted Arbiter(s) (TA), cryptographically verifiable transactions, and e-bonds. If the anonymous entity authorizes an illegal transaction, as adjudicated by the TA, the e-bond is forfeited and a payment is made to the SP.

4.7 Privacy-Preserving Database Queries

Privacy-preserving database queries and privacy-preserving data mining seek to allow the computation of certain information, while protecting other information. For example, the simplest case of symmetrically private information retrieval allows a client to access an individual database record without learning anything else about the database, and without the database learning which record the client accessed. Many software vendors do not reveal the source code. If the source code is private then one cannot verify correctness, check for Trojan horses, etc. Worse, if source code development is outsourced to third parties, the issue of correctness becomes even more problematic, as trust is implicitly extended to the third party even if the existence of this party is unknown to the software consumer.

An approach that could be followed is to use a layered design and ramp schemes [4]. (A ramp scheme is a secret sharing scheme in which parties may learn some, but not all information.) In this approach the designer will give to each entity:

– the logical layout of the subroutines, and
– the source code of a subroutine,

such that no entity has access to more than one block (a block being the logical layout or a source code subroutine). Each entity will verify the correctness of the block and if correct certify it.

Although this approach may seem valid, it is easy to construct a counterexample, as shown in the following example.

> Suppose a corporation wants to design a secret block cipher algorithm. To prevent it to become public, it splits the algorithm in two and reveals to different experts half the scheme. Each expert group can now analyze each part and certify it is a block cipher. However, if the second part is the inverse function of the first, then the total is not a secure block cipher!

So the certification of each part is not sufficient. One needs extra information about their inter-operability. The question then is how to guarantee this while maintaining as much privacy as possible.

A question is whether the work on privacy-preserving database queries can help address this issue. One solution is to use general secure distributed computation [25], which allows splitting the problem without the need to reveal even a single line of source code. However this solution is completely impractical for realistic program sizes.

Privacy-preserving computation on large databases is a new and exciting area that shows a lot of promise in helping to balance privacy with information disclosure [32,1]. However, much work remains to be done, such as understanding how to integrate query restriction [12] with privacy-preserving computation and how to combine provably private cryptographic methods [32,21,22] with very efficient randomization methods [1].

4.8 Emerging Approaches: New Models for Accountable Privacy

Good models are important tools for describing problems and their solutions formally. For each of the accountable privacy tradeoffs discussed above, a model is necessary. For some properties, such as privacy, we already have a model: Shannon's secrecy model, which is based on probability theory. Other models are based on computational complexity or economics. Although early research with computational complexity models led mainly to impractical schemes, recently several practical protocols have been proposed. One should, therefore investigate models based on probabilistic and computational complexity approaches. However, not all properties of accountable privacy have been formally modeled as yet. The work of Millen and Wright [35] may be a useful starting point for addressing liability.

We propose that the overarching goal for addressing the conflicting requirements of privacy and accountability must be to: (a) identify the guilty and, (b) protect the innocent. We posit that it should be possible to enable entities

Models for Accountable Privacy

1. e-Bay Model
   – Weakly authenticated accounts
   – Trust based on ongoing trust ratings
2. Business Model
   – Weakly authenticated accounts
   – Security is based on profit, i.e. it is more profitable to adhere to the rules
3. Insurance Model
   – Weakly authenticated accounts
   – Liability covered by third party insurance
4. Credit Card [Accountability Partner] Model
   – Weakly authenticated accounts
   – Institution or a groups of trusted users vouch for the protected entities
5. Deposit-Based Model
   – Weakly authenticated accounts
   – Entity provides deposit to guarantee liability in lieu of insurance

Fig. 4.

to maintain a "reasonable level" of privacy as long as they do not abuse the privilege. Specifically, we seek tools that allow decision makers to evaluate the quantitative meaning of the reasonable level of privacy and what constitutes abuse. There is extensive existing research in security, cryptography, and trust infrastructures that, when coupled with new models, methods, and techniques can provide adequate accountability while concurrently balancing security with needs for privacy. Several emerging technologies offer attractive prospects for accountable privacy. We conclude this paper by listing some basic, emerging technology models in Figure 4.

References

1. R. Agrawal and R. Srikant, "Privacy-preserving data mining", *Proceedings of the 19th International Conference on Management of Data*, ACM Press, 2000, pp. 439–450.
2. M. Bellare and S. Goldwasser, "Verifiable partial key escrow", *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997, pp. 78–91.
3. T. Beth, "Zur Sicherheit der Informationstechnik", *Informatik-Spektrum*, Vol. 13, Springer-Verlag, 1990, pp. 204-15 (in German).
4. G. R. Blakley and C. Meadows, "Security of Ramp Schemes", *Advances in Cryptology, CRYPTO 1984*, Springer-Verlag, 1984, pp. 242-68.
5. D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In *Advances in Cryptology — Crypto '99, (LNCS 1666)*, pages 338–353. Springer-Verlag, 1999.
6. G. Brassard, C. Crepeau, and J.-M. Roberts, "All-or-Nothing Disclosure of Secrets," *Advances in Cryptology: Crypto '86*, Springer-Verlag, LNCS 263, 1987, pp. 234-238.

7. M. Burmester, Y. Desmedt and J. Seberry, "Equitable Key Escrow with Limited Time Span". *Advances in Cryptology - Asiacrypt '98* LNCS 1514, Springer-Verlag, 1998, pp. 380-391.
8. David Chaum, "Blind Signatures for untraceable payments", *Crypto '82*, Plenum Press, New York, 1983, pp. 199–203.
9. David Chaum, "Security without identification: transaction systems to make big brother obsolete", *Communications of the ACM*, Vol. 28, No. 10, 1985, pp. 1030–1044.
10. David Chaum, "Showing Credentials without Identification: Transferring Signatures between Unconditionally Unthinkable Pseudonyms", *Auscrypt '90*, LNCS 453, Springer-Verlag, 1990, pp. 246–264.
11. David Chaum and Torben P. Pedersen, "Wallet Databases with Observers", *Crypto '92*, LNCS 740, Springer-Verlag, 1993, pp. 89–105.
12. F. Chin and G. Ozsoyoglu, "Auditing and inference control in statistical databases", *IEEE Transactions on Software Eng.*, Vol. 8, No. 6, April 1982, pp. 113–139.
13. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *Crypto '94, (LNCS 839)*, pages 257–270. Springer-Verlag, 1994.
14. David Davenport, "Anonymity on the Internet: why the price may be too high", *Communications of the ACM*, Vol. 45, No. 4, April 2002, pp. 33-35.
15. D. E. Denning and D. K. Branstad, "A Taxonomy of key escrow encryption systems", *Communications of the ACM*, Vol. 39, No. 3, 1996, pp. 34-40.
16. Y. Desmedt, "Securing Traceability of Ciphertexts—Towards a Secure Software Key Escrow System", *Advances in Cryptology—Eurocrypt '95*, LNCS, Springer-Verlag, 1995, pp. 147-57.
17. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer, 2004.
18. Carl Ellison and Bruce Schneier, "Ten Risks of PKI, What You Are Not Being Told About PKI", *Computer Security Journal* , Vol. XVI, No. 1, 2000, pp. 1–7.
19. A. Fiat and M. Naor, "Broadcast Encryption", *Advances in Cryptology–Crypto '93*, LNCS 773, Springer-Verlag, pp. 480–491, 1994.
20. Uriel Feige, Amos Fiat, and Adi Shamir, "Zero-knowledge proofs of identity, *Journal of Cryptology,* , Vol. 1, No. 2, pp. 77–94, 1988.
21. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N. Wright, "Secure multiparty computation of approximations", *Proceedings of 28th International Colloquium on Automata, Languages and Programming*, LNCS 2076, Springer, 2001, pp. 927–938.
22. M. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection", *Proceedings of Eurocrypt 2004*, LNCS 3027, Springer, 2004, pp. 1–19.
23. A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Election", *Advances in Cryptology-Auscrypt '92*, LNCS 718, Springer-Verlag, 1992, pp. 248–59.
24. J. Furukawa, H. Miyauchi, K. Mori, S. Obana, and K. Sako", "An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling", *Financial Cryptography 2002, 6th International Conference*, LNCS 2357, Springer-Verlag, 2003 pp. 16–30.
25. O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game", *Proc. of 19th STOC*, pp. 218-229, 1987.

26. D. Gollmann. "What do we mean by Entity Authentication?" *In Proceedings of the 15th IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, 1996, pp. 46–54.
27. M. Hirt and K. Sako, "Efficient Receipt-Free Voting Based on Homomorphic Encryption", *Advances in Cryptology—Eurocrypt 2000*, LNCS 1807, Springer-Verlag, 2000, pp. 539–556.
28. S. Jarecki and V. Shmatikov, "Handcuffing Big Brother: an Abuse-Resilient Transaction Escrow Scheme", *Advances in Cryptology—Eurocrypt 2004*, LNCS 3027, Springer-Verlag, 2004, pp. 590–608.
29. L. R. Knudsen and T. P. Pedersen, "On the difficulty of software key escrow", *Advances in Cryptology—Eurocrypt '96*, LNCS 1070, Springer-Verlag, 1996, pp. 237–44.
30. K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *Eurocrypt '98, (LNCS 1403)*, pages 145–157. Springer-Verlag, 1998.
31. Kaoru Kurosawa and Takuya Yoshida. Linear code implies public-key traitor tracing. In *Public Key Cryptography*, volume 2274 of *LNCS*, pages 172–187. Springer, 2002.
32. Y. Lindell and B. Pinkas, "Privacy preserving data mining", *J. Cryptology*, Vol. 15, No. 3, 2002, pp. 177–206. An earlier version appeared in *Crypto 2000*.
33. S. Micali, "Fair Cryptosystems", *Advances in Cryptology, Proceedings of Crypto' 92*, LNCS 740, Springer-Verlag, 1992, pp. 113–138.
34. S. Micali and R. Sidney, "A simple method for generating and sharing pseudo-random, with applications to Clipper-like key escrow systems", *Advances in Cryptology—Crypto '95*, LNCS 963, Springer-Verlag, 1995, pp. 185–196.
35. J. K. Millen and R. N. Wright. "Reasoning about Trust and Insurance in a Public Key Infrastructure," *Proceedings of 13th IEEE Computer Security Foundations Workshop*, IEEE Computer Society, July 2000, pp. 16–22.
36. Donald Rumsfeld. US Secretary of State, Comments to the press, Sept 12, 2001, `http://www.defenselink.mil/transcripts/2001/t09122001_t0912sd.html`.
37. K. Sako and J. Kilian, "Receipt-free mix-type voting scheme", *Advances in Cryptology—Eurocrypt '95*, LNCS 921, Springer-Verlag, 1995, pp. 393–403.
38. Didier Samfat, Refik Molva, N. Asokan, "Untraceability in mobile networks", *Proceedings of the 1st annual international conference on Mobile computing and networking*, Berkeley, California, 1995, pp. 26-36.
39. Fred B. Schneider. *Trust in Cyberspace*, National Academy Press, 1999.
40. B. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computers and Security*, 11(6):581–583, October 1992.
41. Gene Spafford, "Protecting Personal Information in Academia," *Computing Research News*, May 2001, pp. 3, 4, 12. `http://www.cra.org/CRN/articles/may01/spafford.html`.
42. Mike Spreitzer and Marvin Theimer, "Providing location information in a ubiquitous computing environment" (panel session), *December 1993 ACM SIGOPS Operating Systems Review, Proceedings of the fourteenth ACM symposium on Operating systems principles,* Vol. 27, No. 5, 1993, pp. 270–283.

Accountable Privacy (Transcript of Discussion)

Rebecca N. Wright

Florida State University, USA

Initially Mike and I had the idea that we would each take sides; one would be accountability, one would be privacy, and we'd debate, but actually both of us wanted to be on the privacy side. So instead I'll talk for a while about this apparent conflict between privacy on the one hand and accountability on the other. We'll look at a number of existing technologies, some open questions, and although I guess there's no new research in the paper now, the hope is that looking at things through this lens might ultimately lead to new solutions. We're all aware that change in technology is making privacy much harder: we have much more use of computers, computers are heavily networked, storage costs are going way down, and computers are getting more powerful, so we have increased ability to process and store large amounts of data. So whereas it used to be we had to work hard to make information public - that's what a lot of computer science was about, getting information from one place to another in a reliable and persistent way - now the infrastructure has evolved to make that very easy. What's become harder is to keep the information private, understanding where it flows, when it flows, and the fallout of it all. And the issue of privacy has become much more critical as public awareness increases of this new world that we're in and as potential misuse starts to be on the rise, and also I think as these conflicting roles increase: the desire for privacy on the one hand, and the desire for using this information on the other hand.

Privacy has a social context, so if I ask each one of you what is privacy, you'll have a different answer, and if I ask you, do you consider the fact that our conversation is being taped now to be a privacy violation, some of you may say yes, some of you would say no, some of you may have different ideas about why it's relevant to privacy. But many people do think privacy of content is important, even though we may disagree about exactly states are the violations. Yet we also think that use of information is important, so few of us - well few in the world, and maybe those few are all in this room - would advocate a society in which no data is collected, and no collected data is ever used. You have the choice yourself to try to live outside of any kind of data collection; it's quite difficult to do.

Understanding some kind of proper balance between collection and use of information, and complete privacy, is really a social decision. It's a policy decision, a decision that different communities make differently, and even within a community people may disagree with the decision that's been made. So, for example, here in Britain there will be cameras at many intersections, and you could say that Britain as a society has agreed that the value of having that, and so being able to reduce motor vehicle accidents and certain kinds of crime, is worth the privacy that's given up in having your picture taken as you go through the intersections.

So there really are policy decisions to be made, and technology is relevant because technology can support policies. One possible course of action in technology is to try and enable as wide a range of social policies as possible. Another direction is to say, here are the policies I like, I'm going to try to combine technologies that support those policies, or to make policies possible that were not possible without that kind of technology. So although privacy's a social concept, there's definitely an interplay with technology because technology can sometimes find new points on the trade off, you find solutions that have more privacy and more accountability than you can get without technology.

Privacy may be a matter of protecting your identity, so I may or may not be person X; or speech, I may or may not have said X; or indeed I may or may not have done some activity; I may or may not have been at a particular location; I may or may not have known X; and of course all of these could be within a particular time frame. And so if you look at different technologies that protect, for example, anonymity of speech, or of identity, usually you can divide the systems into those that may say who you are but protect what you said through encryption and those that might have pseudonyms and the messages are completely clear so *what* you said, or what you did, e.g. browse through some web pages, is completely open, but *who* did it, is not. There are different ways of trying to accomplish goals like this.

And if you want to weaken it a little bit and not have full privacy, or strengthen perhaps in the case where you want to allow someone to give some kind of selective disclosure, then you may add some kind of deniability: I am not saying who I am, but I am not person X or someone in some list Y; I'm not divulging what I said, but I never said this particular thing X. They have to be a little bit careful with this from the privacy perspective, because imagine if you have a solution that allows general deniability: then someone like a judge may say to you, OK, prove you didn't say X, and you would say, well I won't prove that to you, and he would take that as evidence that you did in fact say X.

Matt Blaze: There seems to be a distinction between *I* can prove that I *didn't* say something, and *you* can't prove that I *did* say something. The second one sounds like the intuitive definition of deniability.

Reply: Maybe that is general deniability, and this is some kind of selective disclosure; the terminology might not be quite right.

Matt Blaze: But I think there's also a deeper issue with this. What you can prove I did, and what I can prove that I did, are intuitively different to me in some way that I am having a little trouble pinning down[1].

Reply: Right. So for example, if you look at the various kinds of digital signatures, you get separation between the typical kind of digital signature that you can turn around and show someone else, versus the sort of dedicated confirmer where at the end you'll believe that I said it but it doesn't have any kind of evidentiary value, or provide a thing that you can show someone else. So there are some things that maybe you'd like to be able to prove to particular people in particular settings, but not necessarily all the time, and not under demand to a judge, who could then put a binary search on what you did.

Tuomas Aura: How do you totally prove that you didn't do something?

Reply: For that one I feel there is some hope. The one I think is really interesting, and probably impossible in some formal sense, is how you prove you didn't know something. At the time you do something, you commit to some description of what you did, or at the time you knew something you commit. Then later you can maybe prove certain properties in a controlled zero knowledge way about some subset that these actions, or representations of information, are in or out of.

Tuomas Aura: What they need is some kind of alibi.

Bruce Christianson: Yes, the usual approach to an alibi is proving that you weren't there because you were somewhere else, and that you didn't say something because you said something else.

Reply: That's right, and so here you maybe would want some kind of set you'll prove that what you did is in.

Virgil Gligor: The solutions may be quite simple but undesirable, such as you log everything everyone does.

Reply: That's right.

[1] Editors' note: if Alice has general deniability (in Rebecca's sense) then for every assertion x either Sx (Alice said x) or else PNSx (Alice can prove that it is not the case that Alice said x). Thus writing Q for NPN we have CQSxSx where C as usual denotes Polish material implication (i.e. Cpq means either q is true or p isn't) and Qx interprets as "x is consistent (for Alice)".

Following the judges mandeamus, either Alice stands in contempt of court or else CPNSxSPNSx. This is equivalent to CRQSxQSx, where R = NSN is non-denial, and we already have CQSxSx, whence CRQSxSx. Assuming at the meta-level that the court adopts CRQSxSx as a rule of inference, it follows that in the posited case (although not in general) Matts two notions are coterminous: Alice can prove NSx exactly when the court cannot prove Sx. This is precisely the danger of general deniability, as Rebecca points out.

Worse, substituting PNSx for x in CQSxSx and contraposing gives CRQSxPRQSx whence (via CRPSxRQSx and CPRQSxPSx) the court can deduce CRPSxPSx.

Ross Anderson: As an example,"shibboleth" is proof that I am not an Ephraimite. The Bible describes the fight between the Gileadites and the Ephraimites. The Ephraimites tried to escape across the River Jordan, but the Gileadites identified them by asking them to pronounce the word "shibboleth". Whoever said "sibboleth" was immediately put to death[2].

Reply: Right, in some very specific cases you have specific tokens that you can use to prove these kinds of things. Now I'm going to turn to accountability for a minute. Usually when we deal with accountability it's very much tied to identity. We have all kinds of ways of binding identities to entities; certificates, tokens involving registration authorities, either straightforward public key certificates with an identity and a key, or they might be credentials that say something like I am in some set, like I'm over 21. And as with some of the work we heard about earlier today, it is possible to have pseudonyms rather than actual names, but even then there's something that you can pick up; there's some replication of identity.

Accountability is usually about associating actions, or things that have been said as an action, with these entities. If you want to go more towards privacy then, rather than logging everything, you only actually make those associations if certain trigger events occur. This could be of the e-cache type where the event itself creates the ability to do the binding, or it could where an authority has the ability always to do the association but only will do it under certain conditions, and has to be trusted, or somehow controlled, to act at the right time.

But the goals of doing this action are usually so that the responsible parties can also be identified and successfully prosecuted, so that victims can recoup losses, or if you can't do that, at least the extent of the damage can be verified and under control. So if you can find ways that maybe do less association of actual identities but still give you recouping or damage control, maybe that's enough so you can give up this strict reliance on entities. And then there's a whole other area of associating intellectual property with entities that could also be part this.

The balancing of privacy and accountability is where the social aspects come in. To understand what balance is desired one has to understand the context, and often if you look at the notion of the rights of the individual and the needs of society inherently it seems intuitive that there will always be conflicts between the desire for anonymity and the desire for identification. The desire of confidentiality versus the desire of verification disclosure, freedom of action versus attribution of action, your privacy with your activities on-line versus cyberforensic evidence, and free speech versus liability and copyright issues.

Mike Burmester: As Rebecca said, privacy is to do with the rights of the individual, whereas accountability is to do with society. The rights of the individual can be determined in different ways, that have to do with policies, and politics, and other things, and it's not up to us to judge the right or wrong, but to somehow find technologies, or mechanisms, that can implement them the way it is

[2] Judges 12.6.

designed. Even that is still rather interesting because sometimes it is difficult really to understand, to see how things work, unless you discuss them in detail.

Anonymity is commonly used for protecting the identity of the individual. On the other hand the individual can use this anonymity to accomplish malicious acts, so malice accomplished anonymously cannot be deterred by retribution. There has to be some kind of balance here. There's a natural conflict and this extends into free speech, for example, and enforcing copyright laws.

As Rebecca pointed out, often the differences have to do with intention, and intention is much harder to determine whether there is abuse or not.This perhaps highlights the whole approach here, the idea that if the user has abused the system then society has the right to somehow get whatever there is back from the abused material.

Ross Anderson: But if a message from Al Quaeda appears on a website somewhere, then the NSA will simply insist on the decryption and identification of every other message that comes from the same source. So if you're going to use Trusted Computing mechanisms, for example, to implement something like this, then there's going to be a user requirement for much more widespread powers of search and indexing of the encyphered material, as well as just access to specific named content. Similarly if a dirty old man is observed holding a conversation in a junior chat room, the police will demand indexing and retrieval of every message he has ever sent or received previously.

Mike Burmester: I think it is an open question, how far do you go, do we reveal everything, or do we reveal just enough.

Andrei Serjantov: At the moment this problem is essentially solved by lack of resources. If we all start saying bad things, then not all of us will get arrested, but this is only because it's difficult at the moment to go and grab people, and arrest them, and prosecute them etc, etc. By allowing people to decrypt things very easily, this becomes a lot easier.

Mike Burmester: I'm not proposing *very* easily, I'm proposing only if there is proof that criminal activity took place. You go through the court, and the court proves it.

Andrei Serjantov: Well you're effectively proposing a very easy way of identifying people, which is something that in the online world should be the equivalent of what happens in the real world at the moment.

Mike Burmester: Eavesdropping.

Andrei Serjantov: Right, so eavesdropping from your phone is hard, because someone has to go and put the tap on it, while decrypting a message is easy.

Mike Burmester: Maybe decrypting should require an approval from the court, to make it equally hard.

Tuomas Aura: Getting approval from the court is easy, because the police don't need evidence to prove that the crime has occurred: That's what they are trying to investigate. For example, there was a remailer in Finland run by a private person, and they closed down eventually when the Church of Scientology in California went to the local police and claimed that in order to solve some particular crime they had to get the identity, the name of the person. Then the police from Interpol made a request to the police in Finland, and they could just say this is a standard Interpol request, and the law doesn't provide any special anonymity protection for remailers. This wasn't abuse, this was that someone suspected something. Every day the police investigate crimes when they are not sure whether one happened or not.

Mike Burmester: I'm not the lawyer, but the idea would be here you should have some evidence and not just suspicion.

Matt Blaze: There's a real risk of making a very similar mistake to the one that I think was made during the key escrow nonsense. I think one of the fundamental mistakes in the key escrow debate was framing it as a strictly technological issue. And in particular, surveillance - in law enforcement - and it's more so of law enforcement than the judicial system, is largely outside of the scope of interception technology. So, for example, law enforcement depends very heavily on the ability of the police to arrest people and question them, and identify them, and conduct very traditional kinds of investigation. It's not clear to me that it's necessary to design a communications and computing infrastructure that replicates all of the real world functions of law enforcement within the system. Technology has to be coupled with policies and other things; technology alone can never do exactly what you want. Whether technology does it or not, those other mechanisms are quite powerful and will assert themselves as society policies exist.

Reply: But this is why I think ideally we want some combination of things like, for example, double-spending where it's the bad act that releases some information, you don't just want a paper trail that says, I am allowed to decrypt this because whatever. The equivalent of a private conversation in a private room should still always exist, and you should know when you're potentially in that situation, and when you're essentially not.

Ross Anderson: But Rebecca, there was another point on key escrow there, which is that most people assume that privacy is about content, whereas in fact privacy in law enforcement is almost exclusively against traffic data.

To first approximation no content is used, it's all itemised phone bills and stuff like that. I took this line during the mid-90s, it was very unpopular then but there is masses of evidence now. If you're going to build a system which shields traffic data from investigators so that everybody's itemised phone bill is encrypted up to the eyeballs, that would be an interesting technological problem, but even given key escrow, how could you support the kind of searching and correlation that would be required under a court order? It's very hard to imagine any purely mathematical crypto mechanism that would do that, it would just wholly be an itemised phone bills database on a trustworthy machine with some people who are liable to arrest and imprisonment if they abuse their powers of search.

Mike Burmester: But these are all mechanisms which try to address this kind of balance problem, whether desirable or not.

George Danezis: I guess in order to address all the privacy questions when you talk about anonymity and the capability of the individual, if you put forward technical mechanisms for the first side, you should always put forward technical mechanisms for whoever's going to revoke to be accountable too.

Mike Burmester: I agree, but the problem of the citizen abusing the system, and of the system abusing the citizen are both accountability. You can cheat on privacy as an individual, and the system, the state, can cheat on accountability; and the technology should be there for preventing the state to cheat on accountability.

Bruce Christianson: There's an interesting question about how much privacy the state needs. If there's an investigation, and somebody's line is being tapped, then clearly you don't want them to know that their line is being tapped because they're liable to say something different. There are other kinds of investigation where if the person wasn't aware that their house has been searched, then the police can't use stuff that they get as evidence. I'm trying to make the argument that privacy isn't always an individual thing, and accountability isn't always a social problem.

Mike Burmester: I agree with you, it is more complex, but essentially here privacy I think has to do with individuals, and accountability is the measure which somehow controls the abuse of the individuals.

Bruce Christianson: Sure, and therefore abuse by the state is a major threat.

Mike Burmester: Freedom of movement and location tracking, CCTV's are very common in Britain, it's a serious privacy problem, but on the other it does control petty crime, petty crime usually moves out of the area where there are CCTV's. The problem is that there have to be mechanisms to decide when they're abused. Bonded privacy comes low level, what happens is if you abuse privacy you pay a price, you forfeit your e-bond.

Essentially it should be possible to enable certain types of technologies which will somehow maintain a reasonable level of privacy provided the entities are used legitimately. And the question is, what response do you refuse in different circumstances, in different countries, in different societies, there's a different definition of abuse.

Bruno Crispo: So these approaches maintain a reasonable level of accountability as long as they don't abuse the privilege of accountability.

Mike Burmester: Yes. Yes.

Richard Clayton: I feel unhappy with this subtext here, that we have to build into technology something that will somehow make it weaker in order to give this ability for people to investigate abuse. I think you're not looking at the whole system, because the system's not just technology, it's the people who operate it, who are usually really inept at doing this, or divulge stuff all by themselves, particularly if they're manipulated. So even if you give them the best anonymous channel in the world, you tell them a plausible enough lie and ask them their address, they will tell you, and nobody needs to break the technology.

Mike Burmester: I know, but determined criminals will not do what you have just suggested.

Richard Clayton: I suggest that criminals are just as dim as everybody else. The second thing is we're changing our society in order to take a different view of things which are in some sense anonymous. For example, our media tend to not to amplify anonymous statements; lots of papers will now not run single source stories because they're not prepared to look though they've been manipulated.

Mike Burmester: Newspapers and TV stations have to be accountable because if they do the wrong thing, they are in serious trouble, but if you go on the Web you can find all sorts of sourced information, and there's no accountability there.

Richard Clayton: Well we're beginning to understand that as a society and to disregard it. Why do you have to break the technology?

Mike Burmester: There's a scale; one solution is you just pretend, you hope for the best. Decent people will not watch pornography. Another model is that, if you don't behave then you lose trust; that's e-Bay model.

Richard Clayton: I don't think that's true anymore because e-Bay's model is based on the fact that there's traceability through the credit card information which it has. You can't sign up, you can't sell anything, or buy anything on e-Bay, unless you use a credit card.

Mike Burmester: Yes, but it is in your interest to get more money by behaving honestly than by cheating.

George Danezis: Enron excepted. [Laughter]

Mike Burmester: Another model is Insurance; you're covered by third party liability; a lot of business works like this. Or a credit union: some institution or group of trusted users who are prepared to vouch that you will pay. Credit card companies are prepared to pay for this; they can get two and half percent here, and they are extremely good at doing it. Deposit based model, you pay a certain deposit that you would forfeit if you don't behave well.

Hackers, and terrorists, and thieves are becoming extremely sophisticated. So we have to find technologies that will be able to balance what we call legitimate privacy against the accountabilities of society.

Pasi Eronen: I would object to this premise that hackers and criminals are more sophisticated; I don't think they are.

Mike Burmester: It's not very easy to design viruses nowadays; seven years ago it was. To design a virus these days that won't be traced is not easy. You can do it but you have to be much smarter.

Matt Blaze: I had a class full of students this semester and the first assignment was to write an exploitable Buffer Overflow, they all did that. The second assignment was to write a virus that would pass the anti-virus software, and all of them did by the following week. Maybe my students are smarter than everyone else, but...

Pasi Eronen: From society's point of view it's not important if there are a couple of smart people who can do advanced things, it's whether the average attacker is becoming more and more sophisticated.

Mike Burmester: I disagree with that because a smart hacker could do far more damage. If you design a virus, it all depends how you use the virus.

Andrei Serjantov: Yes, but there is some evidence that all of these viruses haven't actually had payloads. We all know how to design viruses with far worse consequences for the world, yes, so there is some evidence that smart people don't actually design destructive payloads.

Mike Burmester: Or haven't as yet, because there's no money in it. Losing your post as a professor will cost you more than what you would gain by doing it.

George Danezis: But the worst terrorist attacks we've had in the past few years were not using cybertechnology.

Mike Burmester: No, because you hack into power stations.

Pasi Eronen: Well it's still amazing how much gets written about cyber terrorism without there ever having been a single cyber terrorism instance.

Reply: Part of the response to terrorism is that some of the planning is done through electronic communications, and an informed society can now monitor all electronic communications. Our response to that is that they must also use encryption anonymity, and so the question is, is society served by having only anonymity and encryption, or is that part of a larger picture?

Matt Blaze: But Rebecca, I think the fallacy there is the assumption that the only way to violate anonymity is with mechanisms that are in the communication system. The police, and the government, still have an enormously powerful toolkit for fighting crime; they can arrest people, they can get witnesses, they can get informants, and they have wiretaps. So if the state says that this is a closed system then analyse it as a closed system.

Reply: I agree. I hope you don't see this talk an advertisement or endorsement of key escrow, it's not, but it is to say there's a whole system, including the social parts, you know, there is technology that supports doing things in a way that gives some privacy and an accountability.

Pasi Eronen: The big problem with key escrow, you can have crypto systems with key escrow, but bad guys use something else. It's like CCTV.

Mike Burmester: The problem there is the shift. The petty crime moves from the neighbourhood where you have these closed circuit TVs to the next neighbourhood.

Ross Anderson: This is to some extent a re-run of the debate in the mid 90s. I am concerned about the wisdom of putting a large effort into trying to do inference-control in databases using cryptographic means. There's only one instance that I know of where this has been attempted for real, and that's a medical database in Iceland where the promoters of the database sought to reassure the Icelandic population by saying, we will encrypt the geneological database, the genotypic database, and the medical records database separately, using separate keys, which are kept by the Icelandic Information Commission in Reykjavik, and this means that nothing can go wrong. We're using encryption, they even pointed, look there's a copy of Bruce Schneider's book on the shelf of the guy who was writing this code. But of course this is the wrong level at which to do such protection, the real issues in database privacy are to do with inference control - whether we permit a number of queries to home in on a particular individual - and the system is almost completely vulnerable to that. They had used the wrong protection mechanisms, for show and for propaganda only, and they had not protected what it was prudent and sensible to protect. If you start trying to port crypto into environments where crypto is inappropriate, then it doesn't give you any leverage, you're just wasting money.

Reply: The system can be abused, whatever mechanism you use, whether it's crypto or anything else. But if it's crypto then it's more vulnerable because people will not realise it was abused.

Toward a Broader View of Security Protocols

Matt Blaze

Department of Computer and Information Science
University of Pennsylvania
blaze@cis.upenn.edu

Computer and network security researchers usually focus on the security of computers and networks. Although it might seem as if there is more than enough insecurity here to keep all of us fully occupied for the foreseeable future, this narrow view of our domain may actually be contributing to the very problems that we are trying to solve. We miss important insights from, and opportunities to make contributions to, a larger world that has been grappling with security since long before the computer was invented.

This position paper initiates and advocates the study of "Human-Scale Security Protocols" as a core activity of computing and network security research. The *Human-Scale Security Protocols (HSSP)* project treats "human scale" security problems and protocols as a central part of computer science. Our aim is to identify, stimulate research on, analyze, and improve "non-traditional" protocols that might either have something to teach us or be susceptible to improvement via the techniques and tools of computer security. There are compelling security problems across a wide spectrum of areas that do not outwardly involve computers or electronic communication and yet are remarkably similar in structure to the systems computer scientists routinely study. Interesting and relevant problem spaces that computer security has traditionally ignored range from the very serious (preventing terrorists from subverting aviation security) to the trivial and personal (ensuring that a restaurant serves the same wine that was ordered and charged for).

We use the term "human-scale security" to refer here to high-level security protocols (such as commercial transactions) performed manually by people and to systems and objects intended for direct human interaction (such as mechanical access controls and paper documents). It is distinct from (but related to) the study of the various financial and legal protocols that are analyzed in economic terms, e.g., with a game theoretical model of behavior, and with the aim of designing systems that encourage fair play. Rather, we are concerned here with the often informal protocols that have evolved to prevent outright cheating or crime, as well as with the (often *ad hoc*) non-computerized security mechanisms and practices that protect the physical world. An important characteristic of these protocols and systems is that their design and operation do not depend on, and are not motivated by, electronic computers or communications systems.

Human-scale security systems are relevant to us first because they form the basis (the "root") of trust in the complex systems used for society's basic functions. The trustworthiness of any system (computerized or not) ultimately depends on the integrity and reliability of the people who built and run it, on the security of physical objects, on the soundness of information, and on procedures carried out by human beings (who employ whatever explicit or implicit interfaces the system provides). Yet we often have only informal, *ad hoc* metrics for the security of these basic system elements, and even less of an understanding of how their security properties compose than we do for computing systems. These systems often fail in ways that mimic common security breaches in computers, with similar results and for similar reasons.

Secondly (and conversely), for all their *ad hoc* properties, human-scale security systems appear to have much to teach us. Protocols and systems implemented and used directly by people tend to be heavily optimized for efficiency as well as for performance against the perceived threat model and risk. Their evolutionary development process is often much slower (and more informal) than that used to produce computer systems, with optimizations typically discovered by users seeking to reduce their effort and expense and with countermeasures against specific vulnerabilities introduced only after attacks become a perceived practical risk. Some of the resulting protocols seem to be quite good, perhaps close to optimal for their applications.

The study of human-scale security seems to have much to offer computer and communications security research and practice. How secure are these protocols when analyzed with the methods and against the threat models of computer security? How can well-optimized and highly risk-sensitive human scale systems be adapted to improve computer security protocols? How do human-scale security elements compose? How do they interact with computer security? Can we adapt the trust management techniques from computing to specify and enforce better security in human-scale systems?

1 Human-Scale Security Protocols

There has been relatively little work on human-scale protocols by computer scientists and cryptologists, at least in the half century since we became distracted by the invention of the electronic computer. However, the relatively few example of serious computer science and cryptologic investigation into the subject that do exist are in fact quite encouraging.

Two recent papers from the computer science literature illustrate the kinds of analysis advocated here. The first provides a striking example of how an obvious attack from the physical world can expose a much less obvious, yet fundamentally similar, vulnerability in computer networks. The second introduces an attack against a human-scale system that seems entirely obvious when examined in computer security terms but that remains quite obscure otherwise.

1.1 Denial of Service and Burglar Alarms

Our example from the former category is the delightful CACM paper (and keynote address) by Needham ten years ago on the problem of denial of service in (physical world) burglar alarm systems [19]. The central insight here was that an alarm system can be compromised in two ways: the direct and most obvious approach, which involves preventing the alarm signal from being sent (e.g., by cutting wires), and a more subtle, indirect approach, which involves overwhelming the system's capacity to respond by repeatedly triggering perfectly valid alarm signals.

To perform this second kind of attack, a burglar might set off the alarm (e.g., by rattling the door), wait in the bushes for the police to respond, and set the alarm off again as soon as they leave, repeating until the police stop responding altogether. In other words, Needham observed that we can deny service not only by preventing the system from working, but by letting it do exactly what it was designed to do. (Here the attack seems *more* obvious when the system is analyzed in physical terms than it does when analyzed in abstract terms, since anyone actually installing, responding to, or trying to defeat an alarm system would intuitively consider the capacity of the police to respond as an integral part of the system). The attack illustrates a tradeoff in the design of such systems: greater sensor sensitivity increases security only until it becomes possible for an attacker to deliberately manipulate them, at which point it becomes a serious liability.

The subsequent recent DoS attacks on the Internet exploit a similar property of the IP architecture: one need not cut the wire associated with a network link when one can simply persuade the hosts that are connected to that wire to overwhelm one another with traffic.

1.2 Privilege Escalation in Mechanical Locks

Our example from the later category is drawn from our own initial work on human-scale protocols: the privilege escalation vulnerabilities in master-keyed mechanical locks[6]. Here, the central insight is the modeling of a conventional master-keyed pin-tumbler lock as a online authentication oracle. This is a natural first step from a computer security perspective; a lock is, after all, an "online" service that accepts or rejects keys. An obvious question to ask is whether the "advertised" interface to the service can be exploited to leak information about its secrets (in this case, the configuration of the valid key).

Ordinary (single-key) locks appear to do well under such an analysis. A lock (an ideal, *abstract* lock without mechanical imperfections, at least) will not open unless all of its tumblers are set to their correct secret depth by the key. The "oracle" gives the same response to an "almost correct" key (one that sets most of the tumblers to the correct depth but has at least one at the wrong depth) as it does to a "completely incorrect" key (one that sets all tumblers to the wrong depth). The best attack against an ideal lock appears to be exhaustive search of the keyspace. This should be reassuring for lock users, since this search would be exponential in the number of tumblers. Lock attackers must resort, therefore, to exploiting vulnerabilities in the implementation (e.g., by "picking" the lock).

Master keyed locks, on the other hand, immediately crumble under such an analysis. (A master keyed lock system is one in which locks can be opened by at least two keys, a unique "user" key that opens a single particular lock, and "master keys" that can open all or some of the locks in the system). Here the oracle can be exploited efficiently and easily to solve the (ostensibly exponential) problem of amplifying the rights of an ordinary user key into those of a system-wide master key

The details of the attacks are beyond the scope of this paper, but depend on the fact that the tumblers in master-keyed mechanical locks have two correct (and secret) depths, one for the user key and another for the master key. The tumblers are "evaluated" independently of one another when a key is inserted; the lock will open with a key cut completely as a user key, completely as the master key, or *with some tumbler positions cut for the user key and others cut for the master key.* This latter property allows the holder of a user key to issue a small number of queries to the lock (via specially cut test keys) to learn the master key's tumbler depths one by one.

As obvious as this attack may be in computational terms, it is in fact quite obscure if locks are modeled only in mechanical terms, as they have been for hundreds of years. Most computer scientists and cryptologists quickly see several such attacks once the problem is set up and described in this way, but it requires remarkable and unusual insight for a locksmith to discover or recognize any problem at all, given the mechanically-oriented analytical tools ordinarily employed in that domain. (And indeed, it isn't entirely clear how widely known this attack was within the trade; some locksmiths claim not to have known about it all, while others assert that "everyone" who "needed" to know such things already did. All we can be sure of is that any locksmiths and lock makers who did know of the attack must not have considered it to be very serious, since they failed to warn potential users about it, develop or migrate to different designs, or even document it.)

An interesting property of this attack is that it arises from a basic design weakness in how locks are master keyed, not from any flaw in their implementation or manufacture. Any lock master keyed with the standard techniques, no matter how well-made, suffers from this vulnerability; it is more analogous to a "protocol failure" than to an "implementation bug."

This small result is encouraging; aside from exposing a practical yet little known weakness in locks, it provides an accessible demonstration of several otherwise rather abstract computer security principles. It suggests that the methods and tools of computer security have something to contribute beyond the security of computers.

2 Some Human-Scale Security Problems

Aside from the examples in the previous section, it is not difficult to find human-scale systems whose characteristics and requirements are similar to those of computing and communications security.

What follows is a rather scatter-shot sampling of human-scale security problems that might be susceptible to a computer science style of analysis or that seem to have something to teach us (or, perhaps in most cases, both). These examples are intended merely as starting points to illustrate the relationship between computer security abstractions (and vulnerabilities) and human scale systems and to stimulate further discussion and research, not as as a comprehensive survey of the problem space.

2.1 Ordering Wine in a Restaurant

The procedure in restaurants for ordering a fine (or not so fine) bottle of wine follows a largely standardized set ritual over several rounds of interaction. The wine ordering problem appears to have several familiar (and difficult) requirements: secret communication (not revealing to eavesdroppers – the guests at the table – the price range of the wines being advised about), integrity (ensuring that the correct bottle is delivered, opened and not switched for a cheaper product), and non-repudiation (making it difficult for the customer to claim that he or she had really ordered, and believed to have consumed, a cheaper bottle when the bill is presented).

The protocol appears to have evolved over many years, and seems to be well optimized for efficiency (it requires only a few round trips) and against practical threats (to the point that today hardly anyone seriously worries about fraudulent wine downgrades, although it must have been a problem at some point in culinary history). There seems to be much to admire about this protocol, and it appears to be amenable to more formal notation and analysis. What can we learn from it?

2.2 Paying the Check in a Restaurant

As well-optimized as the restaurant wine ordering protocol may be, the familiar protocol for settling the bill with a charge card is less encouraging. The common protocol requires at least six "flows" between the server and the patron (three round trips):

$$\begin{aligned}
P\ (\mathit{patron}) \rightarrow S\ (\mathit{server}) &: \text{Request bill} && (1)\\
S \rightarrow P &: \text{Calculate and present bill to P} && (2)\\
P \rightarrow S &: \text{Examine bill;}\\
&\quad \text{if incorrect complain and have bill recalculated;}\\
&\quad \text{if correct summon server} && (3)\\
S \rightarrow P &: \text{Collect bill and card from P;}\\
&\quad \text{run charge;}\\
&\quad \text{return card, charge slip and bill to P} && (4)\\
P \rightarrow S &: \text{Examine charge slip and bill for consistency;}\\
&\quad \text{if incorrect complain and have charge invalidated;}\\
&\quad \text{if correct sign and summon server;}\\
&\quad \text{exit} && (5)\\
S \rightarrow P &: \text{Collect signed charge slip} && (6)
\end{aligned}$$

This protocol involves separate interactions for summoning the server to obtain the bill, examining the bill, summing the server to collect the the credit card, examining the charge, and returning the signed charge slip. Intuition suggests that these steps are needed to protect the patron against incorrect charges and the restaurant against invalid or over-limit cards.

In fact, at least one round trip can be eliminated and its steps collapsed without any apparent loss of security for either party:

$P \rightarrow S$: Request bill and present card to S (1)
$S \rightarrow P$: Calculate bill and run charge with card;
present bill and charge slip to P (2)
$P \rightarrow S$: Examine bill and charge slip;
if incorrect complain to have bill recalculated and charge invalidated;
if correct sign charge and summon server;
exit (3)
$S \rightarrow P$: Collect signed charge slip (4)

This smaller protocol appears to have essentially the same properties as the conventional protocol. It provides the same opportunities for errors to be detected and corrected, if at the expense of requiring a charge invalidation for any detected errors in the bill. But here the patron need wait for only one round trip and can leave at step (3). This “obvious” optimization is apparently not so obvious to the users of the protocol, given the motivation of parties in both roles to reduce latency and idle time.

2.3 Railroad Seat Checks

Many American (and perhaps other) passenger railroads employ on-board ticket collectors, who sweep through trains as they pass between stations and exchange the tickets of newly-boarded passengers for *seat checks* that indicate their paid destinations. The ticket collector collects and destroys the seat check just prior to arrival at the stations for which they are encoded. (Passengers without seat checks can therefore be presumed to have just boarded and are asked for their tickets).

The encoding of seat checks is an interesting problem, since they need to be easy to read at a glance, easy to encode, and must make it difficult to modify the encoded destination to one farther (and more expensive) than the one for which originally issued. A rather ingenious encoding has evolved, passed down entirely as folklore among railroad workers and used extensively in practice on many railroad lines. Seat checks are small printed slips of cardstock paper. An unmodified card indicates the train’s final (and most expensive) destination. A small vertical tear halfway down the card’s length indicates the second to last stop. A torn off corner indicates the third stop. And so on, with the seat check more and more mutilated for closer (cheaper) destinations. The encoding has the intended property that it is easy to read and encode quickly but difficult to modify to obtain fraudulent travel to a farther station; it implements a paper-based monotonically decreasing counter.

There is a vulnerability in the seat check protocol as described, however. It is possible to obtain unlimited travel to any nonterminal station simply by purchasing a ticket for one station beyond the attacker's intended destination and retaining the (uncollected) seat check for use on the next trip.

Although this could be fixed, e.g., by encoding the date on the seat check, the protocol has another problem familiar to computer security practice: it cannot be easily upgraded. Because it is passed on only as folklore, as opposed to being written into standard procedure, the railroads have no centralized mechanism for deploying a newer, more resilient scheme should seat check fraud ever become an actual problem. Only when the users (train crews) themselves recognize the problem does the protocol have any chance of evolving or being upgraded.

2.4 Smuggling and Aviation Security

A number of the protocols used to prevent the introduction of contraband (e.g., weapons) on commercial flights have surprisingly subtle vulnerabilities if not analyzed and implemented very carefully. For example, prior to the tighter security imposed after September 11, 2001, it was possible to smuggle a weapon onto a domestic U.S. flight by exploiting a weakness in the *extra* security given to international arrivals and without requiring the failure of any security checkpoint to provide proper screening. The vulnerability was simple: the attacker checked in in, say, London, which has good security screening of passengers and luggage. On arrival in, say, New York, the attacker clears U.S. Customs and connects to a domestic flight. The protocol allowed a weapon to be smuggled onto the final flight, because the Customs screening (an extra security layer) allowed passenger access to checked luggage. Although the passengers may have been screened properly for weapons an London, the luggage was screened only for explosives, and it was possible to remove a weapon from the luggage (in the Customs hall) prior to boarding the domestic connecting flight.

A solution is to require passenger screening after clearing Customs. This appears to have been known to the security authorities, but not uniformly implemented or understood at a local level; post-customs passenger screening was not universally required or enforced at US airports prior to September 2001.

Other complex interactions still exist in the composition of customs inspections and aviation security that may permit various kinds of smuggling. In particular, the handling of lost baggage appears to permit a smuggler to avoid risk under certain circumstances.

The techniques of cryptographic protocol analysis seem directly applicable to aviation security and customs inspection protocols, and a more systematic analysis could possibly detect other, as yet undiscovered, vulnerabilities or expose and make explicit the requirements and assumptions.

2.5 Drug Testing in Sports

Certain sports players (e.g., in professional and amateur competition) are subject to random or routine screening for performance-enhancing chemicals. The protocols for these tests are surprisingly involved, and appear to have been designed to prevent both cheating by the player (substituting test samples or spoiling the test validity) and the tester (who might attempt to either aid or "frame" a player by substituting a tainted sample). The complexity of these protocols raises serious questions as to their soundness; they seem ripe for analysis and study.

2.6 Voting in Democratic Elections

There is considerable controversy in the United States surrounding the introduction of computerized voting systems to replace mechanical and paper ballots in national and local elections. The lively debate focuses on whether various proposed products and systems are trustworthy enough, and indeed on whether computerized systems can ever meet the security, privacy and robustness requirements of democratic elections. Many computer science and cryptologic researchers have embraced this problem, contributing both technical analysis as well as political commentary to the debate.

The central question on which both the current scientific research as well as the public policy debate on electronic voting focuses is how computerized systems can be made to mimic the properties of traditional physical ballots. However, this leaves behind a number of equally interesting, and critically important, questions about the underlying application.

How secure are paper ballot systems in the first place? How do they fail? How do they scale? How well do they meet their stated security requirements[13]? Many of the recently developed cryptologic techniques and models for analyzing anonymous communication systems, distributed computation, and so forth, would seem to rather directly applicable here.

3 Human-Scale Security in Mainstream Computer Science Research

To the extent that human scale protocols appear in the computer science literature, they are usually in the form of (often strained) analogies illustrating some computational principle rather than as the focal point of evaluation. Several areas of recent computing research, however, touch upon or are relevant to human-scale protocols.

3.1 Trust Management

Trust Management[9][10][7], introduced in 1996, is concerned with the specification, checking for compliance, and enforcement of security policy in complex, decentralized systems. Although existing trust management systems, languages and tools operate at a relatively low level of technical policy (e.g., [14]), the framework and mechanism appears to be useful for managing security policies in systems with human-scale components.

3.2 Security Engineering and Software Vulnerabilities

Many aspects of software systems research are concerned with identifying, repairing, and preventing, implementation errors ("bugs") that introduce security vulnerabilities. Much of the motivation for such advances as restricted execution environments, "safe" program languages (e.g., Java), and dynamic and static program memory analysis tools, is the prevention or containment of errors that allow software to be misused in unexpected and dangerous ways. Much of modern software engineering research on program correctness is now focused particularly on the security implications of incorrect programs.

A related, and relatively recent, branch of computing sometimes called *security engineering,* focuses on the development and application of sound engineering practices for building systems with good security properties. Anderson's treatment of the subject, for example [3], views security in broad terms, drawing on many human-scale systems for inspiration and as examples. Similarly, US DoD security standards often include specific human security requirements (e.g., *two-person control* for access to certain systems).

However, human-scale systems are generally regarded by security engineers as providing accessible examples, rather than as focal points for study or as integral parts of the system. In particular, no general security engineering metrics for evaluating or quantifying the security of human-scale system components are in common use, and those that do exist appear to be drawn at least as much from folklore and intuition as from engineering experience.

3.3 Trusted Computing and Secure Booting

So-called "trusted computing" architectures (such as TCPA, etc), aim to provide a "secure" computing environment based on a small tamper-resistant cryptographic module. Certain smartcard-based applications (e.g., [12]) attempt similar functionality for specific applications (such as electronic cash, file decryption, etc.)

An essential element is of this area is the notion of bootstrapping from small trustworthy elements into complex systems. Secure booting (e.g., [5]) uses this concept for general computing, but the idea can be applied more generally.

Human-scale systems frequently exploit a very similar kind of "trust amplification" in which untrustworthy components are contained or managed by more trustworthy ones.

3.4 Hardware Security and Cryptographic Key Material Management

The security of especially sensitive data (particularly cryptographic key material) is often delegated to "tamper-resistant"hardware. This hardware is often assumed to be secure by software designers, but recent work (e.g., [2] [17] [18] [15] [16]) suggests that such assumptions are frequently proven to be unjustified.

An unfortunate side effect of assumptions that secure hardware is indeed secure (because we lack the tools to analyze it) is that when they are discovered to be wrong (whether by the analyst or the attacker), the result is generally catastrophic for the systems that use it. Often the entire security model rests on the assumption that the sensitive data managed by "secure" devices cannot possibly be compromised, and often there is no viable recovery plan should these assumptions prove false.

Similarly, human-scale protocols and systems are often assumed (without justification) to be trustworthy or abstracted out of the security analysis of complex computing systems that include them. A more integrated security analysis toolkit might accommodate more diverse system elements, or at least allow for the possibility of failure in human-level components.

3.5 Economics and Risk Management

There has recently emerged a vibrant research community focused on the relationship between economics and computer science, often either investigating security problems directly or drawing talent from the traditional computer security and cryptology research communities.

Economics is especially relevant to human-scale security protocols not only because commercial transactions motivate many of the problems, but because the parties in these protocols can often be assumed to follow economic models of behavior. For example, high latency and idle waiting times are usually not well tolerated by people, and are likely to cause perceived un-needed rounds of interaction to be optimized out (intentionally or not) from some protocols, even at the expense of reducing security and increasing risk.

3.6 Human Factors

Almost all security systems have a user interface of some sort, but *Human factors* research on computer security has been remarkably limited. Although such research is beyond the scope of this paper, it acutely highlights the need for it.

4 Future Work: The Human-Scale Security Protocols Project

We believe human-scale systems are interesting and important. In this section, we speculate on some future research directions for a "Human-Scale Security Protocols (HSSP)" effort. The objective of this project is to establish the study of human-scale activity, applications, and systems as an integral part of computer security research and practice. In particular, we expect to use HSSP as a starting point to motivate three (complementary) courses of research:

– Analyzing the security properties (especially with regard to how they succeed and fail) of a range of human-scale protocols and applying the lessons learned toward improving computer security methodologies. (What can computer science learn from these systems?)

– Applying the tools and techniques of computer security and cryptology in novel ways to analyze, attack and improve the security of human-scale systems. (What can computer science do for human-scale security systems?)
– Developing stronger ties between the security properties of underlying system components (including their physical environment, hardware, software and networks) and the requirements of the applications that are built upon them ("bootstrapping" trust). (What is the relationship between human applications and the security of the systems that implement them? Can their requirements and properties be quantified or at least made more explicit?)

The dual nature of the project's objective – establishing a sound intellectual foundation for the study of human-scale security in computer science as well as making specific contributions to the security of various systems – means that some research here must be exploratory in nature while others must be focused on specific seminal subproblems.

4.1 Taxonomies of Threats and Security Properties of Human-Scale Applications

It seems especially useful to establish comprehensive "taxonomies" of threats (and risks) in human-scale applications, as well as of the security properties that are required of or that characterize the systems that implement them. Such taxonomies would be useful for analysis of both human- and computer- based systems (and especially systems based on a combination of the two) in a coherent, more uniform manner.

A longer term (and more ambitious) goal is to develop prudent security engineering practices that can integrate both kinds of systems, without requiring radically different treatments for human-scale and computational components. Existing work, e.g., [1] or [4] on security protocol engineering does not seem to be completely applicable to the design of hybrid systems that include human scale components.

4.2 Trust Management at the Human Scale

Our previous work defined the trust management and compliance checking problems as a model for security policy and enforcement and established an abstract "trust management layer" as a system security element in distributed systems.

We are expanding our study of trust management to make it a more useful abstraction for human scale applications.

A basic principle of our trust management tools (such as PolicyMaker [9] and KeyNote [8] [11]) has been the *decentralized* nature of the policies for which they check and enforce compliance. That is, a policy can be made up of signed assertions from many different entities, who need not know the details (or even of the existence) of the top level policies of the ultimate endpoints that use them.

This property makes it relatively simple to adapt these systems to new applications, provided that the security policy enforcement points are capable of executing the trust management engine. This is not usually a problem for even very modest computing platforms (e.g., KeyNote runs even on early-generation Palm Pilots), but obviously makes it difficult to integrate into human-scale protocols that depend on people to carry out parts of the task. In particular, the basic service of the trust management system is *compliance checking*; it allows or disallows access, but does not have any means for providing output to or receive input from a human user during the actual evaluation process.

This seems an unfortunate limitation, since many of the most important security critical systems (with complex security policies) include a human-scale component. For example, imagine a comprehensive physical security access policy, which includes both electronic access controls as well as human guards who check identification and interrogate individuals who request access, with different identity requirements for different kinds of access under different circumstances. It would be useful express as part of the security policy the identity documents and questions for the guards to ask under different circumstances, and for the trust management engine to be able to display these questions (and receive the answers) during the compliance checking process. Current tools do not support such a model of interaction; all input to the query is passed to the compliance checker as a "batch," returning a single answer based only on the query.

Indeed, we are aware of no security policy management tools that can capture interactive human-scale operations along with those enforced directly by people.

To make trust management tools useful for such applications, we intend to extend the KeyNote toolkit to allow interactive negation as part of the query evaluation process. In addition, we intend to develop KeyNote trust management policies that capture a variety different human-scale security scenarios. In particular, many human scale protocols have implications not just for security, but for privacy as well. The security and privacy requirements may be in conflict (as discussed, e.g., in the previous section).

4.3 Analysis of Specific Human-Scale Systems

We are examining several specific human-scale systems, with the aim not only of investigating their security properties, but also to motivate and identify other problems for subsequent, larger-scale study.

Small-Scale Transaction Protocols. Small-scale transactions (such as, e.g., the restaurant rituals discussed in the previous section) are a rich source of human-scale security requirements, protocols, and problems. In addition to the protocols discussed in the previous section, we will identify other appropriate transaction protocols, formalize their apparent requirements, identify the security properties of their components, and analyze the existing protocols as well as alternative versions. Of particular interest for these protocols is a better understanding of how they fail and of their relationship to risk management strategies.

Physical Access Controls. Physical access controls (e.g., locks, safes and vaults [21], [20]) are a rich source of security requirements and designs. As previously discussed here and in [6], mechanical locks are in fact readily modeled as computers, and indeed, are being replaced or managed by embedded micro-controllers in many applications. Traditional analysis of these systems has focused on their physical properties and implementation (e.g., vulnerability to forced entry or lock picking), rather than on abstract properties of the design or the application.

Inexpensive and low-power embedded micro-controllers have been used in security and access control systems for many years. Electronic locks, of course, do not have mechanical tumblers and are therefore not usually directly vulnerable to mechanical manipulation. Of course, this does not at all imply that electronic locking systems are inherently more secure than their mechanical counterparts. The underlying lock mechanism is still mechanical and may be subject to mechanical bypass. The electronic control mechanism may be vulnerable to attack, e.g., through the introduction of RF or power faults or via emission monitoring. And, of course, electronic locks have at their root software of increasing size and complexity as they become more sophisticated (and as they are networked into centralized control systems). Needless to say, the software used in electronic lock systems is no less subject to bugs, vulnerabilities and protocol failures than the typical (buggy) software of similar complexity used in other applications. Indeed, security system software may well be considerably worse, since it is often purpose-written and may be subject to only limited scrutiny and testing.

We expect to continue our study of master-keyed mechanical locks. An open problem in [6] was how to better structure the keyspace of master keyed locks to limit the information available to the attacker who can either disassemble a lock or use a lock as an oracle. Preliminary results suggest that although there are fundamental tradeoffs that limit the size of systems that can be implemented without leaking information to the attacker, it is possible to build medium-scale master keyed lock systems with "false" cuts that resist our attack and are otherwise secure.

We can also expand this study to include software access control systems, with the aim of identifying software, hardware, and interface interactions that can lead to vulnerabilities similar to those occurring in general purpose computers and networks. Conversely, we will attempt to identify design strengths in these systems, especially defenses against known attacks, that can be generalized to apply to general computing systems. (The limited interfaces of access control systems is sometimes cited as a strength compared with general purpose computers, but results such as those of Anderson, Kocher, and Kuhn perhaps suggest otherwise).

Sensors and Alarm Systems. As noted in the previous section, alarm systems (and sensor system networks generally) provide an interesting framework and motivation for a number of computer and network security problems, most notably denial of service. The convergence toward the use of general purpose platforms (computer and network) for these applications makes the relationship between computer security and physical security all that much more relevant.

Furthermore, as alarm and area security systems collect and record more and more data, the interactions and tensions between security policies (which encourage collecting and storing more data) and privacy policies (which encourage ignoring or destroying the same data) become more acute. We hope to develop an integrated policy model that can capture security as well as privacy requirements at the same time (e.g., trust management for privacy and security simultaneously).

5 Conclusions

We have have focused too much of our attention on computers and electronic communication systems. Small-scale, mechanical, physical, and paper security systems deserve our attention and scrutiny.

In then end, it seems likely that most of these protocols and systems have more to teach us than the other way around. Any protocol implemented by human beings, especially those evolved over time, is likely to be heavily optimized and carefully tuned against risk, whether by conscious analysis or *ad hoc* trial and error. How can we ignore them?

References

1. Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.
2. R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.
3. Ross Anderson. *Security Engineering.* Wiley, 2001.
4. T. Anderson, S. Shenker, I. Stoica, and D. Wetherall. Design guidelines for robust internet protocols, 2002.
5. W. A. Arbaugh. *Chaining Layered Integrity Checks.* PhD thesis, University of Pennsylvania, 1999.
6. M. Blaze. Cryptology and physical security: Rights amplification in master-keyed mechanical locks. *IEEE Security and Privacy*, 1(2), March/April 2003.
7. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer-Verlag Inc., New York, NY, USA, 1999.
8. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System Version 2. Internet RFC 2704, September 1999.
9. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. of the 17th Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, Los Alamitos, 1996.
10. M. Blaze, J. Feigenbaum, and M. Strauss. Compliance Checking in the PolicyMaker Trust-Management System. In *Proc. of Financial Cryptography '98, Lecture Notes in Computer Science, vol. 1465*, pages 254–274. Springer, Berlin, 1998.
11. M. Blaze, J. Ioannidis, and A. D. Keromytis. Experience with the KeyNote Trust Management System: Applications and Future Directions. In *Proceedings of the 1st International Conference on Trust Management*, pages 284–300, May 2003.

12. Matt Blaze, Joan Feigenbaum, and Moni Naor. A formal treatment of remotely keyed encryption. In *Proceedings of EUROCRYPT 98*, pages 251–265, 1998.
13. Federal Election Commission. Voting system standards, 2002.
14. A. D. Keromytis. *STRONGMAN: A Scalable Solution To Trust Management In Networks.* PhD thesis, University of Pennsylvania, 2001.
15. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Proc. CRYPTO 99*, 1666:388–397, 1999.
16. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. *Proc. CRYPTO 96*, 1109:104–113, 1996.
17. M. Kuhn. Cipher instruction search attack on the bus-encryption microcontroller ds50002fp. 47(10), October 1998.
18. Markus G. Kuhn. Optical time-domain eavesdropping risks of crt displays. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 3–18, 2002.
19. R. M. Needham. Denial of service: An example. *Communications of the ACM*, 37(11):42–46, November 1994.
20. M. Oehlert. *Safe Technican's Reference Manual.* AOLA, 1997.
21. M. W. Tobias. *Locks, Safes and Security.* C. Thomas, Ltd., 2000.

Toward a Broader View of Security Protocols (Transcript of Discussion)

Matt Blaze

Penn State University

In keeping with the tradition of this workshop this is actually a position paper in the sense that it represents work that's not done and ideas for work that could be done, rather than simply a paper that got rejected from a bunch of other conferences. [Laughter]

I had a realisation during a talk from Dave Clark. He pointed out that Plato and Socrates actually managed to communicate their ideas quite well without Powerpoint. I am apparently not that good, but I made the decision that I will endeavour not to use Powerpoint, or at least will feel guilty about it when I do.

The thesis of my position paper is that we, as computer security people, are wasting far too much of our valuable time thinking about computer security, when in fact we should step back and think a little bit more broadly. We've missed important opportunities, both to learn things that will improve computer security, as well as to contribute to the security of other kinds of things, because we've decided that there are entire subject areas that are really not part of our focus and not appropriate for us to be thinking about.

First of all, computer scientists and cryptologists are not the only ones who have been thinking about security protocols throughout history. There are in fact quite a few very old security protocols (many related to commerce, many related to the security of private information), that have existed for hundreds, and sometimes even thousands, of years. They have the very nice property that they tend to be optimised very heavily for security against actual practical threats, and for performance on the computing platforms (that is people, and mechanical devices) that they are run on. People have very little tolerance for their security protocols not working so they optimise for security, and as they also have little tolerance for wasting time they optimise for performance. I believe we can, and should, study these protocols as we can probably learn quite a bit from them.

Secondly, and perhaps conversely, computer science is not the only field with interesting security problems that are susceptible to the kinds of analysis that we are comfortable performing. So I think that in addition to learning something, there's a very reasonable possibility that we can contribute something to other fields. The way we model protocols, and the way we abstractly think about protocols, in particular just the very simple notion of separating a design from an implementation for the purposes of studying it, is rather unique to the practice of computer security, but could readily be applied to many other disciplines. I think there is a very real potential contribution that we've largely forgone making, and I'd like to persuade you to think more broadly. Really this is the important thing to take away from this talk.

Alarm Systems. The motivation for this came from an example from the physical world, something on the human-scale, that led to a very direct insight about computing systems. It was a paper by Roger Needham in 1994, and I believe also a keynote address at the first CCS conference, on denial of service[1]. I read this with great interest, and was surprised, and disappointed, that this paper about denial of service wasn't about denial of service in computing. Why is Roger talking about these crazy alarm systems which obviously having nothing to do with computer science, or the analysis of security protocols? After reading the paper and talking to Roger about it, the insight just struck me like a ton of bricks. It was an enormously important moment in the evolution of my thinking about protocols. His central observation was that there are two basic ways to prevent an alarm system from working, that is, to violate the security of an alarm system.

The first kind, I'll call it a type A attack, is to interfere with the alarm signal, to prevent the alarm signal from reaching it's destination. You might cut the wire, you might send a false signal to say there is no alarm going on right now, or what have you. This is a kind of head-on direct attack against the system. It is a fairly obvious line of attack, and it's the one people think of first. Alarm systems really take quite considerable pains to be hardened against a type A attack that aims to interfere with the transmission of the signal. We try to make sensors sufficiently sensitive, so that they will be set off before you can cut the wire to them. We might have both positive and negative signals, a keep alive signal that says, the wire has not been cut, and if it goes away, that's equivalent to sending an alarm signal, and so on. We might use redundant communications paths to make sure that the alarm monitor will receive the signal. That's the attack that we think of first, that's how computer scientists think of denial of service.

Then there's the attack that anybody who wants to violate an alarm system would actually do, which we'll call the type B attack. This embraces the fact that the sensors are going to send signals, and exploits that fact to subvert the overall system: overwhelm the capacity to respond to alarms. Walk up to the door, rattle it, set off that sensitive sensor, hide in the bushes, wait for the police to show up, the police decide nothing is wrong and leave. Walk up to the door, rattle it, wait for the police to show up, hide in the bushes, and when the police leave, rattle the door. Iterate this a few times, eventually the police stop responding altogether, now break the door open. This is a much less obvious attack to a computer scientist; the first time I had considered such an attack was when Roger pointed it out. And it has this annoying property, annoying if you care about alarm systems as opposed to just studying alarm systems, that there's a fundamental trade-off. Finding a fundamental trade-off gives you a PhD in computer science, but it just gives you a headache if you're trying to build something for real. The trade-off is that increasing defences against the type A attack increases your susceptibility to the type B attack. The more sensitive you make the sensors the easier it is for the adversary to use them against you.

[1] "Denial of Service: an example" CACM 37(11) 42-46.

The interesting thing is that all alarm installers, and police alarm responders, and burglars, have long understood that there are these two kinds of attack and these fundamental trade-offs. Computer scientists, on the other hand, tended not to have this understanding of denial of service. Evidence of that has been in the late 90s, since the publication of Roger's papers, virtually all denial of service attacks against the Internet infrastructure have had to do with overwhelming the capacity of our systems, rather than with preventing the transmission of bits, cutting wires, and so on.

Chris Mitchell: Does that mean that Roger was responsible for these attacks?

Reply: I'm happy to blame Roger, I think he would be delighted to take the blame for this.

Mechanical Locks. For a second example, I want to go in the opposite direction, with insight in computing possibly being useful in the physical world. I started studying last year, or maybe it was the year before, the question whether we could apply cryptologic techniques to things other than computing and communication systems that use computers. The first question I looked at was what's the complexity of attacking a mechanical lock? The first basic thing we would do as computer scientists is to abstract out the mechanical stuff - you know, how precise are the things inside the lock, what are the manufacturing tolerances - from the fundamental design. So I made the decision, let me assume a perfect lock, and let's analyse how good a perfect lock would be.

The best attack against an abstract lock, other than brute force, which in the lock world means something other than exhaustive search, is exhaustive search. [Laughter] That's nice because the complexity here is exponential in essentially the size of the key: the number of different heights the key might be cut at raised to the number of positions. That's good news, that's what we look for in security, exponentials are very nice. But, under this analysis, master key locks, where there's not just a key to the individual door, but a key that will open all the doors in the entire system, completely fall apart. It turns out it's relatively easy to take the information in a regular key and use that to learn the master key for the entire system. There is a rights amplification, or a privilege escalation, attack, that is very powerful, which ends up being essentially linear in the product of the number of pins and heights. With the security factor that the actual locks use, this attack is entirely practical: it costs about $2 to compromise a typical master key system that costs tens of thousands of dollars to install. Very bad news for the users of master key locks because this is not a problem that's solved by getting better locks, it's a fundamental weakness in the design.

The second interesting thing I discovered after I did this was the realisation that more computer scientists, prior to the publication of my paper, appear to know about this attack than locksmiths. Locksmiths believe this attack was very subtle and strange, and that only a weird computer person, or a really, really good locksmith might think of something like this. But this attack from a computer science perspective just leaps right out at you as soon as you start thinking about it as a computer scientist. If you think of a lock as an on-line authentication oracle, and you think, well let me ask it some questions, like we would if this was a proposed crypto system, then you'd just get this attack right away. In fact, virtually everybody who has gone to a decent engineering school in the United States has either figured out an attack like this, or something equivalent.

So that's an example that suggests that perhaps computer scientists might actually be good for something. Perhaps the way we think about things might have some useful generality that goes beyond designing our own protocols: we don't seem to be very good at designing protocols in computing systems, but maybe at least we can become locksmiths, and alarm designers, and be pretty good at it.

What I'm advocating is the idea of what I'm calling human-scale security, and I think that this is a legitimate branch of computer science. Alarms and master key locks are my two motivating examples, and what I'd like to do is explore the question of whether or not these instances are the tip of an iceberg. I think it is reasonable to suggest that there may be many other protocols that have fundamental insights back and forth between computer science and other aspects of the human-scale world.

Open Security. Now there are some barriers to conducting a course of study on this. First of all, the academic approach to security seems to make people nervous. People tend not to trust systems that are provably secure. People don't really trust abstract designs in an intuitive way, in the way they trust large bank vaults with lots of guards, and airport security that makes you take your shoes off before they let you get on the plane. Those intuitive things make people feel more comfortable than any kind of formal analysis that says, security is good in this way. We tend to trust protocols that evolved naturally more than protocols that were designed, maybe there's a good reason for this. Discussing security openly the way that we do in our community, this idea that we publish things and let everybody learn from them in the hopes of improving our security, because more good people will use it to improve things than bad people will use it to violate security, and it's something that computer scientists generally seem to understand now pretty well, but is not something that has an intuitively obvious appeal. Certainly our community, by virtue of having conferences like this one, buys into this view. It's interesting that it was once also that way in the mechanical security world. I'd like to just give you three quotes over a period of 150 years that will indicate a kind of recession into a dark age of thinking about security in the human-scale world.

First, Alfred Hobbs is my hero, he deserves to be yours too. Alfred Hobbs was an American who among other things was an engineer. He was an amateur engineer back when you could be one of those, and he studied mechanical locks. The Bramah lock company of London had had a long standing offer from 1804 that anyone who can defeat their lock, which was in their store window, would be given two hundred guineas, which was a fair amount of money back then. No one had broken the Brahmer lock which had stood for 50 years without being defeated or picked.

Alfred Hobbs came over on the fast boat and built his picking tool for the Brahmer lock. He went to the great exhibition, and went to the Brahmer booth in London, and then in 48 hours had the Brahmer lock open. He then went to the Chubb booth, and the Yale booth, and did the same thing to their locks, but they didn't have a prize. Alfred Hobbs in the 1850's was the darling of the lock community. Everybody said, you have got to get your lock past Alfred Hobbs if you want it to be taken seriously. It turns out no-one could, and we still use those same basic designs up to this day. But one of the things Hobbs did was write his memoirs, he wrote out how he did these things; how he defeated the Brahmer lock, the Chubb lock, and the Yale lock. He anticipated that this might vex some people, and so he included in his book a preface, in which he said:

> "A commercial, and in some respects, social doubt, has been started within the last year or two [that would be 1851 or 1852[2]] whether or not it was right to discuss so openly the security or insecurity of locks.
>
> Many well meaning persons suppose that the discussion respecting the means for baffling the supposed safety of locks offers a premium for dishonesty by showing others how to be dishonest. This is a fallacy, rogues are very keen in their profession and already know much more than we can teach them respecting their several kinds of rogueishness. Rogues knew a good deal about lock picking long before locksmiths discussed it among themselves as they have lately done. If a lock, whether it had been made in whatever country, or by whatever maker, is not so involved with that it has hitherto been deemed to be, surely it is in the interest of honest persons to know this fact because the dishonest are tolerably certain to apply the knowledge practically, and the spread of the knowledge is necessary to give fair play to those who might otherwise suffer by ignorance. And I cannot too earnestly urge that an acquaintance with the real facts will, in the end, be better for all parties."

Now other than the fact that this is well written, this could easily have been written a few years ago, about Internet vulnerabilities. This is very modern, or at least we think of this as the modern view of the question. Should we talk about these things? In the aggregate, more knowledge is better, and it turns out that we weren't the first people to think of this. So Alfred Hobbs ended up representing the mainstream of locksmith thought for somewhere between zero and one hundred years.

The next kind of bit of evidence that I can get on how thinking has progressed in the locksmith community is from 1953. There is a book called the Art of Manipulation, which remains in print until this day, is still the main book if you want to learn how to manipulate safe locks. If you want to be a safe cracker you've got to read this book. It was by Lentz and Kenton, and they put in their 1953 preface:

> "It is extremely important that the information contained in this book be faithfully guarded so as not to fall into the hands of undesirables. [That

[2] See http://www.crypto.com/hobbs.html

> would be you guys.] And we also suggest, after you become proficient in the art of manipulation, to destroy this book completely so as to protect yourself and our craft [and our future sales presumably]."

So somewhere things just went terribly wrong in the lock world.

Now if we go fifty years forward from that, after my paper on Master Key Locks was published, the locksmiths just went absolutely berserk. They couldn't imagine why somebody would do something as horrible as what I did. There's a magazine called The National Locksmith, which is the main trade publication in the United States for locksmiths, and I think they didn't realise that I was a subscriber because they were publishing monthly guest editorials from March through July last year, denouncing me. It was as if they invited people, "if you've got something bad to say about what Matt Blaze did, send it in and we'll feature it." I was the cover story of this magazine for months. And if I'd lost the keys to my house and I needed to call a locksmith I would have been in trouble. I probably still would be, I'd have to wear a disguise because they have published my picture.

Frank Stajano: Anonymous credentials.

Reply: Yes, even in some kind of anonymity. Billy Edwards, who's a fairly prominent locksmith, I guess he's probably the Alfred Hobbs of today, it's very sad, said:

> "Blaze's master keys paper shouldn't have been published, because the only people we'll educate are the dishonest who will use it to compromise security, locksmiths don't have to be surreptitious, no we can't call him a moron because he is obviously intelligent, after all he did grasp the concepts of master keying. [Laughter] We can however see that he is an inexperienced amateur when it comes to physical security, in his computerised world it is a simple thing to fix a security problem, you just load some software in." [Laughter]

So, this is essentially what we're up against, because this is frankly mainstream thinking that makes great intuitive sense to most people. My parents sided with Billy Edwards over me on this; they don't know why I would violate the security of their locks by openly discussing it, and they're kind of required to love me. So I think there is a significant culture clash, but in spite of that let's press ahead.

In the Restaurant. We're going to look at a couple of protocols. How many people have ordered a bottle of wine in a restaurant? There's a very elaborate security protocol associated with doing this. Let's think about the security requirements. There's a requirement for secrecy, for confidentiality. The patron's guests never learn the price of the bottle of wine, there's only one copy of the wine list with the prices, but there's a discussion about selecting a bottle of wine in the right price range that's held completely out in the open. The patron can't repudiate the order after the bottle of wine has been opened and consumed. At the end of the protocol it's very hard to claim that you didn't know which bottle of wine you had ordered. Similarly the waiter and the restaurant can't readily switch a cheaper bottle for one that you were actually paying for.

The protocol starts when you say, I'm thinking of this bottle of wine, and the waiter says, well no, you don't want that bottle of wine with this food, you want this one. The information the waiter got from that was the price of the first bottle and they're supposed to make a suggestion for an appropriate wine in the right price range. Then if you approve the order, they show you the bottle, you again agree that that's the bottle you ordered, they open it within your sight, they decant it within your sight, you sniff the cork, and you sample the wine, and then everybody agrees that this is the wine that's been ordered, it gets added to your bill, and the wine is left in your presence throughout the meal.

This seems to protect all parties involved in very nice ways, and you could write this protocol down in pseudo protocol analysis language. Now this protocol appears to have quite a bit to admire about it. There are three rounds and it's very difficult to violate its security. Apparently fraudulent wine ordering was a big problem in restaurants. It's not a problem anymore, this protocol's apparently successful, it looks as if it's been optimised pretty well. I'm sure this protocol's very old, the oldest reference I was able to find for it was the manual of dining car operations for the Pennsylvania railroad in 1927, so this protocol has been around for at least 75 years, and I'm certain quite a bit longer. Here we have an example where it's been tuned to the threat very nicely, and appears to be rather successful at that.

Now, when it comes time to pay the bill, everything just breaks down completely. There's a much more recent protocol that has been motivated by the existence of credit cards, so we know this protocol can't be more than 40 years or so old, since the credit card became a common-place thing for paying in restaurants. This protocol appears to be very poorly optimised both against security requirements and for performance purposes, as it requires two and a half round trips; asking for the bill, examining the bill, providing your card, which may involve a wait, examining the card slip, approving it, and then finally signing it. If you're not lucky the original itemised bill isn't provided at the same time as the credit card slip is provided, so you don't actually have anything to compare against when deciding whether to sign the bill.

In fact you can easily optimise this with no loss of security for anyone, and in fact, possibly quite a bit of a gain in security, to only one and a half round trips, you only have to wait for one round trip, by simply taking out your credit card, handing it to the waiter and asking for the check, that appears to protect you in all of the exactly same ways, and protects the restaurant in all of the exactly same ways, as the original protocol, and has precisely the same weaknesses.

Fabio Massacci: You know I think I can explain why it's this way.

Reply: Yes, people are dumb, it takes a while to optimise the protocols.

Fabio Massacci: No, no, if you revocate the bill, so to speak, you have to challenge some item. But once he has swiped the card, the merchant will be charged four percent, and then if you revoke it he will not get the charge back (depending on his contract). So the revocation of the charge card is costly, but the revocation of the bill is free of charge.

Reply: Ah, so then the question is, at which stage of the protocol are there revocations? I suspect that analysis has never actually been performed. My point here is not how to optimise things in a restaurant, but just to note that the problem seems to be susceptible to very similar analysis.

Mark Lomas: There's a better protocol used in the fish counter in Harrods. If you sit there and order your fish, they will present you with a bill without you asking for it, and say, if you agree with it, sign it, and hand over your credit card. You never see a credit card slip, because it goes round and back in one.

Reply: One round only, so that's optimisation, they want to get you out of that seat as quick as possible.

Richard Clayton: It's long been known as a problem in cryptography in discussing these things that the people sitting at the front, the professors and so forth, go to smarter restaurants than everybody else. The dining cyptographers shows that very clearly. The difficulty here is that in fact you can summon the bill without the waiter actually coming across, they can come with the bill the first time, but they don't necessarily know which way you're going to pay for it.

Bruce Christianson: You wave your credit card.

Ross Anderson: This is an example of technological lock-in. Had you been using credit cards with zip-zap machines for a hundred years, this optimisation would no doubt have been found and then you wouldn't have this charge back problem. Now the charge back problem you can fix by changing merchant agreements that are locked in by 21,000 contracts. What people did try to do for a while was to say, when we move to smartcards then you will have this radio connected terminal that you take to the guy's table and use, plunk it down and do it there on the spot. But again we seem to have been culturally locked away from that, because that's a piece of back office equipment, it's not elegant, and the waiter won't put it there in the midst of the silver and the glassware.

Reply: I'm willing to make a prediction that within a hundred years this protocol will be universally optimised.

Other Protocols. Let me mention a couple of other protocols. I have a couple of students looking at the drug testing protocol being used in professional sports, at Olympic level. I don't know whether we can find a vulnerability in this protocol or not, from a cryptographic point of view, but it appears to be susceptible to many of the same kinds of analysis, and it appears to have been designed by someone who's familiar with cryptographic protocols, aa it is described in ways that are very similar to the ways we describe cryptographic protocols. That is something I found rather interesting and surprising.

Another example, going back to the railroads, is proof of payment. Most railroads in the United States operate on collecting tickets from every passenger on the train. What most railroad ticket collectors do is trade the ticket for a temporary single ride only ticket called a seat check. You can't modify it to go further than you originally paid. But there is a security vulnerability in this protocol. Does anybody know, can anyone identify what that security vulnerability is?

Virgil Gligor: What about replaying some sort of destination?

Reply: Right, exactly so. Virgil has obviously defrauded the railroads before. The simplest attack is to purchase a ticket for a destination one stop further than the one where you actually intend to get off. When you get off the train you take your seat check, which has not been collected, so you can use it on your trip the next day.

Andrei Serjantov: Just before each destination the guard collects the ones that are due for the current destination, right.

Ross Anderson: In British railroads they don't use seat checks at all, the conductor simply remembers who's joined.

Reply: Yes, you need to have smarter train conductors here and in the USSR.

Let me just close with this example, international airline security. Prior to September 11th, it was relatively straightforward to circumvent the controls against contraband on airplanes, simply by picking an international flight with a connection. They now require one to go through passenger screening after customs, but that wasn't universally enforced prior to September 11th.

Virgil Gligor: A similar example was the Lockerbie PanAm Flight 103 where in fact the luggage was checked in at Malta, and the person who accompanied the luggage left in Frankfurt, yet their luggage went on to Lockerbie.

Reply: So again this kind of protocol seems to be quite susceptible to the kind of analysis that we do. If you analyse when people have access to what, this kind of an attack falls directly out of the protocol; so you see it's customs who have access to the checked baggage. Adding a security check, the customs check, which required passenger access to luggage, decreased the overall security of another part of the system. You wouldn't necessarily see that if you just analysed this in intuitive terms, you might think, adding more security, well that's obviously good.

So my point here is that this is a ripe area, it's one in which there are quite probably a few problems for us to think about, and I would encourage you all to stop wasting your time with computers and start thinking about all the other kinds of trouble you could make.

Andrei Serjantov: I couldn't help noticing that in your first slide, there was a type C attack: for an attendant to be the police, as in Oceans 11. So, alarm raised, the police guard comes in and catches the criminal. Should we be thinking about that kind of thing, man in the middle, as well?

Reply: I think one of the most powerful tools at our disposal, that other fields don't have, something in our arsenal that tends not to exist elsewhere, is that we naturally abstract out design from implementation. That obscures certain details that are very important, but it also exposes other details in ways that can make them quite clear. I think you might see man in the middle attacks in ways that you wouldn't if you looked at the system as a collection of human skeleton bones.

Audience: Other fields must have procedures for analysing security of their own, not in general but in particular. I know some security people who, if they're dealing with a floor plan will colour it (this is a red area, this is a green area, etc.) so you can see that there's a wire, or a pipe going from green to red they know exactly whether that's OK. If they colour the airline situation with a multitude of colours so checked baggage was one, and carry-on was another, they would have discovered this problem. I was just wondering if other fields have such methods.

Reply: To the extent they do I think they're either ad hoc or relatively private within a small community. Certainly, I have been reading literature in the physical security world, and I've seen very little systematic analyses of those things.

Ross Anderson: There's sophisticated technology in the oil refinery design security model. If you design an oil refinery, you have thousands of pipes of different types. There are some types that you mustn't mix, you mustn't put sewage water into drinking water obviously, that's also a domestic plumbing requirement, and so you use different types of pipe, different brands, and different materials. In an oil refinery it's very much worse because most of the pipe that you've got is eight inch thin steel and some may be carrying hot methane, others may be carrying cold crude, others may be carrying ammonia. There is a great big long table of what you mix with what, and what absolutely you must never put near something else. And people write very complex programmes which are embedded in the CAD/CAM systems that they use in this environment in order to see that complex set of rules is obeyed. I sometimes fear that if we end up with a pervasive trusted computing environment, then something like this is in our future too.

Bruce Christianson: I'd go one step further than that. Yes, abstraction is a very powerful tool that computer scientists practice using a lot, and which other people use but really not to the same extent. But I think the other thing that distinguishes the computer scientist from many other professions is that we don't accept that there is a definitive model of a system. We want to have lots of different models, for lots of different purposes. We don't want unify them all, or we'd end up with a model that's at least as complicated as the system we're trying to understand[3]. It's because we know the system is too complicated to understand, that is why we try and model it in the first place. In a lot of physical sciences, and a lot of engineering disciplines, there is a received model, and that's the model that you do your security check on. Anything that breaks the abstraction boundary on that definitive model, you won't catch. And I think that's our great ability as computer scientists, to be able to change between models without losing money in the process.

Reply: Right, yes, although I want to caution us against being too confident that the direction of the contribution will be entirely us saving the world, because look at how successful we've been at securing computing systems. I think that we have much more to learn from studying these systems than the other way around. But we are looking at a two-way flow of value here.

Richard Clayton: I think you've been quite restrained in not talking about elections. Elections have a very complex set of protocols which are essentially a series of fixes layered on top of each other for dealing with frauds from the past in order to prevent those frauds being perpetrated in the future. Whereas a lot of people are going out and designing electoral systems in the electronic world without understanding what has been done in the past.

Reply: My computer just froze and I can't show the slide that I have on it. [Laughter]

There's been enormous amounts of work on this question, can we make computerised elections as secure as paper ballot systems. That is a focal point of the huge debate in the United States and in many other countries right now. But there's been almost no analysis of how secure, in some formal computer science sense, these paper election systems are. I think we don't have a good sense of that at all. There seem to be seem some intuitively nice properties to these systems; one that we don't have an analogue to in computer science is that you can commit fraud in a paper election, but the amount of damage you can do seems to be roughly proportional to the amount of effort and risk that you put into it. So stuffing a ballot box with one ballot is less risky than stuffing it with a thousand ballots, and so on. That's a property that doesn't seem to carry over very well into computerised elections, and I think doing analyses of both of

[3] After all, what we computer scientists think of as "the system" is usually itself a model or representation of something else. So if unifying all our partial models of the system gave us a unified model which were simpler than our eventual design, then we should have done better had we just animated this unified model instead of implementing what we did.

So either we're incompetent designers anyway, or we failed to model some vital aspect of the system behaviour, or else the unified model is useless as a vehicle for understanding the system design, because it's more complicated than the thing it's being used to explain.

In contrast, show an empirical scientist a number of models with overlapping domains, and their immediate instinct is to try to unify the models.

these types of system in a more sophisticated security model would be a useful contribution.

Ross Anderson: Perhaps another set of protocols to look at are those that surround the retail trade. If you buy something and is unsatisfactory you can end up having a conversation with the retailer, if it's cheap. If it's expensive you end up getting some kind of paperwork which enables you to have a conversation in writing with his boss. Similarly, if I go to the market and want to buy a banana I can if I'm impulsive, foolish and wicked, simply grab a banana and run off, and perhaps the stallholder will run after me, and perhaps not.

Bruce Christianson: If he does, your accomplice will then steal the whole stall full of bananas.

Ross Anderson: That doesn't matter because even a whole stall load of bananas only costs a few pounds, but I cannot do that by going to a goldsmiths. Pick up a five thousand pound ring and run off with it and they will pull a switch operated doorlock, and you run and bash your nose on the glass. So your whole set of protocols evolved around things like retail, auctions, and all things that go on in the High Street, which are being replicated at best badly in e-commerce.

Reply: By the way I can also just mention Ross' book here. Ross' security engineering book[4] is currently one of the very few references in computer science that talks extensively about the relationship between human human-scale systems and electronic systems, and there deserves to be much more thought on this subject.

[4] R.J. Anderson, Security Engineering, Wiley, 2001.

Privacy, Control and Internet Mobility

Tuomas Aura and Alf Zugenmaier[⋆]

Microsoft Research, Cambridge, United Kingdom
tuomaura@microsoft.com, zugenmaier@docomolabs-euro.com

Abstract. This position paper explores privacy issues created by mobile and wireless Internet access. We consider the information about the users identity, location, and the serviced accessed that is necessarily or unnecessarily revealed observers, including the access network, intermediaries within the Internet, and the peer endpoints. In particular, we are interested in data that can be collected from packet headers and signaling messages and exploited to control the users access to communications resources and online services. We also suggest some solutions to reduce the amount of information that is leaked.

1 Introduction

As laptops and hand-held devices replace stationary desktop computers as the mainstream personal computing device, the average Internet user and host become increasingly mobile. The mobile devices stay connected to the Internet via business and domestic broadband networks, wireless hotspots, and cellular data links, bringing closer the dream of universal connectivity.

The connectivity is implemented with the help of a number of autoconfiguration and mobility protocols. The user, however, is not exposed to all this complexity. The devices do their best to hide the protocol details from the user and to provide seamless universal access to online services. We can expect them to get better at this as the mobile access technologies mature. Ideally, the user remains connected to the same services regardless of the location and access network.

While transparent connectivity is a powerful tool, it is worth considering the potential adverse consequences that it has for the users. One such consequence is that the users expectations of privacy may not match what goes on at the network protocol level. As the users and their mobile devices move seamlessly from one access scenario to another, packets traveling over the access links and across the Internet may reveal information that the user would prefer to keep confidential. The same information may also be used to control the way the user communicates and to differentiate the services and prices offered to individual users.

The information that we consider in this paper is limited to the users identity, location and the services they access. Protection of critical application data is outside the scope of our discussion. That is, we are interested only in information that is carried in the packet headers or revealed by standard signaling messages.

[⋆] Alf has moved to DoCoMo Euro-Labs, Munich, Germany.

IPv6 is particularly important for our discussion because many new mobility and security solutions have been designed for the next-generation Internet protocol. Nevertheless, we also consider IPv4 where appropriate because IPv4 will be around for quite a while and mobile hosts will have to alternate between the two technologies to gain maximal access to the Internet. It is also interesting to compare the new and old protocols from the privacy point of view. It should be noted that we limit the discussion to the most common standard Internet protocols and medium access protocols. It is quite possible that proprietary protocols and extensions either reveal additional information or help to hide it.

The privacy of mobile users has previously received quite a lot of attention. See, e.g, [4] for identity protection and [5,6,10] for location privacy. We refer the reader to [3] and [17] for comprehensive overviews. This paper adds the control aspect to the discussion. We provide novel viewpoints to the privacy implications of various networking technologies. We consider practical privacy enhancements to Internet protocols rather than theoretically strong anonymity mechanism. See, e.g., [15] for an overview of strong anonymity.

The rest of the paper is organized as follows. Sections 2-3 discuss the kinds of information that may be collected, the entities that may collect it, and the reasons why they might want to do so. In Section 4, we explain some common mechanisms for privacy protection and discuss their limitations and potential improvements. Section 5 concludes the paper.

2 Information That May Be Gathered

When thinking about their privacy, users are typically worried about two kinds of exposure. First, users do not want to reveal names and other identifiers that enable someone to uniquely identify them. Second, users are wary about exposing personal information, such as their address, sex, age, job, and salary. Users are particularly reluctant to reveal the personal information if it can be linked to the unique person. These concerns arise from experience: most personal information is currently collected for marketing purposes and the goal of the collectors is typically to link a person to an address and other demographic information.

In mobile Internet access, it is also possible to identify the user, either by name of by another permanent identifier. If the user is paying for network access or online services, the service provider typically knows the users name and other personal details. Otherwise, Internet protocols reveal many identifiers that can be used to uniquely identify the device or user, although rarely the users name or other personal details. Usually it is sufficient to find out the identity of the device used because most mobile devices belong exclusively to one user, or to a small group of users. The available identifiers include:

- device identifier, e.g., the network interface MAC address,
- host identifiers used by mobility protocols, e.g., the Mobile IP home address or the HIP host identifier, and
- application-layer identifiers, e.g., HTTP cookies or a media player serial number.

Device identifiers are only visible at the link layer while host identifiers are typically seen both at the local link and by the mobiles correspondents across the Internet. The application-layer identifiers are intended only for the correspondents. They may be visible at the local link if the packets are not encrypted, although sniffing is practical only for standardized protocols and identifiers.

The information gathered about users is likely to be somewhat different from the usual demographic data in which advertisers are interested. Instead of an address or a salary range, the network elements can learn the information in network signaling messages and in packet headers. The information available to observers from such sources includes:

– the mobiles location, i.e., current IP address,
– protocols and applications used,
– web sites visited and other online services contacted, and
– VPN gateways of the users organization.

In this paper, we consider the users privacy to be at risk when an observer is able to connect a communication event to a particular device or user. This information may, in fact, be more interesting to the network operators or advertisers than demographic data about the user would be. They can, for example, tell apart business and entertainment use. Moreover, even if the observer cannot link the particular identifier or event to a person, it may be sufficient to track pseudonymous users and their communication habits.

3 Common Observers and Their Motives

Traditionally-recognized threats to privacy come from such observers as direct marketers, nosy individuals, and overbroad law enforcement access. The common feature of these observers is that their main goal is to gather information about the target individual or group. These threats are not fundamentally different from the problems that existed in the pre-Internet society. The threats to privacy have been highlighted in the Internet age because digital data formats and computer networks enable more efficient data aggregation and because the users habits have not yet adjusted to the new medium.

Discussions on Internet privacy often focus on one or more traditional data collectors and on potential offline misuse of the information collected online. We, however, argue that this focus is preventing us from seeing another serious threat to user privacy and freedom: the use of the online information for controlling the users access to communications resources and online services. This threat arises from the pressure for network and service operators to maximize their revenue by price and service differentiation and market segmentation. Clearly, the operators can charge more if they know how much each individual connection or data packet is worth to the user. In order to be able to do so, the operators need detailed information about the protocol and services that each mobile node is using.

Fixed-line Internet access is usually priced by the bandwidth or by the time spent online regardless of which services the users access. The same is not true for GPRS and other mobile networks. There are complex pricing structures in place where the cost of Internet access depends on the protocol, application, and server accessed. Telephone-network operators sometimes block access to online services that compete with their own offerings. Mobile phone operators routinely charge different prices for IP traffic depending on the application-layer protocol, the server contacted, and the location of the mobile. The effect of such restrictions is twofold. First, the operators can directly control the users Internet access without the need to first gather demographic information and then using that information for market segmentation. This enables much more accurate price and service differentiation, to the extent that each new protocol and destination address is allowed only if the user pays a separate subscription for it. Second, the users find themselves unable to access arbitrary services on the Internet, including new and innovative ones, until the network operator has a service plan and a pricing structure in place for them. Thus, mobile users are unable to reap the full benefits of universal Internet access.

To summarize, we are concerned about two kinds of information gathering:

- collection of business, personal or law-enforcement
- intelligence about the users for offline use, and
- using communications metadata to exert fine-grained control over the users online activities.

The network elements where information can be collected include:

- routers and servers in the access network,
- other nodes in the access network,
- name servers and other network infrastructure,
- middle boxes such as NATs, proxies, firewalls, and other traffic filters and shapers,
- mobility infrastructure, and
- peer endpoints, such as web and email servers.

The item that is notably missing from this list is the core network routers. Theoretical and extremely paranoid discussions of Internet privacy sometimes assume that the observer may be anywhere in the network. It is, however, impractical to implement a Dolev-Yao-type omnipresent attacker on the Internet. Instead, the noteworthy observers are either located at key nodes on which the communication depends, or they are at edge nodes such as servers and other mobiles. The former will appeal to casual and covert observers while the latter are the best places for service providers and authorities to establish maximal control.

The mobile user typically accesses Internet via a wireless LAN, wired LAN, or cellular phone network. In a broadcast access network, such as a wireless LAN, other hosts can listen to the traffic between the access point and the mobile host. Even if the end-to-end communication is encrypted, the packet headers and signaling messages are typically visible to the local nodes.

The nodes in the local network can identify the mobile by link-layer and IP-layer identifiers that are in the unencrypted part of the packet. These include the link-layer address and Mobile-IPv6 home address. These addresses are visible to all nodes on the network if the LAN is unprotected by link-layer encryption, or if the encryption is based on a shared key. Regardless of link-layer encryption, the wireless access router always sees the IP-packet headers and signaling packets that have not been encrypted in the network layer. The router can, for example, learn the IP addresses of the mobile's correspondents. Moreover, the home address (in a Mobile-IPv6 home address option and routing header) cannot be encrypted at the network layer because IPsec security associations between the mobile and its correspondents are identified by the home address.

Wireless LANs that encrypt data on the link-layer using a different key between each mobile and the base station are equivalent to switched wire LANs: only the access router is able to see the IP packet headers. The mobile's link-layer address is still visible to everyone.

Most mobiles access a DNS server provided by the access network, which is typically configured with the DHCP protocol. The DNS server is able to record the names of the online servers contacted by the mobile. Even if the mobile connects to a VPN gateway and uses DNS services via a VPN tunnel, it may still rely on the local DNS server to resolve the VPN gateway name. This means that the DNS server in the access network learns the name of the organization to which the user belongs, and may enable it to identify the mobile with some accuracy. Furthermore, DNS requests are made recursively, which leaks the mobiles approximate location to the remote DNS servers. Thus, even if the actual data connections are forwarded via anonymizing proxies, the source of a DNS request may reveal the mobiles location to the peer endpoint.

In addition to the access routers and name servers, there are other bottlenecks on the network that can see and control all traffic to and from a mobile host. These include NATs, Web proxies, and firewalls and other traffic filters. In many cases, the existence of such middle boxes has the effect of preventing the hosts in the access network from using some features of the Internet protocol suite. For example, a NAT prevents the hosts from acting as servers, except for specifically approved protocols, and a firewall can be used to filter out protocols and destinations not endorsed by the network operator. Together with information acquired from authentication and sniffed signaling, such middle boxes enable fine-grained control of the services afforded to each mobile.

Mobility agents, such as Mobile IP [13] home agents, HIP [12] forwarding agents and SIP [14] registrars, are another point at which location information can be collected. The mobility agents may also be able to differentiate between different applications and, thus, control the mobiles Internet access. Indeed, such control is one of the main functions of a SIP server. SIP servers enable two Internet nodes to find each other even if neither one of them has a permanent IP address. SIP servers are often provided by a network or VoIP service operator. They typically perform accounting functions and, therefore, identify and authenticate the user. Unlike SIP, Mobile IP does not specify a mechanism for differentiating between applications but, in the reverse-tunneling mode, the home agent could implement similar control. Moreover, location information collected by the mobile can be used to vary prices depending on the mobiles location.

The peer endpoints can, of course, also collect information on the mobile node. Since this is the privacy treat model that has received most attention in the past, it suffices to mention that mobility does not change the threats all that much. Only the location of the mobile is added to the data that the mobiles correspondents can learn about it. Advertising-based services, including most online publications, need to know their customer in order to target advertising, and the location is one valuable additional piece of information. Mapping the IP address to a geographical location is, however, relatively unreliable.

4 Some Simple and Effective Privacy Solutions

In this section, we consider simple and inexpensive privacy solutions that exist in current networks. We are interested only in mechanisms that can be deployed locally with little cost in either infrastructure or communications overhead. We suggest ways of using these mechanisms effectively to improve the privacy of mobile users.

4.1 Roaming vs. Mobility

Roaming is a term more commonly used in cellular phone networks than in TCP/IP. Nevertheless, it is useful in making a distinction between different types of mobile access. Roaming means that a mobile host gains access to the Internet via some other access network than its usual network. For example, if one takes a laptop computer from work to home and connects to the Internet via the network at ones home, the laptop is roaming there. The reader might wonder what is so unusual about roaming. The answer is that, in mobile phone networks, roaming cannot be implemented without a complex inter- operator charging system and contracts drawn by lawyers.

For the more orthodox type of engineer, roaming is not true mobility. Mobility means that that mobile is moving from one network to another. Moreover, mobility is associated with two technical challenges that the engineer expects a mobility system to solve. First, connections between the mobile should survive the movement, even across the boundaries of physical networks. Second, the mobile should be reachable wherever it goes. That is, other network nodes should be able to contact the mobile. Solving these two problems requires fairly sophisticated technology, such as the cellular mobility management protocols or Mobile IP.

From the privacy perspective, roaming, regardless of its technological inferiority, is more attractive than mobility. Consider a mobile host that attaches to the Internet via whatever access network is currently available, obtains an address at the access network, and behaves in every way like a local node at that network. It can access many Internet services, such as web servers, without identifying itself or telling anyone that it is a mobile host.

On the other hand, Mobile IP(v6) and other mobility protocols require the user to identify itself to the servers and peer hosts by some permanent identifier, so that the same mobile can be recognized when it reconnects from a new location. Many mobility protocols also require the mobile to register its location in some central or distributed database in order to implement the reachability.

It is clear that user privacy would be better protected by using the mobility protocols only when their features are needed and simple roaming access at other times. The reason why this is rarely done is that it is difficult for the computer to know what level of service the user needs, and how soon the user intends to move. It is also difficult for the IP, transport or middleware layers, where mobility is often implemented, to know whether a particular application requires mobility or just roaming. Quite clearly, it would help to make the applications more aware of mobility, and the protocol stack more aware of the application requirements. It may, however, take a while before the protocol stack layers and applications communicate with each other about privacy requirements.

4.2 Anonymizing Proxies and Mobile IP Tunneling

Forwarding packets via a proxy hides the individual sender within the group (anonymity set) [2]. The most common type of proxy is an HTTP proxy. They are commonly used at the boundary of corporate networks and hide the sources of individual web requests. This is useful even in a stationary network: an observer on the Internet (usually the web server) knows that a request came from the specific organization but it cannot tell which member of the organization made it.

Some web proxies (e.g., Anonymizer) exist for the sole purpose of anonymizing web access. In that case, they may be located far from the user who sends requests to the anonymizing proxy via an encrypted tunnel. They also try to attract large and diverse groups of users, with the aim of hiding not only the identity of the individual user that makes a request but also the organization from which the request originated.

The way we usually think about the effect of the proxies is that they hide an individual in a group. If the group is large enough, it is impossible to distinguish individual access patterns in the pool of web requests. In a mobile access network, however, the proxies have another effect that is perhaps more significant: a proxy at the access network hides the existence mobile nodes that are connected to the local network behind the proxy.

One can also see a NAT [16] as a proxy that handles individual IP packets. It hides the individual nodes within the network and presents only a single IP address to the Internet. In addition to having the same privacy benefits as a web proxy, a NAT has the effect of hiding the number of IP hosts in the network, as well as the structure of the network behind the NAT. Obviously, the existence of any mobiles is hidden as well. A limitation for both the NAT and the web proxy is that if the number of nodes behind the proxy is small, the anonymizing effects become weaker. (We will have more to say about NATs and IPv6 below.)

The Mobile IP home agent is also a kind of proxy between the mobile node and the Internet. If the mobile uses reverse tunneling, all packets are routed via that home agent. The effect of this proxy is, however, different from the proxy in the access network. It hides the mobiles location, and even the fact that the mobile is a mobile, while revealing its the home address (i.e., permanent identifier) to the peer endpoints. This may sometimes be exactly what the mobile user wants.

Mobile IP reverse tunneling implements full mobility, as defined in the previous section. So how is it able to hide the mobiles location? The answer is that reverse tunneling is a trade-off between performance and privacy. By sending all communication via the home agent, the path taken by the packets may be much longer than it would be if the packets were sent directly.

To be honest, the original trade-off was between performance and technical complexity but, at least in Mobile IPv6, that is no longer an issue as implementations of route optimization exist. It would be desirable to make the choice between reverse tunneling and route optimization (i.e., direct communication between the mobile and the peer endpoint) depending on the application requirements. Unfortunately, the state of art in Mobile IP(v6) implementations does not yet allow this.

4.3 Mobile IPv6 Route Optimization and IPsec

IPsec [9] encryption and Mobile IPv6 [8] do not combine well to hide the mobile users identity. The reason is a kind of chicken and egg problem: it is not clear whether IPsec should run on top of Mobile IPv6 or vice versa. On one hand, the messages exchanged by the mobile node and its correspondents contain the mobiles home address, which identifies the mobile, in the home address option or routing header. Therefore, the headers of these messages should be encrypted to hide the home address. On the other hand, IPsec identifies endpoints based on their IP addresses. Therefore, it would be technically more elegant to run IPsec on top of Mobile IPv6 so that the IPsec policies and security associations can be indexed by the permanent home address rather than by the changing care-of address.

One solution, although not an ideal one, is to create a new security association between the correspondent and each new care-of address. The encryption would hide not only the home address but also the fact that a mobility protocol is being used. The tunnel would be encrypted but it need not be authenticated. Indeed, authentication is not even possible below the Mobile IPv6 layer because the only available identifier for the mobile is the care-of address. The session key for the encrypted tunnel between the care-of address and the correspondent could be created using an opportunistic key exchange, or with a unidirectional authentication and key exchange protocol.

An opportunistic key exchange is one that does not bind the session key to any identifier that has meaning on the application layer but, instead, uses whatever (even weak) credentials are available at the IP layer to prevent man-in-the-middle attacks. Encryption based on such a key exchange is useful for privacy protection of the content data and header data above the IPsec layer (including the care-of address) because increases the cost of privacy violations.

A unidirectional authentication and key exchange, on the other hand, assures the mobile about the identity of the correspondent but gives the correspondent no information about be mobile. The asymmetry is similar to the way the SSL protocol us usually used. Since the goal is to protect the mobiles identity, the asymmetric authentication is sufficient. A limitation of both the opportunistic and the unidirectional protocols is that the resulting session key must not be relied upon for strong authentication of encrypted application-layer data.

This kind of excessive layering of IPsec is, of course, not an ideal solution. The fact that we have to resort to such trickery calls into question the whole Mobile IPv6 and IPsec protocol architectures. Since there is such intricate interaction between the secure tunnel and the endpoint-mobility protocol, it would make sense to combine the two into one protocol. Indeed, mobility could be implemented by changing the addresses of the endpoints of IPsec tunnels, instead of the ends of the degenerate kind of tunnel that Mobile IPv6 implements.

The above idea appears in the novel HIP and MOBIKE protocols, but also in the standard Mobile IPv6. The IPsec security association between the mobile node and its home agent is updated dynamically when the mobile moves so that it can always be looked up by the mobiles current care-of address. This makes it possible to send binding updates and tunneled data to the home agent thought an encrypted IPsec tunnel, which hides the home address and the fact that Mobile IPv6 is being used. Since the security association between the mobile and its home agent is typically configured manually or by some out-of-band protocol, there is no key exchange that could divulge the mobiles identity outsiders. If a dynamic key exchange is needed, it is important to use a protocol, such as IKEv2, that can hide the endpoint identifiers from passive eavesdroppers. It is also important that the key exchange protocol does not insist on using the IP address as an endpoint identifier (as most IKEv1 implementations do).

4.4 RFC-3041 Addresses and DHCP

Apart from the permanent Mobile IP home addresses, most IP addresses are assigned dynamically either by DHCP or stateless autoconfiguration (the latter only in IPv6). Nevertheless, all IP addresses enable observers to correlate packets sent by the same host as long as the host keeps using the same address. Moreover, typical IPv6 addresses have a structure that essentially includes an embedded permanent identifier.

The interface identifiers in IPv6 addresses were originally created by expanding the 48-bit link-layer (MAC) address of the network interface card to a 64-bit EUI-64 [7]. Anyone who sees such an address can recover the MAC address and, thus, identify the network interface card. The unique identifier embedded into the IPv6 address can be used to track a mobile device and user even outside the access network where one would not expect the link-layer address to be visible.

A solution was proposed in RFC 3041 [11]: the interface identifier can be selected randomly instead of using the EUI-64, and the value can be changed periodically. This makes no difference to the functionality of the address as the only real requirement for the interface identifier is that it is different for each node in the network. It is very unlikely that two randomly generated addresses collide. RFC-3041 addresses are particularly attractive for roaming mobile hosts that move from one access network to another because they can create a new random address at each access network and, thus, hide the link between the addresses.

IPv4 addresses cannot be randomized in the same way. On the other hand, IPv4 addresses do not have any part that uniquely identifies the host. DHCP servers typically allocate the same address to the same host every time it asks for an address, unless it has already been given to someone else. One possibility is to change this behavior so that the DHCP server randomizes the address assignments. This prevents peer endpoints and observers outside the local network from correlating connections with earlier ones when the mobile returns to a previous access network.

It is interesting to note that IPv6 hosts can anonymize their addresses independently, without any help from the network infrastructure. In IPv4, it is the network that needs to randomize the address assignment. This difference is, of course, a direct consequence of the fact that the preferred method of assigning IPv6 addresses is stateless autoconfiguration, where the host selects its own address, while IPv4 addresses are usually assigned by a stateful DHCP server.

The randomized addresses only help when the mobile is communicating with remote nodes across the Internet. Nodes on the access network can observe the mobile hosts link-layer address, which is a permanent identifier for the network interface hardware. The obvious solution would be to randomize also the link-layer address. Unfortunately, there is no consensus for doing this. In fact, the trend is quite the opposite: since Ethernet MAC addresses are used as a weak form of host authentication in many wireless LANs, the hardware manufacturers and driver writers are making the changing of MAC addresses more difficult than it used to be.

4.5 NAT and IPv6

The main reasons for deploying a network address translator (NAT) are that it allows multiple Internet hosts to share a single IP address and that it acts as a kind of stateful firewall by preventing Internet nodes from sending unsolicited packets to the local network behind the NAT. The fact that the ISP cannot differentiate pricing based on the number of nodes on the local network is particularly important for home users. Such per-host charging would, for example, prevent users from taking mobile devices from work to roam at the home network.

From the privacy point of view, there is also the advantage the NAT hides the structure of the local network from the Internet. Observers on the Internet cannot tell whether two connections from the NAT originate from the same host or from two different hosts. If the NAT hides a more complex intranet that is composed of multiple local links, the subnet prefixes and, thus, the topology of the intranet, are also masked by the NAT. In addition to preventing the ISP from charging per node, an advantage of the NAT for mobile users is that it makes it much more difficult for Internet nodes to observe the arrival and departure of mobile nodes to and from the intranet.

A disadvantage of a NAT is that it makes providing global services from behind the NAT cumbersome and, in some cases, impossible. For example, it is not possible to have two web servers at home using the standard port 80. Also, NATs prevent the deployment of mobility and security protocols, such as Mobile IP and IPsec, which are both either blocked by a NAT or require complicated NAT-traversal solutions.

IPv6 solves the problem of IP-address shortage, which makes it less acceptable for ISPs to differentiate pricing based on the number of IP addresses used in each network. Eventually, more users will want to have server computers at home, such as web and file servers. They will also want to use IPsec, VoIP, and other protocols that are currently blocked or made inconvenient by NATs. Thus, there will be a need to allocate an IPv6 subnet prefix to each home, rather than a single IP address.

Many network administrators are, however, so used to the security provided by a NAT that they find the deployment of NAT-less IPv6 to be a threat rather than a promise. If this is the case, it is worth remembering that a stateful firewall can implement the same policies as a NAT and provide additional flexibility, e.g., by authorizing any intranet node to act as a server. The privacy aspect of NATs is, on the other hand, not as easy to replicate in IPv6 without actually bringing back the limitations of a NAT.

The solution we propose is to use more addresses instead of less. That is, an IPv6 hosts should use large numbers of temporary addresses and change them frequently. In fact, we propose using a new IP address for each TCP connection that originates from an IPv6 host. Using a new random or pseudo-random address for each connection makes it more difficult for an observer on the Internet to correlate two connections from the same host. It also prevents the observer from counting the hosts in the local network. Note that this solution is more useful for home users and other small networks with only a single subnet prefix than it is for large networks with multiple prefixes.

In order to prevent the TCP port numbers from revealing the association between the different hosts, it is better to either use the same port (e.g. 12345) or a completely random local port number for all TCP connections initiated by an IPv6 host. For UDP-based protocols, a similar approach can be taken, although it is necessary to understand the higher-layer protocol or application to know when it is appropriate to generate a new IP address.

If IPv6 hosts routinely use a large number of addresses, that may also guide the ISP charging policies away from counting the number of addresses, which will help create an IPv6 Internet without the unnecessary complications caused by NATs. This would make it much easier to deploy home servers and mobility and security protocols.

4.6 Random Addresses and CGA

Some network-layer signaling protocols for IPv6 rely on cryptographically generated addresses (CGA) [1] for message authentication. CGAs are IPv6 addresses where some bits of the address have been generated by computing a one-way hash of the mobile hosts public signature key and some additional parameters, including a random modifier. This key is then used to sign signaling messages. CGA-based authentication is being standardized for neighbor discovery and there are proposals for using CGA for authentication of mobility signaling and for end-to-end key exchange.

The problem with a CGA is that the address and the public key may be used to identify the mobile. If the mobile uses the public key to authenticate end-to-end signaling such as Mobile IPv6 binding updates, the only solution is to change the public key. However, if the mobile does not use signatures for end-to-end signaling, it can randomize its address by re-randomizing the modifier parameters in the hash input. (For those familiar with the CGA details: the re-randomization prevents the mobile from using long hash extensions but that hardly matters for mobiles that move frequently.)

The public key, of course, can also reveal the mobiles identity to the nodes in the same local network. This only makes a difference if the mobile is already randomizing its link-layer address or changes network interface cards occasionally. In that case, the solution is also to replace the public key and to generate a new CGA whenever the mobile moves from one access network to another.

5 Conclusion

In this paper, we discuss the privacy implications of transparent wireless and mobile Internet access, with focus on information present in packet headers and in signaling messages. We discuss the ways in which user identity and location can be revealed by the network protocols, usually without the user being aware of the disclosure, and the ways in which various network elements can use communications metadata to control the users Internet access. We also suggest simple technical solutions that reduce the amount of information leaked in order to preserve the users privacy and to retain more control at the end nodes.

References

1. Tuomas Aura. Cryptographically generated addresses (CGA). In *Proc. 6th Information Security Conference (ISC'03)*, volume 2851 of *LNCS*, pages 29–43, Bristol, UK, October 2003. Springer.
2. David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
3. Alberto Escudero-Pasqual. *Privacy in the next generation Internet: data protection in the context of the European Union.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, December 2002.
4. Digital cellular telecommunication system (phase 2); Security related network functions. ETSI TS 100 929, European Telecommunications Standards Institute, November 1999.
5. Hannes Federrath, Anja Jerichow, and Andreas Pfitzmann. MIXes in mobile communication systems: Location management with privacy. In *Proc. Information Hiding Workshop*, Cambridge, UK, 1996.
6. Marco Gruteser and Baik Hoh. On the anonymity of periodic location samples. In *Proc. Intl. Conference on Security in Pervasive Computing*, volume 3450 of *LNCS*, pages 179–193, Boppard, Germany, April 2005. Springer.

7. Robert M. Hinden and Stephen E. Deering. IP version 6 addressing architecture. RFC 3513, IETF Network Working Group, April 2003.
8. David B. Johnson, Charles Perkins, and Jari Arkko. Mobility support in IPv6. RFC 3775, IETF Mobile IP Working Group, June 2004.
9. Stephen Kent and Randall Atkinson. Security architecture for the Internet Protocol. RFC 2401, IETF Network Working Group, November 1998.
10. Dogan Kesdogan, Hannes Federrath, Anja Jerichow, and Andreas Pfitzmann. Location management strategies increasing privacy in mobile communication. In *Proc. 12th International Information Security Conference*, pages 39–48, Samos, Greece, 21–24 1996. Chapman & Hall.
11. Thomas Narten and Richard Draves. Privacy extensions for stateless address autoconfiguration in IPv6. RFC 3041, IETF Network Working Group, January 2001.
12. Pekka Nikander, Jukka Ylitalo, and Jorma Wall. Integrating security, mobility, and multi-homing in a HIP way. In *Proc. Network and Distributed Systems Security Symposium (NDSS'03)*, pages 87–99, San Diego, CA USA, February 2003.
13. Charles Perkins. IP mobility support. RFC 2002, IETF, October 1996.
14. Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. SIP: Session initiation protocol. RFC 3261, IETF, June 2002.
15. Andrei Serjantov. *On the Anonymity of Anonymity Systems.* PhD thesis, University of Cambridge, Cambridge, United Kingdom, March 2004.
16. Pyda Srisuresh and Matt Holdrege. Ip network address translator (NAT) terminology and considerations. RFC 2663, IETF, August 1999.
17. Alf Zugenmaier. *Anonymity for Users of Mobile Devices through Location Addressing.* PhD thesis, University of Freiburg, 2003.

Privacy, Control and Internet Mobility (Transcript of Discussion)

Tuomas Aura

Microsoft Research, Cambridge

This is work not yet done by myself and Alf Zugenmaier at Microsoft Research here in Cambridge. With Alf being a privacy person we're starting to think about the kind of threats there are to a wireless mobile user's privacy. We thought these threats actually matter to the user, and so what can we do about them? Especially, what can we do at the time of attack, or when designing network protocols to improve the users privacy? This is a position paper about the kinds of things we think probably matter and should be done.

The two main hypotheses that we have are, first of all, that what matters for mass observation is signalling messages and applications in the Internet. The content of the data is pretty much irrelevant because, in order to observe the content of data, the observer has to actually understand what this data does. It won't be a human doing this work; it has to be someone automating the observations, tracking the user and so on. And also if the users are interested in protecting their application metadata, then there are security protocols that can hide this, and they can use encryption. But what we really should look at are the signalling messages and applications which are often ignored when thinking about privacy, as well as by people looking at things like wireless card MAC addresses.

We made the other hypothesis by asking about the motives for someone to observe users and their traffic over the network. This hypothesis is that people are not so interested in hiding what they are doing from the police or from their neighbours, hiding how many members there are and so on, though these kinds of things often come up in privacy discussions. What really seems to matter more to people, and also to those who are trying to track us, is whether this information that you can collect on the network can then be used in some way: information which comes from you, or from your network access, or your access to services. And so for this reason, I don't know if the protocol I am using should be included under the umbrella of privacy, which services I access, and so on. Is it a privacy matter if someone can stop me from doing what I want to do with my Internet access? No matter whether it should be called privacy or not, it's something that matters to the user, and to the attacker. Of course what we should then look for is how to fix the real problem behind this.

We'll take a look at some simple techniques that you can use to move control to the user over their own Internet access.

Here is the type of information that you might want to protect. None of this information is in the application data, the actual payloads, or the email you are sending, or web page you are downloading, but it is somehow in the protocol metadata. Location is obviously something that we might worry about wanting to hide, and which websites one has been accessing. But there are also things like which other services have you used, and maybe, and more surprisingly, which protocols and applications you have used for the Internet access. It might tell quite a lot about you, for example, if you open a VPN connection somewhere, that means that you are a kind of business user, you have somewhere a company network to connect to, so you're not just surfing for entertainment.

So, these things might tell about the affiliation of the user to someone who's looking at the traffic. For example, if I surf on the network then you can probably see my laptop will try to connect to Microsoft email server, and that sort of thing. You can tell who I work for, and you can also tell that I'm a business traveller, and I would be willing to pay more for my wireless access here, and Matt there might put up the prices.

So there's lots of information in the metadata and what we really haven't done yet is properly to analyse what kind of information is going to be in the protocol.

Matt Blaze: Are you assuming the attacker is just looking at traffic, or doing something more active?

Reply: The attacker is just looking at the packets. An attacker might be at the local network where the mobile laptop is, or the attacker might be anywhere on the Internet.

Matt Blaze: But is the attacker doing something active, like profiling your computer to figure out what applications are installed on it, and find out all about you like that?

Reply: No, I think that's a secondary framework.

Matt Blaze: So that's not in your remit?

Reply: Yes, if it's easy to do that then I will include that, but in the first step we are concerned with passive observation.

Bruce Christianson: Is it part of your threat model that the attacker might have control over the pricing structure?

Reply: Yes. The question here is really who are those persons and what are their objectives, so we probably shouldn't call them attackers.

Bruce Christianson: It's a sleight of hand.

Reply: Yes, that's a good way of putting it. What really matters with this kind of information is, and this I guess is obvious to everyone, somehow connecting the information to some kind of identifier, or identity[1], that you can either track, serialise (over multiple consecutive accesses), or use to connect to that person. Typically with identifiers you concentrate on one thing, so some people look at the MAC addresses, and I've been worried a lot about IPv6 addresses, but

[1] Dieter Gollmann, these proceedings.

basically to the observer it doesn't matter what kind of identifier it is, as long as they know they have something to connect together the pieces of information about the user.

So a known identifier could be a MAC address, IP addresses or mobility protocols which typically give you a permanent identifier that you can use anywhere no matter where you are on the net. For example mobile IP uses a home address, so it's a permanent ID address where the mobile is always based. All the packets sent to and from the mobile now have the home address somewhere. It's obviously a way of broadcasting who you are, in all the packets that you send you're telling everyone this permanent identity. Then there are application-level identifiers, which we haven't really considered seriously because there is already enough in the applicators mobility signal.

Obviously strong anonymity systems, like mixes, are research topics, good for a PhD, but they don't matter in practice. I think if we really want to improve people's privacy, we should do better to minimise the amount of data that we send on the network, and so reduce the risk that it is available for someone to observe and collect. This might be by protocol design to remove unnecessary permanent identifiers, or to use them less. A good example is the GSM network where (originally in order to save bits) they decided not to use the mobile node's identifier in every packet. They use a temporary identifier instead, and this turned out to be an important privacy measure; you can't track someone's mobile phone by just listening to the packets in the air because they use the temporary identifier that changes every now and then.

Another general principle that seems to help is to have some kind of indirection between the user's identity and the services that they access. If we have this indirection step, then after traditional authentication of identity, we can forget about the long-term identity, and give anonymous authorisation to the user, who can go and access the services[2].

Audience: Who should need anonymity protection, I mean?

Reply: Against me, or against Matt who is driving your Internet access there, and all your packets go through his protocol analyses.

Matt Blaze: Like a relay.

Reply: And he's just entertaining himself there, reading your email. These people may obviously not be bad, not what we would call attackers, but people who may have a "legitimate business reason" for looking at your traffic.

The classic privacy threat is the big brother that is everywhere and trying to get us, or at least me. A common example is marketing people, because they want to know who I am, they want to sell things to me, they want to know as much as possible about me, and they especially want to know about large groups of people. Another example is government, and the police I guess. In order to do their job they want to have access to information. Their real motive in this threat

[2] See Chowdhury, Christianson & Malcolm, *Anonymous Authenication*, these proceedings.

model is to collect data about individuals or groups. This data is collected on line on the Internet but it is not really used on the Internet; it is used offline. For example, if the police knock on your door at 5 o'clock in the morning, or if you get junk mail to your physical mailbox. The network is a source of information, and then this information is put into good or evil use, however you feel about it, somewhere outside the computer network. Of course this is an old problem, but the fact that we have the Internet, and all this information technology that makes it much more efficient to collect data, means it may become a much larger problem.

But regarding whether it is good or not to collect this data, the argument is in both places. You can say, well it is good for the economy in the market, it is good for our security too, for the police to be able to access your data, to know where you are and what you are doing if they need to. Obviously there's a trade-off there, and there's maybe no right level. There is this kind of opponent, the user and someone else, and they have different interests.

The new threat I don't have a good name for, so today I'm calling it service differentiation. Typical opponents for you would be the service and network operators, like your mobile phone operator, or your wireless LAN operator, your ISP, your email and webmail provider. They are businesses and their goal is to maximise their profits. One way to do that is to differentiate the service quality and pricing between users and the different services you're accessing. But in order to do that they need to understand what you are doing, what kind of services you are accessing, what matters to you, and what matters to each particular user. For this reason they, first of all, need to observe what you are doing online, and then they need to be able to control your behaviour online. The simplest case is block for hello access, but also maybe they trust the quality of service they provide to you.

The difference to the previous case is that the data is both collected and used online. This is why it matters more, because it's not some kind of complex system where you have to assume that someone collects this data, processes it, mines it, and then uses it for some purpose that it wasn't originally intended for. Here the data is only being used for the purpose that it is intended for. For example, you send a packet and it's classified for quality of service routing based on the headers. Well that's what the headers are for aren't they? What these networks want to do is to make it easier than before for all these service providers to differentiate the services, and it's not necessarily in the interests of the consumer or user. The user would like to have flat pricing structures, or would like not to differentiate between private use and business use. It's common for ISPs to say you have to pay ten times as much if you use your home line for business purposes rather than for entertainment.

I am not claiming that something here is good or bad. It maybe a sensible business model for someone who operate these services. But on the other hand if I'm thinking of myself as a consumer I am on the opposite side of the table, and I would like to prevent them from knowing what I'm doing. If I can do something about that, then I'd like to do so.

There are various little boxes on the network that do something funny to your traffic. These are good places both for observing the traffic, and filtering and controlling what can get through. The common examples are Nets and Web Proxies. For example, my ISP intercepts all ACP data tracking, and sends it to their own proxy, so instead of the 512 kilobytes per second that I'm paying for, I only get one web page every minute because their proxy just happens to be a bit slow of course. That's the way they control the bandwidth use. It may be a good idea for them, it's not a good thing for me so if I can get around that I would like to.

Of course all kinds of firewalls exist nowadays. Designers increasingly provide you with what they call a firewall, that means they filter out more and more protocols as they come in, maybe voice over IP kind of traffic. I'm not feeling very impressed with that.

And then there is the mobility infrastructure, home agents and mobile agents. You have this one home base per user. You need this home base; providers depend on it. We were hoping to get rid of this home environment and have mobility without it, and then I discovered that providers are really worried about the fact that a user might use a different, unapproved home base, home bases that they didn't provide for the user. But it's understandable because actually certain products in this model, charge for the home base and registration for mobile IP. And I guess one of the main reasons for it is to have a place where you can do an audit of a charge for the connections.

Bruce Christianson: To what extent is the mobile network provider going to continue to have control over the applications that are used? At the moment a lot of the things that they can potentially do rely crucially on the fact that there is no possibility of tunnelling or encapsulation.

Reply: Yes. This is something that we have to state, build the Internet protocols right. We could prevent someone from say building application to application, the typical example is LAN access. The same company could set up wireless access points that are running 2G and 2.5G and 3G mobile phone networks, and then they sell you mobile phones that claim to have Internet access, but in fact the only websites you can access are the stores where you can buy ring tones and wallpapers, and there's absolutely no way of accessing anything else.

Bruce Christianson: Perhaps there needs to be some regulatory provision like the old FED3, that says you must be divested from doing both things.

Controlling Who Tracks Me

Denis Bohm[1], Mik Lamming[2], Robert N. Mayo[2], Jeff Morgan[2], and Kan Zhang[2]

[1] Firefly Design, Los Altos, CA USA
[2] Hewlett-Packard Laboratories, Palo Alto, CA USA

Abstract. Many Ubiquitous Computing applications require tags and sensors that track the daily activates of people and things. For example, tags might be placed on objects to allow a system to remind people of the object's location. In another application, caretakers of the elderly may wish to monitor the daily activities of those under their care, in order to track decline in functioning and offer appropriate medical care. These forms of tracking, however, raise privacy concerns. A person does not want to be tracked by everybody, but rather only by those that are trusted and only during certain times. We present a method of building tags and tracking devices that ensure a person is tracked only by those that have been granted that right, and only during the times specified.

1 Background

More and more portable or wearable devices are appearing on the market with the potential to communicate wirelessly. Almost in lock-step public awareness of their vulnerability to tracking is rising, and we are beginning to see dire warnings in the press about the potential invasion of personal privacy[1]. While tracking is not a new phenomenon (ref: war-time spy-radio emission detection; Visa card transactions) the risk versus benefit has been relatively easy for users to assess and control. One reason for this has been that the wireless infrastructure that enables communication, and thus tracking, has been very expensive to install on a large scale, and has therefore remained in the hands of relatively trustworthy large enterprises such as cellular telephone service providers, and credit card companies. This is changing, and now individuals have the capability to install their own limited infrastructure (802.11 hotspots) or carry devices capable of making spontaneous ad-hoc connections to any compatible device that is in range[5].

At present the threat to individuals is limited because the infrastructure is not yet widespread, tracking software is not yet very sophisticated, and users can in any case easily switch off their portable device. With the introduction of passive wireless devices such as radio frequency identification devices (RFID) the threat is becoming more apparent. These devices are called "passive" because they require no internal power, and are stimulated to reveal their unique identity code simply by being scanned by an RF reader device located a few inches or feet away.

Manufacturers are keen to incorporate these devices into the items they make to facilitate stock control and have been surprised to discover civil liberties groups turning on them. The reason for this response is that once an RFID has been embedded in an item of clothing for example, the wearer can be tracked passing by any RFID reader. Since these readers are installed at natural "choke points", doorways for example, vendors are able to recognize and identify individuals as they enter the store and exploit this information - presumably to the wearer's detriment. At present readers are normally used in locations where stock control, and security issues are important, but as prices fall, and tracking software becomes more available, we can expect RFID tracking by small organizations, or individuals to become practical. As this scenario unfolds we anticipate public concern will increase and an opportunity will emerge for solutions that give the user more control of what information they reveal about themselves.

2 Who Needs Tracking Anyway?

One problem with this scenario is that the public is being systematically exposed to the downsides of this new technology before they have the opportunity to experience the benefits. We have been exploring various "everyday" applications of technologies capable of tracking, not just the location, but potentially the context and behavior of the wearer and have become convinced that there are many substantial benefits[2]. In the medical arena alone there seem to be many benefits to tracking daily activities[3], but we face an uphill struggle to convince potential beneficiaries that the downsides of personal tracking can be controlled and contained. We fear that unless we can rise to the challenge many of the benefits will not see the light of day for many years. In effect, we will be throwing the baby out with the bathwater. On the other hand, if we can find a solution that balances the risks with the benefits we may be able to open up a new industry in personal tracking and data mining.

3 SPECs

To explore these and other opportunities we developed a tiny computer called SPEC[4]. It is designed to be worn, or attached to a place or a thing. Its functionality is very limited - in essence, it notices and remembers when other SPECs are nearby. It takes the first tentative steps towards providing a minimal level of continuous situation awareness.

SPEC uses a PIC micro controller and a real-time clock for time stamping observations, which it can store in 32KB of memory. It communicates with other nearby SPECs using a 40kHz infrared carrier with OOK at 2666 baud and Manchester encoding giving an effective data rate of 40 32-bit words per second, and a range of 4-8 meters. The current user-interface uses a single green LED used to attract the user's attention, and a single button.

Each SPEC broadcasts a unique 32-bit identifier (ID32) every 2 seconds (to conserve power this interval is increased automatically when the set of SPECs in proximity isn't changing). They also listen continuously for the ID32 broadcast from nearby SPECs. When a new SPEC is sighted, a sighting record is created, time stamped with the start time, and stored in a history. Each record describes an interval during which a particular ID32 was repeatedly sighted. If sightings cease for more than a specified interval (2 minutes in the current system), then the sighting record is time stamped with the end time and closed.

4 Selective Tracking

To get a minimum level of privacy, each SPEC broadcasts time-varying identifiers pseudo-randomly derived from its own unique secret master key. Rather than computing the time-varying identifiers from the master key directly, we derive them in a hierarchical manner. For example, suppose the master key is Km. We first compute the year key K2004 for year 2004 as K2004 = H(Km, 2004), where H() is a secure hash function like MD5 or HMAC. Next, the month key K200401 for Jan, 2004 is computed as K200401 = H(K2004, 01) and day key K20040120 for Jan 20, 2004 as K20040120 = H(K200401, 20) and so on. For initial experimentation purposes we chose to stop the key hierarchy at the hour level and the hourly keys (or certain encoding of them) are used as broadcast identifiers.

In such a way, a SPEC can selectively give out its year, month, day or hourly keys to its friends so that they can recognize it for the validity periods of the corresponding keys without having to reveal the master key.

We are aware of the shortcomings associated with such a simple scheme. For example, a SPEC can be tracked by anyone within an hour since the broadcast identifier stays the same throughout an hour. For the same reason, a SPEC can also be impersonated within an hour. Moreover, SPECs can be impersonated at another location by relaying observed identifiers to remote locations - a man in the middle attack.

We are currently examining more sophisticated designs in an attempt to find a balance between privacy and usability. The difficulty lies in the low power constraints of SPECs, and the infrastructure-less environments that SPECs operate in. SPECs are wearable peer-to-peer devices designed to operate autonomously over a long period of time (in years). Their usefulness for studying context-sensitive computing depends heavily on their portability and being able to operate continuously. Hence, minimizing power consumption is always a top concern. We are further constrained by the peer-to-peer autonomous nature of SPECs - there is no real-time connection to a supporting infrastructure.

All in all, we believe the complex interplay between power-utilization, user-needs, and security/privacy provides us with a challenge that is central to ubiquitous computing.

References

1. BBCNews.co.uk 'Radio tags spark privacy worries.' 11-Nov-2003
2. Weiser, Mark (1991). 'The Computer for the Twenty-First Century'. Scientific American, pp. 94-10, September 1991.

3. Forbes.com. 'Digitally Monitoring Mom' 14-Aug-2003
4. Lamming, M. and Bohm, D (2003). 'SPECs: Another Approach to Human Context and Activity Sensing Research, Using Tiny Peer-to-Peer Wireless Computers'. UbiComp 2003: Ubiquitous Computing: 5th International Conference, Seattle,WA, USA, October 12-15, 2003, Proceedings
5. BBC News.co.uk. 'New mobile message craze spreads' 4-Nov-2003

Controlling Who Tracks Me (Transcript of Discussion)

Robert N. Mayo

Hewlett-Packard Laboratories, USA

Our group is looking for applications in everyday life that involve information processing. Not Internet connected things, not things that are on the network, but things that are in everyday life. For instance, you may forget to buy milk; that's an information processing problem. If your car gets a flat tyre, that's probably not an information processing problem, but there may be information processing that can assist you.

One basic idea is to track your daily activities for your personal benefit. For instance, you may use devices that give you location awareness, or detect categories in your daily routine. If every Tuesday you throw your laundry in the car and go to the dry cleaners, then a device could conceivably detect this, and if on Tuesday morning it finds you in your car and you don't have your laundry with you, it lets you know, beep, and tells you that something's different; and you can either choose to ignore it or run back and get your laundry.

Another application is to create a lifetime diary. There are a few people in the world actually doing this, maybe a dozen. They take every single piece of data they ever encounter in their entire life and store it in a single place, and hopefully use it for future applications. Maybe you might mine it for personal data: you may find that everyday that you have a caf mocha you're grumpy the next day. I don't know how you measure grumpy automatically, but things like that. A more down to earth application that we're looking at in the short term, is assisting people that care for Altzheimers patients. If you're caring for an Altzheimers patient, you have to be constantly vigilant monitoring their activity. Some things we might be able to help with would be, monitoring hydration, which is a big problem, making sure they take their medications, discovering patterns of long term decline, eye behaviour, falling down, things like that.

To investigate these we've built an apparatus to gather data and see if these applications have anything to them. Some requirements for this: the device must be unintrusive; people aren't going to carry a big thing, so it has to be small enough that it doesn't change daily applications; and furthermore it can't really use any built-in infrastructure.

You'll see the privacy implications right away I think. We have these little tiny devices and they basically transmit IDs every couple of seconds, and then other devices recognise the ID. Here's this student who wears all this stuff everyday, and he's not a laughing stock; he's cool in his own way, but his problem is he always takes his scooter to school and then often forgets it and walks home with his friends. So let's see if we can get him to not forget his scooter. This scooter and his backpack each broadcast IDs every couple of seconds. The little device that he carries on his shirt broadcasts his ID, there's one in his garage, there's also one in his desk at school, and these things work patterns, and they see that he went to school with his scooter but locked his desk for more than ten minutes, and doesn't have the scooter, then it will remind him to go back and get it.

The basic data that these things collect looks like this, at this time you are in this location where you saw this item near you. You can aggregate and collate this to get reasonable sounding things like, you were in the dining room for four hours except for a 20 minute excursion to the kitchen, or something like that. And of course if you can do this then so can other people, so privacy could be a real barrier. We have all heard the news media about putting a hitchhiker inside someone's sweater and then every time they go to a mall someone's going to be tracking them. That's a totally legitimate problem.

So what are the typical privacy desires for this sort of system? I think information is money in a way. I don't want to be tracked, I don't want to give my information out to anyone, except when it's to my advantage, in which case I'm willing to trade, to downgrade my privacy in exchange for something else that I value. But on the other hand I do want to track everybody and everything as long as they will let me because I like that information. We're assuming that people are willing to explicitly grant track information to certain parties when it's to their advantage, for example, an Altzheimers patient may want to be tracked by their care givers but not by their snoopy neighbours, and not by someone else's reminding system, like that Kyle guy with the scooter, who certainly shouldn't be able to track this Altzheimers patient just because he uses the same hardware devices. And certainly we don't want to be tracked by bad guys using complex computing hardware, and things like that.

The capability of the hardware for trace in the system interacts with what sort of security design we can do. In particular, the devices have to be small, and convenient to use implies that they have to be pretty low power. To give an example of what that means, the processor is capable of doing a hash of small messages. That's not too bad, it takes about 42 milliseconds, and draws 2 milliamps. So one of these little console batteries which we call we've all seen has the capability of doing about ten million hashes. For communication, transferring 32 bits takes about 28 milliseconds at 25 milliamps; that means there's about a million of those in one battery if it wasn't doing anything else. The bottom line is, one hash uses enough power for transmitting 4 bits, which means in my mind minimising communication is your number one goal, but they need security or privacy schemes for use with this.

So here's the basic idea. We'll transmit an ID periodically, like every two seconds, but we're going to be changing that ID so it cannot be tracked, and without the key you won't be able to track it, but with the key you can. So here might be a transfer of what you would see if you had the proper keys, and without the keys you just see random numbers. So one advantage of this is there's no router communication, a given stack just sits on your shirt and broadcasts an ID.

Matt Blaze: So there's an assumption you're making that this is the only way to identify one of these things is by the bits it's transmitting; there's no other signature that identifies it?

Reply: Yes. There's bit level timing, and very physical things that will identify it which we'd like to get rid of in some future version but I don't know far we can go that way. Certainly we can vary our clock and change the bit timing and things like that, but there may be a hardware signature that people can still attack.

So a typical stack just sits on something and broadcasts its ID every two seconds; it never really does any round trip communication; we can't afford a handshake, or any sort of exchange of information, and also we don't have a whole lot of bits to put in this ID. We'd like a full 64 bit ID, so it's unlikely to have collisions, meaning two random IDs would never be the same. But that costs communication bandwidth, and costs power, and the lower the power the less often you have to change the battery, or the lighter the device. So we did some calculations and chose 32 bits as our initial tradeoff, one collision a year. If you get one false piece of data what are the implications of that? We'll find out I guess.

So the idea is to generate these IDs so that they appear random to someone who doesn't have a key, in particular, you can't correlate one from one time in a role with another from another time in a role, and you can't map these back to your identity. Many of the applications we're considering are time-limited: you may have a school field trip where kids are getting on the bus and visiting some park or something, and there will be chaperones. In that application we'd want everyone to be able to track everyone for that ten hour period, but after the trip is over we don't want a given chaperone to be able to track an arbitrary kid the next day, we want it to be limited to that ten hour period.

We start off by assigning a master key to each SPEC, and then we generate a key for the time period we're in: 9.30 would be year 2004, month 4, day 27, and 15 minute interval number 38 in a day, and we compute a key which grants privilege to track us for an entire year by taking the hash of the master key of the year 2004, and then we want to refine that to a month level we simply hash in the month, and so on.

So to grant someone permission you give out some set of keys which covers the time interval that you want to grant and then on the air you can just transmit the lower 32 bits of this key. This is a five and a half day conference, you might give someone two day keys and six hour keys that cover the interval of time, and then during that time we could use those keys to generate our ID, and then they can listen on the air for IDs and see if they match one that they generated.

OK, so we can see a number of weaknesses in this approach, some of which we know how to enhance our scheme to solve, some that we don't. We've chosen a 15 minute period; this means I can be tracked over 15 minutes. We think that's acceptable, we'll find out, but you can't track that persona back to my identity. However there's environmental data that we don't see any way around which may merge these personas. For instance, if I'm monitoring a room and I see an ID for 15 minutes and then I see another ID for the next 15 minutes, two things could happen: the guy may have left the room and a different person come in, or that person may have just been sitting in the room and changed their ID; well that's much more likely. You could assume that the person just changed their ID and now you can correlate those personas, and then let's say, later on, they used their credit card or something while you're tied into their identity, that would be a bad thing.

In this particular scheme you can also impersonate people, really, it's pretty easy. If you're a firm that has these recognition keys they allow you to generate your IDs and transmit them too, even though that would be considered bad behaviour. A man in the middle could do the same thing, and even a very low level impersonation is possible, you could just listen for an ID on the air, transmit it over a network, and transmit it somewhere else, to a totally different location without really doing any processing on it at all.

Matt Blaze: There's another weakness, because you need a lot of people to have these things. It's the same weakness as with any anonymity scheme if you're the only one using it. So in an environment where there aren't many users you really stand out.

Reply: Yes, we hadn't thought about that but that's obviously true.

Mike Roe: Also, if, say, you have bad codes on machines tracking me and containing my diary, you know, the fact that that particular PC is asking about me, asking in a particular sense about a particular location, is giving away where I am.

Reply: In the simplest form these devices are just devices; they don't actually have any network connection, so they store their diary inside them; there's no network activity other than these IDs going on. Now, when you get home you might plug in a physical cable, create a secure connection to a PC and upload your data that way. In fact there is limited storage, so eventually you need to upload the data to some more permanent store, say once a day. We currently do that with a secure link, but you do it when you're at home; it could easily be secure by using a cable or something. In the future we want to make that upload happen over access points in wireless networks, and we're thinking of using a mix network type thing to try to hide your identity when you're doing that. But we're not near that point really, at this point it's only devices that you are carrying.

Mike Roe: OK, but when I go home and plug my badge into my PC and try to build my diary, it sees that we need these other badges now to represent your IDs, so it's got to get the key to there. If my PC is seen requesting a key for this room, that reveals the fact that I've been in this room, because in order to build my diary up I have to ask it to get the key.

Reply: Right. We're still experimenting with various ways of setting up initial keys. A couple of them again involve only the devices themselves, for instance, if I meet you and you're carrying a device and I want to exchange keys with you we can both put them in our hand and shake them, and maybe shaking them for a certain period of time has changed the life of certain keys. Right now in fact we do plan to do exactly what you've said, and we'll have to consider that. But the goal has to be somehow to enable key exchange to happen in the field because there isn't any infrastructure. That initial key exchange will give you enough information that you can contact the other person's computer later on and exchange a full set of keys.

Richard Clayton: Instead of the device chirpily saying, here I am, which works well with some of the tracking things that you're talking about, conferences and so forth, the more private way of acquiring the daily routine is for the device to sit and just listen where you are, and then at the end of the day the device downloads where it has been into your machine, you can sit and work out what the patterns are and deduce what's going on, so that the next time it can tell you about taking out the laundry on Tuesday or whatever.

Reply: So you're suggesting we receive only but not transmit. That's good, except when you're monitoring around other people that are also doing the same thing, in which case it's a symmetric situation, you need to identify them, and they need to identify you.

Bruce Christianson: In the symmetric case, you can hash a sequence number to the 15 minute slot identifier. When somebody else comes in the room that's the logical time for both badges to jump to transmitting the ID based on the next sequence number. Then it's impossible for an observer to tell which person has walked out of the room subsequently. Similarly if somebody comes and stands next to you, that's the logical time for both devices to change what they're transmitting so then the attacker can't track anybody.

Reply: That's a very interesting idea; we didn't consider that. Bits are expensive but we wouldn't need to transmit more bits.

Matt Blaze: Right now your device is, if I understand correctly, just for transmitting a kind of identifier heartbeat.

Reply: That's the only transmitting they do; they also receive.

Matt Blaze: So it seems that, as technology improves, I'm going to want to do one of two things: I'm either going to want to make these things really small, RFID-size devices, or I'm going to make them more sophisticated about what they're transmitting. Have you considered how your protocols and your architecture would change if these devices were transmitting environmental census stuff, and UPS data, whatever things you might want to do?

Reply: No, we haven't really considered that. We've considered putting sensors on our device and logging that internally, but there's currently no networking going on. That's actually something we probably will want to do.

Matt Blaze: So I've been furiously searching Google and I saw this wonderfully horrific device intended for paranoid parents. It was a very cool looking watch, you know, bright colours, that you lock on the wrist of your child, and it has a little GPS device and it logs its most recent location. You can call it up presumably using one of the cellular phone protocols, UPRS or whatever, and find out where your kid is. The thing that I found particularly both wonderful and horrible about it at the same time was that you could also do these remote unlock functions, and let your kid take it off. I guess there's some password mechanism, or something; I was trying to find the details on this, but you seem to have tapped into a market that is recognised here, the paranoid parent market.

Tuomas Aura: In the UK they use the same kind of devices, these are for prisoners.

Matt Blaze: Yes, I assume that this must be the same company that makes those little ankle bracelets for people on home detention.

Bruce Christianson: It's not enough to know where they are, you want to know who they're with, that's the next step.

Frank Stajano: Wasn't there a Schwarzenegger movie where the thing will explode if you tried to remove it?

Matt Blaze: I guess that's the following step.

Reply: The real problem that I see for this technology is this lifetime diary. Many people in the field of pervasive computing believe that we're moving towards a model where people store their data forever. And that may include photographs, their GPS receiver, then they may throw their coordinates into a database, actually I have met someone in the UK who has been storing his GPS coordinates for three or four years now. He enjoys looking at it on a map, and he also claims that it provides him benefit, he often needs to go to a place that he's been to before but instead of calling up a map of directions he just needs to get close and his GPS thing shows him where he's been in that area before and he can get there.

Frank Stajano: It's my former colleague Alan Jones.

Reply: Yes. So once you've put too much stuff in the diary, a lot of people get very nervous. I had a project like this a few years ago, and the reactions that people had were pretty strong, they felt that the mere existence of this lifetime compilation of data about them was an incredible threat to them. They weren't really worried about people stealing their diary and reading it, and using encryption to track their data, but what they're really worried about is people forcing them to turn over the data, people know that it's there. Right now when people ask you a question socially you can always say, no, I don't have that data. A lot of people will say I don't know, but if people know the data's logged basically you have to say, I refuse to tell you, and while that seems like a subtle difference, it's actually a huge difference and people are worried about it. OK, I have this diary of my life, what if the laws changed and suddenly I want to deny that I went to college or something, because all intellectuals are going to be killed, well, you know, the diary's there, the proof's there, and people will threaten me and I'll have to turn this over. So it would be really nice if the diary contained the same properties that your human memory does, and you could selectively claim to forget things. I don't have any idea how to do that. The real problem is to make people comfortable having this lifetime diary exist, and the only way I can see of doing that is trying to make it be very similar to the way your own memory exists, and give you the same control. A big red button which destroys all your data has its problems too.

George Danezis: It it seems that an obvious solution is to give people the choice not to carry such gizmos with them, but right now as we speak, every cellular phone on the European standard here has location data associated with it that are being retained by phone companies for quite some time. The granularity is not down to that of GPS, but it is down to cells which in the city are pretty small. There have been articles in the press about that; there have been campaigning organisations, and I have seen some circles in which people turn phones off, trade unions, and stuff like that, where people are really aware of the very confrontational environment they might be finding themselves in.

Reply: Yes, that is harder to get at. If you put a gun to my head right now and said, turn over all your GSN coordinates, you know, I just can't do it. But with a tyrannous device, a gun to my head will work, I'll say, OK, my password's boo3; go for it. I think the notion of who has control over access is part of the reason this bothers people.

Chris Mitchell: Your point about social threats seems to be the thing I'm most uncomfortable about. The government getting all of my data is, OK, a possibility, but not very likely, but my wife asking me embarrassing questions about, why weren't you home last Thursday, or, what were you doing three months ago on such and such a day, is much more likely. It may not be anything terribly malicious but you just don't want to reveal that you told a little white lie about why you weren't at home. You were in fact working late, and you were really meant to be home, and you said, oh well my car broke down, which is a far much more acceptable excuse than, I just had a lot of work to do. (Laughter). And you wouldn't be able to lie about these things anymore which is just a complete disaster. So we need faulty technology that we can blame and say, well actually the damn thing didn't work that night.

Mark Lomas: There's a market opportunity here, you sell an alibi service. Here are all the readings that came from my little device. I'll send them to you electronically so you can put them into your device.

Matt Blaze: But the human brain has this property that it neatly, at least with current technology, is the only device that truly only you can read correctly. It's possible that we'll get better biomedical technology and be able to build lie detectors that work well, but it's unlikely for a while. It seems in order to have a life diary that people will trust, you need something with this property that only you can reliably read it, that the owner has the most reliable view of it, and that seems to be something that might be a fundamentally difficult problem.

Reply: Yes I think it really is hard. What I intend to do is to come up with a taxonomy of all possible things that could improve this problem, even if I can't solve it. Maybe you don't log facts in this diary but you log basically retrievable clues for your own memory, and then you seed it with lots of false retrieval clues too which mean nothing to you but which make the diary useless as evidence in a court or something like that. There's a 90% jump, would that even be useful to someone, I don't know.

Virgil Gligor: You know why Hilary Clinton never kept a diary: because it could be subpoenaed. Any of the information that you have, it doesn't matter in what store, about yourself, is subject to a subpoena, so consequently a lot of people don't keep diaries. And in fact, for example, if you look at the President of the United States, the Secret Service has to be aware of where the President is every 15 minutes just like your system, so it's got a built in tracking organisation. Personally I think this is extremely dangerous, at least in the US. Your diary becomes a matter of the court, the court might not be sealing it, it becomes part of the public record in your district, so an extremely dangerous thing for legal reasons, not for technological reasons.

Reply: Yes. I'm bothered by that very much, but whether I work on this or not it seems natural to store all your data and just let it grow, and get backed-up. We might develop a system which forgets things automatically; as time goes on it just abstracts things out and just remembers only the salient facts, whatever that is.

Virgil Gligor: This is even worse than the normal danger because, for example, when you are in court and you are asked what did you on such and such a day, you can always say, without lying, I don't remember. Nobody can prove that you lied when you said I don't remember, but you cannot do this with that system; there's no plausible deniability.

Matt Blaze: On the advice of counsel I do not recall.

Bruce Christianson: But as well as the record of what he was actually doing, in the secret service log, there's also a record of what the President was supposed to be doing. This doesn't say, well actually he's in the bathroom with Monica Lewinski, it says he's attending to correspondence in the Oval Office. What's missing is this cover story that says, actually my car broke down and I was sitting on the hard shoulder for fifteen minutes.

Mike Bond: I think certainly with email people have adapted and developed social techniques so that email is now what we want it to be in terms of plausible deniability. Oh that email is too deep in my archives, I can't dig it out, I can't find the right key, sorry, my spam filter filtered your email out, that's why I didn't reply. And similar things in time will develop the location data to counter the social threats.

Chris Mitchell: I'm not so sure. I believe Microsoft is busy reorganising the file system so that, in the next version of Windows, everything will be all nicely sorted for you. The trouble is I'd keep everything very well organised because I want to be able to find messages, but now that's what I'm worried about.

Mike Bond: So there's a tradeoff. The more you get out of the system, the less plausible deniability.

Bruce Christianson: Mike Roe's thesis[1] about threat/service duality needs to be thought about more, and applied to the next generation of software.

[1] Michael Roe, *Cryptography and Evidence*, PhD thesis, University of Cambridge Computer Laboratory, 1997.

BLIND: A Complete Identity Protection Framework for End-Points

Jukka Ylitalo and Pekka Nikander

Ericsson Research NomadicLab,
02420 Jorvas, Finland
{Jukka.Ylitalo, Pekka.Nikander}@nomadiclab.com

Abstract. In this paper, we present a security framework that provides identity protection against active and passive attacks for end-points. The framework is based on a two-round-trip authenticated Diffie-Hellman key exchange protocol that identifies the end-points to each other and creates a security association between the peers. The protocol hides the public key based identifiers from attackers and eavesdroppers by blinding the identifiers. We complete the identity protection by offering location privacy with forwarding agents. To our knowledge, our privacy enhanced protocol is the first denial-of-service resistant two-round-trip key exchange protocol that offers identity protection for both communicating peers.

1 Introduction

The current structure of the Internet is very much the same as if a person's name would be defined by his or her current location. Let's say that a spy moves from Moonlight Street to Shadow Street. Since identification is bound to locations, his *identifier* changes due to movements, but his actual *identity* stays the same. M, who used to know his spy, James Blind[1], as Mr. Ten Moonlight Street, does not recognize him anymore. Now being identified as Mr. Twelve Shadow Street, he must convince M, in one way or another, about his actual identity, i.e, he still is the same person as he was before. Since there is no equivalent of the human face in the current Internet, convincing M is not a particularly easy task.

Several mobility protocols have solved the naming problem by using *home addresses*[2]. Each end-point is assigned a static address, its home address, which is used to identify the end-point independent of its location. This solves the basic naming problem; our spy is still known to his boss M as Mr. Ten Moonlight Street. However, even in the real world, the actual location is needed for reachability. It would be really hard for our spy, usually living in a hotel at Moonlight Street and currently walking at 12 Shadow Street, to prove to M that his name is Mr. Ten Moonlight Street c/o Twelve Shadow Street, without assurances. By

[1] The name corresponds the Fully Qualified Domain Name (FQDN).

[2] In a way, the naming convention resembles human naming conventions in the medieval times, when people were named after their home town, e.g., William of Ockham.

sending a challenge message to a given location and waiting for a response, M can check if there is a spy named with the given location[3].

Unfortunately, the existing naming convention causes many privacy problems that are related to location names. The presence of location information in a message reveals the location of its recipient and alledged sender[2]. At the same time, location names are used to identify end-point, and thereby allow an end-point's action to be traced. Our spy would definitely not want his identity nor location to be revealed to outsiders[4].

Our framework uses a cryptographic name space, based on the use of public keys, that separates the location and end-point identifier roles of location names. Continuing our analogy, the approach would bring our spy a genuine name, one cryptographically bound to his real identity. In a way, our Mr. Ten Moonlight Street would no longer be named after his location, but by a self-signed photograph of his face. However, the new naming convention causes privacy problems that are no longer related to location names, but the use of public keys[4][5]. In this paper, we focus on solving that problem[5].

We introduce a privacy enhanced authenticated Diffie-Hellman protocol. The protocol provides complete identity protection, requiring an initiating party only to possess a hash of the full public key of its peer at the start of the protocol run. Basically, the protocol scrambles the photograph of our spy in a way that only his old acquintances are able to recognize him. The protocol protects both parties from *passive and active Man-in-the-Middle and polling attacks*, unless the attacker is able to find the public key with the help of location information. Therefore, we complete our protocol by presenting forwarding agents that provide location privacy for end-points.

The rest of this paper is organized as follows. Section 2 defines the framework terminology. In Section 3, we define the privacy problems that we are addressing in this paper. This description is followed by a detailed problem statement in Section 4. In Section 5, we present the privacy protecting key exchange protocol. The forwarding agent and location privacy is discussed in Sections 6 and 7. Section 8 concludes the paper.

2 Framework Terminology

In our framework, a logical *end-point* is a participant in an end-to-end communication[6]. Each end-point generates a public key pair that works as a global

[3] In the Mobile IPv6[1] terminology, this is called Return Routability (RR) test.

[4] The current IP mobility practices[3][1] reveal both the end-point's identity (home address) and location (care-of-address) to outsiders. In the case of Mobile IPv6 route optimization, the identity and location also to the servers and peers the end-point is communicating with.

[5] The location and identity privacy problems can be solved by using privacy proxies, such as one provided by Zero Knowledge systems. Such usage necessitates that the proxy is trusted to keep the user's identity and location secret. However, using generic privacy proxies falls beyond the scope of this paper.

End-point Identifier (EID) [6]. The owner of the private key owns a specific *identity*, while the corresponding public key works as an identifier for the end-point. This defines the naming trust relationship between an identity and an identifier without name certificates. A cryptographic hash of the public key (EID) is called a *fingerprint*. The fingerprint represents a consistent format for protocols, independent of the whatever public key technology is used.

A *host* is an environment for an end-point(s). The end-points use an Application Programming Interface (API) to communicate with other end-points. *Transport Layer Identifiers (TLIs)* represent the EIDs in the communication API and at the transport layer. The local TLI presentation depends on the instantiation of our framework.

A *location name* (i.e. a locator) defines the topological point-of-attachment of an end-point in the network. A multi-homed host offers several topological point-of-attachments for end-points. The introduction of EIDs clarifies the role of locators. For example, IP addresses become pure topological labels, naming locations in the Internet, while the EID identify an end-point. The location names are bound dynamically to EIDs. Furthermore, a *connection* is a communication link between two end-points. It is bound to EIDs, instead of location names. An end-point may change its location without breaking connections.

The *key exchange protocol* uses the EIDs for mutual authentication and to generate end-to-end security associations (SAs) between two end-points. The end-point that initiates a protocol run is called the *initiator* and the responding end-point is called the *responder*. The security associations are used to protect the connections. In addition, the SAs are also used with *mobility management protocol* (out of the paper's scope). The mobility management protocol is used to update the binding, at middle boxes and peer nodes, between EIDs and location names. The key exchange must take place before end-points can update their address bindings.

Our framework contains a logical protocol layer between the OSI transport and networking layers. In the current Internet TCP/IP architecture, the processes are bound to transport layer sockets, and the sockets are identified using IP addresses and ports. In our framework the connections are no longer named with locators but with EIDs. The new abstract end-point identifier layer translates the TLIs to locators. The set of associated locators can belong the different families. This binding, between EIDs and locators, is simultaneously dynamic and one-to-many, providing for mobility and multi-homing, respectively. In the rest of this paper, we discuss the framework from the Internet architecture point of view.

3 Privacy in IP Based Communications

In the current Internet, IP addresses are used to identify end-points and name their topological locations. IP addresses together with public keys, used in the key exchange protocols, reveal directly the location and identity of an end-point.

[6] For example, Cryptographic Generated Address (CGA) [7], Host Identity Protocol (HIP) HI[8].

3.1 Public Keys as a Privacy Problem

Using public keys as identifiers is a source of privacy problems. Firstly, a public key directly and strongly identifies an end-point[9]. Secondly, if an end-point has just a single public key, using it repeatedly, it is fairly easy to link together all the transactions made by the end-point. However, an end-point may have several public keys instead of just one. Some of the public keys can be used as more permanent identifiers, allowing the end-point to be recognized. At the same time, some other keys can be anonymous, being temporary and periodically replaced.

It is important to make a difference between anonymity and identity protection. If an end-point uses an unencrypted identifier, it deliberately reveals its identity to outsiders, breaking identity protection. On the other hand, one can openly use an anonymous public key and remain anonymous.

In identity protection, one of the goals is to prevent malicious nodes from tracing any identity. Therefore, if we are able to offer complete identity protection for any type of identities, public or anonymous, the role of anonymous identities is changed. They are no longer needed to protect from man-in-the-middle or eavesdropping attackers but from legitimate peers.

3.2 IP Addresses as a Privacy Problem

To keep the size of routing tables small enough, the Internet addresses are distributed hierarchically [10]. That is, address prefixes and network topology are kept in rough synchrony, thereby allowing the routers to store less information than they otherwise would be forced to. At the same time, this practice binds the IP addresses to the topological locations in the network.

While the primary purpose of IP addresses is to make packet delivery possible, they are also used directly by the transport layer protocols, including TCP, UDP, and SCTP. In all of these, IP addresses are used for naming the transport layer sockets. That is, each communication context is named by IP addresses together with protocol and port numbers. This necessitates using static IP addresses, or connections will break.

As long as a user is using a static IP address, it is possible to link her actions together and form a profile about her. With some little help, it is often even possible to link this profile to her real life identity[4][11][7].

The address tracking and profiling problem is slightly mitigated by the current practice of using dynamic IP addresses and especially Network Address Translation (NAT)[12]. However, with IPv6, the privacy situation is likely to detoriate since NAT is less likely to be used. Furthermore, even if NAT is used in IPv6, the translation will typically be one-to-one, without multiplexing several hosts behind a single address.

[7] Other techniques for user tracking, such as HTTP cookies, fall beyond the scope of this paper.

3.3 Our Position

The identity and location privacy problems are related in the sense that identity privacy does not protect a node against all kind of threats. In some cases, the location may reveal confidential information to a malicious person. E.g. someone is in the bank vault or someone is alone in the park. In such case, the attacker does not care of the actual identity of the end-point. On the other hand, IP addresses that are stored in the DNS together with public keys give information about end-points identity. Therefore, an end-point having a DNS record must protect its location together with its identity.

In our framework, cryptographic end-point identifiers are used to identify the communication end-points. They have no permanent relationship with locations or IP addresses. IP addresses, on the other hand, are used to identify only the topological locations, not end-points. We will show how it is possible to fully hide the used cryptographic identifiers from outsiders and integrate them with Network Address Translation (NAT). As a consequence we obtain a security framework where an end-point can control both its identity and location privacy.

4 Problem Statement

As pointed out by Molina-Jimenez and Marshall[11], if it becomes possible to use random IP and link layer addresses, the problems related to IP address tracking more or less disappear. To be more precise, if an IP address no longer acts as an end-point identifier (see Section2), it is possible to take advantage of address translation at end-hosts and middle boxes (discussed later). As a consequence, the fact that it remains possible to keep track of IP addresses and find out their geographical location doesn't matter that much any more. The focus is moved to the end-point identifiers, public keys and fingerprints, that must be protected.

For the purposes of the rest of this paper, we define *end-to-end identity privacy* to denote the situation where any given two end-points are able to communicate, using their public identities, without having to disclose the identities to outsiders.

4.1 Identity Protection

The identity protection problem arises in all two-round-trip Diffie-Hellman key-exchange protocols that use separate public-keys for mutual authentication.[8] To obtain DoS protection, the responder must not generate the Diffie-Hellman shared secret before the initiator. Therefore, the responder must defer key generation until it has received two messages from the initiator. Consequently, if the responder sends its public key in the second protocol message, the public key must be transmitted in clear. Thus, the responder's public key can be safely transmitted only in the last (fourth) protocol message. As a result, the initiator can completely authenticate the responder only once it has received the last message.

[8] Protocols based on public-key encryption (e.g.[15]) are vulnerable to CPU related DoS attacks and are therefore out of the paper's scope.

The initiator, in turn, is able to generate the shared secret after receiving the second protocol message, i.e., the first message sent by the responder. Hence, the third protocol message may contain the initiator's public key in an encrypted form. In any typical Diffie-Hellman protocol, an active attacker can easily find out the initiator's public key, because the second message cannot be fully authenticated.

One way to solve the problem is to enhance the session key generation with a secret that is initially known only to the authentic end-points. Incidentally, the same method also protects the responder from active attacks. In other words, even an active attacker is not able to find out the responder's identity by sending spoofed first messages, i.e., by launching polling attacks [13]. In other words, all identifiers, fingerprints included, must be hidden in a cryptographically strong way to obtain identity privacy.

4.2 Privacy Protection vs. Denial-of-Service Protection

The design of any public key based key exchange protocol will eventually face a trade-off between denial-of-service and identity protection properties. For example, Bellovin et.al. [5] stated: *"We remark that it is essentially impossible, under current technology assumptions, to have a two-round-trip protocol that provides DoS protection for the responder, passive identity protection for both parties, and active identity protection for the initiator."*

In this paper, we present how the introduction of a new cryptographic name space can help to mitigate the problem. The new cryptographic end-point identifiers, together with the techniques discussed in this paper, make it possible to obtain complete privacy protection from both passive and active attacks for both end-points, in a two-round-trip DoS-resistant protocol (See Section 5.2).

To be more precise, if an attacker is able to learn the public key of the participants through some external means, e.g., if the public key can be resolved from reverse DNS, it remains impossible to provide active identity protection. However, whenever the attacker cannot use the IP addresses to find out the public keys, the public keys remain private even from active attackers.

4.3 Three Scenarios

Identity privacy can be divided into three scenarios, which differ on the a priori knowledge of the participating parties. In the first scenario, both of the parties have a priori knowledge about each other's identifiers, e.g. fingerprints, but possibly not the actual public keys themselves. For example, the parties have been in touch with each other earlier and remember the fingerprints but not the public keys. In the second scenario, the initiator knows the the responder's identity, but the responder may have no knowledge about the initiator's identity. In the third scenario, neither of the parties know each other's identity beforehand. In this case, the parties may use temporary identities, i.e., short term public key pairs, or public identities, e.g., published in a directory service.[9]

[9] The scenario where a responder would know who is going to contact it, but the parties taking contact do not know the identity of the responder, is not possible in practice.

In the first and second scenarios, we are able to protect both identifiers from both passive and active attacks. Typically, the initiator learns the responder's host identity (public key or fingerprint) from a directory service. Additionally, in the first scenario, the responder has some configuration information that contains the fingerprints of potential initiating parties as well. However, neither of the parties may lack access to the full public keys, requiring them to learn the actual public key during the handshake.

In the second scenario, the initiator has a priori knowledge about the responder's identity, but the responder is oblivious of the initiator's identity. However, even in that case, the initiator's identity may be important *afterwards* (e.g., for auditing purposes), and therefore, we want to support the case where the initiator is able to use its public long-term identity without revealing it to anyone else but the right responder.

5 Key Exchange Protocol Supporting Identity Protection

In this section, we present a two-round-trip authenticated Diffie-Hellman Key Exchange Protocol that protects the initiator's and responder's identity. Our solution is based on the idea of *blinding* the cryptographic hash (fingerprint) of the public key used in the exchange. That is, instead of directly using the hashes of the public keys to index the session, the parties create scrambled versions of the fingerprints and use each scrambled value only during one protocol run. This makes it impossible to correlate independent protocol runs.

Our blinding method is based on the assumption that at least the initiator has a priori knowledge about the actual or potential fingerprints of the responder (see Section 4.3). The parties must know their own fingerprints, of course.

Before starting our protocol, the initiator computes *blinded fingerprints* for both end-points, using a *nonce* to hide the actual *plain fingerprint*. The blinded fingerprints are generated using the following formula:

$$fingerprint_I^{Blinded} = SHA1(nonce_I || fingerprint_I^{Plain})$$
$$fingerprint_R^{Blinded} = SHA1(nonce_I || fingerprint_R^{Plain})$$

The initiator generates a fresh nonce for every base exchange. The blinded fingerprints are changed for every key exchange, which makes it hard to trace the usage of plain fingerprints.

In order to derive a plain fingerprint from a blinded one and the nonce, the plain fingerprint must be already known. That is, if a party that knows the nonce has a list of possible plain fingerprints, it can test these, one by one, to see if any of them matches with the blinded fingerprint. This allows the initiator to use the same plain fingerprints all the time between a given pair of end-points, while the blinded fingerprints, used in the key exchange packets on the wire, will be stored only for the life time of one connection. On the other hand, if only the nonce and the blinded fingerprint are known, it remains computationally infeasible to find the plain fingerprint.

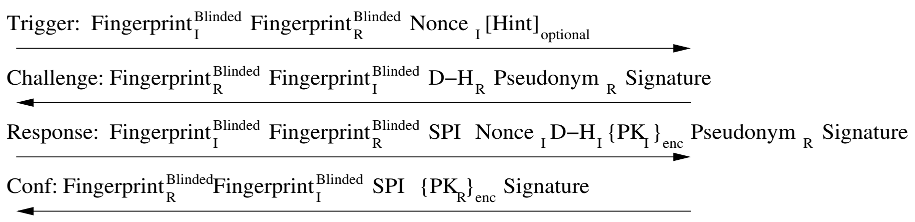
Trigger: Fingerprint$_I^{Blinded}$ Fingerprint$_R^{Blinded}$ Nonce $_I$ [Hint]$_{optional}$
Challenge: Fingerprint$_R^{Blinded}$ Fingerprint$_I^{Blinded}$ D–H$_R$ Pseudonym $_R$ Signature
Response: Fingerprint$_I^{Blinded}$ Fingerprint$_R^{Blinded}$ SPI Nonce $_I$ D–H$_I$ {PK$_I$}$_{enc}$ Pseudonym $_R$ Signature
Conf: Fingerprint$_R^{Blinded}$Fingerprint$_I^{Blinded}$ SPI {PK$_R$}$_{enc}$ Signature

Fig. 1. The key exchange

5.1 Sending Trigger Packet

The first packet contains the nonce and corresponding blinded fingerprints. The packet structure is shown in Figure 1. In addition to the nonce, the packet may also contain an optional *hint* that makes it easier for the responder to obtain a correct plain responder fingerprint than by trying out them all. The hint discloses k lowest bits of the plain responder fingerprint.

The idea behind our approach is that the responder finds out its plain fingerprint by repeatedly attempting to generate blinded fingerprints from the plain fingerprints it knows, using the given nonce. A typical responder has only a few public keys at most, and it is able to easily find its own plain fingerprint even without a hint. Thus, in most cases, the initiator does not have to include the hint, and therefore the responder obtains complete identity privacy.

Basically, the hint protects the responder from DoS attacks. A busy server (e.g., a proxy) with thousands of own fingerprints may drop the 1st packet if it does not contain a large enough hint. However, the k value must be so small that the hint does not statistically give enough information to bind it to the plain fingerprint[10].

The responder does not need to know the actual identifier of the initiator before it receives the 2nd packet from the initiator. Thus, there is no need to use a hint for the plain initiator fingerprint in the trigger packet.

5.2 Sending Challenge Packet

Upon receiving a trigger packet, the responder sends a temporary pseudonym and starts the Diffie-Hellman exchange in the challenge packet (see Figure 1). The pseudonym is implicitly bound to responder's public key, because the challenge packets are created and signed beforehand. To select a suitable challenge packet, the responder needs to be able to find out the correct plain fingerprint from the blinded one. The responder does not send the public key in the challenge packet, but signs the packet with its private key.

[10] The framework is based on that assumption that most of the end-points in the Internet will store their own public key identifiers into the directory services. Therefore, once there is any substantial number of public key owners, it is statistically really hard to use a small size hint to find the right fingerprint among all of the worlds fingerprints.

Later when the initiator sends a response packet, it will contain the responder's pseudonym. The pseudonym acts as an index for the responder's public key. In this way, the responder does not need to resolve its plain fingerprint twice. The pseudonym is a local random number. It does not give any information about the responder's actual identity.

It is possible to include a challenge puzzle (e.g. HIP[8]) to the challenge packet. After solving the puzzle, the initiator sends the result back to the responder in the third message. The puzzle protects the responder from DoS attacks.

5.3 Forming the Session Keys

Since the challenge packet does not contain the responder's public key, the initiator may not be able to verify at this stage that the packet is signed by the correct responder[11]. In this case, the initiator defers the verification of the signature until it receives the 2nd packet from the responder. If the initiator finds out later that the signature in the challenge packet was invalid, the initiator's identity is still not revealed to passive or active attackers, as we will shortly see.

Basically, it is possible that a malicious node sends a spoofed challenge message to the initiator in trying to disclose the initiator's identity. The initiator may not have the public key of the responder at this point, therefore being unable to verify the signature in the challenge packet, but it knows the responder's correct plain fingerprint. The malicious node, in turn, does not know either of the plain fingerprints. To take advantage of this, the initiator generates the required key material using its own blinded fingerprint and the responder's plain fingerprint, known only to the correct responder:

$$KEY_1 = SHA1(KEY_{DH}||fingerprint_I^{Blinded}||fingerprint_R^{Plain}||1)$$
$$KEY_n = SHA1(KEY_{DH}||KEY_{n-1}||n)$$
$$KEY_{MATERIAL} = KEY_1||...||KEY_n$$

While any eavesdropper learns the blinded fingerprint, only the correct responder knows the responder's plain fingerprint. Therefore, a malicious node is not able to form the session key even if it has gained access to the Diffie-Hellman key through an active attack.

5.4 Sending and Receiving Response Packet

The initiator uses the key material to encrypt its public key in the response packet (see Figure 1). The response packet includes the same nonce that was sent in the trigger packet and the pseudonym that was sent in the challenge packet.

When the responder receives the response packet, it uses the pseudonym to find out its own public key. The responder generates the key-material in the same way the initiator did earlier, and uses the key-material to decrypt the initiator's public key.

[11] The initiator may only have the responder's fingerprint, and not the public key itself.

As an additional verification step, the responder computes the initiator's plain fingerprint from the decrypted public key and verifies that the blinded fingerprint is a correct one.

5.5 Sending Confirmation Packet

At the final stage, the responder sends its encrypted public key in the confirmation packet to the initiator (see Figure 1). The initiator has a state related to the blinded fingerprints, and therefore it is able to easily look up related plain fingerprints and key-material. The initiator decrypts the public key and verifies that the expected responder's plain fingerprint corresponds to the received public key. Finally, the initiator verifies the signatures of the challenge and confirmation packets.

In addition to the presented first and second scenarios, it is possible that neither of the end-points know their peer's public key or fingerprint beforehand. The initiator knows an IP address of the responder, but nothing else. This kind of scenario is vulnerable to certain man-in-the-middle attacks, since there is no security relevant information at the start of the protocol run.

5.6 Security Analysis

The first and second scenarios correspond to one-way and two-way authenticated Diffie-Hellman key exchanges. The parties do not need to know the full public keys of their peers before the protocol starts; hashes or other one-way derivatives of the public keys are enough. This is different from earlier identity protecting protocols that require the parties to know the full public keys before the protocol starts.

The blinded fingerprint protects the responder from the polling attack[13], where an attacker impersonates an initiator, since only an initiator knowing the identity of the responder may compute the blinded fingerprint. The only efficient way for an attacker to learn the responder's identity is to find out the mapping between the IP address and public key. If an attacker is not able to map IP addresses to identifiers the end-points obtain full identity privacy. In Section 7, we present a mechanism that hides the end-point's actual location; offering location privacy.

In the first and second scenarios, a Man-in-the-Middle (MitM) knowing the Identity of the actual responder can solve the responder's plain fingerprint from the blinded one. However, without such a priori knowledge, an attacker must guess the correct fingerprint, which is statistically very difficult. Therefore, the only way for an active MitM attacker to succeed is to map the recipient IP address, in the key exchange packets sent by the initiator, to the identity of the responder. Since the protocol protects the responder from polling attacks, such identity knowledge must be obtained in another way, e.g. from DNS. Therefore, the DNS should not contain reverse look-up table for identifiers.

In the second and third scenarios, the question is about partially or completely *opportunistic* identification, where the initiator or both of the end-points learn their peer's identity during the protocol run. That is, one or both of the parties do not have any a priori knowledge about the identity of their peers. Naturally, the third scenario, where both parties are initially oblivious about their peer's identity, is vulnerable to active attacks.

6 EID Enabled Network Address Translation

An EID enabled NAT device translates IP addresses, acting as a router for end-point identifiers. The NAT device associates the connection state with the EIDs. It is able to multiplex several connections on a single address based on the end-point identifiers.

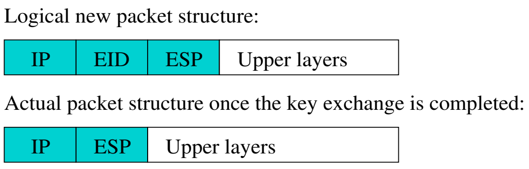
Logical new packet structure:
IP
EID
ESP
Upper layers
Actual packet structure once the key exchange is completed:
IP
ESP
Upper layers

Fig. 2. The packet structures

In a preferable instantiation of our framework the existing IP or IPSec packet structures are not changed. As a consequence, the EIDs are not present in the regular traffic between two hosts. They appear only in the key exchange messages. However, at the logical level, the end-point identifier name space imposes changes to the logical packet structure. That is, each packet must logically include both the end-point identifiers and IP addresses of the sender and recipient. If IPSec is used, the Security Parameter Index (SPI) values can be used as *indices for end-point identifiers* [12] , resulting in packets that are syntactically similar to those used today. This is illustrated in Figure 2.

A NAT device learns the SPIs together with the EIDs during a key exchange. The SPIs and the IP addresses are used together to act as shorthands for the EIDs. The NAT device needs the SPIs to properly demultiplex any packets arriving to a shared IP address. Whenever the packets are integrity protected with ESP, the recipient is always able to verify that a received packet was sent by the peer no matter what the source and destination addresses are.

Basically, SPI multiplexed NAT (SPINAT) works in the same way as port multiplexed NAT (NAPT). If a specific SPI is already in use, the SPINAT device replaces the value with a new one. When there are several NAT devices on the path, all of them may not have the initially assigned SPI value available to be used with the connection. Therefore, it may become necessary to change the SPIs several times along the way. Thus, the SPI values in key exchange messages cannot be encrypted or included into the signature. The SPINAT technique does not require any tunneling headers.

[12] It is also possible to use, e.g., flowid for the same purpose with IPv6.

7 Location Privacy with Forwarding Agents

A Forwarding Agent (FA)[14] is functionally similar to an EID enabled NAT device. In fact, functionally the two different devices are equal for all practical purposes. However, a FA is not necessarily located between two different addressing domains (such as the public Internet and a privately addressed intranet), but it may just be conveniently located almost anywhere, even with a single interface[13].

The forwarding agents in our framework, take advantage of the end-point's multi-homing properties. A multi-homed end-point having several IP addresses is considered to be present at several locations at the same time. In functional terms, the end-point is able to receive packets sent to several different IP addresses. On the other hand, if an end-point has leased an IP address from a forwarding agent, the end-point is also able to receive packets sent to the forwarded address. Thus, in a sense, the packet forwarding agent can be considered to dynamically provide a virtual interface to the end-point, and that the end-point is *virtually present* at the location of the forwarding agent. The situation is illustrated in Figure 3.

It must be noted that a forwarding agent always translates only one IP address, and never both the source and destination addresses. When a forwarding agent is passing a packet to an end-point that has leased a virtual interface, only the destination address is changed. What was previously the virtual address now becomes the real address. For packets sent by the leasing end-point, the situation is reversed.

The FA assigns an IP address from a pool of addresses for each virtual interface lease. In the IPv4 world the shortage of public IP addresses forces the FAs to overload IP addresses, just like SPINAT devices do. An end-point must negotiate a key exchange with its peer via the forwarding agent to obtain location privacy. Therefore, an end-point must make a bi-directional lease. The lease consists of following information:

$$(fingerprint_{dst}^{Blinded}, addr_{dst}, fingerprint_{src}^{Blinded}, addr_{src}, lifetime)$$

The forwarding agent provides location privacy by hiding the real location of the node. The peers are able to see only the virtual address, not the real address(es) of the end-point.

7.1 Complete Privacy

From the privacy point of view there are trusted and untrustworthy forwarding agents. Untrustworthy FAs may allow anonymous leases, while trusted FAs may require the initiator to identify itself during the lease. The responder typically has a trust relationship with its long-lived forwarding agent. In such a case,

[13] The FA discovery protocol is out of the paper's scope. However, the discovery can be based on anycast addresses or DNS queries.

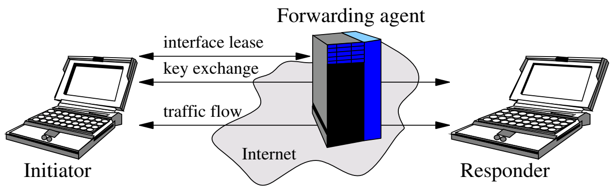

Fig. 3. The virtual interface model

the forwarding agent trusts the responder not to establish extra states and the responder trusts forwarding agent not to disclose the responder's identity.

The initiator has to acquire a virtual interface from a forwarding agent to obtain complete identity privacy. The initiator may negotiate a lease with an untrustworthy forwarding agent using some temporary identifier during the lease. The lease contains a pair of blinded fingerprints that are used in the communication with the actual responder. The communication between the initiator and the responder goes via the forwarding agent. The identity privacy protects the end-points against Man-in-the-Middle attackers including the forwarding agent. Furthermore, the responder or a man-in-the-middle, on the responder's side of the forwarding agent, are not able to learn the topological location of the initiator.

Basically, both peers may obtain complete identity privacy without knowing the other's topological location. In a typical case, the responder leases a virtual interface, and publishes the virtual address in the DNS. The lease may have long lifetime, e.g., months. Rendezvous servers in different mobility architectures have this kind of forwarding agent role.

An initiator can make a lease from a suitable forwarding agent and hide the actual destination of the responder. As a result, the initiator obtains identity protection against active attacks, i.e., impersonating the responder.

8 Conclusions

In this paper, we have presented a framework offering identity and location privacy for end-points. The main focus has been on making the guessing of identities as difficult as possible by blinding the public keys and hiding the topological locations of the end-points using forwarding agents. The presented key exchange protocol provides passive and active identity protection for both peers unless an attacker is able to find out the public keys, utilizing location information. Thus, if the attacker can learn the identity with the help of IP addresses, no protocol can provide identity protection.

Our protocol does not require the parties to initially know the public keys of each other. It is sufficient that only the initiator knows a hash of the public key of its peer. The actual public keys are transmitted as a part of the protocol, but in such a way that even an active attacker is not able to learn them. To our knowledge, this is the first time this level of combined end-to-end security and privacy has been achieved in IP networks.

Acknowledgments

We would like to thank Gonzalo Camarillo, Petri Jokela, Tony Jokikyyny, Miika Komu, Goran Schultz, Vesa Lehtovirta, Vesa Torvinen, and Jari Arkko, for their valuable comments.

References

1. Johnson, D., Perkins, C., Arkko, J.: Mobility Support in IPv6. Internet Draft, work in progress. June, 2003.
2. Lamm, S.E., Reed, D.A., Scullin, W.H.: Real-time geographic visualization of world wide web traffic. World Wide Web Journal, The Web After Five Years, Summer 1996.
3. Perkins, C.: IP Mobility Support. RFC 2002. 1996.
4. Escudero-Pascual, A.: Privacy in the next generation internet: Data protection in the context of the european union policy. Ph.D. dissertation, Royal Institute of Technology, Stockholm, Dec. 2002. [Online]. Available: http://www.imit.kth.se/˜ aep/PhD/
5. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ionnadis, J., Keromytis, A., Reingold, O.: Efficient, dos-resistant, secure key exchange for internet protocols. ACM Computer Communications Review, Nov. 2002.
6. Saltzer, J., Reed, D., Clark, D.: End-To-End Arguments in System Design. ACM Transactions on Computer Systems, vol.2, Nov. 1984.
7. Shea, G., Roe, M.: Child-proof Authentication for MIPv6 (CAM). ACM Computer Communications Review, vol.31, Apr. 2001.
8. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Host Identity Protocol. Internet Draft, work in progress. Feb. 2004.
9. Nikander, P.: An architecture for authorization and delegation in distributed object-oriented agent systems. Ph.D. dissertation, Helsinki University of Technology, Helsinki, Mar. 1999. [Online]. Available: http://www.tml.hut.fi/˜ pnr/publications/PhDThesis.pdf.
10. Fuller, V., Li, T., Yu, J., Varadhan, K.: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. RFC 1519. Sept. 1993.
11. Molina-Jimenez, C., Marshall, L.: True anonymity without mixes. In Proc. IEEE Workshop on Internet Applications '01, San Jose, CA, July, 2001.
12. Srisuresh, P., Holdrege, M.: IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663. 1999.
13. Perlman. R.: Understanding IKEv2: Tutorial, and rationale for decisions. Internet Draft, work in progress. Feb. 2003.
14. Nikander, P., Ylitalo, J., Wall, J.: Integrating Security, Mobility, and Multi-Homing in a HIP Way. In Proc. Network and Distributed Systems Security Symposium, NDSS'03, San Diego, CA, Feb. 2003.
15. Abadi, M.: Private Authentication. In Proc. 2002 Workshop on Privacy Enhancing Technologies, Springer-Verlag, pp. 27–40. 2003.

BLIND: A Complete Identity Protection Framework for End-Points (Transcript of Discussion)

Jukka Ylitalo

Ericsson Research Nomadiclab, Finland

Matt Blaze: In the JFK paper[1] we made the kind of throw away assertion, a very trivial assertion, that the reason that you can't have identity protection for both parties in the presence of an active attack is that someone has to reveal their identity first, unless you know something about each other that's secret in advance. Assuming certificates and so on is the means for identification, someone has to identify first, you can't say, I'll only tell you who I am if you tell me who you are.

Now that was an assertion of a trivial fact rather than a proof. I'm wondering if you believe, under the assumptions we had, that that was true?

Reply: It depends how you interpret your statement, but that is more or less where I was coming from; I wanted to hear your argument about this.

Matt Blaze: I'll tell you mine only if you tell me yours first. [Laughter]

Reply: OK. If the initiator is the attacker and doesn't have *a priori* knowledge of the responder's public key, the responder just replies with this packet, it is decrypted using the key material that was generated with the plain fingerprint of the responder, so the initiator cannot disclose the responder's public key. On the other hand if the responder is the attacker, and the initiator just sends the first packet, this doesn't disclose anything, but it uses the same way to create the key material using the responder's plain fingerprint that is known only to the responder and to the initiator. So in this way the responder was the attacker but he doesn't know who is the correct responder or who the initiator is going to be.

Matt Blaze: Right, but then I no longer know what my opinion is in that case.

Reply: Anyway, as I said, the focus changes from this protocol to how we can find out who owns this blinded fingerprint, and how can we solve the puzzle.

Matt Blaze: Yes, so you've moved the problem.

Reply: There are weaknesses of course, but we can make the guessing so very hard it becomes very difficult.

Pasi Eronen: So basically it seems that an attacker who guesses who these parties might be can use a protocol to verify the guess, but if his guess is wrong then he doesn't get anything?

[1] William Aiello et al, Efficient DOS-Resistant, Secure Key Exchange for Internet Protocols, LNCS 2467, pp27-448.

Reply: Right, yes.

Matt Blaze: So if the population is small?

Reply: Yes, if the population is small, there are difficulties of course, but this was intended only to be used with a large population.

Privacy Is Linking Permission to Purpose[*]

Fabio Massacci and Nicola Zannone

Department of Information and Communication Technology
University of Trento - Italy
{massacci, zannone}@dit.unitn.it

Abstract. The last years have seen a peak in privacy related research. The focus has been mostly on how to protect the individual from being tracked, with plenty of anonymizing solutions.

We advocate another model that is closer to the "physical" world: we consider our privacy respected when our personal data is used for the purpose for which we gave it in the first place.

Essentially, in any distributed authorization protocol, credentials should mention their purpose beside their powers. For this information to be meaningful we should link it to the functional requirements of the original application.

We sketch how one can modify a requirement engineering methodology to incorporate security concerns so that we explicitly trace back the high-level goals for which a functionality has been delegated by a (human or software) agent to another one. Then one could be directly derive purpose-based trust management solutions from the requirements.

1 Privacy Protection and Cleaning Ladies

Consumer privacy[1] is a growing concern in the marketplace. While the concerns are most prominent for e-commerce, the privacy concerns for traditional transactions are increasing as well. Some enterprises are aware of these problems and of the market share they might loose if they do not implement proper privacy practices. As a consequence enterprises publish privacy statements that promise fair information practices[2].

There are a number of risks to an enterprise if it does not manage its personally identifiable information correctly. Recently, many countries have promulgated a new privacy legislation. Most of these laws incorporate rules governing collection, use, store and distribution of personally identifiable informa-

[*] This work has been partially funded by the IST programme of the EU Commission, FET under the IST-2001-37004 WASP project and by the FIRB programme of MIUR under the RBNE0195K5 ASTRO Project. We would like to thank P. Giorgini, M. Pistore, and J. Mylopoulos for many useful discussions on Tropos.

[1] Privacy is the right of individuals to determine for themselves when, how, and to what extent information about them is communicated to others (Alan Westin).

[2] The OECD defined a set of privacy principles in 1980. The document OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data is considered to contain the core requirements for managing privacy today.

tion. It is up to an organization to ensure that data processing operations respect any legislative requirements. If an enterprise breaches trust, that is, it uses data for other purposes, then it can be sued. Further, business relationships are built on trust. Organizations that demonstrate good privacy practices can build trust. Organizations with no privacy practices will turn away customers.

The last years have seen a substantial increase in privacy-related security research[3]: we have a number of dedicated workshops (e.g. PET, WPES), a number of European and US projects and standard initiatives such as P3P. In the realm of cryptography the work on anonymizing networks and transactions has a long history since Chaum's first proposals. One could even say that trust negotiation's birth itself [11] was spurred by privacy concerns of non-disclosure of sensitive credentials to unknown strangers.

However, the current set of solutions is slight unsatisfactory: we must either struggle on keeping privacy by a complex cryptographic infrastructure or be involved in complex protocols for trust negotiation.

Not long ago, at a Cambridge seminar, Robert Morris, formerly from the US NSA, spoke about the cryptographic role of the cleaning ladies, inviting people to consider the actual perimeter of one's secure systems. We would like to use the example with a slightly different view.

Indeed, from the standpoint of access control and privacy protection, the problem of the cleaning ladies admits no solution. The cleaning lady must have access to the room and even when the room is not occupied to avoid disturbing people while at work. There is no way to prevent her from looking at the papers laying on the desk, even searching through them, doing any possible action on the unattended desktop and leaving unnoticed.

Yet, we do allow cleaning ladies and we are perfectly happy with many others similar problems. One possible explanation may be in the the actual economics of trust in cleaning ladies wrt the risk of privacy losses [12]. However, we believe that the solutions lies in the implicit permission model that we use in the physical world, but so far we have not implemented in trust management protocols.

So it is useful to consider how the legal profession defines the *power of attorney*: with the general power of attorney the individual appointed attorney in fact is vested with unlimited powers for an indefinite amount of time (unless otherwise explicitly specified). The validity of this of power of attorney ceases only with a specific written revocation or with the death of the person that has granted it. This is the way we used as delegation of "identity" works in trust management systems[4], and is also the idea behind usage of data by systems. Once the system has the data, it will be used. We can read privacy policies to guarantee that the data will not be misused but we have no way to know that this policy will actually be enforced by the system. Privacy polices are added *after* the system has been built.

[3] A comprehensive and updated literature survey can be found at `http://www.freehaven.net/anonbib/date.html`.

[4] This is the A speaks for B paradigm of SDSI/SPKI [6].

What is interesting for our purposes is the *special power of attorney*: the individual appointed attorney is vested solely with the power needed to carry out a specific affair (i.e. the sale or purchase of a real estate or a car). Therefore, in the formal contract it is necessary to indicate exactly the particular power that the principal intends to give to his attorney. Actual documents state quite explicitly the powers but also the goal for with these powers have been delegated. If I have to sell a house I may do all that I deem fit, but everything got tagged by that purpose and if my action is clearly not necessary I might be asked to pay for incurred losses. This is the missing important twist: in our trust management and data processing systems we have implemented only the description of powers but not the purpose.

Back to the cleaning lady, in giving her the door's key we have stipulated (or better our administration has) a contract that she has the permission of entering the room with the goal of cleaning it. Any other action would constitute breach of contract and would result in fees or contract resolution.

That's our claim: *we feel that our privacy is protected because the permission we are granting is linked to a purpose.* Notice that the the purpose is not at all security related but rather a functional goal of the system.

There are already solutions to link permissions to purpose in databases, such as the work on Hippocratic databases [1], but we found no proposals for other methods for data and credential management in distributed trust management or trust negotiation. However, it is not difficult to add yet another field to an X.509/SPKI/etc. certificate format.

The intriguing issue is how can we link the permission in a credential doled out by the security sub-system to the actual functional goals of the entire system?

The strategy that we envisage is the following:

– find a "traditional" requirements methodology in which functional goals can be made explicit;
– enhance the methodology with security-related features such as trust and delegation that are linked to the explicit goals;
– each time a delegation of permission must be foreseen by the system designer, she can trace back on the design the goals and make that explicit.

2 Related Work

The last years have seen an increasing awareness that privacy plays a key role in system development and deployment. This awareness has been matched by a number of research proposals on privacy. Next, we present some of those adopting solution to link permissions to purpose.

Platform for Privacy Preferences[5] (P3P), developed by the World Wide Web Consortium (W3C), is an emerging standard whose goal is to enable users to gain more control over the use of their personally identifiable information (PII) on web sites they visit. P3P enables web sites to express their privacy practices

[5] http://www.w3.org/TR/P3P/

in a standard format that can be retrieved automatically and interpreted easily by user. P3P provides a way for a web site to encode its data-collection and data-use practices in a machine-readable XML format known as a P3P policy.

P3P policies provide contact information for the legal entity making the representation of privacy practices in a policy, enumerate the types of data or data elements collected, and explain how the data will be used. In addition, policies identify the data recipients, and make a variety of other disclosures including information about dispute resolution, and the address of a site's human-readable privacy policy. In other words, P3P policies represent the practices of the site. Each P3P policy is applied to specific web resources listed in a policy reference file. By placing one or more P3P policies on a web site, a company does not make any statements about the privacy practices associated with other web resources not mentioned in their policy reference file, with other online activities that do not involve data collected on web sites covered by their P3P policy, or with offline activities that do not involve data collected on web sites covered by their P3P policy.

Agrawal et al. [2] propose an server-centric architecture for P3P. The P3P protocol has two parts: *Privacy Policies*, an XML format in which a web site can encode its data-collection and data-use practices, and *Privacy Preferences*, an XML format for specifying client privacy preferences. In the server-centric architecture, a web site first installs its privacy policy in a data system. Then database querying is used for maching a user privacy preference against privacy policies. Finally, web site sent result of matching preference against policy to the client, and the client requests web page if policy conforms to his/her preference. This approach is different from ours, because it does not explain the permission purpose that is implicit in privacy policies of the web site. Further, the client have to choose without knowing all the available policies of the web site. For example, there are web sites that provide services only if a client agrees a certain policy, but they show multi-policies without specifying which policy is sufficient to get the service. In this case the client could grant more permissions than necessary.

In summary, as a first step towards managing privacy, organizations publish privacy promises. The P3P statements can be used by a P3P client (e.g. the Internet Explorer 6 web browser) to notify the user automatically whether the privacy policy of the enterprise matches that configured by the user. However, this is not sufficient to guarantee the enforcement of the promises that enterprises have made, and has resulted in privacy violations, even from well meaning companies. In fact, P3P is an language for expressing privacy promises on web sites, but it cannot be used to enforce them within an enterprise.

The Enterprise Privacy Authorization Language[6] (EPAL), developed by IBM, enables an enterprise to formalize the exact privacy policy that shall be enforced within the enterprise. It formalizes the privacy promises into policies and associates a consented policy to each piece of collected data. This consented policy can then be used in access control decisions to enforce the privacy promises made. The EPAL policy language categorizes the data an enterprise holds and

[6] http://www.zurich.ibm.com/security/enterprise-privacy/epal/

the rules which govern the usage of data of each category. An EPAL policy is essentially a set of privacy rules. A rule is a statement that includes a data user, an action, a data category, and a purpose. A rule may also contain conditions and obligations.

Next, we present an architecture for implementing privacy management based on EPAL. During submission of PII, the privacy management system (by submission monitors) will create the submission records. This data is a permanent record of when PII was submitted, what privacy policy version was in place at that time, and what the users preferences were. Later, when PII is to be accessed, the privacy management enforcement monitors ensure that only data accesses are allowed that conform to the privacy policy. They also create access records that record which user accessed the data and for what purpose. The combination of submission and enforcement monitors allow the enterprise to prove that it is a good data keeper and gives the enterprise some assurance that it is enforcing its stated privacy policy.

EPAL aims at formalizing enterprise-internal privacy policies. This requires a vocabulary that formalizes the privacy relevant aspects of an enterprise. It also includes a hierarchy of purposes for which the enterprise collects data. On the other hand, P3P aims at formalizing privacy statements that are published by an enterprise. The goal is to define a machine-readable equivalent for the human readable privacy promises that are published as a privacy statement on a web page. Unlike EPAL, P3P defines a global terminology that can be used to describe the privacy promises for any enterprise. Although P3P is well suited for expressing policies, it is not as suitable for expressing an internal enforceable privacy policy. EPAL on the other hand is designed specifically to express an internal privacy policy that can be enforced by an enterprise privacy management system. IBM is currently investigating how to project a P3P policy from EPAL.

There are already solutions to link permissions to purpose in databases. Following this approach, Agrawal et al. show that the database community has the opportunity to play a central role re-designing databases to include responsability for the privacy of data as a fundamental tenet. Inspired by the Hippocratic Oath, they propose to call Hippocratic databases [1] those databases that have privacy as a central concern. Agrawal et al. propose the key principles for such Hippocratic database systems, distilled from the principles behind current privacy legislations and guidelines. Particularly, such principles highlight that the purposes for which PII has been collected shall be stored with that information in the databases, that the purposes associated with PII shall have consent of the owner of data, and that the PII collected, used, and stored shall be limited to the minimum necessary for accomplishing the specified purposes.

Hippocratic databases can be useful to add enforcement dimension to P3P. As we have seen above, a P3P policy essentially describes the purpose of the collection of information along with the intended recipients. The policy description uses data tags to specify the data items for which the policy is being stated. P3P's concepts of purpose and retention can be mapped directly into analogous concepts in Hippocratic databases. Thus, from a P3P policy it is possible to generate the corresponding data structures (i.e., a privacy-policies table) in the Hippocratic database system.

What is still missing in these proposals is capturing the high-level privacy requirements, without getting suddenly bogged down into security solutions or cryptographic algorithms. If we look at the requirements refinement process of many proposals, we find out that at certain stage a leap is made: we have a system with no privacy features consisting of high-level functionalities, and the next refinement shows encryption, access control and authentication. The modeling process should instead makes it clear why encryption, access control and authentication are necessary. This work is a step in this direction closing the gap between the functional and privacy requirements and the trust management architecture that is now emerging as the standard way to implement security in distributed systems.

3 Goal-Oriented Security Engineering

The basic building block is a "traditional" requirements methodology in which functional goals are explicit: the Tropos framework. It is an agent-based software engineering methodology [3,4] that strives to model both the organizational environment of a system and the system itself. It uses the concepts of actor, goal, plan, resource and social dependency for defining obligations of actors (dependees) to other actors (dependers). *Actors* have strategic goals within the system or the organization and represent (social) agents (organizational, human or software), roles etc. A *goal* represents some strategic interest of an actor. A plan represents a way of doing something (in particular, a plan can be executed to satisfy a goal). A resource represents a physical or an informational entity. Finally, a *dependency* between two actors indicates that one actor depends on another to accomplish a goal, execute a plan, or deliver a resource. In the sequel, when the distinction between goal, plan or resource is not essential we use the term *service* to denote any of them.

Tropos has been designed with cooperative information systems in mind and therefore it is already equipped with a methodology for reasoning about *functional delegation* of goals. Goal delegation arises quite naturally among cooperative, rational actors: every actor pursues its own goals, goal partitioning is a standard divide-and-conqueror strategy, and usually in a collaborative environment there are enough hierarchy and trust relationships, so that an actor is likely to find some other one to delegate a subgoal. When considered from an actor coordination perspective, goal delegation has two main facets:

- *Delegation of commitment.* This means that the delegatee should embrace the intentions of the delegater, trying to fulfill the goal as it was one of its own. From delegater point of view, this requires a kind of trust: the delegater has to believe that the delegatee is trustworthy and will honestly try to achieve the goal.
- *Delegation of strategy.* Delegating a declarative goal instead of an operational plan means that the delegater is interested only in the resulting outcome and not in a specific way the delegatee fulfill it.

The incorporation of security features in Tropos is not trivial and is discussed in another paper [7]. It is essentially based on the following intuition: in the dependency relationship it is implicitly assumed that if I delegate the execution of a service to somebody else I'm implicitly also the owner of this service. This implicit assumption is no longer true when also security requirements and not just functional requirements are part of the target.

In this modeling framework, four relationships (beside the old functional dependency) can be singled out:

Trust (among two agents and a service), so that A trust B on a certain goal G;
Delegation (among two agents and a service), whenever A explicitly delegates to B a goal, or the permission to execute a plan or access a resource;
Offer (between an agent and a service), so that A can offer to other agents the possibility of fulfilling a goal, executing a plan or delivering a resource;
Ownership (between an agent and a service), whenever an agent is the legitemate owner of a goal, plan or resource.

Note the difference between trust and delegation. Delegation marks a formal passage in the requirements modeling: a TM certificate will have to be eventually issued for the delegatee when implementing the system. In contrast, trust marks simply a social relationship that is not formalized by a "contract" (such as digital credential). There might be cases (e.g. because it is impractical or too costly), where we might be happy with a "social" protection, and other cases in which security is essential. In this model, there is no relationship between trust and delegation.

The requirements engineering methodology proposed in [7] specifies how to derive the trust management system (aka the delegation relationship) from the general requirement:

1. design a trust model among the actors of the systems;
2. identify who owns goals, plans, or resources and who is able to fulfill goals, execute plans or deliver resources;
3. define functional dependencies (functional delegations) of goals among agents building a functional model;
4. define a trust management implementation (e.g. based on the Delegation logics by Li et al. [8,9,10]) in which the delegation of permissions is defined.

In [7] it is also shown how one can use Datalog and the DLV system [5] to model check the correctness of the implementation wrt the previous two model or the consistency of the functional model with the trust model.

However, in [7] there is no notion of Goal-linked permission, though the framework have all necessary machinery, simply because all considered target trust management systems for the last phase have no notion of Goal-linked permission.

4 A Basic e-Health Case Study

Here we show a very basic case study, based on a modelling an e-Health service. We consider the following actors:

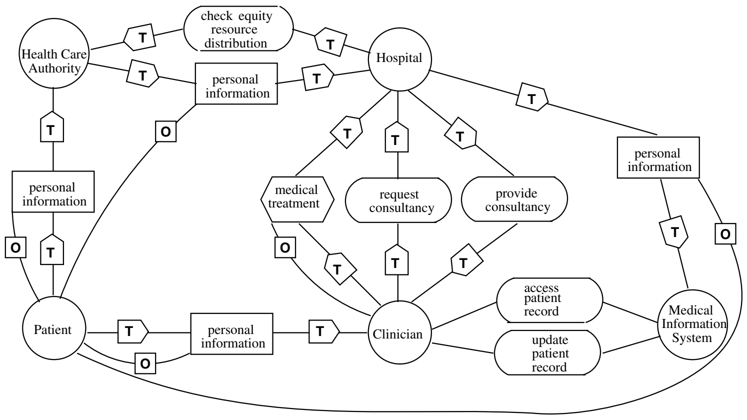

Fig. 1. Health Care System trust model

- *Patient*, that depends on the hospital for receiving appropriate health care;
- *Hospital*, that provides medical treatment and depends on the patients for having their personal information.
- *Clinician*, physician of the hospital that provides medical health advice and, whenever needed, provide accurate medical treatment;
- *Health Care Authority* (HCA) that control and guarantee the fair resources allocation and a good quality of the delivered services.
- *Medical Information System*, that, according the current privacy legislation, can share the patients medical data if and only if consent is obtained. The *Medical Information System* manages patients information, including information about the medical treatments they have received.

In order to provide rapid and accurate medical treatments, clinicians need a fast access to their patient' medical data. Similarly, HCA needs a fast and reliable access to the data in order to allocate effectively the available resources, and guaranteeing then that each patient can receive a good quality of medical care. Furthermore, HCA wants to be sure that the system cannot be defrauded in any way and that clinicians and patients behave within the limits of their roles. To the other hand, the obvious right of the patient to restrict access on his/her medical data and moreover, to be able to use some safeguards on the privacy of these data, should be taken into serious consideration. The patient's consent must be requested, and he must be notified when its data is shared.

Figure 1 and Figure 2 show respectively the trust model and the functional model. Actors are represented as circles, goals by ovals, plans by polygons and relationships by labelled arrows. The ownership relationship has an edge labelled by **O**. We use trust (**T**) to model the basic trust relationship between agents and permission (**P**) to model the actual transfer of rights in some form (e.g. a digital certificate, a signed paper, etc.). This is the delegation of permission. The plain **D** is used for dependency, that is the functional delegation.

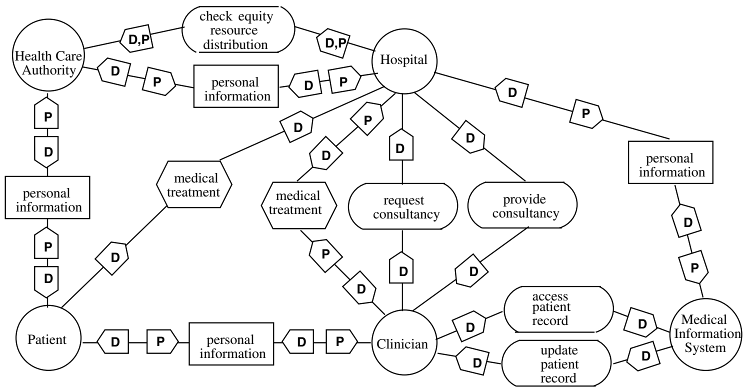

Fig. 2. Health Care System functional model

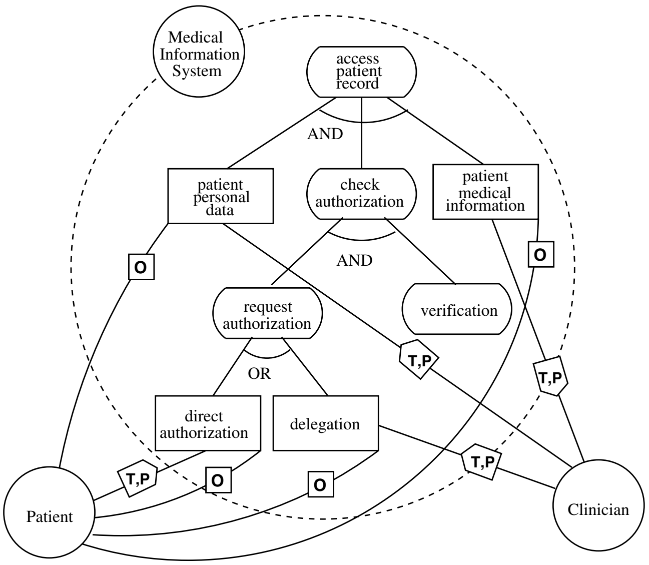

Fig. 3. Rationale Diagram

In the trust model *Patient* trusts *HCA* and *Clinician* for his personal information, and *HCA* trusts *Hospital* for it. Further, *Hospital* trusts *HCA* for checking equity resource distribution. *Clinician* trusts *Hospital* for medical treatment and for requesting specific professional consulting, and *Hospital* trusts *Clinician* for providing such consulting and for patient personal information. We also consider the trust relationship between *Hospital* and *Medical System Information* for patient personal information. Notice on top of Fig. 1 that there is a trust relationship between two actors (HCA and Hospital) on a resource that is owned by neither of them.

In the functional model, *Patient* depends on *Hospital* for medical treatments, and in turn, *Hospital* depends on *Clinician* for such treatments. To provide accurate medical treatment, *Clinician* can request specific professional consultancy to *Hospital* that depends on other *Clinicians* for this consultancy. *Hospital* delegates the goal of checking equity resource distribution to *Health Care Authority*. *Clinician* and *Health Care Authority* need patient personal information to fulfill their service. Thus, *Patient* delegates them his personal information. Further, *Health Care Authority* re-delegates these data to *Hospital*.

Notice that it is not necessary for the owner of the data to delegate the data directly to the entity that will use the service. For example here the patient can delegate the usage of his personal information to the HCA with a certain depth so that HCA can eventually re-delegate it to the actual service provider.

This trust-functional-TM implementation process is not static: requirements can be refined, new actors can be introduced, delegations can be split or passed further down the line, etc. To clarify refinement analysis in Tropos, one can use rationale diagrams that explain relationships among actors and decompose high level goals into subgoals as in Figure 3. This makes even more explicit how permissions should be linked to high-level goals.

5 Linking Permission to Purpose

Now we have all the necessary machinery to ensure the patient that his privacy will not be violated. It is of course possible to specify in details all possible delegations in this model. Indeed, this is what has been done in the paper [7]. The formal analysis carried out there has shown that this process is extremely error prone and even the expansion of delegation certificates to include blacklists may not be sufficient to rule out certain delegation paths.

In this setting is may seem unnecessary to link permission to goals: after all, few lines above we have just defined delegation as a ternary relations between a pair of agents and a service (which might be a goal). This is not what we meant by tagging permissions to purpose. The three-place delegation is a delegation of a permission. Simply the framework makes it possible to delegate something better than a simple action such as exec, or a resource. So I can delegate a plan (which is just a big composite *fixed* action) or a goal in which case I simply delegate all possible plans that can fulfill the goal. If we go back to the special power of attorney we have only mentioned the how but not the why.

The *direct base case* for linking permission to purpose is the following:

1. The owner and the depender of a service are the same actor
2. The service is a primary service of the depender and has not been obtained by refinement from other goals[7].

[7] In the Tropos methodology only goals can be refined. We stick to this line here, though from a security perspective also resources may be additionally refined for fine-grained access control.

In this case the delegation of the service from the owner to the provider of the service is tagged by the service of the functional dependency of the owner.

In the *reverse base case* the owner and the dependee of the service coincides. Also in this case the credential should be tagged with the goal of the owner but it is worth noting that the emission of this credential is not trivial. Indeed, the owner of the service should have some reason to delegate a service besides a cooperative spirit.

Notice that this is not really the case in our e-health example. For example the Hospital do need the patient personal data (of which the patient is at the same time the owner and the dependee). However, in this case the patient personal data is obtained during the design process as a refinement of the patient's own goal of obtaining care which has been delegated to the hospital. In most practical cases this is the typical format: an high-level goal of agent A is delegated to agent B and after a suitable number of recursions and refinements a subgoals is delegated from C back to A.This is typically providing information for actually fulfilling some other task necessary for the overall goal.

In the *general case* we have that

1. Owner A, depender B, and dependee C of a service are all different actors,
2. The service S is a derived service from another goal G_d of the depender,
3. The service is also a derived service from a possibly different goal G_o of the owner.

The simplest solution is to add the *conjunction* $G_d \wedge G_o$ of the owner's goal and the depender's goal to the digital credential. There might be cases in which the designer may want different schemes.

After the linkage between permission and purpose has been put in place, we can check that a delegation chain is also an appropriate delegation of purpose by checking that each goal along the delegation is a subgoal of the initial base case and each step is also appropriate. This can be easily done within the same datalog framework used in [7].

An interesting question is whether we should have also noted in the delegation credential the actual agent names to which the "purpose"-goals belonged. So far we could not find a situation in which such information is needed and cannot be reconstructed from the delegation chain.

6 Conclusions

In framework we have proposed, privacy is considered during the whole process of requirements analysis modeling trust and delegation relationships between the stackholders and the system-to-be. In this way, the framework we propose allows to capture privacy requirements at an organizational level, and hence, to help designers to model privacy concerns throughout the whole software development process.

In this paper we have discussed how a trust management (sub)system can accommodate the notion of purpose of a permission by linking it directly with the functional requirement of the overall information system. This makes possible to capture the high-level privacy requirements without taking cryptographic algorithms or protocols for trust negotiation into considerations.

There are a number of open questions that we have not answered and that are worth discussing:

– which *format* can be used for goals in certificates? A string field costs nothing to include but in this way all possibilities that we have listed of linking goals to subgoals would be lost as only equality would be tested. So a semantic web solution may be used but this may be costly from a processing perspective.
– are *distributed implementations* possible? and in particular can existing implementations of Trust Management systems be ported to this new framework (at least under the assumption that somebody has already magically derived the goal-oriented trust management implementation)? How effective and essential should purpose verification in the credential chain be?
– how *history* should be presented in the purpose? One possible solution is no history at all, just mention the last delegation step; another solution could be let history be recovered by the chain of credentials so that one could finally check if all credential have been gathered for the appropriate purpose.
– what kind of *automatic support* could be available for the automatic synthesis and validation of the trust management implementation.
– can we define a similar process for the definition of Hippocratic databases?

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB'02)*, 2002.
2. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An Implementation of P3P Using Database Technology. In *Proc. of the 9th Int. Conf. on Extending Database Technology*, volume 2992 of *Lecture Notes in Comp. Sci.*, pages 845–847. Springer-Verlag Heidelberg, 2004.
3. P. Bresciani, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An Agent-Oriented Software Development Methodology. *J. of Autonomous Agents and Multi-Agent Sys. (JAAMAS)*, (To appear).
4. J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Inform. Sys.*, 27(6):365–389, 2002.
5. T. Dell'Armi, W. Faber, G. Ielpa, N. Leone, and G. Pfeifer. Aggregate Functions in Disjunctive Logic Programming: Semantics, Complexity, and Implementation in DLV. In *Proc. of the 18th Int. Joint Conf. on Artif. Intell. (IJCAI'03)*. Morgan Kaufmann Publishers, 2003.
6. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*, September 1999. IEFT RFC 2693.
7. P. Giorgini, F. Massacci, J. Mylopoulous, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proc. of the 2nd Int. Conf. on Trust Management (iTrust 2004)*, Lecture Notes in Comp. Sci. Springer-Verlag Heidelberg, 2004.

8. N. Li, B. N. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. on Inform. and Sys. Sec. (TISSEC)*, 6(1):128–171, 2003.
9. N. Li and J. C. Mitchell. Datalog with Constraints: A Foundation for Trust-management Languages. In *Proc. of the 5th Int. Symp. on Practical Aspects of Declarative Lang. (PADL'03)*, 2003.
10. N. Li and J. C. Mitchell. RT: A Role-based Trust-management Framework. In *Proc. of DARPA Inform. Survivability Conf. & Exposition (DISCEX'03)*, 2003.
11. K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting Privacy during On-line Trust Negotiation. In *Proc. of the 2nd Workshop on Privacy Enhancing Technologies*, 2002.
12. P. Syverson. The paradoxical value of privacy. In *Proc. of 2nd Annual Workshop on Economics and Inform. Sec. (WEIS 2003)*, 2003.

Privacy is Linking Permission to Purpose (Transcript of Discussion)

Fabio Massacci

University of Trento, Italy

The latest trend on privacy, is that you go to a work service you feel respects your privacy, then you're happy and you give them the data.

If there's some problem with calculating pass codes, according to date of birth, place of birth, I create then I'm happy nobody can actually track me in this way. This is actually like the lost and found office of a railway station, and it's OK I'm able to receive information.

But if the product information is really relevant for a service, if you want some goods delivered at your house, then it has to be your name, and your address, because when the postman comes there, he will look at the name on the door, he will need to see your name otherwise he will not deliver the goods. There's no way they can enter by any kind of cryptographic protocol because they need to see the raw data, and once they have it they have it.

We could say that we will use credentials, but although credentials are very good for information groups, they're not that good if you want physical things delivered to you. The basic idea of the credential is that you give your identity to some issuer who then gives you what can be called a non credential. Essentially you just form your free properties like your name, your identity, to some bound properties. It could be some hidden identifier which allows you to browse certain New York Times web pages, and so on. But the problem is that you just pass trust from the server to the credential issuer; there does always have to be somebody who asks your identity. There's no point in that. And actually it may be even worse, because for this idea to make business sense you must have many servers for each credential issuer. This is because if each company has its credential issuer, and its server, it doesn't make much sense. It's just like giving the same the information to the same people.

So I could say, OK, then there's no solution. You can go home all happy. But instead we relied on an actual practical problem we were dealing with. It was defining the policy for the health authority, which, according to legislation, has to write down a policy for genetic material, which was that first of all what's called delegation of permission has to be privacy-respecting. So one of the problems that you'd probably get in real life is that permission never actually fits. The cleaning lady, for instance, where did she go in the room? You have no other way to let them clean the room, but once they are inside then they can look at the waste basket, or they can look at the papers on your table, and that's it, basically you have no way to get round it. But one of the points is that when you do contracts for a cleaning lady, or for submitting genetic material, you have the purpose and the permission. So you have the permission that says that what you can do, and the purpose. And the purpose is really the relevant thing. The cleaning lady is allowed to have the key of the room to clean the room, not to look at the paper. If you come inside and you find her looking at the paper that's a breach of the contract, even if she has the permission to enter the room.

And indeed, in the real wide world, there is also a well known legal procedure for doing that. I think it's called a special power of attorney, so if I wanted to send somebody to the US to buy a house, I can go to a notary and write down, you, X, have the power of attorney to buy me a house. I give you all the power to do all necessary things, but only things necessary to buy a house. I can make it cover things like opening bank accounts, closing bank accounts, or paying money, and so on, for the purpose of buying a house. The same things are true for the privacy statement. When you submit genetic data, and actually when you submit genetic material, like placenta at childbirth (the mother can decide to donate the placenta of the newly born baby for genetic analysis, and for transplants), you write on something and you specify the purpose for which this genetic material can be used, including by yourself. So there is the permission and the purpose in this piece of paper.

The protection that the law gives you is something after the fact. If you give the data to somebody else, in this case once they have it, they have it, but the law says, since they got it for certain permission then you're somehow protected by the law if they misuse it. So you can ask for correction, or redress, and so on. Actually, our idea is that the privacy can be enforced because we link the permission to the purpose.

It doesn't make sense that we want to be anonymised and so on. In reality, if I give you some money for buying a fire extinguisher, you don't need any other information from me because I just give you the permission to buy the fire extinguisher. If you want to do anything else besides that that's why we use cash. That's the basic statement that we want to make: If the private information is used for the purpose for which you have given it, then the user is happy. Private information can be used for other purposes then, which is the reason why we are so concerned at putting our data on the web. US law has a big issue on this subject. For European citizens that have their private data held in the US, there is an agreement that is called a safe haven, in which US companies essentially declare they will respect the EU privacy legislation. The users may be unwilling to provide this information if the information can be used beyond their purpose.

One interesting point is that if you look in the literature there is something that's slightly wrong conceptually in the proposals. They can't specify very well all the other purposes for which your information can be used, like marketing, analysis, development; they can only specify what data is actually used.

If I'm a user I don't give my data for administration and development. I usually give you the data for something else, because development of your business is not my main interest.

The actual business use is the one for which we have given the data. But if I ask you to find the actual business use of this policy, the answer is you have no clue. What's wrong? Is the language not expressive enough? Maybe, but I believe that it's something slightly different. If you look at your data there is no statement of what you do use the data for, which is the only legitimate use.

The focus is on modelling. What you would like to model, in some sort of a development architecture, is a function on things for which the web service has been set up, and then you can have additional goals, so we use this for genetic analysis, or for marketing, or for scientific analysis, and so on.

Having the purpose is not enough if the purpose is not what you actually submitted. If you provide genetic data then you can add lots of interesting information that maybe you don't want the receiver of the data to have. Our idea is that if you have a way to specify the functional requirements from the service for the user on the system then we can say, OK, the user has certain goals and certain purposes.

Let's generalise the notion of credential to a chain. So this credential essentially mentions that this is the information that I give you and besides we also have the goal for which the permission can be used.

So I delegate this to you and then you have permission; and then at each time you can do a refinement purpose. So I am the bank, I retrieve your address for getting all the bank's services, then I need you to delegate this to the shipping company, which is now a different purpose, that is reading and printing your address as a permission, but the purpose is shipping some material. So in a sense it's just a credential that's slightly more definite.

The interesting point is that when you go to court and there's a claim that says you have used my genetic data wrongly, then you must show the court that you have a valid chain of credentials starting from the donor, actually from the purpose that the donor has given you the data, up to the point of use, showing the controls. So once you have this then we can say, yes, OK, you have a valid delegation from the patient to the hospital, go and get the record and provide this access for his own health for providing care to himself. Any other access to the record is forbidden.

George Danezis: The salt and pepper really is missing from this slide because I don't see the conflicts. Our trade as computer security people is actually to manage conflict. Unless you explicitly include the conflicts that will induce people to try to get in the way of each other, and cheat on each other, and steal from each other, and kill each other, then it's just about communicating totally, and being polite to each other. So the question is. . .

Bruce Christianson: What's the threat model?

George Danezis: And what are the technical mechanisms to avoid?

Reply: If I have given someone the data for doing certain things, I expect that if he allows anybody else to access the data for any different purpose then that's a violation of trust. In a sense everything else that I've not given the data for is an attack on the system.

Andrei Serjantov: Yes, but hang on, attackers don't work like that. Suppose I've given my data to the hospital, and then somehow the insurance company that provides my medical insurance gets hold of it, by breaking access control, or whatever, and suddenly my premium's gone up by fifty percent, what am I going to do? How do I find out what happened; how I am going to prove it to anybody?

Reply: If they don't have legitimate access to this data they have to delete it, period.

Andrei Serjantov: Yes, but they are an attacker. They are interested in pricing me, and gaining money from me.

Bruce Christianson: We're assuming the existence of some regulatory authority that can hold insurance companies to account.

Reply: The assumption is that once you give the data to somebody else they have it.

George Danezis: Even in this case I think that you cannot get out of the fact that you need to provide some technical mechanisms in order to support the work of this regulatory authority, such as providing trails of evidence.

Virgil Gligor: Also some degree of originator control on the data.

Bruce Christianson: Sure, but providing an audit trail of this kind is a different problem to the kind of audit trail that we would customarily provide, and I think that is the point that Fabio was making.

Reply: Yes, that's what I'm saying; there has to be a chain, OK, there has to be an audit trail so that technology has to support what I've given you the data for. You cannot just say I'm giving you the data, period, and the data's there.

Virgil Gligor: I have two comments. One is that linking the permission to the purpose may not be a sufficient mechanism for a different reason. What you'd like to do is to always operate with the least permission possible for the purpose. So you have to tailor your permission in your system to match the purposes very tightly, otherwise the mechanism is subject to unintended effects during the failure of the mechanism. In other words, you not only have to control what happens in a normal mode, but you have to control what happens in a failure mode. For example, our news data goes bad, how much damage can he do? So we always want to design systems where that linkage is done with a least permission in mind.

The second thing is that when you click on various buttons and you get permissions for various stated actions, it's generally not enough just to know exactly the purpose, but it's also necessary to know the implications of that action. I'm told by lawyers that when you are given options for various stated actions you are supposed to explain to the person who takes those options what the implications are. That goes beyond a privacy matter; it's basically truth in advertising in some sense. But actually to do that is not easy.

Reply: Is there any technological solution for that problem?

Virgil Gligor: Well you can inform the person. Here is the purpose, and by the way if you give me permissions for this purpose here are some of the implications that may fall out of that.

Reply: But this would be in place if you specify what the purpose is, and from the purpose you could infer the actions that are possible.

Virgil Gligor: Well you can extend the meaning of the purpose with the implications of the purpose, but generally that's not part of the definition of purpose.

Bruce Christianson: But this is saying there needs to be a change in the way in which we describe or model a security policy.

Tuomas Aura: I don't think we need to assume an adversarial model where you have an enemy that's trying to get your data and that's why you need an access model, to prevent them from getting the data or trying to misuse your data. That's not the only reason why you need some way of auditing that data. Let's start from a simpler model and assume that the businesses who collect your data and keep it in a database have a good legal reason for keeping it there; that they want to confirm the reason why they have the data, and that they want to do the right thing, and use the data only for the purposes that they promised to use it. Even in that case we don't have the technological support currently to be able to do this.

The first problem would be, in a big corporation, for example, that if you had a database of data you would always have the marketing people getting their fingers into it. We want to mark clearly that this should not happen, and make sure it doesn't happen if you have promised that it won't.

Bruce Christianson: Yes, absolutely right; that's the real problem. There's no provenance attached to the data. This was the thing that we got wrong back in the 1960s when we thought that computer hardware would last a long time, and this month's data would be replaced by next month's data very soon. In actual fact computers last about six months these days but the data lasts forever. Data collected for one purpose is often not sufficiently reliable to be used for another purpose; if you don't know the purpose for which data was collected, you don't know whether it's safe or not to use for another purpose. Even if your motives are pure you still need to have that provenance.

Virgil Gligor: So the other problem which I think is extremely important is how understandable the purpose is. In the US court system, the average the person is supposed to understand; so for example, if you have a very complex description of the purpose and you click the "I understood the purpose" button, if the average person would not understand the purpose, the fact that you clicked the button is legally meaningless.

George Danezis: I agree that there is a very valid field of research that is about managing data. I think most of computer science actually is about managing data properly, but computer security within that field also specialises in providing security properties to people namely, properties in the face of an adversarial situation.

Tuomas Aura: The first step of security when you give me something to keep for you is to prevent me from accidentally giving it to someone else. The next step after that is to keep me honest.

George Danezis: So the thing is, how do people who use such mechanisms perceive them? Do they perceive them as good management, and good business practices, which I agree they should be there, and we don't know how to do that, or do they perceive them as security mechanisms?

Pasi Eronen: I don't think there's actually this rigid difference between good management and security mechanisms, because not all security mechanisms are implemented in software or hardware. The security mechanisms that are implemented by procedures, and people, and management practices, you can call that good management but you can also call that good security. I think that what you are actually talking about here is somewhere in between. There are parts of this that are implemented in software and hardware, or could be, and parts of it are implemented in good management practices.

Tuomas Aura: Another way to say this would be, the computer system is an implementation of your business model. Then the question is, does that also implement the privacy policy that you have promised.

George Danezis: What I'm concentrating on is not really whether it is implemented by an automatic system or a human system. Matt, for example, described yesterday[1] a lot of protocols that are implemented over social interaction. What I'm saying is that in order to design such protocols one has to start from the point of view that there will be an adversary, and consider what kind of roles this adversary will have. I think it may also be controversial because private data is being used all the time; we see this in the newspapers, so clearly third parties are naturally interested not just in managing but also in abusing this data. Unless we actually start from that then our mechanisms, whether social or automatic, are going to be weak.

Bruce Christianson: But the question, and it's a good question because it's deep, is whether, from the point of view of mechanism, it's culpable to make this rigid distinction between an external malicious adversary and the internal malicious adversary of incompetence. There are arguments which suggest that often a unified approach will get you further than trying to orthogonalise those threats. There's a very cynical pragmatic point of view that, if you can wrap part of your software, or part of your system interface, up in the line that this is value added to your business plan, rather than this is an essential security feature, then it's much less likely to be turned off or worked around. I realise this argument is cynical, but I don't think that's a knock down rejoinder. There's a real argument to be had about how helpful it is to orthoganolise the different

[1] Matt Blaze, *Toward a Broader View of Security Protocols*, these proceedings.

types of threats. Clearly there are some cases where you really do want to know, was it an accident or was it deliberate. Why is that alarm tripping, is it an evil door-rattler like Matt Blaze, or is it an incompetent door rattler like me?

Matt Blaze: But there are other cases where that distinction between us isn't important.

Bruce Christianson: Exactly[2]. I think this issue is a genuine question which we really ought to think about a bit more.

Rebecca Wright: Another point is that there's not usually just one notion of an attacker. The company that collects the data might discover a user who says, I think you do not honour the obligations that you've given to me; the company can say, yes I did because here is this contract we agreed on. In some sense here an attack by the user that they're protecting against by just having good management practices.

[2] And there are even cases where the distinction itself isn't so sharp.

Establishing Trust with Privacy

Laurent Bussard and Refik Molva

Institut Eurécom[1]
Corporate Communications 2229, route des Crêtes BP 193
06904 Sophia Antipolis, France
{bussard, molva}@eurecom.fr

Abstract. In pervasive computing environments, *a priori* trust among parties is lacking. New mechanisms are required in order to build trust without relying on existing relationships. We present a solution to establish trust based on a history of previous interactions among parties. Past interactions can be proven while assuring the untraceability and anonymity of provers.

Introduction

With the advent of self-organizing systems such as ad hoc networks or pervasive computing, security protocols have to meet a new requirement for establishing trust among parties that have no *a priori* relationship like a shared naming structure or a common organization. Trust establishment in this context calls for a brand new paradigm with respect to classical scenarios whereby entities build trust based on some existing security association. We suggest in this paper a cryptographic protocol through which parties can build trust based on the history of their interactions with other parties. This protocol allows a prover to get a proof of history or the evidence that the prover was involved in some interaction with another party. During further interactions, other parties consider the prover trustworthy based on the verification of the proof of history.

Privacy is an essential requirement for such a protocol since providing proof of history to several parties without privacy would severely expose the behavior of the prover. The history-based trust establishment protocol thus assures the anonymity of the prover and the unlinkability of interactions using the proof of history. Moreover, the prover can choose to show only parts of his history. The proof of history is based on a signature mechanism and the trust establishment protocol is a challenge-response protocol based on this mechanism. The signature mechanism is an extension of group signatures.

1 Problem Statement

First we describe the general trust establishment requirements then we show how trust can be established while preserving user's privacy.

[1] Institut Eurécom's research is partially supported by its members: Bouygues Télécom, Cegetel, France Télécom, Hasler Foundation, Hitachi, STMicroelectronics, Swisscom, Texas Instruments, and Thales.

1.1 Trust Establishment

Trust is generally defined as *the belief that one can depend on something or someone*. There are clearly notions of uncertainty and risk. A resulting property is that *if party B trusts A then B can grant A some rights*. Likewise, any information that enables B to decide whether he can grant A some rights is part of the trust establishment process.

Trust is generally a derivation of some *a priori* trust. For instance, C trusts B and then C trusts A if B says A is trustworthy. In a web of trust, B is a friend of C, and in the simple public key infrastructure (SPKI), B is the hierarchical authority of C.

However, in environments based on new paradigms such as peer-to-peer, pervasive computing, or ad hoc networks, there is a lack of *a priori* trust among parties and thus a new mechanism in required to build trust in *a posteriori* fashion based on monitored evidence. The following two alternatives appear to be suitable concepts on which to build *a posteriori* trust:

– Reputation (statistical evidence).
– History of interactions (provable evidence).

In this paper we focus on the latter case. A interacts with B and receives some credential proving that this interaction occurred. Subsequently, A can prove B or another party C that this interaction occurred in the past. As a result of this proof, B or C can grant A some rights.

1.2 Privacy Protection

Proving history exposes the behavior of the prover (A) and thus is a potential threat on the privacy of A.

Our architecture for protecting the untraceability of users in such a context relies on three layers (see Figure 1). First, it is necessary that the network layer does not rely on static addresses. In a personal area network (e.g. Bluetooth), MAC addresses should not be visible or should change regularly; in a WAN, IP addresses should be kept secret by using, for instance, mixes [5]. Second, the credentials that are delivered to the users should not be traceable, i.e. with SPKI or X.509 the signature that is attached to a credential can be used as a unique identifier. Schemes relying on blind signatures or on signatures of knowledge can solve this problem (see related work in Section 4). At the application

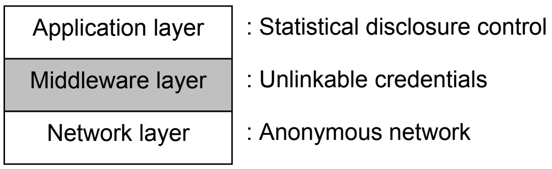

Fig. 1. Three layers for defining unlinkability during trust establishment

layer, the attributes that are revealed have to be carefully chosen in order to avoid traceability based on the attributes. It is obvious that identity certificates enable traceability of holders even when the credential is unlinkable. However less precise attributes can also be used to trace some person in a small group (e.g. birthday, office number). Statistical disclosure control [8] aims at controlling what information can be revealed without threatening user's privacy.

In this paper, we only focus on the second layer and propose a new scheme for unlinkable credentials dedicated to trust establishment. Other schemes such as [3] could be used as well with some restrictions (see Section 4).

2 History-Based Signature

This section shows the interactions necessary to build a provable history and to use this for history-based trust establishment. Users collect pieces of evidence for their activity and store them as a provable history. In order to assure their non-transferability, the pieces of evidence are implemented as credentials attached to a valuable secret (equivalent to the private key in public key infrastructures). Credentials can define group membership, location-and-time stamps, recommendations, etc.

When signing a challenge or a document, the user chooses some credentials in his history, modifies them, and signs with those credentials. Credentials have to fulfill the following requirements to build a provable yet anonymous history: non-transferability, i.e. credentials can only be used by the owner of some valuable secret; anonymity, i.e. use of history-based credentials should not reveal the identity of the prover; and unlinkability, i.e. it is not possible to link different signatures based on the same credential.

History-based signature is an extension of the group signature scheme described in [4]. Alice (A) is the signer. She collects some credentials to subsequently prove some history. For instance, A holds a credential to prove that she has been in some place. When A is traveling or visiting partners, she collects location stamps. A has credentials to prove some membership, e.g. employee of a company, member of IEEE computer society, partner of some project, member of a golf club, citizen of some state, client of some bank, customer of some airline. A can alternatively show some recommendations: when she collaborates with other entities, she receives credentials. All those credentials define A's provable history. Each credential can either be used as a proof during a challenge-response protocol or as an attribute associated with a signature.

2.1 Certification

To initiate the system, the signer A has to get some certificate proving that she has a valid secret, i.e. a secret linked to her identity. This phase of our protocol is similar to the join protocol of the Camenisch's group signature scheme. However, we use a modified version because a coalition attack exists against the former version of the scheme [1]. We define the following elements: $n = pq$ where p and

q are two large primes; $\mathcal{Z}_n = \{0, 1, 2, \ldots, n-1\}$ is a ring of integers modulo n; $\mathcal{Z}_n^* = \{i \in \mathcal{Z}_n \mid \gcd(i, n) = 1\}$ is a multiplicative group; $G = \{1, g, g^2, \ldots, g^{n-1}\}$ is a cyclic group of order n; g is a generator of this group G; $a, b \in \mathcal{Z}_n^*$ are elements of the multiplicative group.

Table 1. Creation and first certification of A's secret x

A		**B**
		private: p_b, q_b, d_b public: $n_b, e_b, G_b, g_b, a_b, \lambda_b$
1.1) chooses random secret x' $x' \in_R \{0, 1, ..2^{\lambda_b - 1}\}$		
	1.2) $y' = a_b^{x'} \mod n_b$ $\longrightarrow$	
	1.3) $\xi \in_R \{0, 1, .., 2^{\lambda_b - 1} - 1\}$ $\longleftarrow$	
1.4) computes $x = x' + \xi$ $y = a_b^x \mod n_b$ commits to $z = g_b^y$		
	1.5) y, z $\longrightarrow$	
	1.6) PK$[\alpha \mid y = a_b^\alpha]$ $\longleftrightarrow$	
		1.7) verifies $y \overset{?}{=} y' \cdot a_b^\xi$
	1.8) $\text{cert}_{1b} = (y+1)^{d_b} \mod n_b$ $\longleftarrow$	

In Table 1, A generates some secret x with the help of a CA or group manager B. Moreover, A receives a certificate on this secret x: $\text{cert}_{1b} = (a_b^x + 1)^{d_b} \mod n_b$.

2.2 Obtaining Credentials

Once certified, A can visit different entities that will provide credentials such as proofs of location, proofs of interaction, recommendations, etc. A provable history is a set of credentials. Table 2 shows how A can get a credential from C. The identity of A is not known but C verifies that this entity is certified by some known CA or Group manager B. It is always necessary to have some trust relationship with previous signers when providing credentials or when verifying history. In this example, C has to trust B otherwise the protocol of Table 1 has to be run once more. However, when an entity D needs to verify the signature of A on some document, D only has to know C.

Two interactive proofs of knowledge (PK) are done in step 2.3). The first one proves that y_2 is based on some secret. The second one shows that this secret

Table 2. Obtaining some credential to build history

A		**C**
private: $x, (a_b^x + 1)^{d_b}$		private: $p_c, q_c, d_c, d_{c_1}, \dots d_{c_k}$ public: $n_c, e_c, e_{c_1}, \dots e_{c_k}$, $G_c, g_c, a_c, b_c, \lambda_c$
2.1) $y_2 = a_c^x \mod n_c$ $\tilde{g}_b = g_b^r$ for $r \in_R \mathcal{Z}_{n_b}$ $\tilde{z} = \tilde{g}_b{}^y$ (i.e. $\tilde{z} = z^r$)		
	2.2) y_2 $\xrightarrow{\hspace{6em}}$	
	2.3) pk_2: PK$[\alpha \mid y_2 = a_c^\alpha \wedge \tilde{z} = \tilde{g}_b{}^{(a_b^\alpha)}]$ pk_3: PK$[\beta \mid \tilde{z}\tilde{g}_b = \tilde{g}_b{}^{(\beta^{e_b})}]$ $\longleftrightarrow$	
		2.4) $t \in_R \{0, 1, \dots, 2^{\lambda_c} - 1\}$ $\text{cert}_{1\text{c}} = (a_c^x + 1)^{d_c}$ $\text{cert}_{2\text{c}} = (a_c^x + b_c^t)^{d_h}$ $\text{cert}_{3\text{c}} = (b_c^t + 1)^{d_c}$ where $d_h = \prod_{i \in S} d_i$
	2.5) t, $\text{cert}_{1\text{c}}$, $\text{cert}_{2\text{c}}$, $\text{cert}_{3\text{c}}, S$ $\xleftarrow{\hspace{6em}}$	

has been certified by B. Indeed, $\tilde{z}\tilde{g}_b = \tilde{g}_b{}^{(\beta^{e_b})} = \tilde{g}_b{}^{(a_b^\alpha)}\tilde{g}_b = \tilde{g}_b{}^{(1+a_b^\alpha)}$ and thus $1 + a_b^\alpha = \beta^{e_b}$. It means that A knows $\beta = (1 + a_b^\alpha)^{d_b}$ that is a certification of α, which is also the discrete logarithm of y_2 to the base a_c. In other words, y_2 has been computed from the same secret x.

In step 2.4) A receives a new credential $\text{cert}_{2\text{c}} = (a_c^x + b_c^t)^{d_h} \mod n_c$ from C that will be used to prove some history. b_c as well as a_c are elements of $\mathcal{Z}_{n_c}^*$, x prevents the transferability of credentials, and t is different for each credential to forbid a user from combining multiple credentials (see Section 3). The attribute value, be it a location or a recommendation, is defined using a technique akin to electronic cash: $d_h = \prod_{i \in S} d_{c_i}$ where S is a set that defines the amount or any attribute. Construction of d_h is given in Section 2.4. Two other credentials can be provided: $\text{cert}_{1\text{c}} = (a_c^x + 1)^{d_c} \mod n_c$ is a certification of the secret that can replace $\text{cert}_{1\text{b}}$. To avoid a potential attack (see Section 3), we add $\text{cert}_{3\text{c}} = (b_c^t + 1)^{d_c} \mod n_c$.

2.3 Using History for Signing

This section shows how Alice can sign a document as the holder of a set of credentials. Alice knows a secret x, the certification of this secret ($\text{cert}_{1\text{c}}$), and some credential that is part of her history ($\text{cert}_{2\text{c}}$). Using these credentials, she can compute a signature on some message m. A generates a random number $r_1 \in_R \mathcal{Z}_{n_c}$ and computes five signatures based on a proof of knowledge (SPK):

$$\hat{g}_c = g_c^{r_1},\ \hat{z}_2 = \hat{g}_c{}^{y_2},\ \text{and}\ \hat{z}_3 = \hat{g}_c{}^{(b_c^t)}$$
$$spk_1 = \text{SPK}[\alpha \mid \hat{z}_2 = \hat{g}_c{}^{(a_c^\alpha)}](m)$$
$$spk_2 = \text{SPK}[\beta \mid \hat{z}_2\hat{g}_c = \hat{g}_c{}^{(\beta^{e_c})}](m)$$
$$spk_3 = \text{SPK}[\delta \mid \hat{z}_3 = \hat{g}_c{}^{(b_c^\delta)}](m)$$
$$spk_4 = \text{SPK}[\gamma \mid \hat{z}_2\hat{z}_3 = \hat{g}_c{}^{(\gamma^{e_{h'}})}](m) \quad \text{where} \quad e_{h'} = \textstyle\prod_{i \in S'} e_i \text{ and } S' \subseteq S$$
$$spk_5 = \text{SPK}[\epsilon \mid \hat{z}_3\hat{g}_c = \hat{g}_c{}^{(\epsilon^{e_c})}](m)$$

The signature of message m is $\{spk_1, spk_2, spk_3, spk_4, spk_5, \hat{g}_c, \hat{z}_2, \hat{z}_3, S'\}$. The signatures of knowledge spk_1 and spk_2 prove that the signer knows $\text{cert}_{1\text{c}}$: $\beta = (1 + a_c^\alpha)^{d_c} \mod n_c$. The signatures of knowledge spk_1, spk_3 and spk_4 prove that the signer knows $\text{cert}'_{2\text{c}}$: $\gamma = (a_c^\alpha + b_c^\delta)^{d_{h'}} \mod n_c$. To avoid some potential attack (see Section 3), we added spk_5 to prove the knowledge of $\text{cert}_{3\text{c}}$. spk_3 and spk_5 prove that t was generated by C: $\epsilon = (1 + b_c^\delta)^{d_c} \mod n_c$.

Note that when this scheme is used as a challenge-response protocol, signatures based on a proof of knowledge can be replaced by proofs of knowledge.

2.4 Encoding Attribute Values

In the protocol, Alice only discloses a part of her attribute: she receives $\text{cert}_{2\text{c}}$ with attribute S and signs using $\text{cert}'_{2\text{c}}$ with attribute S'. A flexible mechanism is necessary for displaying part of the attributes.

Each authority that delivers certificates (time stamper, location stamper, group manager, etc.) has a public key: a RSA modulo (n), and a set of small primes $e_1, \ldots, e_m$ where $\forall i \in \{1, \ldots, m\} \mid \gcd(e_i, \phi(n)) = 1$. The meaning of each e_i is public as well. Each authority also has a private key: p, q, and $\{d_1, \ldots, d_m\}$ where $pq = n$ and $\forall i \in \{1, \ldots, m\} \mid e_i \cdot d_i = 1 \mod \phi(n)$.

A signature $SIGN_{(S,n)}(m) = m^{d_h} \mod n$, where S is a set of integers $i \in \{1, \ldots, m\}$ and $d_h = \prod_{i \in S} d_i$, can then be transformed into a signature $SIGN_{(S',n)}(m) = m^{d_{h'}} \mod n$, where S' is a subset of S and $d_{h'} = \prod_{i \in S'} d_i$. The attribute value is encoded as a set S defines as follows: $S = \{i | 1 \leq i \leq m \text{ and } b_i = 1\}$ where b_i is the ith bit in the binary representation of the attribute. The signature based on set S can be reduced to a signature based on any subset $S' \subseteq S$ based on the following property:

$$\begin{aligned} SIGN_{(S',n)}(m) &= \left(SIGN_{(S,n)}(m)\right)^{\left(\prod_{i \in \{S \setminus S'\}} e_i\right)} \\ &= m^{\left(\prod_{i \in S'} d_i \mod \phi(n)\right)} \mod n \end{aligned}$$

Thus, an entity that received some credential $\text{cert}_{2\text{c}}$ is able to compute $\text{cert}'_{2\text{c}}$ and to sign a document with this new credential as follows:

$$\begin{aligned} \text{cert}'_{2\text{c}} &= (\text{cert}_{2\text{c}})^{\prod_{j \in \{S \setminus S'\}} e_j} \\ &= \left((a_c^x + b_c^t)^{\prod_{i \in S} d_i}\right)^{\prod_{j \in \{S \setminus S'\}} e_j} \\ &= (a_c^x + b_c^t)^{\prod_{i \in S'} d_i} \mod n \end{aligned}$$

This technique ensures that part of the signed attributes can be modified. A location and time stamper can certifies that some entity has been at a given place at a given time: `[time: 10:15 AM`, `date: 24/05/2004`, `latitude: 43.6265°`, `longitude: -007.0470°]`. If a location and time stamper provides the following credential to Alice:

```
[10|15, 24|05|2004, 43|62|65, -007|04|70]
```

Alice can sign a document with a subset of this credential:

> `[10|XX, XX|XX|XXXX, 43|62|65, -007|04|70]` to mean that the document is signed by *someone that was in the building someday around ten o'clock*.

Or

> `[XX|XX, 24|05|2004 43|XX|XX, -007|XX|XX]` to mean that the document is signed by *someone who was in the South of France the 24th of May.*

Similarly, a company can qualify customers as *Platinum, Gold, or Silver*; a state can provide digital Id cards to citizen to certify gender, name; an employer can provide credentials that define role, access rights; and a partner can define recommendations. In all those cases, the ability of selecting which attribute is displayed is very important to protect privacy when enabling trust evaluation.

3 Security Evaluation

The security of the scheme is based on the assumption that the discrete logarithm, the double discrete logarithm and the roots of discrete logarithm problems are hard. In addition it is based on the security of Schnorr and RSA signature schemes and on the additional assumption of [4] that computing membership certificates is hard.

Our proposal is based on the group signature scheme of [4], whose join protocol is subject to a collusion attack [1]. Modifications suggested in [7] and that prevent this attack have been taken into account (see Table 1). The security rests on a well-defined number-theoretic conjecture.

The first requirement for history-based signature scheme presented in this paper is that credentials can only be issued by the legitimate issuer B. This is expressed in the following.

Proposition 1. *Credential unforgeability - It is infeasible to generate an unlinkable credential without knowing the private key of the issuer B.*

Proof. In order to encode attribute values, a set of different e_i and d_i are used with the same modulo n. However, the common modulus attack does not apply here because d_i's are kept secret and known by a single entity as with the standard RSA. Because there are multiple valid signatures for a given message, this scheme seems to make easier brute force attacks that aim at creating a valid signature for a given message: an attacker can choose a message m and a random $d_R \in_R \mathcal{Z}_n^*$ and compute a signature $m^{d_R} \mod n$. If e_i and d_i are defined for $i \in \{1, \ldots, k\}$, there are 2^k valid $d = \prod_{i \in S' \subseteq S} d_i$. The probability that a random d_R be acceptable is 2^k times higher than with standard RSA where $k = 1$. However, even if the number of possible signatures for a given message increases, it is necessary to find out the set S (i.e. e_R) corresponding to the randomly chosen signature. In other words, the attacker has to test whether $\forall S' \subseteq S \mid m \overset{?}{=} (m^{d'})^{\prod_{i \in S'} e_i} \mod n$. There are 2^k possible sets S' to check and thus the security of this scheme is equivalent to RSA.

In some cases, the signature scheme can allow combining attributes of two credentials in order to create a new one: naive credentials $(a^x + 1)^{d_{h_1}}$ and $(a^x + 1)^{d_{h_2}}$ could be used to create $(a^x + 1)^{d_{h'}}$ where $S' \subseteq S_1 \cup S_2$. If h_1 states that Alice was present from 8 a.m. to 10 a.m. and h_2 states that she was present from 4 p.m. to 6 p.m., it is necessary to forbid that Alice could create a h' stating that she was present from 8 a.m. to 6 p.m. To avoid this attack, a unique secret t is associated to each credential. Hence, without knowing $\log_a(b)$, the certificates $(a^x + b^{t_1})^{d_{h_1}}$ and $(a^x + b^{t_2})^{d_{h_2}}$ cannot be combined.

The second requirement for unlinkable credential scheme presented in this paper is that it is not possible to link the signature of A on some message to the identity of A and that it is not possible to link different signatures computed with the same credential.

Proposition 2. *Signature anonymity and unlinkability - the credential scheme assures the anonymity of the signer and the unlinkability of signatures.*

Proof. Linking two signatures $\{spk_1, spk_2, spk_3, spk_4, spk_5, \hat{g_c}, \hat{z_2}, \hat{z_3}\}$ and $\{spk'_1, spk'_2, spk'_3, spk'_4, spk'_5, \hat{g_c}', \hat{z_2}', \hat{z_3}'\}$, i.e., deciding whether these signatures have been issued by the same user A, is only possible by deciding whether $\log_{\hat{g_c}}(\hat{z_2}) = \log_{\hat{g_c}'}(\hat{z_2}')$ or deciding whether $\log_{\hat{g_c}}(\hat{z_3}) = \log_{\hat{g_c}'}(\hat{z_3}')$, which is equivalent to solving the discrete logarithm problem that is considered a hard problem. The signatures generated by the credential scheme are therefore unlinkable. The party B, be it a certification authority or a group manager, certifies the secret x of A and knows the identity of A. Other parties have no information on the holder of x and thus A can sign anonymously.

As mentioned in Section 1, when the same credential is used in two signatures, the attribute revealed should be different or not precise in order to prevent linkability. Statistical disclosure control [8] is a promising direction to determine at the application level, what information can be revealed without violating unlinkability.

The third requirement for history-based signature scheme presented in this paper is that a signature cannot be generated without holding the correct credentials. This is expressed in the following.

Proposition 3. *Signature unforgeability - It is infeasible to sign with respect to some attribute without holding a credential with this attribute.*

Proof. In the random oracle model, spk_1 proves that the signer knows his secret, spk_3 proves that the signer knows a credential's secret, and spk_4 proves that the signer knows a credential corresponding to both secrets. That is, spk_1 and spk_3 respectively show that

$$\hat{z_2} = \hat{g}^{(a^\alpha)} \quad \text{and} \quad \hat{z_3} = \hat{g}^{(b^\delta)}$$

and therefore:

$$\hat{z_2}\hat{z_3} = \hat{g}^{(a^\alpha + b^\delta)}$$

Whereby integers α and δ are known by the signer. On the other hand, spk_4 proves that

$$(a^\alpha + b^\delta) = \gamma^{e_{h'}}$$

for some γ that the signer knows. Under the hardness assumption on the unforgeability of credentials, this can only happen if the signer received a credential.

The fourth requirement for history-based signature scheme presented in this paper is that credentials cannot be transferred. This is expressed in the following.

Proposition 4. *Credential non-transferability - Credentials are strongly linked to a valuable secret of the holder and thus cannot be transferred.*

Proof. Even when the signature of a message cannot be forged, a desirable goal is to be able to assure that it is not possible to find another message with the same signature. Violation of this property with our protocol would require the generation of two pairs (x, t) and (x', t') so that $a^x + b^t = a^{x'} + b^{t'}$. In order to prevent transferability based on such generation of equivalent pairs, cert_{3c} and spk_5 were included in the protocol. Computing (x', t') from a credential based on (x, t) would thus require computing $x' = \log_a(a^x + b^t - b^{t'})$ which is equivalent to solving the discrete logarithm problem. Our protocol thus assures that the credential received as a proof of context or as a recommendation cannot be transferred. A proof that the generation of equivalent pairs is equivalent to a difficult problem (e.g. the discrete logarithm problem) would allow for important simplifications of the history-based signature scheme.

4 Related Work

Common attribute certificates such as X.509 and SPKI [6] are based on public key infrastructure and thus cannot be made unlinkable because the public key can be recognized.

The schemes presented in [3,2] already allow for privacy-preserving attribute verification. However, the target of these schemes is anonymous attribute certificates and untraceable access control. Credentials defined in [3] rely on pseudonyms and thus cannot be used for non-interactive signatures. Credentials defined in [2] do not assure non-transferability and have to be used as one-time credentials to ensure untraceability. The one-time property of these credentials does not suit multiple interactions that are required by our scenario.

Table 3 compares different *unlinkable* credential schemes. *Selective disclosure* means that the user can choose to reveal a part of the attribute when signing.

Table 3. Comparison of different schemes

Properties	**Group signatures**	**Brands' credentials**	**Camenisch Idemix**	**History-based Sig**
Unlinkable	×	×	×	×
Selective Disclosure	-	×	×	×
Complex Disclosure Schemes	-	×	×	-
Non-transferable	PKI	Biometrics	AoN or PKI	PKI
Number of Use	∞	1	1 or ∞	∞
Anonymity Revocation	-	-	×	-
Credential Combination	-	×	×	×
Non-interactive Signature	×	×	-	×

For instance, a credential asserting that its holder is thirty years old can be used so that it only discloses that its holder is older than eighteen. More complex disclosure schemes are defined in some related work. *Non-transferability* means that a user cannot let another person use one of his credentials without revealing some valuable data: a private key in public key infrastructures (PKI) or all pseudonyms and credentials in "all or nothing" (AoN). The *number of use* while ensuring unlinkability is another important parameter: to build history, it is mandatory that credentials could be shown multiple times. *Combining credentials* when signing is another important issue. Indeed, a document signed by a doctor that have seen a given patient is different than a document that is signed by a doctor and by someone else that have seen this patient. *Non-interactive signature* is necessary when the scheme is used to sign documents, i.e. when the verification of the signature has to be done off-line by an unknown party.

Conclusions

This paper introduces a *history-based signature* scheme that allows to build trust based on the evidence of previous interactions without revealing them. This scheme can be useful in two ways: first it allows anonymous signatures, e.g. some article can be signed by an *art critic that visited some museum one week ago*. Next it can be used in a challenge-response protocol to prove one's history. For instance, previous successful collaborations can be asserted. User privacy is protected by displaying only necessary attributes and by avoiding traceability.

This paper addresses privacy concerns in trust establishment protocols in terms of unlinkable credentials. However, we neither tackle selective disclosure control, i.e. choosing which attribute value can be revealed without threatening the privacy, nor anonymous communications, i.e. it is assumed that a privacy-preserving communication infrastructure exists. Further work will investigate schemes that jointly address privacy-preserving communications, unlinkable credentials, and statistical disclosure control in order to establish trust from scratch while preserving privacy.

References

1. G. Ateniese and G. Tsudik. Some open issues and new directions in group signatures. In *Proceedings of Financial Cryptography'99*, volume 1648 of *LNCS*, pages 196–211. Springer-Verlag, 1999.
2. Stefan Brands. A technical overview of digital credentials. Technical report, Credentica, 2002.
3. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer-Verlag, 2001.
4. J. L. Camenisch and M. A. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO '97 Proceedings*, volume 1294 of *LNCS*, pages 410–424. Springer-Verlag, 1997.
5. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
6. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Rfc 2693 – spki certificate theory, 1999.
7. Zulfikar Amin Ramzan. Group blind digital signatures: Theory and applications, 1999.
8. L. Willenborg and T. de Waal. *Elements of Statistical Disclosure Control*, volume 155 of *Lecture Notes in Statistics.* Springer Verlag, 2000.

Establishing Trust with Privacy (Transcript of Discussion)

Laurent Bussard

Institut Eurécom

This talk is about trust establishment with privacy, so I will briefly explain what we mean by trust, what we mean by trust establishment and why privacy is important in such a context.

There is one, more or less common, definition of trust: it is a belief that one can depend on something or someone. When we go to a more precise definition everybody disagrees; but at least there is the notion of uncertainty, the notion of risk. So, what we define as trust is the data we need to collect in order to give rights to someone. If A trusts B then A will grant some rights to B.

There are basically two types of trust establishment. The first, and more common one, is to establish trust on pre-existing, or *a priori*, trust by means of some derivation or delegation mechanism. So when A trusts B, or some group G, it's a given relationship, then A will trust C if B says that C is trustworthy. So B can give friends and e.g. web of trust in PGP, or B can believe some authority, e.g. certification authority is the key of public key infrastructures. Those solutions are generally based on some form of authentication.

Another approach is to build trust from scratch, without *a priori* trust. This is more and more necessary in privacy computing, the floor on which I'm walking appears to be floor and so on. So when trust is built from scratch, there are two approaches. The first is a kind of statistical approach – if 100 people say that I'm nice we can assume that I'm a nice person – generally called reputation. We propose another approach, which is to use some provable evidence in order to create some history. If I can prove to you that I went to this place, that I was in France, and so on, it will increase the trust you will have in me.

The principle is as follows. There is some interaction between A and B. A will be able to prove to B, or to another entity, that this interaction occurred in the past. So if that's sufficient for B and for this group, then this group will grant some rights to A. The problem with this approach is that it's strongly privacy threatening because I will reveal what I have done in the past in order to build trust. Often my history will be too precise, and it will be possible to link different types of information. I reveal first to you some information to build trust with you, and different information to another person, so both together can combine this information in order to bring my whole history and get more information.

So what we want to do is to ensure privacy in these two cases. First we want to ensure unlinkability, so we cannot link the proof of history I am giving to a specific interaction. We also want to ensure anonymity, and untraceability, meaning that A cannot be traced by the person to whom he showed some proof of this. The mechanism we use is a signature scheme. B can be an authority, as

I said previously, or the key. So it is as follows: this person goes to some service, e.g. the location-stamper, and gets a history address, or some credentials, that prove that he was in this place at this time. And subsequently this person has a different set of history credentials. He will be able to prove to someone else that he has done something in the past in order to get a new credential, or he will be able to use directly part of the credential to sign some document, for instance, he could sign a web page as a reporter that was in this room at this time, assuming that there is some location and time stamp service.

So we want proof of context, some kind of group membership. I shall not go into detail about our scheme but basically what we have done is an extension of the group signature, where we do not sign as a member of group G but we sign as someone with a given history. So as it's a signature, it can be used as a proof in a challenge response protocol, or it can be used as a non-interactive signature. It's anonymous, and it's non-transferrable, in fact it's equivalent to the private key of the user. We can selectively disclose some attributes, for example if I have the proof that I was in this room at this time, I can prove that I was in the UK at this time. The credential can be shown more than once, and we can combine multiple credentials.

So, to give an example, if Alice receives credentials saying that she was today at the Security Protocols Workshop, then she will be able to sign as someone that was in this place at this time, or as someone who was in Cambridge on Monday around noon, or as someone who was in the south of England on the 25th April, or combine with other credentials to sign as a member of group G that state that you're the person.

So to conclude, we are able to establish trust by protecting the server's privacy. It's an interactive construction of the provable history, there is no link to identity, and no link to specific events. When I show you that I was in the UK today there is no link to the fact that really I was in this place. Future work will certainly include some high level view of this, in fact we have the possibility to use statistical disclosure control for that kind of thing.

I think that strong privacy is really required because when we collect data, some data will be revealed. There will be people inside that will sell this data, and so on, and it's really difficult to prove. There is no non-revocation, it's really difficult to prove that this data is coming from a given entity. If someone raided my address I would not be able to prove that this address was swindled by this company or another one. But another reason why it is really important is that with privacy computing there are more and more daily interactions, and we need fine-grained trustability, for instance a printer.

Unfortunately strong privacy is very expensive, there is a cost in terms of computational complexity, there is a communication cost when you are talking about these approaches, and there is a deployment cost, for example, to lose two days to provide a MAC address which you are trustable on.

So my question is, who will pay for that kind of untraceability?

Andrei Serjantov: You say that collected data can't be tagged, but there's enough entropy in addresses: various people have inserted various bits of difference in their addresses to see where the actual address information was collected from. That provides a certain amount of entertainment if you're into such things. So then, if you will get it back, there's a bit more hope for tagging data. If you don't get it back, coming back to my insurance premiums example in Massacci's talk, well then you just don't have it. I totally agree with that.

Reply: You are thinking of some kind of water marking of the data that I provide.

Andrei Serjantov: Well it doesn't have to be as difficult as that, if you just insert some entropy into your address then that's enough.

Without extra entropy you can't do it for other things than addresses. Phone numbers, for instance, you know are probably more of a problem.

Bruce Christianson: Usually if you put digits on the end of a phone number it will still work.

Andrei Serjantov: Right, but you as the user being called probably won't discover that.

Bruce Christianson: Only if you can have access to the underlying data.

Andrei Serjantov: You can request it, so there's some mileage in that.

Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System

Bogdan C. Popescu, Bruno Crispo, and Andrew S. Tanenbaum

Vrije Universiteit - Department of Computer Science,
Amsterdam, The Netherlands
{bpopescu, crispo, ast}@cs.vu.nl

Abstract. In this paper we describe Turtle, a peer-to-peer architecture for safe sharing of sensitive data. The truly revolutionary aspect of Turtle rests in its novel way of dealing with trust issues: while existing peer-to-peer architectures with similar aims attempt to build trust relationships on top of the basic, trust-agnostic, peer-to-peer overlay, Turtle takes the opposite approach, and builds its overlay on top of pre-existent trust relationships among its users. This allows both data sender and receiver anonymity, while also protecting **each and every** intermediate relay in the data query path. Furthermore, its unique trust model allows Turtle to withstand most of the denial of service attacks that plague other peer-to-peer data sharing networks.

1 Introduction

Freedom to exchange information derives from the freedom of speech; unfortunately, there are many countries where this basic human right is not guaranteed. Turtle is a peer-to-peer data sharing architecture that makes very hard to restrict the freedom to exchange information by either technical or legal means.

When designing Turtle, we were inspired by the way people living under oppressive regimes[1] share information deemed “hostile” by their government (this can be books, newsletters, video and audio recordings, or even political jokes). Because of the potentially very serious consequences raising from being caught possessing/distributing such material, no single individual is willing to share it, except with close friends. Experience has repeatedly shown that, even in the most repressive environments, this “friends-to-friends” delivery network is remarkably effective in disseminating information, with relatively little risks for the participating parties; if one chooses his friends carefully, the chance of being caught doing the forbidden exchanges becomes very small.

The idea behind the Turtle is to take this “friend-to-friend” exchange to the digital world, and come up with a peer-to-peer architecture allowing private and secure sharing of sensitive information between a large number of users, over an untrusted network, in the absence of a central trust infrastructure.

[1] One of the authors was born in a country where the government’s track record on civil liberties used to be less than stellar.

The rest of this paper is organized as follows: in Section 2, we give a high level description of the Turtle architecture, including the protocols for query propagation and result retrieval. In Section 3 we look at technical, security and ethical implications of our design choices. In Section 4, we review related work, and in Section 5 we present our conclusions.

2 The Turtle Architecture

To bring the discussion to a more formal level, we will introduce a system model. For the Turtle architecture, this consists of a large set of nodes $\mathcal{N}$, and a large set of data items $\mathcal{D}$. We assume that behind each Turtle node i there is a human user (the node's **owner**) who has a subset D_i of all items in $\mathcal{D}$, and is interested in obtaining more. However, a user owning a node i is willing to share his data items only with nodes owned by people he trusts - we denote this as $i's$ **friends subset** - F_i. We assume the friendship relation is commutative, for any two nodes i and j, if i is in F_j, then j is in F_i. However, friendship is not transitive (the friend of a friend is not automatically a friend).

Each data item d has an **attribute set** A_d associated with it. The attribute set consists of a number of *attribute=value* pairs describing certain properties of the data item, and are used when evaluating user queries. These are logical expressions consisting of a number of *attribute=value* pairs, connected using the *AND, OR* and *NOT* logical operators. A data item matches the query if the *attribute=value* pairs in its attribute set satisfy the logical condition in the query expression.

Each user establishes a cryptographically secure connection between its Turtle node and all the nodes in its friends subset. Since there is no central trust infrastructure, the shared secrets needed to establish these secure connections have to be agreed-upon by out-of-band means (this can be done using common knowledge based on common past experiences - after all owners of friend Turtle nodes are assumed to be friends in the real-life!). Once established, the inter-friends secure communication links are used for exchanging data items and propagating user queries.

Users search for new data items by sending queries to the Turtle network. The user starts by introducing a query expression and a query depth through the query interface of its Turtle node. The node then creates a 128 bit **query ID**, by generating a 64 bit random number and appending it to the most significant 64 bits of the SHA-1 hash of the query expression (treated as an ASCII string). In this way, the probability that two distinct queries will have the same query ID is extremely small. Once it has the query ID, the node constructs a query packet containing the query expression, the query ID, and a hop count, initially set to the query depth. The query packet is then broadcast over the "friendship" links up to the desired query depth. Upon receiving a query packet, a Turtle node first evaluates the query expression against the attribute sets of all the data items in its data subset. If matches are found, the node reports them back to node that has forwarded (possibly originated) the query. Furthermore, the node decrements the hop count in the query packet, and if the count is still positive, the packet is further forwarded to all the node's friends (except the one from which the packet came).

We can see that propagating a query in the Turtle network generates a *query broadcast tree* rooted in the node originating the query, tree that follows the trust relationships among the Turtle users. The query broadcast tree is also used for delivering query answers, which travel hop by hop up the tree until they reach the root. In order to match queries with answers, each node maintains a query table with queries it has forwarded but for which the query answer process has not yet completed. Each query table entry corresponds to a query broadcast tree the node is part of; table entries are indexed by query ID, and store the address of the node's parent in the tree, the time the query has been received, and a result section, storing all the response packets the node has received from its children nodes in the tree.

A query response packet consists of the address of the responder, the *Final* bit, the query ID, a response hop count, and possibly a *response payload* consisting of number of data attribute sets. The *Final* bit is used for differentiating between partial and final answers. A node receiving a positive answer (query hit) from one of its children in the query broadcast tree, will immediately report it to the parent. A node indicates it has no more answers to forward by sending his parent a response packet with the *Final* bit set. This can happen in the following circumstances:

- The node receives a query packet with a query ID that matches one already present in the query table (we call this a *collision*). Since it is very unlikely two different nodes will generate the same query ID, the most likely cause for the collision is a cycle in the friendship graph which has routed the same packet back.
- A node receives a query packet with a zero hop count (no more forwarding necessary). The node responds with a final packet which also includes the attribute sets of the local data items matching the query (if there are any).
- The node has received final answers from all its children in the tree, and has finished processing them.

A response packet with the *Final* bit set to zero is a partial answer. The following rules apply to partial answers:

- A node receiving a query matching some of the elements in its data set creates a partial answer with the response hop set to zero, and a payload consisting of the attribute sets of all data items that match the query.
- A node receiving a partial answer from one of its children in the tree **changes the responder's address in the packet to its own address**, increments the response hop count, and forwards it to its parent node. The node also keeps a copy of the original response packet (as received from its child) in the response section of the query table (this is needed for the data retrieval phase, as we will see shortly).

The query completes after the originating node has received final answers from all its friends. At this point, the originating node has also accumulated all partial answer packets. The node then sorts through all these partial answers to identify all distinct data attribute sets; these are then presented to the user, much in the same way as the results of a Web search engine query (they can be ranked based on frequency, or hop distance). Once the user selects the result she is interested in, the node can start the *data retrieval* phase.

The retrieval phase consists of selecting a *retrieval path* and propagating the query result along that path. For a given data element d (identified by its attribute set A_d), the retrieval path is the shortest path in the query tree between the root and a node that has d. This path is determined hop by hop, starting with the root node which searches through all response packets and selects the one that has A_d in the payload, and the smallest response hop count. The root then asks the friend from which the selected response packet has been received to retrieve the A_d data item. The friend follows a similar procedure to find the next hop in the retrieval path, and so on until the retrieval request reaches the node that has the actual data item. The data item is then sent to the requester, following (hop by hop) the the retrieval path in reverse order [2].

3 Discussion

There are two basic assumptions we make when proposing the Turtle architecture. First, we assume that continuous, high-speed Internet connections will become ubiquitous in the near future. Looking at current trends, which show increasing DSL/cable modem penetration in the consumer market, not to mention the ever-increasing wireless "umbrellas" that cover large parts of big cities, this first assumption seems very reasonable. The second assumption is that for sufficiently large social communities (a college campus, a country, the world), "friendship" relationships form fully connected graphs. Validating this assumption would obviously involve large-scale sociological experiments, which are beyond the scope of this paper. However, based on the famous "six degree of separation" experiment [15], the moderate success of the PGP infrastructure, and, more recently, the explosive popularity of the "Friendster" service [1], we have reasons to believe that for relationships involving moderate amount of trust, it is very likely to achieve full "friendship" graph connectivity.

3.1 Technical Considerations

We expect that our main application scenario (dissidents exchanging sensitive information) would mostly involve small data items. For this reason, we have designed Turtle as a packet routing overlay. However it should be straightforward to re-design it as a circuit switching network, so it could accommodate large data transfers. The only significant change in the protocol would concern the retrieval path, where nodes would have to establish a "virtual tunnel" connecting the source and the destination. This would also require dealing with issues such as flow control, buffering and quality of service.

[2] This is slow but safe, hence the name "Turtle".

Second, overall system performance can be greatly improved if data items are cached. This can be complemented by "gossip-like" mechanisms that allow nodes to quickly disseminate information about which data items are popular, in order to implement smarter cache expulsion algorithms.

Finally, because the unique trust properties of the overlay (only nodes that trust each other directly interact), it is very easy to enhance Turtle with an economic model that would encourage cooperation and sharing. For example, when sending back a response packet, a node can also include a price tag for supplying the given item, with each node in the broadcast tree adding his "relay fee" to the price tags received from its children. The query initiator can then use the final price tag as an additional selection criteria when deciding which data item to request. These payments can be aggregated over longer intervals, by having Turtle nodes keeping track of the amount owed to/owed by friend nodes. A node can then periodically report the "balance sheet" to its owner, who can settle the matter with his friends by out of band means (e.g. cash exchange).

3.2 Security Implications

From a security point of view, the Turtle architecture raises a number of interesting points:

First of all, Turtle offers good query initiator and query result anonymity: both initiator and responder are known only by their respective friends subsets (trusted nodes). With small modifications in the query/result routing protocol - namely removing the hop counts - it is also possible to achieve complete sender/receiver anonymity. Because all information exchange is done over encrypted channels, the only way for an adversary to link a query initiator to a responder is through traffic analysis. However, there are well known techniques for protecting against traffic analysis [12,7], which can be easily incorporated in our basic query/result routing protocol.

Second, the Turtle network is immune to the "Sybil" attack [6]. Even if a powerful adversary is able to create a large number of malicious Turtle nodes, the effect these nodes have on the correct functioning of the system is minimal, unless the attacker is also able to infiltrate his nodes in the friends sets of correct nodes (but this would require a lot of social engineering!).

Third, Turtle exhibits a very desirable fail-mode property - "confined damage" - meaning that a security break in one correct Turtle node only affects a small subset of all correct nodes in the system (in this case the node itself plus its friends subset).

Finally, due to the way the Turtle overlay is organized, denial of service attacks typical for a peer-to-peer network - such as malicious routing [2], content masquerading (content that does not match its description), bogus query hits (a node answering positive to a query even when it does not have any matching content), and aborted transfers - are much less likely to happen. Because all direct interactions take place between nodes controlled by people who trust and respect each other ("friends"), we expect incentives for random malicious behavior to be very much reduced.

3.3 Ethical Implications

Turtle allows private sharing of data, and protects data providers, data consumers as well as intermediaries from the potential negative consequences stemming from being identified as participants in this data exchange. The main motivation for our work is providing citizens living under oppressive regimes with a safe, private information sharing network. However, it is clear that this same technology can be used for activities of dubious ethical value - such as illegal sharing of copyrighted digital products.

There is a lot of debate regarding the legality/morality of sharing copyrighted information over peer-to-peer networks. It has already been pointed out [5] that the benefits copyright laws have brought to society in the printing press age no longer apply in the case of digital content. It remains to be seen how copyright laws will be changed to adapt to digital technology; nevertheless, it is our opinion that widespread use of fast and efficient file sharing networks such as Kazaa and Gnutella (where almost 100% of the traffic deals with illegal sharing of copyrighted material), if left unchecked, will eventually drastically reduce the incentives for artistic creativity. However, using a Turtle-like network for such activities, will have a less drastic impact. Because of the way Turtle is designed, the speed of the data retrieval is dictated by the bandwidth capacity of the slowest link in the retrieval path. Thus, using Turtle to share large items will be most likely considerably slower than in Kazaa and Gnutella. This "un-ease of use", combined with the fact that at any moment data is only exchanged between friends (thus, the only way to legally pursue copyright infringers would involve morally despicable strategies, such as forcing people to "rat" on their friends), may actually help in defining a new legal framework on what constitutes "acceptable sharing" and "fair use."

4 Related Work

What makes Turtle unique is the bottom-up way it builds its overlay starting from *pre-existing* trust relationships among users. To the best of our knowledge, there are no other peer-to-peer systems that employ this technique.

However, the idea of indirect routing of requests and results in order to achieve sender/receiver anonymity has been around for some time. Chaum [3] has first suggested it for building a mix network for anonymous e-mail delivery. Crowds [12] builds a peer-to-peer mix network for anonymous Web browsing. The same idea is more recently employed in [11] and [7] for building general-purpose anonymizing network layers.

In the context of peer-to-peer data sharing networks, indirect routing for queries and results is being employed by Freenet [4] with the aim of creating an "uncensorable and secure global information storage system". However, a Freenet node acting as a relay in a "forbidden exchange" can only achieve a limited degree of plausible deniability. This relay node exposure, combined with the lack of a-priori trust relationships among interacting nodes, makes Freenet a hard sell for a "dissidents network." The same considerations apply to other anonymizing, censorship-resistant peer-to-peer systems [13,8,10,14,9]: none of them manages to offer an acceptable level of protection to **each and every** entity in the request/retrieval path.

5 Conclusion

In this paper we have described Turtle, a peer-to-peer architecture for safe sharing of sensitive information. In order to achieve strong privacy guarantees, Turtle organizes the data sharing overlay on top of pre-existing user trust relationships. This protects the privacy of both data senders and receivers, as well as the intermediate relay nodes that facilitate the data exchange. Furthermore, Turtle is resistant to most of the denial of service attacks that plague existing peer-to-peer data sharing networks.

As for directions for future work, we have already mentioned possible extensions of the basic query protocol to support retrieval of large data items (through hop-by-hop virtual circuits) and economic models that would encourage cooperation and sharing. It would also be interesting to look at ways to associate sensitivity levels to data items and different trust levels to different friends, which would allow node owners to specify more complex security and privacy policies regarding the data items they share.

References

1. Friendster Web Site. http://www.friendster.com.
2. M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proc. OSDI'02*, Dec. 2002.
3. D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Comm. of the ACM*, 24(2), 1981.
4. I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. Int. Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 46–66, 2001.
5. James A. Dewar. The Information Age and the Printing Press: Looking Backward to See Ahead. http://www.rand.org/publications/P/P8014/P8014.pdf.
6. J. Douceur. The Sybil Attack. In *Proc. of the IPTPS '02 Workshop*, Mar. 2002.
7. Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. of the 9th ACM Conf. on Computer and Communications Security*, November 2002.
8. S. Hazel and B. Wiley. Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer Publishing Systems. In *Proc. of the IPTPS '02 Workshop*, Mar. 2002.
9. A.D. Rubin M. Waldman and L.F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system . In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.
10. R. Dingledine, M.J. Freedman, D. Molnar. The Free Haven Project: Distributed anonymous storage service. In *Proc. Int. Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 67–95, 2001.

11. Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE J. on Selected Areas in Communications*, 16(4), 1998.
12. Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
13. A. Serjantov. Anonymizing Censorship Resistant Systems. In *Proc. of the IPTPS '02 Workshop*, Mar. 2002.
14. M. Waldman and D. Mazieres. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proc 8th ACM Conf. on Computer and Communications Security*, pages 126–135, 2001.
15. Duncan J. Watts. *Small Worlds, The Dynamics of Networks between Order and Randomness*. Princeton University Press, Princeton, NJ, 1999.

Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System (Transcript of Discussion)

Bogdan Popescu

Vrije Universiteit, The Netherlands

Today I will talk about a project which aims at designing a peer-to-peer network for safe and private data sharing. The motivation for this work is a development that threatens to shut down peer-to-peer file sharing networks, and that's a recent tactic by the recording industry to take legal action against peer-to-peer type networks. So first I want talk about the peer-to-peer file sharing phenomenon: in general, its origin, some of the positive social aspects of such a thing, and the tactical attacks that a peer-to-peer network is subject to. I will then focus on a specific attack that motivates our work, namely illegal users being sued, and discuss possible defences. Our solution, which we call Turtle because as you will see, it is slow but safe, cannot reach the performance of general existing file sharing networks, but at the same time we think it does a good job in protecting users against legal harassment.

Peer-to-peer file sharing is a phenomenon that started around 1999 with a company called Napster. It was really a place for facilitating people to directly exchange music in the form of mp3 files, and it was very popular from the beginning because people could get everything for free. At the same time it was also highly controversial because most of this music was actually copyrighted, so the recording industry perceived this as a major threat, and subjected peer-to-peer networks from the very beginning to multiple attacks. Before moving on to that maybe I should answer the question, are peer-to-peer networks any good, should we even work on protecting something that is mostly used to infringe copyrights? And actually I have some good to say of peer-to-peer networks, probably the most important property of such networks is that information cannot be censored. Basically once a piece of data is injected in such a network it takes a life of its own, and it stays there as long as there are people interested in that thing.

We use the example of protecting people sharing music because this is what mostly happens today, but it's easy to extend this to an example where people use a peer-to-peer network to, lets say, share independent political views, and I think it's worth the effort to work on protecting people if they suffer harassment.

So what types of attacks are there against a peer-to-peer network. The simplest attack is to go against the company for running a query service, and this is actually what brought down Napster. To counter this just move to a system such as Novello Kazaa, which has query processing, and this is what's going on today.

Another type of attack is to sue the company that writes the client software, and such an attack has been attempted against the developers of Kazaa in the

US, but again to counter this just move the company offshore, or underground, or whatever. Another form of attack is to attack the content and this basically means to inject bogus content into the network so that users have a hard time finding the actual relay node. There are also various attempts to do content tracing that involve counting how many users share different files, and it is also possible to counter this attack. Finally, the last attack in the arsenal is to sue illegal users, and so far more than a thousand people have been sued in the US for sharing files, and so far this is the biggest threat, and this is what we are trying to counter with our work.

Matt Blaze: You claim that's the biggest problem but, in some sense the first attacks that you mentioned seem to be more problematic, because they're attacks on an infrastructure based on the use that certain people perceive it to be put to, whereas the final attack of going against individual users is an attack on people for doing something that is at least in fact illegal. Those lawsuits may be very heavy handed, they may be misguided, and some of those prosecutions may be false, but at least they're exercising the legal system to deal with something that is in fact illegal. Couldn't we argue that attacks on the whole infrastructure based on one particular user are a bigger problem?

Reply: I see your point. The thing is, if you can sue people for sharing music. . . I mean, there are enough powerful lawyers that can go against anything – then if you share political views there are some pretty high placed people that don't like that, and they can go against you in the same way as people go against for you sharing music, it is the same model basically. There's always somebody powerful enough that doesn't like what you're doing, so if you have a system architecture that just lets powerful people attack you in this way, then they can shut down everything, I think.

Bruce Christianson: The argument is that a system designed so the simplest approach for the opponent is to attack individual users on an individual basis is flawed. That's the argument. The fact that the attacks on the individual users are legal attacks is beside the point.

Reply: Well, it's not flawed, it's what's happening at the moment.

Bruce Christianson: Well it's not optimal from the point of view of good system design.

Reply: Yes, we want to see if we can counter this, and OK, probably there's going to be a whole debate whether this is ethical or not.

Matt Blaze: The problem is that file sharing is illegal, right, and you argued that that's the problem, that the laws against file sharing shouldn't be as strict as they are. Then making it difficult to find the people who are doing it seems like a very indirect solution to the problem, the direct solution to the problem should be a change to the laws.

Frank Stajano: I believe that the point is this. Let's assume that this infrastructure, which is mostly exploited currently for file sharing here, is something some people use for protection against censorship. Now this is a system that technically allows singling out the individual users who participate in that and beating them over the head, then it's not a good enough system to protect against censorship. I want a system where you can't find the heads.

Matt Blaze: Oh I understand that argument, and I agree that that's a desirable property for a system. I was just arguing in this particular example.

Reply: Another point we noticed is that, OK, maybe we want to change the law but that's very hard. Designing a system that allows us to protect against these laws is simpler.

Ross Anderson: Well there's a paper by George and me at the 3rd Annual Workshop on Economics and Information Security, next month[1]. To minimise the incentives for censorship, and also maximise the incentives for resistance, you should have one big system that in effect is a federation of fan clubs. So if an artist decides that it's stupid to go suing all their fans, then fine, all of his fans can share the music, they can love him, they can hopefully buy more of his albums. If they think suing all their fans is what life is about, then let them sue all their fans, and their fans stop buying. Eventually they'll figure out which works work better.

Reply: Yes, and this is good example, it's not artists that use their kindness, it's their record company. I don't think the artist has that much to say about it.

Ross Anderson: But if you have a federation of fan clubs, then the linkage to the artist is there, it's clear, it's definite, and if the record company is suing a particular artist's fans and as a result the fan clubs say, we hate that artist, we're not buying his records anymore, the artist himself is constantly hurt.

Chris Mitchell: It seems to me that the message of your talk, with some caveats, could be taken as there's people breaking the law out there, and they're being prosecuted, and we want to help them avoid being prosecuted. Now I realise that's not your message, but why are you presenting it as if it is your message, using words like attack when you mean a company seeking legal redress for breaches of copyright. Why not say, my objective is to enable peer-to-peer sharing for those people who wish to do it for legal reasons, rather than to stop those people who wish to prevent it for copyright reasons. Why not present the same talk but take out all the stuff designed to annoy record companies?

Reply: Well, it is fun to annoy record companies.

Chris Mitchell: But some of us believe in the rule of law.

Bruce Christianson: Then it is also legitimate to consider the next challenge that may be faced by law enforcement.

Chris Mitchell: We can enable the exchange of information that would otherwise be censored, and I think that's a very reasonable thing to be doing, but

[1] See George Danezis and Ross Anderson, 2005, The Economics of Resisting Censorship, IEEE Security and Privacy 3(1), 45-50.

why not present it as that, why are you presenting it as an attack on record companies?

Reply: This is what people can build, and this particular attack against the users would not work if such a thing were built.

Mike Roe: We're in danger of getting stuck in an argument about the ethics of whether we should be doing this or not, rather than discussing whether or not the protocol actually works. It's always the case that new protocols are presented here in an adversary form.

Reply: Because it's easier to present a talk in this way.

Bruce Christianson: Then let's apply a bit more abstraction for the sake of the argument.

Mike Roe: Without necessarily taking sides as to which of the two participants in this protocol we think should be gaining out of this.

Matt Blaze: Here, let me make everyone happy, if we can design a system that will allow file sharing in this very hostile legal climate that file sharing currently exists in, then it will surely work well for political dissidents.

Reply: Yes, exactly. So why is attacking the user such an effective mechanism? Well research has shown that most content in such a network is supplied by a small fraction of users, and this has led to the so-called "Crush the connectors" strategy from the RIAA, which is very simple: identify users sharing large numbers of files, reveal the content, from there log these transactions, and then they can see the patterns, and the net result of this is that exchanging content with strangers now becomes dangerous because you never know when a stranger is the adversary, so we get in trouble.

So to formalise the biggest threat model, basically you assume there's a peer-to-peer network in which a fraction of all the nodes are controlled by adversaries, and what we need to do is to prevent exposing the nodes, and this exposure can happen in two situations. The first is when a good node exchanges data with an enemy node, and the second threat is passive logging, when a passive adversary is capable of logging all the data exchanged between two good nodes. This corresponds to a situation when the Internet service provider would be obliged by law to log all the traffic exchanges certain people, or all traffic, that orients from a given node.

These are the only threats we're trying to counter, we are not particularly concerned about traffic analysis, since participating in a peer-to-peer network is not by itself a crime, it's only certain types of data exchange that may lead to legal trouble. And for the same reason we don't really care about strong views of anonymity. Before discussing what we propose, I want to talk a little bit about what most people think is the solution to this problem, which is anonymous file sharing. There are a number of systems that have been designed for such purposes, probably the best example is Freenet. The way they work is to just make it impossible to identify the source and the destination of a given data exchange, and this is accomplished using primitives derived from early work on mixed nets and onion routing. And although in theory the adversary has no way to see you with such a system because he can't identify the source or the destination, in practice, as I will show next, this is false.

With an anonymous file sharing system, before reaching its destination data is routed to another peer-to-peer intermediate node, and the property here is that, in this way the source stays hidden. When a node wants to retrieve a piece of data shared by that source, it doesn't stop the source until the last relay.

Now let's see what happens when an adversary tries to retrieve a file shared by the source. First he will go to the last relay, and under our threat model the last relay is exposed, the adversary can just log the IP address of this last relay, and then take it to court for supplying something that's against the law, or undesirable, whatever. This last relay can get himself off the hook by proving somehow that it was just relayed data, but that most likely will involve exposing the previous node in the relay, and that's not desirable. What's really important is that these endnodes can be subject to enough harassment to make it unappealing for them to participate in such a network, and then you end up with very few nodes that are actually linked to a device in a network, and this is why this probably would not work. The important point here is that this exposure has not happened because any of the good nodes have not followed the proper security practices, but is more an intrinsic weakness of the system because it has this open service model. Basically anybody, including the bad guys, can access any piece of data, and this is why we think this may not work.

I will talk now about our solution, Turtle. The idea is rather simple, just make the peer-to-peer overlay based on social links. Two Turtle nodes connect to each other, if there is a social link between their users. Assume that they are friends in real life and they trust each other. And the communication between any two nodes is encrypted, and given the fact that two nodes that connect to each other in Turtle, are rather like people that know each other in real life, they can release encryption keys which fits very well with this peer-to-peer data line.

It's enough to have only one user to start such a network, one user who runs the Turtle node. He invites his friends to the network, and then they're secure between their nodes. This is what happens when a friend's friends join the network, and then friends' friends, and so on. A query is routed hop-by-hop, every node that receives a query broadcasts this query to all his friends on the link where it came from. Query results are routed differently than in the existing network systems, ours have to go back hop-by-hop, because only the friend nodes can be used, so they go back hop-by-hop, on a reverse route.

Bob Mayo: Are you assuming that I would trust someone that is ten hops away as much as the someone who is one hop away?

Reply: It doesn't matter because you never talk to somebody several hops away. You only ever talk with someone you know, you are telling something to your friend and your friend tells that to his friend, you don't deal with his friends, that's the idea. So hits are passed back to the query originator who selects one hit, and then the solution is a virtual circuit, and that's all.

Frank Stajano: You are basing the security on jumping out of the system, you're not just doing in cyberspace you're doing in the flesh. But what's to stop the adversary to put in a person, an informant, a fake file sharer in the flesh? At some point you have to bootstrap this friendship by having gone to a party together, and getting drunk, or something.

Reply: That's where you have to tread carefully now. It all depends on you. But if you are careful enough, if you select your friends carefully, then your chances to be caught are very small.

Fabio Massacci: The nodes at one end may be very careful in selecting their friend, but they have a very different idea of what is a good friend from you so you can be caught anyhow.

Reply: Well it doesn't matter, because if I select my friends carefully, it's only my friends that know what I'm querying, and what I request, it's only *my* friend that can log what I want.

Bruce Christianson: The argument that's at stake here is the containment of failure, the idea is that if you foul up by selecting the wrong people then you only put yourself at risk, you can't put at risk the friends of your friends. This is potentially anti-transferable.

Reply: Yes.

Andy Ozment: I think it's actually worse than contained failure in the sense that if a single node is compromised, then they can follow the links from that person's machine all the way through the network. If we assume rubber-hose analysis here, they're not too picky about evidence, they can actually just...

Bruce Christianson: You might be guilty so we're going to sack you now.

Andy Ozment: Further, the problem with this is, I may be friends with you because you and I have the same taste in music, but we may have very different political tastes. The problem is when friendships are formed, and links are made, based on one assessment of friendship criteria, but people then use those links based on a totally different assessment such that, the friend of my friend may hate my political views so much he's willing to report me, or report this activity within the network, because he just joined this network to get music not to hear dissident political views.

Bruce Christianson: Or he may share your political views but really hate your music.

Reply: I think everybody has a limited number of people who they trust reasonably well not to turn them in no matter what, unless they really do something nasty.

Matt Blaze: It sounds like this has the unfortunate property that this is useful for file sharing by people in reasonably democratic governments where there's some rule of law, because you can't be forced to simply point fingers at people with no personal knowledge of wrongdoing, but quite weak against a repressive government that can beat people up and intimidate them without the rule of law limiting what they can do.

Reply: You are correct now, this doesn't work in totally oppressive regimes there has to be a certain amount of respect for the law. But I think there are many cases like this, I mean there are other things which you'd want to share.

Mike Bond: I think there is a fundamental problem here with this architecture of sharing certain types of information. What if this system allows people to get in contact, and to get, in a relatively easy way, data from people they didn't know too well without having to make good friendship bonds. So I'm friends with my first friend, we can exchange music if we want to, I'm not friends with anyone who's got Star Trek episodes.

Reply: Yes, you just use the six degrees of separation argument.

Mike Bond: You need a way to find the guy who's got the Star Trek episodes without telling your friend on the way that you like Star Trek.

Andy Ozment: Your arguments are that we're safe because all of our friends trust each other in the same way, and that we can get everything because all of friends know somebody who likes something completely different. These arguments, the argument for connectedness, and the argument for security, run very much counter to each other.

Chris Mitchell: A slightly different way of looking at this is, if your objective is to make available news which would otherwise be censored, what's the advantage of this over posting the news on newsgroups, or as many newsgroups as you can find?

Reply: Well we can censor it. It hasn't happened but in theory, if people know that keeping something on their own machine can lead to nasty consequences they're not going to be motivated to do this.

Bruce Christianson: Suppose the model is not just news but comment.

Reply: Exactly.

Ross Anderson: There's a curious possible side effect of a network like this. The latest econometric research suggests that there is no net cost to record companies for file sharing, they lose some sales but they make other sales. If I decide that I want to sample Northumbrian pipe music, I go to a file sharing system, I get someone, and say, yes, that's really nice I'll buy some CDs.

So the arguments for the record companies is that it is if that their enforcement causes people to move to semi-closed systems like Turtle it will cost them money.

Bruce Christianson: Even Turtle can have a search algorithm that works by sending out a key word that's meaningful to Trekkie fans, but to no-one else, and a Trekkie will respond to it.

Reply: I was going to talk a bit about the query ID protocol. Basically every query has this 128 bit random query ID, and every node has to remember the query, and the nodes also have to remember from which other node the hit ID came from, and this is for being able to establish this virtual circuit. Whilst the query issuer has a route to the node that actually has the data, he establishes the virtual circuit step-by-step, without actually knowing more than the next link. So you only have to talk with your friend, you don't have to talk to anybody else.

George Danezis: You could optimise this, because after the tenth hop, it doesn't matter anymore to reveal who the node currently having the query is, because it could have been originated by anyone. So can you optimise after coming back to this exit node, or would that break the protocol?

Reply: Well that may lead to a problem, it exposes the last node.

George Danezis: But it doesn't matter because it could be anyone.

Reply: Yes, it can be anyone but imagine you are the last node and that you have this lawyer coming to your house, saying you have been sharing this data, and you have to come to court. You are not very likely to participate anymore. You can get yourself off the hook by revealing the people who relayed this, but you don't like the harassment, and you are not the first relay in the network.

So what are the security properties of Turtle? Most importantly, each user is his own trust group. Basically, each user decides who his friends are, and this has a number of advantages, the most important one being this local identity which we discussed earlier. If you select your friends carefully the chances to get into trouble are quite small. Another interesting property is that such a network could be immune to the secret agent attacks: with these the adversary just creates a very large number of peer-to-peer nodes, and injects them into the network. In this case the adversary would not only have to create bogus nodes but also create real world identities for them, and make them friends with real people, and that's clearly very hard. There is also good protection against denial of service attacks, when a user injects bogus content, or takes down this node the people he would most likely hurt would be his friends, so we believe that motivation for malicious random behaviour would be reduced in this system compared to existing peer-to-peer networks.

So now the biggest question, will this work? The first issue is connectivity, if we just follow the social net we'll get the same coverage as with typical open peer-to-peer networks.

The second question is, are people on-line enough to keep this social graph. Again that shouldn't be any problem because ADSL is becoming very widespread, and it's not uncommon for people to have their computer on-line all the time. The biggest question is, can people who have lots of friends keep the social graph connected? The question is whether these connectors will be able to cope with having to relay a large fraction of all the data that's been exchanged. We don't know yet if this is going to work or not. Our idea is to probe this social graph, and then run some sort of simulation, basically see how we are starting to perform against a normal protocol such as Kazaa.

Frank Stajano: Can you explain more clearly why the fact that you're exchanging keys off line provides confined damage?

Reply: No, exchanging keys off-line makes no difference, what makes the difference is the fact that you always establish connections to people you trust. The point of exchanging keys off-line is that you don't need a key server, it's totally decentralised.

Frank Stajano: If someone is targeted by the RIA lawyers, what stops these lawyers from saying under subpoena tell us all the people you have linked with, and so on recursively.

Reply: Well, I don't know the law very well, but I think they would have to prove that you have done something illegal to pursue a subpoena for something like this.

Bruno Crispo: But the way in which the system is designed it's more difficult to penetrate the system rather than, for example, Freenet.

Ross Anderson: There are rate limits on our lawyers so courts can only handle so many subpoenas, assume that a lawyer can subpoena a thousand people a year and that's about it.

Bruce Christianson: The property that Turtle has, is that in the other systems it's reasonably straightforward for an agent provocateur to act as a distribution node and to compromise a very large part of the system, whereas here it's much harder.

Andy Ozment: Of course it also increases the value of penetration. Let's say that we have a group of dissidents and we keep our friendship links based entirely on dissidence, then penetrating the network is slightly more difficult, but vastly more valuable because here you have a chain of dissidents and you can just follow up the electronic evidence. Essentially you create a system where you have this electronic memory that we discussed earlier this morning[2], but you remove plausible deniability because you have this connection evidence on an individual's computer.

Bruce Christianson: Your argument is absolutely right if you have a one-issue network, if it's all political opinion, or all "The Darkness is a good band." But using the same network for multiple purposes undercuts that analysis.

Reply: It's also what Matt was saying. If you're in a totally evil regime, then just the fact that you are sharing encrypted stuff with somebody else will lead to trouble.

[2] Bohm et al., *Controlling Who Tracks Me*, these proceedings.

Bruno Crispo: Yes, it's really for dissident networks because essentially you do a new key distribution every time you use the network. My neighbour doesn't have the key to my data.

Ross Anderson: This model could be combined with sufficient node-level deniability. Bruno asks me for a CD by an unpopular band, and I relay the request unknowingly to Frank who provides it, and somebody comes along and says, this Italian song is pornographic, and I say, start again, I don't speak Italian. If that provides me with enough deniability...

George Danezis: Lots of people have mentioned that this might work in soft places, and not hard places. The main problem would be that by traffic analysis you would be able to tell who is participating, but if participating itself is not incriminating because lots of people use it to share lots of different files...

Bruce Christianson: Yes, that's the assumption.

George Danezis: Then it will be truly difficult to penetrate, because beating people up is actually really expensive. If you have a small degree graph, and let's say it takes one day for someone to be beaten up, then it is one day for the very few people round it to go underground, or to start burning their hard disks, or whatever, so it is extremely difficult for the adversary.

Bruce Christianson: But the crucial point is the one made earlier, so long as most of the traffic is for one particular issue, it's much, much harder for the attacker to isolate individuals.

Stephen Murdoch: If one of your friends reported you to the adversary, whoever they are, can you find out who that person was so that you can maybe warn your other friends or harm the reputation of your ex-friend.

Reply: In a reasonably democratic society, if your friends rat on you, a few probably have to go to court and testify against you and then you'll see who testifies against you. But there's no electronic way to detect that.

Bob Mayo: The model for this is the need to establish trust with people. If you violate their trust, you're known, and you will be found at the bottom of the river or something. Presumably not just over one song. [Laughter]

Bruce Christianson: But maybe for a really bad album!

The Dancing Bear: A New Way of Composing Ciphers

Ross Anderson

Cambridge University

Abstract. This note presents a new way of composing cryptographic primitives which makes some novel combinations possible. For example, one can do threshold decryption using standard block ciphers, or using an arbitrary mix of different decryption algorithms – such as any three keys out of two AES keys, a 3DES key, an RSA key and a one-time pad. We also provide a new way to combine different types of primitive, such as encryption and signature. For example, Alice can construct a convertible signature that only Bob can verify, but which he can make world-verifiable using an AES key. We can incorporate even more exotic primitives, such as micropayments and puzzles, into compound constructs.

Previously, there had been two basic ways to combine cryptographic primitives. One could either design a compound primitive, perhaps using the homomorphic properties of discrete exponentiation, or one could embed several primitives into a protocol. Neither is ideal for all applications, and both have been extremely vulnerable to design errors. We provide a third construction that also allows the designer to do new things. We show, for example, how to incorporate cyclic dominance into a cryptographic mechanism, and how it might be used in a digital election scheme. Our new construction not only complements existing ways of composing crypto primitives; it also has the virtue of simplicity.

1 Introduction – Threshold Decryption

One of the great attractions of public key primitives such as factorization and discrete log is that we can often combine them into compound cryptographic operations such as threshold decryption, electronic cash and digital elections (see for example [22]). It is widely believed that public key cryptography is indispensable for such 'fancy' cryptographic schemes, as its homomorphic properties enable crypto primitives to be linked, blinded, and shared.

This belief is mistaken. We now show how to do threshold decryption using conventional crypto primitives such as AES and SHA, in a way that is not only practical, but inherits an established security proof. We will then go on to discuss the wider implications.

1.1 Background and Notation

At FSE 93, Wheeler introduced WAKE, a variable-width block cipher based on an autokeyed table lookup [27]. This was aimed at giving a concrete implementation of the $\{X\}_K$ 'curly-bracket' abstraction for X encrypted by K that one finds in the protocol literature – without the sort of problems with modes of operation documented (for example) in [6]. At FSE 96, Anderson and Biham produced the BEAR construction, which also provides a variable-width block cipher but uses the composition of standard hash function and stream cipher primitives; one can thus construct (say) a 427-bit block cipher based on AES and SHA [4]. This is secure, though in the rather weak sense that a key-recovery attack on BEAR gives a key-recovery attack on the stream cipher and also finds either preimages or collisions for the hash function.

At FSE 97, Rivest remarked that such a cipher can be used with a fixed and indeed publicly-known key as an 'all-or-nothing' or 'package' transform. This was originally proposed as a means of pre-processing plaintext to ensure that an adversary performing a keysearch attack would have to decrypt all of a ciphertext before she could check whether her current key guess was correct or not [20].

At FSE 99, Jakobsson, Stern and Yung showed that if you take a message, add redundancy and an integrity check, then scramble it using a suitable package transform, then it suffices to encrypt just a single block of it. They proved that this construction is secure in the standard complexity model: indeed, it is secure against an adaptive chosen-message chosen-ciphertext attack [16]. They used the term 'ideal length-preserving one-way function' rather than 'package transform', but that is incorrect as the function cannot be one-way. It is better described as a single pseudorandom permutation of the desired length. However, there is no terminology for the combination of random nonce, message and integrity check, processed using such a pseudorandom permutation, so we'll use the term '*grizzle*': the grizzle of M under the nonce N is

$$G_N(M) = \Pi(N, M, c(N, M)) \tag{1}$$

where M is the input message, N is a random nonce, $c(N, M)$ is a checksum and Π is a fixed, publicly-known pseudorandom permutation for each length of input string. In a concrete implementation we might have $c(N, M) = \mathrm{SHA}(N, M)$ and $Pi(x) = \mathrm{BEAR}(0, x)$, that is, x BEAR-encrypted using the zero key.

We will need some more notation. We'll write X_i for the i-th block of X, and assume that the block length b is clear from the context. Thus a Jakobsson-Stern-Yung encryption might be $\mathrm{JSY}(K; M) = \{G_N(M)_1\}_K, G_N(M)_2, \ldots, G_N(M)_k$: we grizzle the message M by the nonce N, encrypt the first block of it, and send that along with the other blocks which we do not encrypt. The Jakobsson-Stern-Yung result is that $\mathrm{JSY}(K; M)$ is just as secure as a classical encryption $\{M\}_k$.

1.2 Basic Construction

The first observation is that we can easily do shared decryption, in which n keyholders must each use their key to decrypt a message. Here, we grizzle the message as before and get each keyholder to encrypt a different block of it. Thus if keyholder i has key ki, the ciphertext is

$$\begin{aligned} c(k1, k2, \ldots, kn; M) = \{G_N(M)_1\}_{k1}, \{G_N(M)_2\}_{k2}, \ldots, \{G_N(M)_n\}_{kn}, \\ G_N(M)_{n+1}, \ldots, G_N(M)_k \end{aligned} \tag{2}$$

In other words, we encrypt the i-th block of the grizzled message $G_N(M)$ with the i-th key and transmit these blocks encrypted, together with the rest of the grizzled message unencrypted. (We assume here and in what follows that the message is long enough, that is, $k \geq n$.)

The Jakobsson-Stern-Yung proof goes across; from the viewpoint of each keyholder, the construction is secure against an adaptive chosen-message chosen-ciphertext attack, whether in the distinguishability or valid-pair-creation model. (In JSY terminology, valid-pair creation is trivially harder for more encryptions; the distinguishability proof implies that if there is one slave the attacker does not control then he cannot distinguish; it assumes nothing about the existence or otherwise of further slaves.)

It was always possible, of course, to perform shared decryption using secret-key algorithms if one used n stream ciphers, or with block ciphers if one were prepared to tolerate constraints on the order of decryption. So far, grizzling seems nothing revolutionary.

The second observation that we can get an (n-1)-out-of-n threshold decryption mechanism by adding a parity block. With the above notation, we compute a simple xor-checksum of the blocks that will be encrypted:

$$C(M) = \oplus_{i+1}^{n} G_N(M)_i \tag{3}$$

and send it along with the message. As soon as $(n-1)$ of the keyholders have decrypted their block, the remaining block may be trivially recovered by xor'ing $C(M)$ with the $(n-1)$ plaintext blocks and the message can thus be decrypted. If fewer than $n-1$ keyholders collude then no information about the plaintext can be recovered.

The third observation is the general case, that is, one in which any m out of n keyholders can decipher the message, but we do not wish any smaller subset of keyholders to obtain any information about the plaintext. To do this, we apply a suitable block-erasure error-correcting code to the grizzled plaintext. One may use an (n, m) Reed-Solomon code over an alphabet of size $q = 2^b$; such codes exist for $n \leq q$ [18]. A more sophisticated solution, which reduces the computational effort required for decoding with large n and m, is given by the recently-developed digital fountain codes of Byers, Luby and Mitzenmacher [7]. Yet another possibility is to use a secret-sharing scheme.

2 General Composition of Crypto Primitives

The above section showed how a combination of grizzling and block-erasure coding enables us to construct threshold-decryption schemes whose underlying crypto primitive is a standard block cipher such as AES. However, we nowhere assumed any property of this cipher, except that the encrypted blocks are the same length.

We can therefore use our construction with quite heterogeneous decryption algorithms, so long as we arrange to use equal block lengths. For example, we might use 2048-bit RSA for some shares, 16-block CBC-encryption with AES for others, and 32-block CBC-encryption with DES for still more. (The use of nonces in the grizzling construction makes IVs unnecessary.)

Heterogeneous keying of threshold primitives appears to be new, and may have uses in backward-compatibility applications. These are notoriously difficult to design in robust ways, and many attacks have been found that exploit backwards-compatibility features, in both protocols and APIs (see, for example, [5,26]).

We'll now look at a few specific examples of new tricks.

2.1 Designated-Confirmer and Convertible Signatures

As well as using different primitives that implement the same function, such as encryption, we can use our construction to compose different functions. Consider for example designated confirmer signatures [11], which allow one or more designated parties to confirm the authenticity of a digital signature without interacting with the signer. Such a construction is now almost trivial: Alice grizzles the message, signs one block of it, then encrypts another block in such a way that Bob can decrypt it. If $\sigma_{ka}(m)$ is a signature of m with message recovery, and $\{m\}_{kab}$ is an encryption with a key shared between Alice and Bob, then we might have

$$cs(M) = \sigma_{ka}(G_N(M)_1), \{G_N(M)_2\}_{kab}, G_N(M)_3, \ldots, G_N(M)_k \qquad (4)$$

It is now straightforward to arrange that the designated confirmer be any quorum of a group of keyholders, and, for that matter, that they can use any decryption scheme with which they are comfortable. This is a significant advantage of our new approach – functionality can be composed simply, compared with number-theoretic constructions where much work may be needed to combine two functions, and where many years can pass with schemes being proposed and then broken in the hunt for a new primitive (see for example comments in [15]). By making complex combinations simpler, our new construction makes them easier to analyze and understand.

3 Protocols

Number-theoretic constructions are not the only way to compose crypto. The other main technique is to use protocols. Ever-more complex functionality can be built up by successive rounds of interaction between two or more principals. However, interaction has its costs, even nowadays, and techniques for making one-pass versions of interactive protocols are generally limited to public-key mechanisms. Our new construction may allow more functionality can be packed into protocols that employ a single pass (or a small number of passes). We will give a practical example, from digital elections, in the next section.

Here, too, grizzling can help reduce the complexity that is the enemy of security. Crypto protocols are notoriously difficult to design correctly and understand completely. It is really important to have design methodologies that keep things as simple and comprehensible as possible [1,3].

3.1 Example - Digital Elections

As an example of how grizzling might be applied in a real application, let us consider digital elections. At present, there are roughly two ways of doing online elections. There is a large literature on using public-key mechanisms to construct protocols that involve some measure of anonymity, such as digital cash and electronic elections; the literature draws heavily on seminal work by Chaum [8,9,10,12], and for a recent survey see Rjašková [21]. These schemes typically return a single bit ('Bush or Gore'). Then there are practical fielded systems that collect complex voting choices, in which the voter is invited to express multiple choices and may even be allowed to write in candidates of his choice. These systems are often flaky, and use at best ad-hoc security mechanisms [14].

A second issue is the human factor. Security usability may be the most neglected subdiscipline of security engineering, yet many existing e-voting proposals assume voters to be sufficiently computer-literate to install special client software on their machines. Our University is currently considering a move from exclusively paper-based ballots[1] to a system that will offer an electronic alternative. One of the strongly-felt requirements is that the new system should not place members of the School of Technology at an unfair advantage. This means that the system must be usable by professors of ancient languages whose computer skills are limited to operating a browser.

The proposed scheme, in its most basic form, is as follows. The voter receives, by physical mail, a voting ticket with a voter number N and an authenticator tag T. The voter number, in our University of 3,500 voters, could be a four-digit number. The tag is a random number too long to be brute-forced during the period of the election, and presented with sufficient redundancy to detect voter typing errors. (In practice, we might have a 72-bit tag with a 18-bit checksum presented as three blocks of five base-64 characters.)

The voter completes an online ballot form with all the various voting options, which is presented by a signed applet. The applet then grizzles the ballot paper, asks the voter to enter his voting ticket, and validates the checksum. It then prints out a copy of the ballot paper, plus a substring of the grizzled version of the paper, to provide a voter-verifiable audit trail [14]. It then exclusive-ors the authenticator tag with the first 72 bits of the grizzled ballot paper, and returns to the web site the voter number N and the ballot paper which is now securely authenticated using the tag.

A surprising number of details have to be fixed to turn this into a properly engineered system. First, the anonymity can be provided by a mechanism such as Onion Routing [25] or Mixminion [13], but this is necessary anyway and the choice of such a mechanism is essentially orthogonal to electronic voting system design. (Simon pointed out in 1996 that an anonymous communications network is needed for anonymous payments, or the counterparty's identity can be determined simply by looking at the traffic data; and if such a network exists, then a much simpler digital cash system based on hash functions becomes possible [24]. Lee first applied this idea to digital elections in [17].)

[1] Believe it or not, we vote in Latin – PLACET for yes and NON PLACET for no!

Second, attention needs to be paid to separation of powers. There should be no single entity who can throw the election, miscount many votes, or break the anonymity of a large number of voters. Thus the anonymous remailers should be managed by more than one entity, while the voting tickets can be generated and mailed out using dual control mechanisms similar to those used to manage banking PINs. (This is not trivial but the details are beyond the scope of this paper; see [2] for more).

Third, we may require that voting be receipt-free, so that no-one can prove to a vote buyer how they voted. This is hard to square with voting at home. Indeed, there was a recent UK scandal in which a senior UK minister was accused of helping voters who were elderly, confused or easily intimidated to fill out postal ballot forms [19]. The best idea we've heard so far is that people should be able to vote as often as they like but only the last vote cast should count.

3.2 Scissors, Paper, Stone

Here is a novel suggestion. Each voter includes in their ballot one word of {scissors, paper, stone}. (For those not familiar with the childhood game, scissors cut paper, paper wraps stone, and stone blunts scissors – there is a cyclic dominance relationship.) If two ballots are received from the same voter, the returning officer will accept the dominant one. This way, even if the minister visits me just before the close of poll and persuade me to vote for her candidate, she has no better than a one in three chance if she chooses the token, and if I can choose it then I can make the vote invalid.

Schemes like these become possible with our ballot as it lets the election designer include arbitrary choices into the ballot paper. It is difficult to see how a classic digital election scheme could encode a cyclic dominance relationship. Such cycles are of great importance in the theory of elections; see for example Sen [23]. (Freedom to design ballot papers is a double-edged sword; care need be taken lest the voter write into the ballot paper a recognition code given to him by a vote-buyer.)

It is not the purpose of this section to design an election system in detail. In fact, we still do not know how to design an election that meets all the theoretical and practical requirements that are thought reasonable. Our goal has been to show that grizzling a ballot paper, before authenticating it with a shared secret, may be an interesting new primitive for designers of real-world electronic election systems. It is striking, for example, that one can do authentication using xor.

4 Conclusion

The invention of public key cryptography caused great interest because it allowed us to do completely new things, and because it held out the promise of doing some existing key management tasks more efficiently. In this paper, we have shown that some of these new things – threshold decryption, designated confirmer signatures, and digital elections – can be done at least as easily by using an already-understood conventional cryptographic primitive, the pseudorandom permutation, in a new way. These three are just examples; we suspect that many public-key primitives of practical interest can be re-engineered using shared-key or heterogeneous mechanisms. We can also do some quite new things, such as encoding cyclic dominance relationships.

The more general importance of the new construction is that it provides a very general way of composing cryptographic primitives. Up till now, engineering a complex crypto function involved either using the homomorphic properties of public-key cryptography to design a custom primitive, or designing a protocol to bind together the required component functions. Both approaches are difficult; they can be fragile; and they have other limitations. Our new construction is comparatively simple and can often inherit existing security proofs. It may help designers reduce the dangerous complexity of existing approaches. It will not replace all protocols or all complex public-key primitives (for example, we do not yet know how to do threshold signature usefully). However, it promises to be a valuable addition to the security engineer's toolbox.

Our work also suggests some further research problems. What, for example, is the best way to construct a fixed pseudorandom permutation of arbitrary length out of a given primitive, such as AES or SHA1? How should we re-engineer the existing primitives and protocols in the literature? What new primitives can we create, and what new applications now become possible?

Acknowledgements. David MacKay suggested the use of digital fountain codes and Eli Biham the use of secret sharing schemes. Three anonymous referees also provided useful comments on presentation.

References

1. M Abadi, RM Needham, "Prudent Engineering Practice for Cryptographic Protocols", *IEEE Transactions on Software Engineering* v 22 no 1 (Jan 96) pp 6–15
2. RJ Anderson,*'Security Engineering – A Guide to Building Dependable Distributed Systems'*, Wiley (March 2001)
3. RJ Anderson, RM Needham, "Robustness principles for public key protocols", in *Advances in Cryptology – Crypto 95*, Springer LNCS vol 963 pp 236–247
4. RJ Anderson, E Biham, "Two Practical and Provably Secure Block Ciphers: BEAR and LION", in *Fast Software Encryption* (1996), Springer LNCS v 1039 pp 113–120
5. M Bond, RJ Anderson, 'API-Level Attacks on Embedded Systems", in *IEEE Computer* v 34 no 10 (October 2001) pp 67-75
6. C Boyd, WB Mao, "On a Limitation of the BAN Logic", in *Advances in Cryptology – Eurocrypt 93*, Springer-Verlag LNCS v 765 pp 240–247
7. JW Byers, M Luby, M Mitzenmacher, "A Digital Fountain Approach to Asynchronous Reliable Multicast", in *IEEE J-SAC, Special Issue on Network Support for Multicast Communication* v 20 no 8 (Oct 2002) pp. 1528 - 1540; earlier version as ICSI Technical Report TR-98-005, and SIGCOMM 1998
8. D Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms", in *Communications of the ACM* vol 24 no 2 (Feb 1981) pp 84–88
9. D Chaum, "Blind Signatures for Untraceable Payments", in *Proceedings of Crypto 82*, Plenum Press, New York, 1983, pp 199–203

10. D Chaum, "Security without identification: transaction systems to make big brother obsolete", in *Communications of the ACM* vol 28 no 10 (Oct 1981) pp 1030–1044
11. D Chaum, "Designated confirmer signatures", in *Advances in Cryptology – Eurocrypt 94*, Springer-Verlag (1994), pp 86–91
12. D Chaum, A Fiat, M Naor, "Untraceable electronic cash", in *Advances in Cryptology – Crypto 88*, Springer LNCS v 403 pp 319–327
13. G Danezis, R Dingledine, N Mathewson, "Mixminion – Design of a Type III Anonymous Remailer Protocol", in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*
14. D Dill, VerifiedVoting.org, `http://www.verifiedvoting.org/`
15. S Goldwasser, E Waisbard, "Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes", at *First Theory of Cryptography Conference* (Feb 04) and MIT TR 329, March 2003
16. M Jakobsson JP Stern, M Yung,"Scramble all, encrypt small", in *'Fast Software Encryption'* (1999), Springer LNCS v 1636 pp 95–111
17. JH Lee, "The Big Brother Ballot", in *Operating Systems Review* vol 33 no 3 (Aug 1999) pp 19–25
18. DJC MacKay, *'Information Theory, Inference and Learning Algorithms'*, Cambridge University Press, 2003
19. S Walters, D Turnbull, "Cabinet Minister in Vote Rigging Enquiry", *Mail on Sunday*, May 4th 2003, pp 1, 8, 9
20. RL Rivest, "All-Or-Nothing Encryption and The Package Transform", in *Fast Software Encryption* (1997), Springer LNCS v 1267 pp 210–218
21. Z Rjašková, *'Electronic Voting Schemes'*, at `http://people.ksp.sk/~zuzka/elevote.pdf`
22. B Schneier, *'Applied Cryptography'*, Wiley 95, ISBN 0-471-11709-9
23. A Sen, *'Collective Choice and Social Welfare'*, Holden-Day and Oliver and Boyd, 1970
24. DR Simon, "Anonymous Communications and Anonymous Cash" in *Advances in Cryptology – Crypto 96*, Springer LNCS v 1109 pp 61–73
25. PF Syverson, DM Goldschlag, MG Reed, "Hiding Routing Information", in *Information Hiding 1996*, Springer LNCS v 1174 pp 137–150
26. D Wagner, B Schneier, "Analysis of the SSL 3.0 Protocol", in *The Second USENIX Workshop on Electronic Commerce*, Proceedings, USENIX Press, November 1996, pp 29–40
27. D Wheeler, "A Bulk Data Encryption Algorithm", in *Fast Software Encryption 1993*, Springer LNCS v 809 pp 127–134

The Dancing Bear: A New Way of Composing Ciphers (Transcript of Discussion)

Ross Anderson

Cambridge University, UK

This talk is about how we go about composing privacy properties. This workshop is about protecting privacy, and authentication and privacy are not only functionally in conflict, they also use different kind of technical mechanisms. Now how do we "compose" cryptographic primitives? I've put "compose" in inverted commas because it is also used by the operating system guys to describe a separate bundle of issues. If you can come up with a better slogan for what's going on here then I would very much like to hear it.

If you want to compose authentication and privacy – say you want to produce a signature which can be verified only in a controlled circumstance so that it can be verified only by particular people, and its verification cannot be traced back to your physical body – then there are roughly two things you can do. You can mess around with mathematics to try to design a compound primitive using the properties of exponentiation, or you can design a protocol. So you can do it with mathematics, or you can do it with computer science, but these are (roughly speaking) the two ways that we know of doing it. There is another idea, trusted computing, that you have the one box which acts as a certain arbiter, but that approach also has its limitations.

Now both of these existing ways of combining stuff are limited. Why? Well you can only do so much with polymorphic properties of discrete exponentiation. While people have managed to design all sorts of really, really clever crypto, it usually doesn't do anything useful. The useful things that we'd like to do we can't think up mathematics that can do it. Protocols are more flexible but they tend to require somebody or something to be on-line, and that's not always feasible. And both of these ways of combining cryptographic primitives are very error prone. If you look back over the talks of the previous security protocol workshops, something like half of them think about how you go about finding and fixing flaws in various protocols.

All I'm presenting today is a new and simple but powerful alternative. It inherits an established security proof, so it can benefit people who like to prove stuff, those things you can do here, and it works with both symmetric and asymmetric cryptography, so you can start with the intelligent things, with AES as well as RSA.

Basically, this old bear that we came up with here eight years ago turns out to have a few more uses than we thought at the time. Suppose that you want to do end to end encryption; you have five directors of a company and they all want to get together to decrypt a message. You can do that with threshold cryptography, or you could do it by simply encrypting something with five different cycles, one after another. If you use string ciphers, then the operations would even commute. Here's a new way of doing it: you package the thing and then the five directors each encrypt a separate block using their own AES key, so that's nothing new, but here is something new, you can now do four out of five threshold crypto by adding a parity check. If you have the blocks that are AES encrypted aligned on let's say 128 bit block boundaries, and the packaging puts in a 128 bit parity check, then four of the directors getting together can decrypt their bits and figure out their chunks, so they can find out what the fifth chunk was simply by using the parity check. And having done that they can then undo the package transform and get at the message.

Can we do better than that? We can perhaps do three out of five, and this becomes technically an order of magnitude more complex. I'm not going to go into all the details, but basically what you'd find yourself with if you can decrypt the first three of these blocks but not the last two is that you have got 256 bits of which about 128 bits are wrong. We want to be able to append appropriate error correction to the message in such a way that at system level it works as a three out of five threshold scheme, but not as a two out of five threshold scheme. The method of error correcting is mildly difficult to manage but recently a technology has come along many thanks to Mike Luby, in the form of some error codes used in a digital fountain scheme.

For current purposes all we need to know is that suitable block erasure codes exist for us to recover say two out of five, or in general K out of N, blocks that have been corrupted, and do so in a reasonably controlled way. So this, for the first time, enables us to do the threshold crypto, in a proper general way, using shared key crypto. You can do three out of five AES decryption.

Mike Roe: We could always do this using K out of N secret sharing on a symmetric key. Then when you reconstruct the symmetric key you decrypt the whole message.

Reply: Sure, but in many applications where people are doing secret sharing, who are doing threshold crypto, they don't trust some individual device to put the blocks back together again. That was the difference between the original Shamir secret sharing back in 1978, and the point made by Yvo Desmedt and others in the mid 1980s, that you could construct key shadows which would operate independently, on a public-key encrypted block, in order to recover content. But yes, you can use secret sharing in addition to error codes in this scheme, so if you've got some bizarre secret sharing scheme with particularly weird properties then you can plug it in and it will work.

Now, what also is new is that until now if you were in threshold decryption you needed to have all the keys the same, so you could have five RSA keys, but you couldn't have three RSA keys and two Diffe-Helman keys. But now the processes have all changed, so you can mix and match pretty well anyway you like. There are one or two issues that you have to think about in terms of the details in the coding, and getting the boundaries right, and worrying about entropy, and so forth, but that's something that we can talk about in more detail off-line.

Perhaps the main point of this is that until now if you wanted to combine different types of operation, not just encryption with RSA, and decryption of one way hash, but say *signature-and-encryption*, then you needed to put together all these clever custom-built structures, and this is what leads to this terrible reputation for being fragile.

What we're offering here is that you can now combine stuff in all sorts of ways that inherit the underlying security proofs. Suppose for example you wanted to have a designated confirmed signature: I want to be able to sign the messages but only Matt can verify a signature. Well I package the message and I sign it, and then I encrypt it in such a way that Matt can decrypt it. There's any number of ways of doing this, shared ASP key, one time pad; I might even get one of these physical one-way functions that they're now building at MIT. I challenge it, the challenge gets the response, X loads the response in here, and then gives the physical one-way function to Matt personally or by courier. All of a sudden I can actually bring him and his physical one-way functions into the normal world of crypto and protocols, which is something that until now people didn't really know how to do, because the physical one-way functions, are, if you like, an instantiation of a pseudo-random oracle. We're not used to working with these other than as hash functions.

Anyway, Matt having received this package can then use his physical one-way function, or his AES key, or whatever, to decrypt. The point about operating on the package is that you can now combine pretty well anything that you want. If you want to play around with micro-payments, or puzzles, or other hardware stuff, there's no reason why you can't do that as well.

What example applications can we come up with? We have thought of one or two. Elections are very topical at the moment, and we've seen all these amazingly complicated digital election schemes in crypto conferences where large mathematical expressions are flying around, but elections are more complex, and also more simple, than these schemes assume. They're more complex in that in most elections nowadays voters have more than a one bit output, Bush or Gore, right? Election papers in most countries are big complex tables with dozens of options, and all sorts of strange rules for choosing one option among a number. All sorts of officials are elected, all sorts of policies are chosen; and so how can you build a digital election system that copes with that kind of complexity and diversity?

How do you also build a system that can be used in the context of human factor limitations by the blind, by the illiterate, etc, etc? We need to be a fair bit more simple. Here's the basic idea. A postal voter will get the ticket with a serial number and a key, this will be printed in thin mail or typed stuff, or if there are blind people it can be printed in Braille, or whatever. The voter then goes to the browser and makes up a package of a nonce, a ballot paper and check stamp. The voter then submits the serial number and key plus package. You don't have to do fancy crypto when you're using package transform, a one-time pad is fine. If you have a ten digit or twenty digit number, or whatever you reckon is big enough to stop key search and sedition on an industrial scale, then that is sufficient.

But you can do some fundamentally new stuff that you just couldn't do with private key crypto. One of the things that we're really worried about in designing elections is that if you have the ability to get a receipt for your vote, you can then sell your vote, or be intimated into voting the way somebody wanted. This is a live political issue in Britain because of our local government elections last year. For the first time everybody was allowed to vote via post, and the Sunday Mail had an expose that alleged that two Cabinet ministers, the Foreign Secretary, and the Secretary of State for Trade and Industry, went into the homes of voters in marginal constituencies as part of the campaigning team, sat down on their sofas and helped them to fill in their ballot forms. One Asian family went to the press and said that it was a very intimidating experience, and actually he intended to vote Conservative but what can you do when a cabinet minister is sitting on your sofa in your front room helping you to fill in the ballot paper for Labour?

How can you build a receipt-free election scheme? Well one of the ways that occurred to me is the old protocol of scissors, paper and stone, which you may have played when you were a kid. Two kids stand in the playground and you whip out scissors or paper or stone. There's then the obvious cyclic relationship, scissors beats paper because scissors cuts paper, paper beats stone because paper wraps stone, and stone beats scissors because stone blunts scissors. What this means is that if you are a voter and you have been coerced once into voting you can then undo your vote, you can change your vote to the thing you really meant to vote for by casting another vote and exploiting this cyclotonomous relationship, so if you're forced to vote for scissors, and you didn't intend to vote for scissors, then of course the stone blunts scissors, and (if you intend to vote for paper) then paper wraps stone.

Now the importance of this from the of view of cryptologic mathematics isn't perhaps so much that I could prove that this becomes in it's present form a practical scheme, but rather that this shows that the mechanism I presented here is strictly more powerful than what we can do with the traditional public key mechanisms. I don't see how you can incorporate a cyclic preference into such mechanisms. This is as far as I'm aware something quite new.

So to sum up, package transforms, and perhaps general variable block ciphers, are much more useful than we thought. They provide us with a third way of tying together different cryptographic functions, where perhaps happening in the protocols are a number of theoretic tricks. You can bind stuff together in previously infeasible ways, such as by putting together schemes which enable heterogeneous threshold decryption. This may be increasingly important as the world becomes more complex, and you have got more and more requirements to interwork with legacy systems without stuff breaking.

One of the things that we found looking at API security for example, is that many of the things that go wrong in systems like the 4758 are there because there are backward compatibility modes to enable the 4758 to work with the old 3848, and if that was exploited in an appropriate way it could reduce the security of the whole system to the security of single DES. In fact a bit worse than that. So our approach gives you a way of creating more precisely controlled backwards compatibility with one-way functions, or package transforms standing between you and the data, and with the possibility of always controlling backward weakness by deciphering something else that's stronger. Hopefully we can avoid some of the complexity, and some of the resulting failures, of these traditional modes of operation. And hopefully there will also be a whole bunch of new research questions opened up, new applications, detailed re-engineering of existing compound primitives, and efficient constructions of primitives for the package transforms themselves. Perhaps in ten years time we will have a listed standard for a package transform, who knows? In the meantime let's go and find lots of applications.

Virgil Gligor: You did not mention the notion of shared public key cryptography[1]. I'm wondering how that would compare that to the maths of the kind of signatures that you do here. In that scheme the complete secret key, which presumably might be the signature, appears absolutely nowhere. Essentially the advantage of those schemes is that only pieces of the private key appear to the holders. To do the verification you don't need anything except the public key, and consequently they have no single point of failure.

Reply: Well this is at least as good as that and in many respects better. With my scheme if you want to do three out of five RSA encryptions, then you have five separate RSA keys; there's a different P and Q in each box. You RSA-encrypt five separate blocks of the package, and there is no mathematical relationship, causal relationship, electrical relationship, or any other relationship at all between these RSA keys, except that as a purely contingent matter they were once used on the surface of this earth to exert five different parts of the same message.

If you are paranoid about somebody coming along and putting his electronic stethoscope on your corporate tower security module and getting out the super secret private key for which the shares are available in various offices, then you may use the Boneh-Franklin scheme, in order to see to it that there isn't a master private key anywhere at any time, but you could just easily use this scheme where there is never even a virtual master private key.

Virgil Gligor: I'm still confused about the advantage of this over the Boneh-Franklin scheme.

Reply: Well this does what Boneh-Franklin does, and you can do threshold RSA decryption without there ever having existed in one place all the bits of the all part master key, but this also does a lot more. For example, you cannot use Boneh-Franklin to do shared decryption between RSA and AES, and there are applications where you really will want to do that.

Virgil Gligor: Can you give an example?

[1] Dan Boneh and Matthew Franklin, *Efficient Generation of Shared RSA Keys*, LNCS 1294, pp425-439 (Crypto97).

Reply: For example, resource limited applications, where you can't do Public key crypto for some reason. Public key crypto is expensive and slow running. If you have got one device that will do RSA say a PL, it's the other device that won't, say an electricity meter, right, they can cooperate on a protocol, because the device that does the RSA does its part of the computation, and the device that does the shared key crypto does its part.

Virgil Gligor: OK, so what happens to an application where you actually don't use public key crypto at all? Will the application ever have to use the entire shared symmetric key as a whole?

Reply: But there isn't such a shared symmetric key Virgil, that's just the point. We're using separate keys. If I do three out of five threshold decryption, there are five separate keys, and they never come within a hundred metres of each other.

Virgil Gligor: Right, so there is no place where they all come together, not even in the box that does the work?

Reply: Correct. In fact, there isn't even a box that does the work. You send a package to the first person who decrypts his part, the second person decrypts his part, the fourth person decrypts his part, perhaps all on separate continents, and then the missing parts three and five are reconstructed through the miracles of tornado code, or secret shares. You don't have to have a super secret master key for this, and if you don't like super secret master keys, you could always use this scheme.

Mark Lomas: Presumably if you're encrypting something then you're encrypting it because you want it to be secret, and you do actually have to put the parts together somehow. You can't do the pieces at arbitrary places because otherwise you need a protocol to move them about.

What Virgil has in mind is the case where you've got a simple box there because this allows you to guarantee that you've got privacy of the information. If you segregate it then you have to build a whole new infrastructure to support the re-combination.

Reply: That may be the case, if your mental model is entirely of incriminating messages which were signed and then encrypted, or the other way around [Laughter], but most of the world isn't like this. Most of the world is about transactions, about quite different and more complex stuff than sending a signed encrypted letter to your girlfriend saying that you shot Admiral Prezak. If you send such a signed encrypted letter to your girlfriend, then you've got to worry about protecting the plain text on the machine where it was.

Mark Lomas: All I'm suggesting is that you said it was encrypted data. I assume you encrypted it for a purpose. Why would I encrypt something if I don't want privacy or secrecy?

Reply: Well most of the cryptography in the world isn't about privacy or secrecy, it's about things like access control.

Bruce Christianson: But if I'm trying to keep a secret I'll probably have five blocks which XOR together to give me the secret, and then I'd go off and encrypt each of the blocks.

Mark Lomas: Whereas if I don't want privacy or secrecy and I just want a signature scheme, then I don't need this encryption.

Reply: The typical application of cryptography nowadays is ensuring that a Hewlett Packard printer can identify ink cartridges from Hewlett Packard.

Mark Lomas: Yes, a signature scheme would satisfy that requirement.

Reply: Well you can actually use a challenge response for that. But the point about the dancing bear is that this is a general mechanism that enables you to combine cryptographic primitives.

Chris Mitchell: I think I understand how you combined all kinds of decryption in a threshold like that. I don't quite understand how you use it to combine other kinds of crypto, such as signature and encryption. How is that semantically different to just taking the data, adding a signature, and then encrypting everything?

Reply: You'd have to speak to a semantics expert on that, because this brings us back to the old argument about whether you encrypt first or sign first. Here you're doing both at the same time, so there's probably a thesis in that for someone. If the block here is still encrypted then you can verify that this is a valid signature but you haven't got the slightest clue what you're decrypting. We have a proof that if you can extract the meaning without doing this encryption then you can break AES.

Chris Mitchell: OK, so you can verify parts of the decryption. But you can't verify that it's a genuine package until you've done the decryption, so you can't tell whether it was a genuine signature on a nonsense, or a genuine signature on a proper package, with only part of the input.

Reply: I'm sure that there's a vast number of combinations, and I know that there will be some things that go horribly wrong, we just don't know what they are yet, because people haven't built enough applications for us to verify. All I'm basically offering you is a new primitive which appears to me extremely promising in that it gives a certain *je ne sais quoi*, that's all.

Identity and Location

Dieter Gollmann

TU Hamburg-Harburg
21071 Hamburg, Germany
diego@tu-harburg.de

Abstract. In some instances, it may be possible to protect access to a user's personal data by authorising the return addresses replies pertinent to that user are being sent to, rather than authenticating the principal making the request. This relationship between a user's identity and authorised locations can offer a new design dimension when trying to improve privacy protection.

1 Introduction

A very efficient way of addressing conflicts between authentication requirements and privacy is to avoid authentication in the first place. There are various reasons why authenticating the user may be perceived to be necessary, but in this note we will focus on access control. In the familiar access control model [6],

$$\text{access control} = \text{authentication} + \text{authorisation},$$

where authentication verifies the identity of the principal making an access request. Access control based on a so-called user identity (uid) is also the standard in operating systems like Unix or Windows, so that for some this type of access control is so natural that hardly any other approach is conceivable.

However, it has already been argued several times, e.g. [2,5], including at previous instances of this workshop [3,4], that in many applications today user identities need not be the most meaningful concept when defining access control policies. Code-based access control is one such alternative, with the Java security model and the .NET security model as prominent examples. We will not pursue this point further, but limit ourselves to cases where user identities would seem to be naturally suited for expressing security policies. We will use e-government applications to show that even such policies can sometimes be enforced without authenticating the user, and postulate that there is a relationship between a user's identity and the locations authorised to receive information pertinent to that user. This relationship offers as a design dimension when trying to improve privacy protection.

2 Identities

For our discussion, the distinction between 'high level' organisational security policies and 'low level' automated security policies made in [7] provides a useful framework.

Organisational security policies concisely state the laws, rules, and practices that regulate how an organisation manages and protects resources to achieve its security policy objectives. *Automated security policies* specify in detail how a computing system prevents violations of the organisational security policy.

The meaning attributed to the term *identity* then depends on the level we are approaching access control. If the focus is on low level operating system mechanisms for access control, *identities* are just unique identifiers. In high level organisational security policies that regulate how people may access resources in an IT system, the *identities* of persons have to be represented and there happens to be a good match between high level entities and low level user identities.

It is a frequent problem in security discussions that these two uses of *identity* are not cleanly separated. There are two issues at stake. If identities are only unique identifiers but our overall policy relies on the fact that the person behind a *uid* can be traced, we would implement an ineffective security mechanism. Conversely, if we assume that identities always must refer to a person, we may miss out on policies that are relevant and useful for our application and request exaggerated controls that cause their own security problems. If all access decisions require the identity of a person to be checked we would raise considerable privacy concerns.

As an aside, the identity of a person or, more precisely, the information identifying a person like name, date of birth, home address, social security number, and the like, is on its own not a suitable access control parameter. All these pieces of information are not secrets kept by the individual, and some are available to the general public. To obtain others needs a little bit of effort. Identity theft as it is commonly called is therefore not difficult and authentication without secrets makes for poor access control. In credit card systems, asking for card number and home address to authenticate the card holder is a weak mechanism indeed.

In the next section, we will discuss some aspects of e-government applications to demonstrate that it may be possible to enforce high level policies referring to 'real' user identities without authenticating the user making the access request.

3 E-Government

In e-government applications, users interact with a government body that administrates sensitive data or provides some service. We will call the two parties *citizen* and *agency* for short. Agency and citizen communicate electronically. That is, the agency receives *requests* and sends *replies*. The agency may keep personal data on citizens but not all services provided by an agency need relate to personal data. The agency has to ensure that sensitive data are only disclosed to authorised parties. In the world of paper documents, this is often achieved by requiring the citizen to appear in person at one of the agency's offices and presenting a document (identity card, or utility bill if you are in the UK) for authentication.

Traditionally, citizens are residents of the agency's country (and for the sake of the argument we might even assume that they have been issued with electronic identity cards.) However, we cannot assume that this is always true. Agencies may well receive requests from foreign citizens or from citizens residing in other countries. In the first case, the agency may have no prior record of the requestor and no information to authenticate the requestor with. In the second case it may be problematic to establish addresses and determine which court has jurisdiction in case there is a later dispute about the legality of a request.

To protect sensitive data, it is furthermore insufficient to just authenticate the requestor. We also have to check that the return address the reply is being sent to meets both parties' security requirements. If it were possible to forge sender addresses an adversary could modify a legitimate request and have the response forwarded to the adversary. Moreover, a careless citizen may launch a request from an insecure site, thereby compromising the information returned. Also, the agency may have its own rules to which sites replies may be sent to.

In the following, we analyze the requirements on securing a transaction for a variety of services.

3.1 Read Request for Personal Information

The agency must *authenticate the return address* it replies to, i.e. the return address has to be authorized to receive personal data for the citizen concerned, there is no inherent need to authenticate the requestor. The citizen could supply in advance a list of authorized return addresses and the agency would check the address given against this list; updates to this list are a special case of the next item. When there is no such list and the return address is specified in the request, the requestor has to be authenticated.

3.2 Change Request for Personal Information

The agency has to *authorize the request*; the agency need not know who the requestor is (as a person) but that the requestor is authorized to make change requests in respect to the citizen concerned; for example, a citizen could authorize a public key by signing an authorization certificate and the agency would check the validity of the certificate before dealing with the request.

3.3 Denial-of-Service, Nuisance Replies

If receiving unasked-for replies is deemed a nuisance, we may demand that requests are authorized. Again, the agency has to check that the requestor is authorized to read data for the citizen concerned but need not know who the requestor is. Alternatively, the agency could establish the requestor's identity (*authenticate the requestor*) and provide it on demand to the party receiving the reply, who then may take further actions.

3.4 Recourse to the Legal Process (Non-repudiation)

If the nature of a transaction is such that a court of law may be asked for adjudication, the agency has to collect evidence about the requestor's identity and place of residence, or even evidence about the location the request is made from.

3.5 Inter-agency Collaboration

Depending on policy, agencies may be required to use the same set of identifiers for citizens so that required cross-checks can be performed, or be barred from sharing identifiers to alleviate privacy concerns.

4 Conclusion

In organisational policies, identities often refer to persons (people and organisations), that is, to entities that could be held accountable if they can be located. However, when an organisation has no authority over its users or cannot reliably locate users, it will be difficult to fall back to non-technical security measures once there is a break of the operational policy – and breaks are bound to occur.

Application-specific identities in organisational policies, like subscriber numbers, sometimes correspond to persons and just constitute a special user representation. Indeed, using such identities may help to protect user privacy. In other cases we may deal with identities that are not linked to persons, a fact one must not forget in the design of the overall security system.

In operational policies, identities are just unique identifiers. They may correspond to user identities in organisational policies, or be a convenient way of grouping permissions. Thus, a *uid* could point to the sender of a request or directly to the permissions allocated to the request.

Although organisational e-government policies will often refer to the identity of a citizen, this identity is often not required in operational e-government policies. We have suggested that in most cases there is no need to establish a citizen's identity, but to check that a certain action (send a reply, amend a database entry) is properly authorized. Authorization will refer to unique identifiers but these identifiers need not be verifiably linked to a person.

In such situations, it is not the origin of a request that is being authenticated, but the return address that is being authorised. This duality may itself have some attraction as a design principle, but discussions about identity and location are not restricted to the example we have discussed and have also emerged, for example, in discussions about security for mobile systems [1].

References

1. Tuomas Aura, Michael Roe, and Jari Arkko. Security of Internet location management. In *Proceedings of the 18th Annual Computer Security Applications Conference*, pages 78–87, December 2002.
2. Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. *SPKI Certificate Theory*, September 1999. RFC 2693.
3. Joan Feigenbaum. Overview of the AT&T Labs trust-management project. In *Security Protocols, LNCS 1550*, pages 45–50. Springer Verlag, 1998.
4. Dieter Gollmann. Mergers and principals. In B. Christiansen et al., editor, *Security Protocols, 8th International Workshop, Cambridge, LNCS 2133*, pages 5–13. Springer Verlag, 2001.

5. Brian A. La Macchia, Sebastian Lange, Matthew Lyons, Rudi Martin, and Kevin T. Price. *.NET Framework Security*. Addison-Wesley Professional, Boston, MA, 2002.
6. Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.
7. Daniel F. Sterne. On the buzzword "Security Policy". In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 219–230, 1991.

Identity and Location (Transcript of Discussion)

Dieter Gollmann

TU Hamburg-Harburg, Germany

The argument set up by Bruce Christianson and collaborators is that authentication verifying claimed identities and privacy are currently conflicting requirements.

Speaking towards the end of the workshop one has the advantage (or disadvantage) that most things have already been said. One thing mentioned by other speakers is that one of the most efficient ways of resolving this conflict is avoiding it, and that's what I want to talk a little bit about today.

By talking about access control it's often a reflex to think, well we have to authenticate identities because that's what we're used to doing, and the claim is that this is not true as often as we think.

We can talk about the identity of a person; are methods, for example, definitely needed if we want to hold the person accountable? Experience I have had in the past, for example, with Microsoft, when they mention the word identities, and a lawyer in the room immediately latches on, which person are you talking about? Even if you're discussing a cryptographic protocol, lawyers have a very clear idea that identities, real identities are real people. You have our world where we use unique identifiers and Unix UIDs, and Windows SIDs, cryptographic keys can be legal identifiers, and there have been complaints that actually calling those an identity rather than an identifier is somewhat misleading.

These are two different ideas of what identities mean. One of my favourite papers is the 1991 paper by by Daniel Sterne on the buzzword "Security Policy" where he explains where there are organisational security policies – the high level rules that govern how a company works – you are most likely to refer to the identities of the first kind I mentioned, whereas when you have an access control policy – the role of access role in your operating system defines – you're more likely to find user identities of the second kind. So already there you find this dichotomy which can hurt you if you're not aware of what is going on.

Fabio Massacci mentioned in his talk that if you want something delivered to you, then you have to tell your address otherwise it won't get there. You can play with mail boxes but if you want courier delivery they will insist on the real mail address with the real telephone number otherwise you don't get the service.

Identity based access control with user identities is the obvious way to confound user identities with the identities of real users, which they not always are, and accountability, to use identities to refer for legal purpose.

Bob Mayo: As other people have mentioned, you can have accountability without knowing who the legal person is behind it. Let's say you're renting a car, and you crash the car, all the car agency needs in that particular case is to be able to extract the appropriate amount of money from a bank account, it doesn't necessarily have to identify you.

Reply: I would not have called that accountability. That's paying a deposit in advance. When I use accountability I rather mean some person has to be held accountable for an action, but you don't need this in many business transactions.

Jari Arkko: So just to pick up an old argument, here maybe it's not so useful to talk about identity. It would clarify things to talk about identifiers instead. Yes, there are physical persons and legal persons, but these are very well understood concepts, at least outside the computer world, and in the computer systems when we are designing protocols these are somehow mapped to identifiers, and then we don't have this kind of identity crisis. It's easy to accept that I might need to have multiple identifiers identified with me. Why not talk about how things are, or how we want to design them? Why should we constantly be worrying about this crisis of identity?

Reply: Yes, right, I totally agree, and we have given up the battle of trying to convince people to use more sensible language. Indeed in cryptography we have identity based encryption, and people have switched more to calling it identifier based encryption.

Ross Anderson: I suppose I have an objection to your assertion that you need identities for accountability. Access to the physical person suffices. If I go into the swimming baths I'm not asked to produce an ID card, if I misconduct myself I am physically arrested. They may then afterwards do an ID as part of the process and determine whether I've got a criminal record so they can decide how long to send me away for, but we do not yet live in a world where you have to present an ID card to go for a swim. Now the government is working on this problem but I do not consider it to be a bug, perhaps I'm strange.

Reply: Yes, I don't say you have to show your identity for everything, but if they take you to court they will ask.

Bruce Christianson: There needs to be a binding between your cyber surrogate and whoever the physical person is in the real world.

Chris Mitchell: I guess accountability isn't always for individuals, there might be corporate accountability.

Reply: Yes, I have been very careful to mention a legal person.

Chris Mitchell: Oh, a legal person could include a corporation.

Reply: In European law it does. In Roman law I think *persona* can also be an organisation, which has a standing in law. Right, that's not what I talk about, I just put out this slide to have some discussion with you, which is normal. [Laughter]

I really wanted to go through an example first. These are quotes on Kerberos, and on services: at times it might be necessary to approach them to allow a service to perform an operation on its behalf, the service must wait to tag on the identity of the client, but only for a particular purpose, let's say, that Bob will need to talk to Ted, that's you, but you need to have those conversations while pretending to be Alice.

Virgil Gligor: That's a proxy description?

Reply: Yes, it's very strange if you ask me, but you find this. [Laughter]

If we move away to the real world, about car rentals, and so on, the examples take on a meaning of their own that gets detrimental to the point we're trying to make. If you want to name services Alice and Bob, then maybe delegating identity would be synonymous for passing an access right word, which is something you have to do, and if you can control it properly, fair enough. Again, a quote, the fast and loose way for Alice to delegate her credentials is to simply ask the key distribution centre for a forward ticket-granting ticket she can give to Bob.

Virgil Gligor: Actually it gets a little worse than that because Alice can give her identity to Bob, with the ability of Bob to transfer the identity further.

Reply: Yes, and my comment is the fast and loose way to let somebody else use my rights is to give my password away. Dangerous language when you introduce the world of people and then use it to describe access control. I think Kerberos has this particularly interesting migration that it started off as a protocol for authenticating real users at MIT, and then it moved into the operating system where it was used to control access between services, and now it migrates back to the users, and the users are supposed to behave like services and processes in an operating system, and I think that is a thoroughly bad concept to have.

Right that was the example, a warning really to control your language, I agree with Tuomas Aura on this. The second example I find more interesting, back a couple of years ago, friends at the German Information Security Agency were working on e-government guidelines for authentication. The idea is you have all these wonderful government services which will now be provided as e-government services[1], and the question is, how does the citizen have to authenticate himself and herself for particular services depending on their sensitivity. What was interesting was when we were working on it we figured out that location verification seemed to be much more important than we thought. We had in mind authenticating the citizen and location came into play. And that's what I want to talk about really.

The model I have is citizens interacting with a government agency, the agency receives requests (please send me this record), and sends replies. These requests refer to personal data corresponding to citizens, and e-government seems to be the natural habitat for identity-based access control, because if I go to an office I have to present my passport, I might even have to bring my registration document because my Austrian passport does not carry German addresses, you can already hear location comes in, and it seems all very natural if you translate all these services into e-government that again it wants identity for its access control.

[1] http://www.bsi.bund.de/fachthem/egov

Slight sidestep, several years ago I had a look at a document for the UK government on identification identity cards, and what struck me when I read the document was that it was really assumed that you were either already born in the UK, or that you were a legal immigrant because there was a record where you had entered the country, it's an interesting closed-system environment. If you now talk e-government in Europe it gets interesting because there may be foreign citizens with no prior record in the country. In particular, if you move from Britain to Germany, because in Germany you have to be registered with the police, but in Britain you aren't, so they're terribly confused. Where do you come from?

If you join a German university you have to provide a record that you have no previous convictions, what do you do if you come from the UK? There is no agency that provides such records for the average citizens.

Ross Anderson: There is now The Criminal Records Bureau[2].

Mike Roe: The charge is something like ten pounds.

Reply: Thank you for that. They would know about my convictions in the UK, but not the rest I've been doing. Actually, in the end, I got a certificate from Austria, no previous conviction (haven't been there), easy to achieve. [Laughter]

Virgil Gligor: The statutes of limitation ran out on you.

Reply: So that's also again a problem related to location if you have this traditional notion of government. Now, security requirements. If someone requests access to my data it doesn't really matter who requests access, it matters where it's been sent to, at least as far as my privacy is concerned. If I can make sure it's only being sent to me, whoever calls for it, or maybe my tax form to my tax accountant, I don't complain as far as protecting my privacy is concerned. So it's not important for this security requirement to authenticate the requestor, it's important to authorise the return address. Same holds for requests for personal information, I don't have to know who's asking to do it, in theory I might not have to tell them who my tax advisor is, I only have to provide them with some credential saying, whoever comes and makes this request I have authorised that person, that suffices to protect me in this respect.

Bob Mayo: Wait a minute, if I want to see if Joe Smith lives in a particular apartment building, I can make a query to some agency for information about him and then they'll mail a yellow letter to that apartment building, I will see the letter go in.

Reply: I was thinking of a cryptographic decryption key as the return address.

Bob Mayo: Is traffic analysis still possible?

Reply: Traffic analysis would also be a problem if I am authenticated when I'm waiting for a request, and I would have to use different schemes anyway to defeat traffic analysis if that's what I'm worried about.

[2] http://www.crb.gov.uk

Next point, requirements. In the scheme I have sketched, all of you could now write to my tax office and say, please return the tax records to Dieter Gollmann and my mailbox keeps filling up. I might be persuaded that now it becomes of interest to me to say, well, to stop this unsocial behaviour I want to know who the people are who are making these requests, so then I can be told later, and take whatever action. But you might say, well the schemes in particular again with these e-government applications, I will not ask for my tax records ten times a minute, and there can be some limitation on how many replies the government agency will send in any particular time. I'm only concerned with denial of service.

If, there is the idea that somehow I should be able to go after people in the courts who are playing these silly games, so that I can go to a court of law and say, please sir, stop this person, find this person, they're doing things that are not desirable, that is the only point where I immediately see an argument for asking the requestor to be authenticated. For all the other security requirements I think we can find other solutions if we try.

Last point, inter-agency collaboration. Interesting topic, depends on how you have set up your administration. Agencies may be required to use the same identifiers to prevent certain types of fraud, and then this scheme whereby I go to my tax office and say, these are the addresses you are allowed to respond to, the keys you are allowed to respond to, would no longer work because I could use different keys with different agencies, and then again we might have to look into the scheme.

On the other hand there might be constraints barring agencies from sharing information. Several years I read in a Danish newspaper a report saying, probably we have overdone privacy protection in our system because people sitting in jail are still able to claim social security benefits because the two authorities have no way of comparing who is still out and who is in jail, so that is something you might discuss when you consider whether you need unique identifiers to authenticate requestors if you're dealing with citizens. But again, from what I said before, I don't think it's necessary in general.

Which brings me to the conclusion of this brief analysis of e-government. In most cases I do not see an immediate requirement to authenticate identity. Instead, I have to check that a certain action is properly authorised. The way I see it, these authorisations will refer to unique identifiers because identifiers need not be verifiably linked to a person. Step back, if we go back to the organisation level and explain who should be allowed to read my data then these policies will be clear to everyone. I don't see that it's absolutely necessary to translate these policies one to one to the operational e-government policies.

Conclusion, a repeat of earlier remarks, these services need not mirror the paper procedures they're substituting. Terms in two different worlds may have two different meanings, you have to be quite careful, and as you go from one level to the other, treat it as translation between languages. That makes you more careful to look out for changes in meanings, which brings me to the end of the talk, thank you.

Security of Emergent Properties in Ad-Hoc Networks (Transcript of Discussion)

Virgil Gligor

University of Maryland

I'd like to talk about emergent properties, something that you are somewhat familiar with from previous research in security, and I'd like to focus on the security aspects of these properties in ad-hoc networks. This work has benefited from discussions with my colleagues, Bill Arbaugh, John Baras and Jonathan Katz at the University of Maryland.

I'd like to define what I mean by emergent properties, and point out that this is actually a known notion, then simply give some examples of emergent properties in different areas of what we might call ad-hoc networks, and examine some of the security concerns that appear. I won't go through detailed solutions to these concerns, I want just to stress my belief that analysing emergent properties for security could become a very profitable research endeavour in the future.

So what are emergent properties? Informally, these are properties that arise from "collaboration" and interaction among nodes (collaboration here may also include competition). They are different from typical protocol properties in at least three ways, and these are not mutually exclusive. First, neither the time nor the locus of emergence of such a property can be anticipated. Second the emergence may be uncertain, so the property may actually be probabilistic, and third, the emergence may be transient, the property may be transient a normal mode of operation; i.e., one need not have an exception for one of these properties to disappear. However, most examples I give below illustrate non-transient properties.

We have had formalisms that define what a system "property" is which typically identifies a property with a predicate that holds on all execution traces of a system. For example, we've had the safety-liveness dichotomy (as used by Alpern and Schneider in 1985) and we also we've had "non-properties," namely properties that fall outside the the safety-liveness framework. Information flow policies are some of the better-known examples of "non-properties". Emergent properties could be either safety of liveness properties or, more often than not, "non-properties."

Most emergent properties that we've seen in the past were actually undesirable. For instance, a typical safety property that might be considered emergent are deadlocks; i.e., neither the time nor the locus of emergence of deadlocks could be anticipated, as they arise from the competition between processes for various resources. Their emergence is clearly uncertain. For example, simulations show that in distributed transaction systems that use locking as the concurrency control mechanism, the probability of a deadlock is proportional to the square of the resource/lock holding time. However deadlock is not transient, in other words, these are a safety property, namely once they set in, they stay with us. Nevertheless, they are an emergent property in the sense that they satisfy the other two criteria.

Another emergent property with which we are familiar is starvation. When we think of starvation we have mind a bunch of typical operating system classic liveness problems such as the "readers/writers problem," the "dining philosophers problem," and so on. In the latter, one couldn't quite tell whether starvation may emerge and if so who would starve, because one couldn't really figure out who the conspiring parties were and against whom they would conspire. However, the possibility of starvation was clear. Again, starvation is a non-transient property: if Dieter Gollman and I can conspire, say, against Mike Roe, we can do so forever.

Another one might be non-negligible information leakage. While we know that information leakage is possible in any system attempting to control information flow and where resources are shared, when it actually happens is usually extremely difficult to detect. I don't know if you've ever tried to detect use of covert channels by examining audit trails, but if you have you undoubtedly realize that it's really extremely difficult to figure out which channel is (re-)used. Even when the set of covert channels in known, one could disguise the use of a channel by information flow in other channels. In this case, emergence could be defined as a threshold property; i.e., if a bandwidth threshold is exceeded, and the answer was yes, otherwise no. Exceeding the threshold is typically a probabilistic matter, and again this is a non-transient property.

The properties that I'm interested in here do not necessarily have undesirable consequences. On the contrary, I'd like to focus on properties that I would like to make happen, properties that I'd like to see emerge. The first example that I give in this area is one that you've seen here two years ago, about trust establishment in ad-hoc networks. I was very pleased to hear Laurent Bussard's talk this morning[1] about trust establishment because it seems that we have very similar ideas. Trust establishment is, in general, an application of an evaluation metric to a body of evidence. Essentially what emerges from applying your evaluation metric is a trust relation. This evaluation can be on or off-line, short- or long-term. In fact already established trust relations may serve as reasonable pieces of evidence.

What are the security concerns, what are the properties that we might want to make emerge? We might want to have an authentication trust-path between any two nodes of a mobile ad-hoc network. This is a non-trivial property because a mobile ad-hoc network, which typically arises as an extension of the internet, may in fact be completely disconnected from the internet, and completely out of touch with any stable trust infrastructure. Essentially what happens here is that each node generates trust evidence on its own, so there may be evidence about node location, configuration, operating system, version number, access control and authentication policies in force, and so on. Each node generates this set of

[1] Laurent Bussard, *Establishing Trust with Privacy*, these proceedings.

evidence and then all the other nodes apply their trust metrics to all generated evidence or parts thereof. The result of the evaluation is used to decide whether a node is trusted for a particular action (e.g., message signature, content).

Clearly we'd like to have as much evidence as needed to satisfy our metrics. There is a number of security concerns here, some of which were pointed out a couple of years ago. An important concern is the integrity and availability of signed evidence, in other words, evidence may be false, some of the nodes may be corrupted, or captured by an adversary; some of the evidence may be stale, as for example, location evidence. This is particularly damaging if we have nodes that act as trust clearinghouse, in other words, nodes that are trusted by just about all other nodes in the and that are used by all nodes to establish trust with other nodes. There is also the potential of denial of access to evidence, because obviously when one gathers on-line evidence in a network one has some request routing problems, and so on.

Furthermore there may be significant privacy concerns regarding the individual and aggregated piece of networked evidence. For example, these concerns arise in what is known as "automatic trust negotiation," whereby a node that acts a server and another as a client, and neither wants to reveal its complete function and policies to the other unnecessarily. The server node might not want to reveal all the access checks that it performs for fear that those access checks might reveal some of its access control policies and relations with other nodes. The client might not want to reveal all his credentials because that might reveal its relationship with other nodes as well. Hence, the client and server nodes start negotiating credentials for policy disclosure: the server discloses more of its access checks if the client discloses more of its credentials, and the other way around. Of course, it may very well be that an effective client-server access path does not emerge because at some point both parties refuse to reveal enough to satisfy the other's party. Or, at some other point in time, some of the credentials of a client may become obsolete and the client-server trust relation that emerged before is now stale.

Another fundamental concern here is the correctness, if you will, of the trust metric a node applies: what is the logic behind such a metric, was it sound and complete in some sense, and how does it relate back to the policies it intends to support? Can we actually use collaborative application of a trust metric, and can we actually compose evidence and metrics. These are all relatively difficult questions to answer at this point. By the way, there is also a possible analogy in some of these areas with the handling and evaluating evidence in the legal domain. In that domain one has a very well elaborated system of metrics for evaluating the evidence and providing advise regarding the trustworthiness or the believability of the evidence.

Tuomas Aura: With this kind of trust, where you suddenly have some kind of limited transit level, you always have the same problem that even though the metrics may be sound within themselves, and well defined and mathematically attractive, they don't really map, there is not even a claim that they map, to some kind of real world concepts, like probabilities.

Reply: Well, for example, there is the work of Kohlas and Maurer[2], which provides a fairly rigorous probabilistic based approach, and some precise metrics for trust. There is some formal work in this area, and I believe it's still an active research area.

Tuomas Aura: And it has been for years.

Pasi Eronen: In the systems I've seen you usually get some sort of real number between 0 and 1, or a binary number, 0 or 1, as a result, is this something that can be meaningfully used for anything.

Reply: In my opinion it's as real as these notions of, for example, *belief* that we have. What we are trying to do here is to find a more precise basis of explaining how trust emerges, and probability theory gives us some degree of precision, just like logic gives us some degree of precision, so, are these are real concepts.

Tuomas Aura: Often security properties map on to some real world things like lots of money, or some medical information, some kind of concrete things that you can observe in nature. I think emergent networks of trust always seem to have many limited assumptions, so they map onto some very limited real world systems. But then there is always the hope that in the complete ad-hoc setting, there will emerge some kind of trust, and I simply don't believe that, though they may revert some nice mathematics.

Reply: The metrics that we're talking about might actually be based on very real practical analysis, for example, to risk analysis in banking. Banks actually do a lot of risk analysis for their correspondent relations with other banks and clients. This is not a simple exercise, and it's all based on some subjective probability-based metrics, and in this sense this is as real as anything we find in practice.

Tuomas Aura: Well yes, in banking systems you are mapping things to physical persons, or at least some kind of established identifiers that have credit history. What you always have to do in these kind of networks is to put a cost on adding nodes to the network, and then usually, there is some value of maintaining the reputation of the node, and so on.

Reply: Reputation may be one of the frameworks, but notice that I'm particularly and deliberately vague about prescribing specific metrics. At some point one has to commit, and we have the examples as you noticed in the paper presented here two years ago[3]. There I gave explicit examples of metrics which appeared to make sense to some extent in practice.

Chris Mitchell: Maybe 'real' isn't the right word, maybe 'useful' is a more sensible word, because we do use some of these bizarre metrics. eBay reputation

[2] R. Kholas & U. Maurer, *Confidence Valuation in a Public Key Infrastructure Based on Uncertain Evidence*, Proc. Public Key Crypto 2000, LNCS 1751, pp93-112.

[3] Laurent Eschenauer, Virgil D. Gligor and John Barras, *On Trust Establishment in Mobile Ad-Hoc Networks*, LNCS 2845, pp47-66.

is an obvious example, run a completely arbitrary metric, and it can be broken, there are all these breaks, but it's nevertheless useful. But it's arbitrary, there's no mathematical basis.

Jari Arkko: But there is money behind it.

Reply: I agree. In the next example we again want to make some emergent properties appear but by design, this time we are not leaving to chance the interaction between nodes. Imagine an array of very large sensor networks, for example, a large number of sensors, maybe 10,000 of them, distributed randomly for obvious reasons on a battlefield. Such a network maybe used when one tries to figure out the position of one's adversary. Here one would like to have basic secure communication between sensors along multiple paths because sensor information may have to be relayed to base stations that may be mobile. So we need connectivity between sensor nodes, and we want to have other properties such as node-to-node authentication. It turns out that some sensor networks have nodes with extreme communication and processing limitations, and consequently one has to employ lightweight cryptography. This might include use of hash functions, random polynomials of low degree that can be calculated efficiently on sensor nodes, and authenticated encryption in one pass with one cryptographic primitive.

Now imagine that we have this very large array of sensors, and if we want to make sure that inter-node communication is secure in some real sense we have to rely on cryptography. Hence, one of the things that one may have to do is key predistribution, in other words the proactive storing of all secret keys needed by a node when the node is manufactured. This may be a dangerous thing to do. Let's take a scenario. Suppose we construct a pool key of random keys, maybe 100,000 of them given that our the number of network nodes is only about 10,000. First, we randomly draw K keys with replacement from this pool and we store them in the first node. Second, we randomly draw another set of K keys and we put them in the second node, and so on. Here K is a small number, say 100-250 keys. We preinstall all these keys and then start scattering nodes around. The scattered nodes will discover their neighbours. An easy way to discover neighbours nodes, but not necessarily the best, is for each node to broadcast the IDs of its installed keys in the neighbourhood. Thus each of the neighbours can figure out which of its keys (if any) are shared with the broadcasting node.

This procedure leads to probabilistic key pre-distribution, and requires only a small number of keys for each node, which is good because the nodes that we're talking about don't have enough memory for storing thousands of these keys. However, there is some disadvantage to being clever here. This key predistribution scheme is obviously probabilistic. As a consequence some pairs of nodes will share the same key. Thus, one of the things that we've lost immediately is the ability to do node-to-node authentication, and I'll argue shortly that we can get it back under realistic assumptions. But anyway, the first question is, given N nodes find this K, which is the number of keys in each node, such that distributed sensor network is key connected.

Andrei Serjantov: Do you need key connected?

Reply: I don't need full key connectivity, but we need connectivity in the sense that any two nodes will have a path of keys so that can ensure secure communication between them.

The second question is, how do you find this minimum K and the maximum pool size P, given the number of sensors, N. For instance, our network might be spread over 70 square miles, with a node's direct communication neighbourhood comprising only about 30 to 40 sensor nodes.

Here the emergent properties are for key connectivity: we want to make sure that we have key connectivity by design. Now *suppose* that our network is a random graph. If this is the case, we are lucky, because we have a well-known Erdos and Renyi theorem which tells us that for certain probability of having a shared key between any two nodes we obtain key connectivity with a given probability and we work backwards to find the average key size K of a node.

Andrei Serjantov: Why is that a random graph?

Reply: Well, remember how you distributed these keys, we took our pool and randomly plug keys into the nodes. If we *approximate* the process of discovering neighbours to the random graph what this result says that given probability p of a link between nodes i and$j$, and N number of nodes, you actually get a target probability P for the node connectivity, which is equivalent to key connectivity. We want P to be large, say, .9999. If we work backwards, setting the target probability of emergence, and working backwards we find p and also the average degree of a node d, which captures the average number of node keys. The point is that for random graphs d is a very small number compared with N, but in general sensor networks are not random graphs (since direct connectivity is restricted to small neighbourhoods).

Ross Anderson: But if I want to communicate to any random node that has a red key, I can just simply send a message out to the network saying, this is a message encrypted under a blue key translate into red please.

Reply: The graph that I am talking about is not necessarily the communication radius graph, it's really the key connection graph, and the reason for which we consider it to be an approximation of a random graph is to illustrate the size of d or K vs. that of N.

Ross Anderson: Well in that case you're talking about the kind of links that I just described, even though they're vastly inefficient.

Reply: No, because nodes are already discovered key sharing. Remember a node broadcasts all the IDs of keys it has to its neighbours, and its neighbours find the keys they have which match the IDs (if any) of the broadcast keys. Thus if two IDs match, we have a shared key.

Mark Lomas: It's analysing the graph after you discover it.

Reply: Right.

Tuomas Aura: It doesn't sound like it is completely random, because they're using the physical connectivity as a constraint.

Reply: I haven't gotten there yet; it actually it gets a little worse in terms of the degree of a node (and meomory size), and it gets a little better in terms of local neighbourhood connectivity. So far I haven't really talked to you about key connectivity in the neighbourhood. Essentially what I want to do is to place this additional constraint which says, on this graph, I want to have high connectivity in each neighbourhood. It's OK to leave the graph as it is, but I want to tweak my parameters to give me perhaps a slightly larger memory requirement, but also give me more connectivity locally. This is because I don't want to traverse a multi-node key path when I actually want to talk to another node locally in the communication neighbourhood (which hears my broadcast).

So I now want to choose a higher degree of a node, in other words I want more keys per node. I do have the trade-off that if I increase the degree of a node for better local communication I'm going to spend less energy broadcasting messages because I sometimes eliminate multiple key hops within a neighbourhood. However, I need more memory to store the additional keys. By the way, if one computes the size of the memory that one needs for 250 keys for a node sensor, one gets a sizable number, say 2 kB for 64-bit keys; however, this is far smaller than approximately 80 MB needed for 9,999 keys within all the nodes of a 10,000 node network for pairwise node keying.

What are the security concerns of this key connectivity in emergent properties? There are some severe ones. One of them is that a node may be captured. Consequently we may need to have a capture detection algorithm, which is not easy to design, but which is the basis for doing revocation of the keys of captured nodes. We have to use key revocation because otherwise we operate with untrusted nodes, which means that node-to-node authentication becomes meaningless in practice.

If we do have such a detection scheme, then we can assume that we detect revoke captured nodes. If, at any point in time, there's reasonable high probability of captured (untrusted) node revocation, then I can remove the undesirable consequences of key collision due to the key distribution. That is, simply hash of a non-unique key and the IDs of the nodes sharing it increasing ID order, and now we get statistically unique keys that could be used for node-to-node authentication. Captured node revocation, however, has to be performed in a way (e.g., centralized) that does not require node-to-node authentication. Note that here I make a virtue out of necessity: node-capture detection and revocation are Necessary in any case, and if I have those, the remaining pool of nodes all are trusted. Consequently, I can use non-unique keys and still obtain node-to-node authentication, admittedly a fairly weak form because it's conditioned by the existence of these additional mechanisms.

Frank Stajano: It looks to me like all the items in the brackets are public, or are known to the people in that environment.

Reply: No, they aren't, clearly the IDs are known to the nodes in that environment, but the keys are not, because what we broadcast is the key IDs, whereas these are nodes IDs.

Frank Stajano: Yes, but anyone who has the same key ID in that environment will know that key.

Reply: Absolutely. But the point is that all the nodes that are left in the network are actually trusted, so when you do node-to-node authentication, you are assuming your adversary is the mythical man in the middle and not your neighbour node.

Chris Mitchell: You can avoid the need for node revocation by increasing the number of keys and arranging it so that the set of keys shared by any two nodes is not a node collection of the maximum number of revoked nodes that ever know that that's a key, you have to design your distribution of keys accordingly.

Reply: Yes, that was already designed by Chan, Perrig and Song. They said, increase the sharing of the number of keys between any two nodes to such an extent at least multiple key sharing between any two nodes, and hash those, and that gives you a unique key pairwise. The problem with this is that it's fairly resilient to node capture, better than what I have here, but only up to a certain threshold of captured nodes. Then the resiliency goes down dramatically. Essentially you now have these trade-offs between making these key connectivity property emerge under a different criteria, such as resilience to node capture, for example. This is still an active area of research, there are obviously many severe concerns here, and some of them are actually described in the Chan, Perrig and Song paper.

There is a couple of further examples of emergent properties in the paper, and one of them is an example I gave here about three years ago. This is an ad-hoc network built on the top of the wired internet. The ad-hoc network arises from the collaboration of multiple domains, and the emergent property arises from the need to establish a common access state among group of application domains; i.e., a state comprising a set of objects to which all domains have access. As it turns out, that this common state has to be negotiated among domains; e.g., the objects are routes between airlines, such as say USAir, United, and Delta. They may have sharing agreements resulting different types of shared routes, US-Europe, or Europe-Middle East, or Middle East-Asia, and different numbers of routes each domain has on those types of routes. Essentially domains end up negotiating routes among themselves, namely accesses to route objects, so that they run shared reservation applications, a accounting applications, auditing applications, and so on.

The question arises, how can we actually negotiate these common routes, and again we have a privacy concern because each of these domains may add some local constraints to negotiation that it cannot reveal. For example, a domain may want to have some routes not be shared with one out of, say, three other domains. If one desires privacy, then one negotiates under local constraints. However, the more one negotiates, the more one may violate the privacy of one's own constraints; e.g., the more it becomes apparent which are the routes that one don't want to give away. One can always get a negative outcome of the common access state negotiation, but if one negotiates a positive outcome it's very likely that one's privacy in terms of maintaining the secrecy of one's local constraints will actually be lost. Furthermore, availability concerns may also arise; e.g., a domain sabotages the negotiation of the common access state by giving all sorts of proposals that are going to be rejected by one or another of the domains. Integrity constraints may also arise; e.g., a one concern might the forced relaxation of some of these local constraints in a way that maintains compatibility with a domain's access policies.

My conclusions are the following. I believe ad-hoc networks, not necessarily only mobile wireless networks, but ad-hoc collaborations will become more widespread than they are now. Some collaboration applications be in mobile wireless networks, whereas others may be over the wired Internet. Some will be collaborations over wireless extensions of the Internet; e.g., over mesh networks. It might also be that whenever you have this size of networks, of collaborative identities, out of the collaboration dynamics you find these emergent properties. We really want to look very, very carefully at the security aspects of these emergent properties. Bob Morris used to say that a system that does not have a specification cannot be insecure, it can only be surprising. So we'd like these emergent properties not to be surprising in some fundamental way, and I believe that security really is a largely unexplored feature of all these properties which can appear at multiple levels in these networks.

Andrei Serjantov: What's the reference to some of the stuff that you've mentioned?

Reply: The common access state negotiation paper was presented at the Security Protocols Workshop in 2001[4].

The paper on probabilistic key pre-distribution paper, which was the first paper published in this area, appeared in the proceedings of the ACM Computer Communications Security Conference in November 2002[5]. Since then here have been at least 40 papers published in this area, so there's plenty to look at.

Chris Mitchell: And there are versions of key pre-distribution published by Li Gong and David Wheeler in early 1990s[6] I believe. I've published a paper on key pre-distribution in 1988[7].

Reply: Yes but these do not account for sensor-network communication restrictions and memory constraints. Actually it turns out that some of these schemes, such as the Blundo scheme that was published in 1991, can be made to work in

[4] Virgil D. Gligor et al, *On the Negotiation of Access Control Policies*, LNCS 2467, pp188-212.

[5] L. Eschenauer and Virgil D. Gligor, *A Key Management Scheme for Distributed Sensor Networks*, ACM CCS 2002, pp41-47.

[6] L. Gong and D.J. Wheeler, 1990, *A Matrix Key Distribution Scheme*, J. Cryptology, 2(2), pp51-59.

[7] C.J. Mitchell and F.C. Piper, 1998, *Key Storage in Secure Networks*, Discrete Applied Mathematics 21, pp215-228.

this environment with some modification. Some of the other people who published in this area subsequent to us, looked at why we discarded the Blundo scheme, and they actually made it work. The reason why we discarded it is we had the trade-off in our mind between the amount of computation we had to do and the memory we had. We wanted to have very little memory, and absolutely zero computation.

Chris Mitchell: But there's quite a bit of work on the common construction of optimal allocation of keys, so you can handle collusion by up to some threshold number of bad guys and still have enough keys to make sure you've got at least one that nobody else has.

Reply: Yes, but this is an entirely different area.

Mike Burmester: I think trust was the basis, somehow, of your talk.

Reply: Our notion of trust establishment was presented at the Security Protocols Workshop in 2002[8], based on prior work done by others, including Kohlas and Maurer, for instance. But there are other papers on trust management, though trust management is a somewhat different area.

Mike Burmester: My feeling is that the metrics may be modelled by some kind of sub-graph, or coloured graph, or a more complicated structure, it won't be a metric number. It will be some sort of element in a lattice, or perhaps a ordered partially set.

Reply: Absolutely. In fact we are keenly aware of some of this work. But what we really wanted to do (remember I mentioned something about lightweight of cryptographic protocols), is essentially very dangerous: we'd like to compute as much of the protocol off-line as we can because then you reduce the attacks which you get if you can actually do free computation of protocol steps on line. For example, you might not have chosen plain text attacks if there is no protocol oracle to provide chosen plain text.

This trade-off there is very difficult, and this is a very dangerous thing to do. Notice what we did: we essentially eliminated the runtime key distribution protocol as we just a broadcast unencrypted information. There is no cryptographic protocol to speak of; everything that one needs as far as cryptographic keys are concerned is stored in sensors. However, if a sensor is captured all its keys are compromised. In contrast, if a key distribution protocol is run on-line and getting keys only when needed, a node capture results in the compromise of a subset of all keys that node will use.

Further, we exploited here a fundamental trade-off. We actually performed very little computation, and we used very little memory. It's extremely difficult to come up with something more efficient than than probabilistic key pre-distribution protocols. What's possible is to come up with pre-distribution schemes that are less efficient but more robust as far as capture is concerned. For instance, with probabilistic key pre-distribution, shared keys between different

[8] Note 3, op cit.

pairs of nodes may collide. Thus capturing a node that has colliding keys with another nodes will cause a fraction of network communication to be compromised. Other schemes proposed have other kinds of behaviours; e.g., they are more resilient to node capture but less efficient or, as in Blundo et al scheme, they may be perfectly resilient to node capture up to a point and then all communication becomes compromised. In other words, resiliency is a step function in the number of nodes compromised. A basic trade-off one has in this area is among computation, memory size, and resilience.

Mark Lomas: The assumption is it takes time to compromise a node, wouldn't it be useful to employ revocation to just throw away all of the key data rather than to negotiate new keys?

Reply: Essentially that's what happens in the work done by Chan, Perrig, and Song. They perform distributed revocation whereby a number of neighbours figure out that a certain node is compromised and proceed to revoke all the shared keys they have with it.

Mark Lomas: No, I'm being more extreme than that, I'm saying as a matter of policy you just refer every single key in the network...

Reply: But then how do you distribute the keys?

Mark Lomas: You set up a correspondence with the neighbours that you can communicate with.

Mike Burmester: It's cheaper to throw in another set of nodes because each such thing costs a dollar.

Reply: Yes, that's exactly the whole point.

An advantage of this scheme is that it's very easy to just extend the network further, or to replace it: just throw in more nodes, whereas with some of the other schemes one really has to be careful because one already has unique pre-established links (i.e., keys), and a node may actually not end up in a neighbourhood of nodes with which it has *a priori* established shared keys. Further, there are very basic trade-offs in terms of energy consumption, ease of network extension, and resiliency (in terms of the fraction of communication compromise by node capture).

Mike Burmester: I think the point is the lifetime of the system is very short, just about 24 hours before something like invading a country, and after 24 hours you're OK.

Pseudonymity in the Light of Evidence-Based Trust

Daniel Cvrček[1] and Václav Matyáš Jr.[2]

[1] University of Cambridge, Computer Laboratory
Daniel.Cvrcek@cl.cam.ac.uk
[2] University College Dublin, Ireland & Masaryk University Brno, Czech Rep.
matyas@informatics.muni.cz

Abstract. This position paper discusses the relation of privacy, namely pseudonymity, to evidence-based trust (or rather reputation). Critical concepts of evidence-based trust/reputation systems are outlined first, followed by an introduction to the four families of the Common Criteria (for security evaluation) Privacy Class: Unobservability, Anonymity, Unlinkability, and Pseudonymity. The paper then discusses the common problem of many papers that narrow the considerations of privacy to anonymity only, and elaborates on the concept of pseudonymity through aspects of evidence storing, attacks and some of their implications, together with other related issues like use of mixes.

1 Introduction

The reasoning introduced in this paper is based on the following general idea of reputation (or trust-based) systems. Systems do not utilize enrollment of users – there is no objective knowledge about their identities [3, 12]. All the system can use is evidence of previous behaviour. Functionally, there are three types of nodes in the system. Requesters/clients are exploiting services and resources offered by servers. Servers may use recommending service of recommenders – nodes that have an interaction history with requesters. Kinateder and Pearson [9] use slightly refined description of recommenders as they define recommender with own experiences and accumulators with mediated evidence.

The system works basically with two sets of evidence (data describing interaction outcomes). First set contains data relevant to a local system. This data is used for real-time risk analysis evaluating security of the local system in various contexts (it may be non-deterministic process in some sense). This set may be also referenced as derived or secondary data.

Primary data is in the second set. As mentioned above, the evidence is delivered (or selected from locally stored data) according to a given request content. Records from this set are used for reputation evaluation to grant/reject access requests. Data in the second set may contain information from third parties representing evidence about behaviour collected by other nodes – recommenders. Note that there may be an intersection between the two evidence sets with implications to privacy issues that we are investigating at the moment.

The approach of reputation systems is rather probabilistic and this feature directly implies properties of security mechanisms that may be defined on top of such systems. The essential problem arises with recommendations that may be artificially created by distributed types of attacks (Sibyl attack [8]) based on huge number of nodes created just to gather enough evidence and achieve maximum reputation.

A node functional model is based on two data flows. Risk analysis is based on locally stored *secondary* data and results of the analysis should adjust rules for access control. This process may be deterministic or probabilistic according to the way the search for behavioural patterns in the evidence data is conducted. Trust analysis is based on primary data about behaviour of a requester and trust values are compared against access control rules.

1.1 Trust and Privacy or Trust vs. Privacy?

We come then to the following question: *How much trust may someone assign to me without compromising my privacy?*

The term privacy is currently understood as the right and ability to protect one's personal space (including personal data) and to prevent invasions into this personal space. It can be then expected that increased amount of evidence necessary for higher reputation implies higher probability of privacy compromise.

A limited amount of evidence/data related to the subject can lead to limited trust level being achieved yet desirable for the subject of trust reasoning. This situation is desirable in mix-based services and it in contrary *increases trust* in the mix. The trust then reflects predictability of behaviour (size of possible recipient set) here.

What do we mean with privacy – anonymity or pseudonymity (for more definitions see below), in other words, is it worth and even possible to require anonymity from some level of mutual trust or would it be better to concentrate on pseudonymity?

2 What Could Privacy Be?

Privacy is not only about subject (user) identity. Let us consider, for example, three questions relevant to user privacy:

- *Who* is the user, e.g. browsing xxx.com from this room?
- *Which* is the user, e.g. browsing xxx.com from this room? (I.e., we are not concerned with that user's identificator as such but with her properties.) This question could be risen by a script on xxx.com to decide whether it is possible to infect the user's laptop.
- *Is there any* user, e.g. browsing xxx.com from this room? This may be a question of a law enforcement agency before entering the room.

We can also identify three most suitable ways to “measure” privacy:

– To what degree is data personally identifiable?
– How certain can we be in linking pieces of evidence?
– To what degree is an action observable?

The Common Criteria [4] class Privacy deals with aspects of privacy in the Trusted Computing Environment and as proposed is composed of four families. Three of these families have a similar grounding with respect to entities (i.e., users or processes) whose privacy might be in danger. They describe vulnerability to varying threats, which make them distinct from each other. These families are Unobservability, Anonymity, and Unlinkability. The fourth family – Pseudonymity – addresses substantially different kind of threats.

Unobservability: *This family ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used.* The protected asset in this case can be information about other users' communications, about access to and use of a certain resource or service, etc. Several countries, e.g. Germany, consider the assurance of communication unobservability as an essential part of the protection of constitutional rights. Threats of malicious observations (e.g., through Trojan Horses) and traffic analysis (by others than communicating parties) are best-known examples.

Anonymity: *This family ensures that a user may use a resource or service without disclosing the user's identity. The requirements for Anonymity provide protection of the user* (bound to a subject or operation) *identity. Anonymity is not intended to protect the subject identity.* Although it may be surprising to find a service of this nature in a Trusted Computing Environment, possible applications include enquiries of a confidential nature to public databases, etc. A protected asset is usually the identity of the requesting entity, but can also include information on the kind of requested operation (and/or information) and aspects such as time and mode of use. The relevant threats are: disclosure of identity or leakage of information leading to disclosure of identity – often described as “usage profiling”.

Unlinkability: *This family ensures that a user may make multiple uses of resources or services without others being able to link these uses together.* The protected assets are of the same as in Anonymity. Relevant threats can also be classed as “usage profiling”. Note that it is sometimes questioned whether this family deals with substantially different threats and applications from those of Anonymity.

Pseudonymity: *This family ensures that a user may use a resource or service without disclosing its user* (bound to a subject or operation) *identity, but can still be accountable for that use.* Possible applications are usage and charging for phone services without disclosing identity, “anonymous” use of an electronic payment, etc. In addition to the Anonymity services, Pseudonymity provides methods for authorization without identification (at all or directly to the resource or service provider).

3 Evidence and Privacy

The crucial question to be answered here is whether it is possible to protect against identification of an individual. What is the cost difference between anonymity and pseudonymity? Can we define relations between amount of evidence and cost of preserving anonymity/pseudonymity? We are (currently) of the opinion that there are higher risks (misuse of evidence that is not personalised) when anonymity is addressed. For it is clear that even if we can reduce the set of suspects such that the set size is 1, i.e. there would be no (or lowest level of) anonymity according to current definitions and perception of anonymity, e.g. [5, 6, 9, 10], we could still have some level of pseudonymity, i.e. probability of linking a personal identity with the given pseudonym less than 1.

3.1 Evidence Store

We come to one of the critical issues here – *Who should store information needed for trust evaluation*:

1. *Server* (service provider) – most convenient for the purpose of local risk analysis and adjustment of rules for access control to local resources. We do not need true client identity, although certain amount of "linkability" is desirable, for that. Basically, only the information computed locally as part of previous (e.g., access control) decisions is needed. However, this holds true only when a service provider is using this information and making the decision on its own, and it is often advisable to use more sources of evidence to increase trust (yet obviously this endangers privacy most).
2. *Client* (subject of information) – ideal from the privacy point of view because I (the subject) can decide what I really want, can and will provide with a service request – i.e., typically the required minimum. On the other hand, client usually does not have enough knowledge to guess volume of necessary evidence. This implies some tricky requirements for correct specification of communication protocols. What seems plausible at the moment is that the service request would consist of a number of request-response interactions. Another approach would be publishing the evaluation criteria for clients' behaviour. For the latter, one can find good reasons both for (transparency like in case of state benefit ratings, etc.), and against (commercial secrets, need to suppress change of behaviour reflecting knowledge of the criteria, security critical information from deployed security mechanisms, etc.).
3. *Distributed storage* – pieces of evidence are spread over a network, e.g. using the Eternity Service [2] (this is a variant of accumulators as defined in [9]). The problem is to ensure that evidence will not remain on server using it to calculate reputation value. One of possible ways may be use of distributed calculation. This however brings question of legitimacy of evidence/results used to get final reputation/trust value.

Another question – is (3) suitable candidate for a compromise between (1) and (2) or not? It may be suitable as replacement of (2) in case that agents have constrained memory, presuming that we are able to address issues of access control, confidentiality and availability.

3.2 Issues of Evidence Store

Initial ideas about attacks partitioned according to holder of evidence data:

1. Evidence held by the service provider – typically about numerous clients, evidence originated at this (one) source. Obviously, situation will be rather different when indirect evidence is taken into account:
   - Privacy – definitely the issue here, main focus would actually be unlinkability when not using any real identities.
   - Integrity is important with respect to server accountability in situations where servers have motivation to change content of data. It is not a big issue when evidence is used just locally but it could be a considerable problem with cooperating servers – recommenders.
   - Another problem is colluding servers – servers may create faked (positive or negative) evidence.
   - Availability attacks – may be interesting to disable recommending functionality, a profound property for distributed environment.
2. Evidence held by the client – it contains data that would probably be integrity-protected (otherwise clients may change evidence) by the service providers, the evidence is most likely only about the particular client. Client would like to increase positive evidence – selling/buying among colluding parties maybe an issue here – cryptography is the only countermeasure. Also, the client would often wish to keep his/her evidence confidential to himself/herself and the services that would request this evidence.

A related question is whether it is advantageous to store complete set of evidence about my previous behaviour and be able/must select the proper subset (ensuring positive response) while requiring a service. Note that selecting "bad" evidence can prompt the provider to lock the service, i.e. making this alias/ID useless to its owner. Note then in turn that this can be useful in case someone would consider leading an obvious related attack.

4 Accidents Happen

One has to consider that building up a credit associated with a given pseudonym usually is not an effortless exercise – the value is determined by the cost (money, time, etc.). It is a loss when a pseudonym must be thrown away and a new pseudonym has to be built from scratch. This cost may be tuned by weights assigned to contexts of evidence pieces.

Issue – should this involve also the requirement to have this new pseudonym not related to the old pseudonym, i.e. take care of such (between-set) unlinkability? It is usually the case, otherwise it is pointless to create new identity. A simplistic view is that negative reputation is not relevant since an agent can just change its name and his trustworthiness is on an initial value. However, there is a risk of someone being able to link two identities of an agent, namely those who would profit from knowing the relation between the *NewID* that the *OldID* (pointing to a negative trust value for the *NewID* rather a zero/initial value). Contradiction between negative (trust/reputation) value and positive "context" value could be worth examining in this context. And obviously, the owner of *NewID* and *OldID* may not wish anyone to link two sets of evidence for privacy reasons.

4.1 Accident Recovery

The question then is – *how hard it should be to create a new pseudonym with good reputation rating*? Good question – we believe that there are two requirements: be as friendly as possible to users, and sufficiently robust against possible attacks. We can set the cost of threshold "just useful" (for a given purpose) pseudonym with associated evidence from the statistical data about system usage. The problem is how to measure the cost [1] so that we can set it correctly. A vital issue to consider is then the cost difference between creating reputation for x and $x + k$ clients.

When a number of pseudonyms exist for a particular user, (s)he may (when deemed suitable to his/her own intentions) decide and link several pseudonyms to give a particular server enough evidence. However, his/her crucial requirement would be that the linkage depreciates as soon as possible – the data is part of one's history that is potentially valuable for a long period of time. So far we are aware of only one way to provide linkage that cannot be (ab)used in the future – to provide a zero-knowledge type of proof of the link existence. Yet while this solution might be used in some situations, on others (where the link as such must be shown) it would be of no use.

5 More Questions Instead of Conclusions

The issues of anonymity and pseudonymity in relation to trust are investigated from various perspectives and in different environments [3, 6, 7, 12]. While we focus on the issue of evidence-based trust (reputation) with respect to clients (of and within) mix networks, there are clearly more areas being (and ready to be) explored.

Anonymous remailers constitute an existing application directly involved in anonymity issues. In fact, this is an application allowing for reasoning about several interesting aspects of reputation. The Sibyl attack [8] is of limited importance here as remailers are usually loosely federated systems with known set of nodes. Each node may locally compute properties of other nodes in the network and identify those sending considerably lower number of messages – i.e., suspected subset of colluding servers trying to reveal identity of mix clients.

Another goal of reputation reasoning can be to measure the behaviour of mix clients. As described in [11], an adversary may launch a statistical attack based on predictable behaviour of mix clients when their sets of recipients are rather small compared to the total number of mix clients. The reputation of clients may be used to identify messages with small predictable set of recipients and bounce them back to senders or generate dummy messages to *decrease* reputation of senders and thus increase set of probable recipients. Note that reputation is a gauge for behaviour predictability in this case.

One related issue with respect to use of mixes that has already been discussed and should be taken into account (e.g. [5]) relates to the use of a mechanism for detection of attacks based on number of colluding nodes – a probabilistic countermeasure of some realistic applicability. The idea is that you know whom you are asking for reference of someone's trustworthiness, but you (and neither a recipient of the request) have no idea about the path of your request – the randomness in the routing could be used to detect cliques of colluding subnets. Yet note that this is in turn essentially unlinkability attack from the privacy point of view.

Last but not least, enormously challenging area for future research concerns plausible ways to "depreciate" links between evidence (data), at least for the server (deletion/depreciation) only. Would fresh temporary pseudonyms (unlinkable to primary identities) used just for one-time links be the right way to approach this issue?

References

[1] A. Acquisti, R. Dingledine, and P. Syverson. On the Economics of Anonymity. In J. Camp and R. Wright, editors, Financial Cryptography, 7th International Conference, FC 2003, SpringerVerlag, LNCS 2742, pp. 84–102, 2003.

[2] R. J. Anderson. The Eternity Service. In Proceedings of Pragocrypt'96, pp. 242–252, Prague, Czech Republic, 1996.

[3] V. Cahill, et al.: Using Trust for Secure Collaboration in Uncertain Environments. In IEEE Pervasive Computing Magazine, pp. 52–61, July-September 2003.

[4] The Common Criteria Project Sponsoring Organisations: Common Criteria for Information Technology Security Evaluation - part 2, Version 2.1, August 1999.

[5] R. Dingledine and M. J. Freedman and D. Hopwood and D. Molnar: A Reputation System to Increase MIX-Net Reliability, In Proceedings of the 4th International Workshop on Information Hiding, Springer-Verlag LNCS 2137, pp. 126–141, 2001.

[6] R. Dingledine, N. Mathewson, and P. Syverson: Reputation in Privacy Enhancing Technologies, In Proceedings of the 12th annual conference on Computers, freedom and privacy, pp. 1–6, ACM Press, San Francisco, California, 2002.

[7] R. Dingledine, N. Mathewson, and P. Syverson: Reputation in P2P Anonymity Systems. Workshop on Economics of Peer-To-Peer Systems, June 2003, Berkeley, California, 2003.

[8] J. Douceur: The Sybil Attack. In 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), Springer-Verlag LNCS 2429, pp. 251–260, 2002.

[9] M. Kinateder, S. Pearson: A Privacy-Enhanced Peer-to-Peer Reputation System. In K. Bauknecht, A. Min Tjoa, G. Quirchmayr (ed.), Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies, EC-Web 2003, Prague, Czech Republic, September 2003.

[10] A. Rezgui, A. Bouguettaya, and Z. Malik. A Reputation-based Approach to Preserving Privacy in Web Services. In Proceedings of the 4th International Workshop of Technologies for E-Services, TES 2003, LNCS 2819, pp. 91–103, Berlin, Germany, September 2003.

[11] A. Serjantov and G. Danezis: Towards an Information Theoretic Metric for Anonymity. In 2nd International Workshop on Privacy Enhancing Technologies (PET 2002), LNCS 2482, pp. 41–53, San Franciskp, USA, April 2002.

[12] B. Shand, N. Dimmock and J. Bacon: Trust for Ubiquitous, Transparent Collaboration. In Proceedings of the First International Conference on Pervasive Computing and Communications, PerCom'03, pp. 153–160, Texas, USA, 2003.

Pseudonymity in the Light of Evidence-Based Trust (Transcript of Discussion)

Daniel Cvrček

University of Cambridge and Technical University Brno

George Danezis: Surely as soon as you link two pseudonyms once, then you can't take that information away from an adversary. But what you said may be theoretically possible if you could prove that you also have another pseudonym that has good reputation without revealing it, which is really I think the property you're looking for.

Vaclav Matyas: Well practically this is what we would like to see, and the only way we've discovered so far is some kind of ZKP, so that you can prove that there are links between the pseudonyms that we operate with. We don't see anything better so far, but we would eventually like to find a different approach.

Pasi Eronen: I guess you could also do something like that with some more trusted computing platform. That could be a third party that links these pseudonyms but doesn't reveal the information to anyone else.

Reply: There may be certain small domains of trust in the system but the domains are not the whole system. There are no globally trusted parties.

Vaclav Matyas: Surely some of the military wouldn't mind to introduce certain level of uncertainty in the system so that you can eventually plausibly deny maybe all your pseudonyms?

Reply: What we are proposing is the possibility to create a number of different pseudonyms not revealing the real identity of user. In typical environments there is a very small difference between the price of creating one, and let's say, two or three other pseudonyms, because you can automate the process too easily.

Secure Processor Consistent with Both Foreign Software Protection and User Privacy Protection

Mikio Hashimoto, Hiroyoshi Haruki, and Takeshi Kawabata

Corporate Research and Development Center, Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan
{mikio.hashimoto, hiroyoshi.haruki, takeshi3.kawabata}@toshiba.co.jp

Abstract. We developed a secure processor that protects foreign software running on an open source operating system (OS). The implementation of such a processor has not been reported yet. The processor is called L-MSP (License-controlling Multi-vendor Secure Processor). In this paper, the relationship between software protection and user privacy is discussed. In general, software protection enables secure distributed computing and copyright protection, but the software might violate user privacy. In the L-MSP system, user privacy is able to be protected by security sandbox and related OS mechanisms which is implemented in an open source. Thus transparency of user privacy protection can be assured on the system.

1 Introduction

Grid computing and other distributed computing have been getting popular, but there are some difficulties for widely spreading of them. One of the major difficulties is data confidentiality and integrity in a grid system.

In grid computing, computation requesters inevitably send their data to many remote systems, so remote system administrators must be reliable for data security [1]. But significant numbers of administrator of general grid computing system are amateur or naive end-users. So, requiring obligation and authentication to those end-users are not favorable for them. In addition, strong authentication is possible to spoil end-user anonymity. This situation can be applied to mobile agent [2].

Foreign software protection may become the solution for the problem. Foreign software protection (software protection) means secure processing of a software on a remote target system which might be managed by hostile administrator. Software protection is also useful for content and software copyright protection.

On the other hand, from the viewpoint of target system administrators or users, foreign software may invade privacy on the system, such as viruses. System users definitely require protection of their system from foreign software. In addition, the protection mechanism is better to be clearly disclosed, because its security flaws are easily inspected. The confrontation between software protection and target system protection is the essential problem in distributed computing.

Given the above background, we have designed a microprocessor which enables foreign software protection on an open source operating system. We call it L-MSP (License-controlling Multi-vendor Secure Processor), since it was originally designed for license control of copyright protection. Through the functional testbed implementation, the design consistency for existing open source operating is verified. There are several proposals of such a processor [3,4], but a functional design has not been reported yet.

This paper is organized as follows: L-MSP architecture and testbed implementation is introduced briefly in next section. Then the significance is discussed using remote computation model of software protection. The relationship between privacy protection and software protection is discussed in section 5, and our conclusion is presented lastly.

2 Introduction of L-MSP

2.1 Basic Idea

L-MSP provides secure process execution for plural software vendors. Figure 1 shows the operation model of L-MSP. Software vendors, indicated left side on the figure, define a secret key for its program (program key). The original plaintext program is encrypted by shared key algorithm using the program secret key.

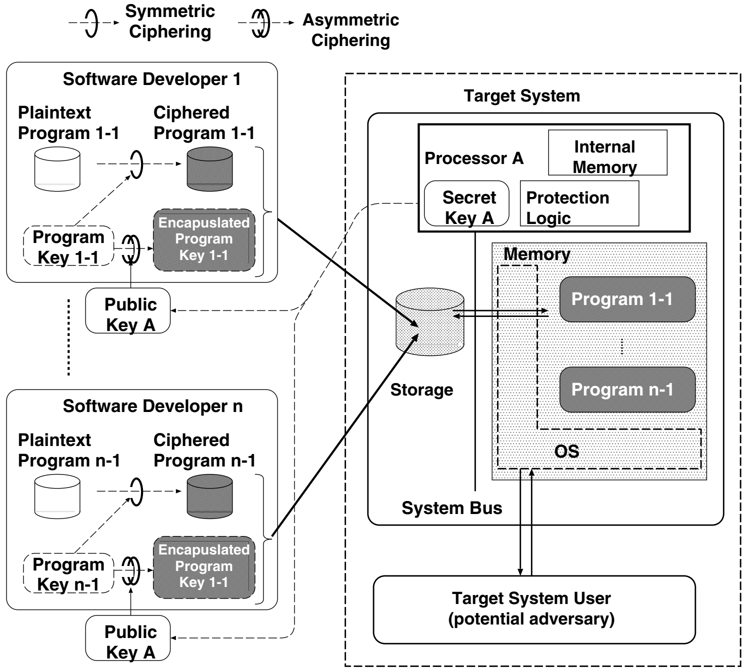

Fig. 1. Operation of Multi-vendor Secure Processor

The program key is encrypted by public key cipher using target processor public key which is corresponding to the processor secret key embedded in the processor chip with tamper proof. The encrypted program key is called encapsulated program key.

These encrypted program and encapsulated program key are sent to the target system. On the target system, this information is placed in external memory. OS can read any content in external memory. But those two informations are secure, because encrypted using unknown key from the OS or target system users. When executing the program, reverse action is done inside the processor using processor hardware which cannot be modified by users.

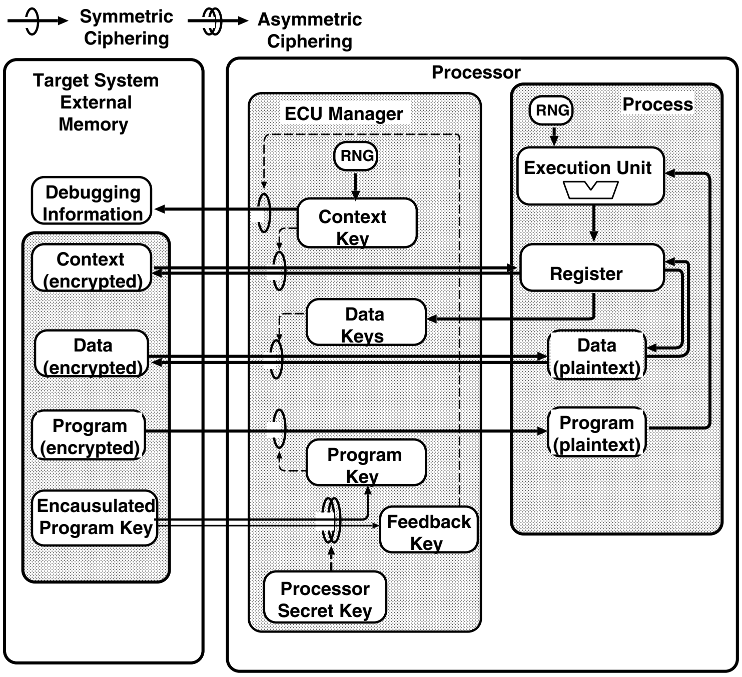

Fig. 2. Process Protection Model of L-MSP

Though this idea which was originally introduced by Arnold [5] works well basically, much more considerations are required to make practical processor compliant to modern open source OS. Our assumption is described below.

The target of protection includes all of process information, those are program (instruction), data, and context (registers). Those informations placed in external memory are encrypted using some ciphering keys shown in figure 2. Encryption and decryption is done by the processor hardware when program or data are fetched by process execution. Context switch is also done by the processor. The context is written to external memory as encrypted format, during the time that the process execution is suspended.

There are several keys and those keys are managed by the processor in connected with specific process. The OS is unable to get those keys which are managed by the processor, but the OS can control process execution and physical memory allocation.

2.2 Architecture

The hardware block diagram of existing processor and L-MSP is shown in figure 3(a) and (b) respectively. L-MSP processor includes additional four functionalities shown below as compared to existing processors.

- Ciphering protection mechanism for external memory information
- Hardware context switch

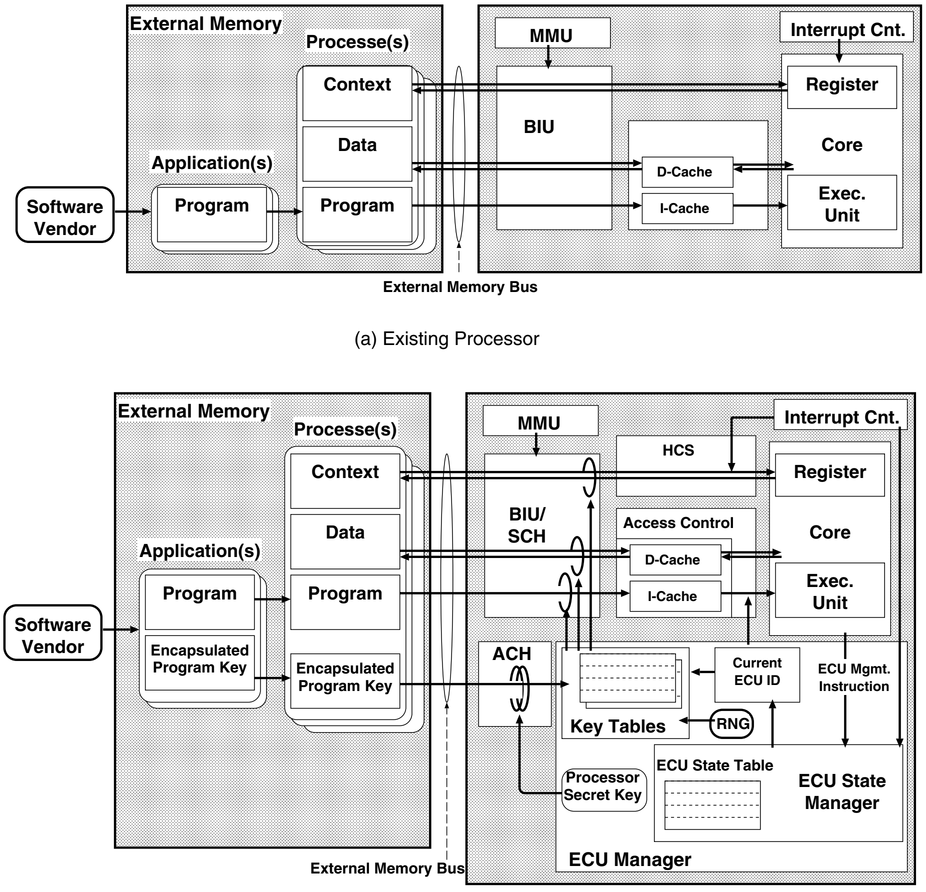

Fig. 3. Hardware Block Diagrams

- Access control mechanism for processor internal information
- Process management function including key management

Ciphering is done in ACH (Asymmetric Ciphering Hardware) and BIU (Bus Interface Unit)/SCH (Symmetric Ciphering Hardware). Access control is done on cache memory and context switch is done by HCS (Hardware Context switch). Process management is done in ECU manager. ECU is abbreviation of Execution Control Unit. ECU is an unit of process management in L-MSP. Protected process has corresponding ECU[1]. ECU manager has several table and maintains them in L-MSP.

OS is able to invoke, stop, resume and discard a process of a protected program, following out the administrators intention. These operations are indicated to ECU manager by issuing special ECU management instructions by OS execution. OS is also able to read and move the process information from external memory to secondary storage and vice versa. But OS is absolutely impossible to get any plaintext fragment of program, data, or context. Because those informations are encrypted by ciphering keys hidden from the OS.

The process is protected by ciphering as if it were a blackbox, but its execution and memory allocation is absolutely controlled by the OS. For example, OS is possible to prohibit unauthorized network access by the protected process using memory access control which is achieved by virtual memory allocation of MMU. This means isolation of system resource management from secret management of protected processes. In other words, foreign software protection and resource protection are orthogonalized.

The design object also includes separation of secret and mechanism for the protection. It is better that the process protection mechanism is publicly disclosed, since the mechanism can be inspected by many people. In addition, it is better that protection mechanism is able to be used by everyone including end-user programmers, not only big industries, since the mechanism would benefit many people as possible.

2.3 Debugging Feature

The L-MSP architecture has many common features with those of previous proposals [3,4]. One major new point is the consideration of debugging. Programs usually have several defects, so failure analysis to improve released programs is done by using process memory images or so-called core dump. But if the context part of core image is encrypted by random number generated inside the processor, even software vendor cannot decrypt the context. This point has not been treated in previously proposed architecture.

In L-MSP, encapsulated program key may include feedback key which is dedicated for debugging. When a special instruction to generate debugging information is issued by OS, the internal random number to decrypt the context is

[1] ECU has its unique ID inside the processor. In general, ECU ID is chosen independently from corresponding process ID, and this relationship may be disturbed. So, discrimination between ECU and process is important for deliberate discussion.

encrypted by the feedback key and then written out to the memory. If target user wants to send the debugging information and core image, these informations are sent to the software vendor of the program by target system user himself. The software vendor recovers the context key using his feedback key. This process is important for developing mature product.

2.4 Testbed Implementation

We have finished the functional design of proposed architecture. The design is implemented on FPGA circuit board (Figure 4). In addition, OS and several sample application are also implemented (Figure 5). Using this testbed system, L-MSP functions, such as on-the-fly ciphering, process management, and secure context switches, are verified. Key features are summarised below.

Processor	modified Toshiba Media Embedded Processor (MeP) [7]
Design Language	Verilog HDL
Hardware	Altera EP1S80 FPGA
OS	modified uITRON (TOPPERS JSP [8])

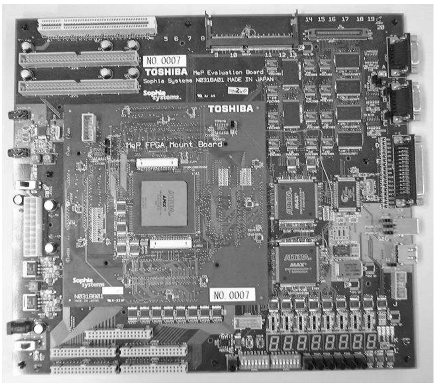
Fig. 4. Testbed Circuit Board

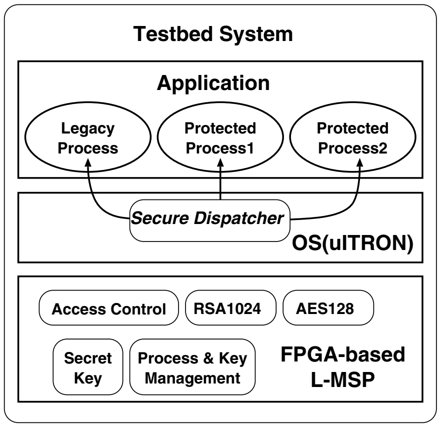

Fig. 5. Testbed Software Structure

Because of hardware restriction of FPGA, following overhead estimation is done on ASIC (0.13μm CMOS technology). Hardware size overhead is around 200K Gates, that includes AES128 and RSA1024 cipher core. Performance overhead is mostly resulted by AES processing delay. In current CMOS technology, the delay is 100ns, equivalent to the delay time of DRAM. The performance degradation depends on cache hit ratio, it is below 10% at 98% of cache hit ratio. This impact is acceptable even for embedded processor for small appliance.

Gate Size	total 400KG
	200KG (original RISC core)
	200KG (additional part)
Cipher Core	AES128 (SCH), RSA1024 (ACH)
Performance	10% degradation of throughput
	at 98% cache hit ratio

3 Remote Computation Model of L-MSP

Process execution of external program in a local system can be interpreted as a form of an remote computation. L-MSP can be interpreted as an Trusted Third Party in target system which is managed by hostile operating system from the viewpoint of software vendor of the program. Figure 6 represents the model schematically.

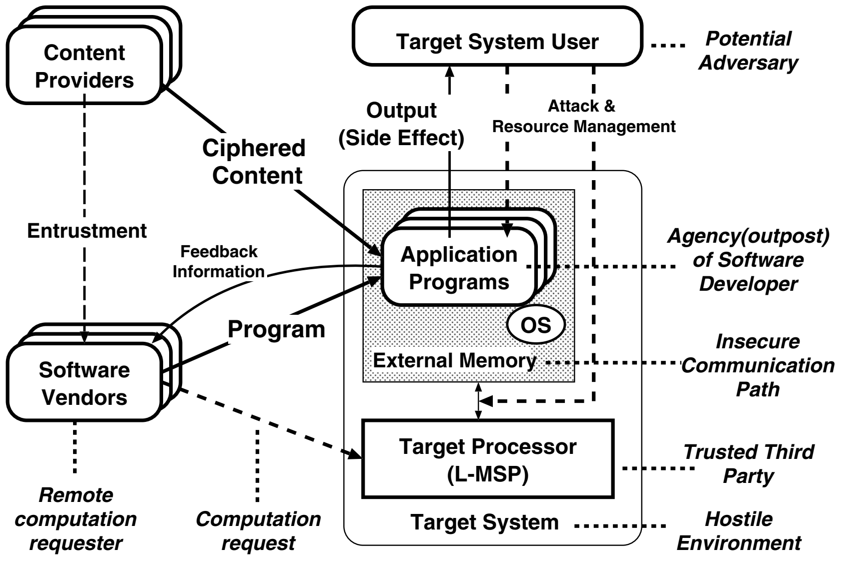

Fig. 6. Remote Computation Model of Foreign Software Protection

Software vendor corresponds to the remote computation requester. Protected program corresponds to the remote computation request. Protected program is sent to the target system, only L-MSP can decrypt and execute the program which is protected cryptographically.

While the process or protected program is running, the execution is protected from the OS and other processes by L-MSP. If the information of protected process is modified, L-MSP cannot prohibit the modification of memory, but can detect the modification by MAC verification at memory access. If L-MSP detects such modification, the process execution is stopped and never be resumed.

Finishing the execution, process can return the computation result using secure communication path established to the computation requester. To establish secure communication path to the requester, the process does authentication. The secret for authentication may be embedded in the instruction stream of the program or in static variable. Confidentiality of the authentication secret from OS is assured by L-MSP. So, the security of computation result can be assured regardless of trust for whole target system or the system administrator.

Process may establish secure communication path to other entities rather than the requester. The secret for the authentication may be embedded in the program initially, or transferred from computation requester using the secure communication path.

Copyright protection is an example of this type of relationship [6]. Content provider entrusts the secret key for the content to software vendor. Software vendor provides playback software to end-users. In copyright protection, the computation result is not necessary to return to the requester, but the process's by-product, or human recognizable output, is delivered to system users. In this case, debugging information is the only result returned to the requester.

The process plays a role of agency of the requester in the target system. In the copyright protection example, the agency enforces the copyright protection rule. System administrator is potential adversary which obstructs the agency. Target system Memory and OS corresponds to insecure communication path between software vendor and L-MSP.

4 Privacy Protection and Foreign Software Protection

4.1 Security Sandbox for Privacy Protection

As described previously, grid system or any kind of distributed computing require sending data to remote system from requester. Unexpected data inspection or modification in the target system are unfavorable for the requester. Those attacks to the computation result are unfavorable too.

Without foreign software protection, these possibilities cannot be excluded, if the requester is not relying on remote system administrators in terms of moral and responsibility. Requirements of moral and responsibility for system administrator are the obstacle for wide spread of grid computing, since almost administrators are naive end-users who avoid the burden of those. Those requirements possibly impair anonymity and privacy of end-users.

On the other hand, a target system faces the threat of foreign program being a virus. Some viruses simply consume the system resources. Other viruses break the system or inspect target system user privacy. System administrator requires to avoid these possibilities.

This confrontation can be resolved using combination of proposed L-MSP and simple security sandbox on the target system. The scheme is presented in figure 7.

Since foreign software is protected by L-MSP independently from target OS, even privileged attack code cannot break the protection. Foreign software is

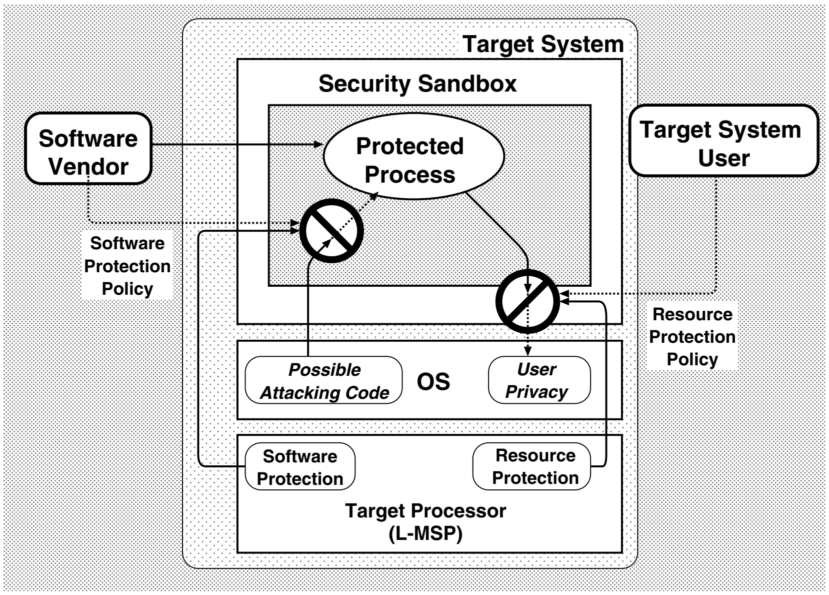

Fig. 7. Software Protection and Security Sandbox

launched using security sandbox, that does the access control to local resource from the software. The access control is based on the OS's resource allocation and processor's resource protection mechanism.

While the access control is adequately configured, user privacy is assured to be protected from foreign software. Even if evil foreign program could get some user privacy information unfortunately, network transfer of it by the program can be prohibited by the security sandbox.

Software protection and resource protection work by following two policies, software protection and resource protection, respectively. It should be noted these are supplied independently by software vendor and target system user.

It is also important that resource protection mechanism can be verified by the target system user himself. In an open source operating system, interfaces between the security sandbox, OS resource management, and processor resource protection mechanism are able to be verified by the user. Because L-MSP has orthogonalized protection mechanism, resource management mechanism can be definitely disclosed to users. This is the substantial feature of the consistency of open source operating system.

If these two protection mechanisms of a secure processor are entangled, OS might be able to crack the foreign software protection using its privilege. Otherwise, the OS wouldn't have enough resource management capability. In latter case, OS couldn't stop faulty process which consumes excessive system resources. The most significant risk of such a system is that foreign process might violate user privacy using security holes which cannot be verified by the user. This might lead to censorship [9].

As for grid system operation, L-MSP may reduce system maintenance cost. In current grid computing system, remote system administrators are assumed to be reliable for data security, and those system requires tight authentication [1]. But in L-MSP system the security of remote computation is resolved into the matter of the security level of L-MSP, instead of target system administrator[2]. So, applying this technology to utility computing servers, maintenance cost of administrators with higher loyalty should be reduced.

4.2 Software Protection and Security Item

The separation of foreign software protection and resource protection is basic principle of the L-MSP system. An attempt to give a meaning of it in security aspect is described here.

Security item is categorized into three items, Confidentiality, Integrity, and Availability. On the target system, foreign software is assured only confidentiality and integrity. The availability is left to the system user.

This seems to be insufficient from the viewpoint of foreign software security. But we have to remember that the management right is owned by target system user. The user have absolute right of system resource allocation.

If foreign software were assured resource allocation and the software were a virus, then there is no way to stop the virus for the user. Even if the software is not a virus, unintentional software fault might cause resource consumption. This may lead the system down. So, software vendor generally will not have responsibility for that.

If software vendor wants to assure the execution, obligation of returning the process result is a possible way of it. In L-MSP system, only confidentiality and integrity of foreign software are assured by the processor. The availability is left to system user, as listed below.

$$\begin{array}{l}\left.\begin{array}{c}\textbf{Confidentiality}\\ \textbf{Integrity}\end{array}\right\}\ \text{Foreign software}\\ \left.\begin{array}{c}\textbf{Availability}\end{array}\right\}\ \text{System user}\end{array}$$

5 Conclusions

L-MSP enables both foreign software protection and resource protection in an open source operating system. Software protection offers more secure ways of distributed computing without trusting target system users.

The significance of software protection becomes clearer using remote computation model. Orthogonalization of software protection and resource protection assures user privacy protection using ordinary security sandbox. In this sense, L-MSP behaves like trusted third party on the target system.

We have examined our design using functional testbed, and estimated hardware size and performance overhead. Both are acceptable for those processors

[2] In this discussion, foreign software is assumed to be secure, that means it has no security hole revealing its own secret.

from smaller appliances to high-end equipment. At this moment, our major concern is how to put the real product of L-MSP into the market.

Acknowledgements

We would like to thank Takashi Miyamori and Takanori Tamai of Toshiba Soc Research & Development Center for helpful discussion and technical assistance. We also thank Frank Stajano for informing us of this workshop.

References

1. V. Welch et al.: *Security for Grid Services*, Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press (2003)
2. W. A. Jansen: *Countermeasures for Mobile Agent Security*, Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press (2003)
3. M. Kuhn: *The TrustNo1 Cryptoprocessor Concept*, CS555 Report, Purdue University (1997)
4. D. Lie, C. Thekkath, P. Lincoln, M. Mitchell, D. Boneh, J. Mitchell, and M. Horowitz: *Architectural Support for Copy and Tamper Resistant Software*, In Proceedings of AASPLOS IX (2000)
5. M. G. Arnold and M. D. Winkel: *Computer systems to inhibit unauthorized copying, unauthorized usage, and automated cracking of protected software*, U.S. Patent Number 4,558,176 (1985)
6. A. M. Eskicioglu: *Protecting Intellectual Property in Digital Multimedia Networks*, IEEE Computer, Vol. 36(7), pp. 39–45 (2003)
7. *Media embedded Processor*, `http://www.mepcore.com/`
8. H. Takada: *TOPPERS Project*, `http://www.ertl.jp/~hiro/hiro-e.html`
9. R. J. Anderson: *Trusted Computing Frequently Asked Questions*, `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`

Secure Processor Consistent with Both Foreign Software Protection and User Privacy Protection (Transcript of Discussion)

Mikio Hashimoto

Toshiba Corporation, Japan

My talk is about secure systems, one particular secure processor, and the relationship between secure processor and privacy protection. We have implemented a secure processor with features consistent with an open operating system and with software protection, where software protection means protection against reverse engineering, I think all kinds of reverse engineering. The second topic is the relationship of secure processor and privacy: I introduce the remote processing model of software protection, and examine the relationship between software protection and the security sandbox which protects user privacy. Finally I will introduce some examples of software protection for law enforcement between anonymous users.

The type of software protection is general-purpose, for example, content protection, GRID computing, and software component protection, and in this sense the adversary is the end user or local system administrator. I have to mention the Trust No. 1 processor proposed by Markus Kuhn[1] which is pioneer work in this area. The protection target is application program or our processes, and these are the two assumptions: operating system is not trusted and is not trustworthy, and even may be hostile, and only processor hardware is trustworthy. We want an open source operating system which has virtual memory support.

One mechanism is standard memory inspection using virtual memory register values at process switching. We assume that the processor chip is trustworthy but some internal resources can't be inspected by the operating system, for example, operating system can't access cache memory. We assume this processor has capability so every user can make signed protected programs. And he can use the protected programs to corrupt other programs.

So many kinds of attack are possible. To produce some counter measures to these threats, we have various mechanisms in this processor. First we put enciphering hardware on the bus interface into the standard memory. When some interrupt occurs, the register content is loaded into standard memory with some encryption. As for cache memory, the access control mechanism is added, and the access control mechanism depends on some process state management action: the operating system issues some special instruction to manage processes, for example, the encryption of Keynote. In Keynote, the program is supplied by encrypted program keys, and the processor had a secret key for public key

[1] M. Kuhn, *The TrustNo1 Cryptoprocessor Concept*, CS555 Report, Purdue University (1997).

encryption, and the program key is encrypted by the corresponding public key. The program key is placed into key table, and that key is used when that program is executed, and each process is managed by a corresponding process ID. We have developed testbed implementation carrying such capabilities data, and we have implemented the processor on an FPGA evaluation board, and the operating system is modified mITRON O.S. So operating system source code can be disclosed to anyone, and everyone can modify the operating system source code, while maintaining the process protection.

Let me move to my second topic: the remote processing model of software protection. I use a content protection example in this discussion, but this model can be applied to other areas of software protection, for example GRID computing or network agents. A content provider supplies encrypted content, and that content encryption key is disclosed to software vendor (and it's trusted then to be clean), and software vendor sends protected program to target system. The program includes a content decryption key and the program is executed by secure processor. This scheme can be interpreted as a kind of remote processing, but in ordinary remote processing the processing result is returned to processing requester, whereas for content protection the side effect of the processing is supplied to the user. In this case the side-effect is output such as video stream, or music, and so on, and the system user corresponds to considerate adversary in this model, and they try to modify the program using operating system privilege, but secure processor protects the protected process from such attacks. We can interpret the role of operating system, and memory, as corresponding to an insecure communication path between software vendor and secure processor, while secure processor protects these programs. But secure processor obeys system-user direction about resource management, so secure processor corresponds to trusted third party between software vendor and system user.

Now then, how the user privacy is dealt with in that system. Usually user privacy is protected by setting the sandbox for some untrusted standard software, and usually, the protection is done by some resource allocation mechanism which is supported by the operating system, and conventional processor, and a system user supplies the security policy into the security sandbox, then user privacy is protected from external software.

Bogdan Popescu: How do you protect the privacy, how do you make sure there are no side channels?

Reply: Secret means secrets stored in the file or some memory. Some process maybe has access to something private, but if this process is put under the resource control mechanism the processes cannot have access to the user's privacy.

Bogdan Popescu: You can always report the user behaviour with regard to that process. You cannot create other files that have nothing to do with that protecting process file, but you can always try and see what the user has done with respect to the process of the application. That's something you cannot protect, or can you?

Reply: In this case I am not trying to discuss the behaviour of the protected process set. While user privacy is protected from external programs, and the processes are protected by a secure processor from attacking using operating system privileges, in this case only confidentiality and integrity are assured by secure processor. Availability is not assured by the processor because availability has a close relationship to resource management.

I now continue to discuss the secure security sandbox. Security sandbox is crucial to user privacy. Open source security sandbox is suitable for protecting user privacy because it is verifiable by users themselves, and as for processor requirements, a separation between software protection and resource management is important. This means software protection and resource management never violate each other. If those two are mixed, they may introduce some security holes from both sides.

I'd like to extend this protection into networking. A protected program may be used to do authentication, for example. Here is a content protection example: A content provider wants to distribute their content only to protected programs, and the protected program has an encryption secret, for the corresponding content, so an honest end-user can use the content, but a pirate cannot make an unauthorised copy of the content. But of course the content provider authenticates protected programs, so processes which have no secrets cannot get that content. In this case no user attributes are supplied to the content provider, the content provider only wants to know if the target process has an authentication secret, and we can extend this case into generic information exchange mediated by protected programs. In this case protection from unauthorised copying is enforced by a protected program, so information exchange would be wider used. This situation makes involuntary exchange of vital information I suppose.

I make a brief summary: We have implemented a secure processor for software protection, the key feature is process protection under an open source operating system. I introduced the remote computation model for software protection. In that model, the secure processor plays the role of trusted third-party. I presented the creation of user privacy and software protection: an open source security sandbox protects the user privacy. And I discussed what software protection brings to the user: information exchange, and peer groups enforced by protected programs while making anonymity possible.

I think this feature brings some benefits to users especially for verification.

Bogdan Popescu: Did you say you have implementation of this? How much slower is your protected process than a normal one, what is the overhead?

Reply: About 100 nanoseconds, for one encrypt operation.

Bogdan Popescu: Oh encrypt, yes, but how much to put this whole thing within the architecture, I mean, the context switch for example?

Reply: You can measure that the performance is half of the current system. But the fact is, if the cache hit ratio figure is 100 percent, there no overhead, and for the usual system this figure is 90%, so…

Bogdan Popescu: If it's in the cache you don't have to encrypt anything?

Mike Roe: Because you only decrypt once you've loaded the cache, so if you're in a program that remains in cache then you're only ever using the decrypted version in cache inside the processor. So you never have that added overhead of external memory.

Virgil Gligor: 100 nanoseconds is for encryption of data in the cache?

Reply: Yes. If the cache ratio is 99%, about 10% of performance degradation. On the hardware side, about 200 kilogate, and if we make such a processor in current technology 1.5 mm^2 is required.

Audience: Do you expect this design to go into production?

Reply: I can't judge the corporate opinion, but I'd like to make some product for the market.

Why Are We Authenticating (Transcript of Discussion)

Mark Lomas

Arabella (Arab Bank Group)
spw2004@absent-minded.com

I think it's important to understand the purpose behind verification before you actually try and design an authentication protocol, because if you don't know what you're trying to achieve there's not really much point.

This is something I worked on a while ago, and I'd like comments on it, but I was reminded of it by a comment that Fabio made yesterday[1]. He referred to the purpose for which we provide data. I would suggest that we should also think about the purpose for which we run an application: unless we understand that I don't think we can meaningfully reason about the authentication mechanisms we're going to be using. I'm going to give a really terrible example, Active X, because it's easy to find fault with it.

I'm going to take it for granted that people know what Active X is, if that's not true I'm happy to describe it. So how do we handle an unseen control?

James Malcolm: What do you mean by an unseen control, just one that hasn't got any visible representation on the screen?

Reply: No, I mean has not previously been seen. You've just downloaded something that hasn't previously run on a computer, or if it has run on your computer, your computer's forgotten about it for some reason.

There's an elaborate authentication mechanism but it doesn't actually seem to serve much purpose; we carefully authenticate the user, who might use a password or who might use a smartcard, in order to check that we've got somebody who's allowed to authorise the Active X control to run. We also authenticate the control itself, and there's this elaborate chaining mechanism of key certificates which essentially ends up with a digital signature for the control, that the computer can use to verify, and if it's happy with that, it will ask the user if it can run it. Unfortunately approval, by which I mean the user approving the running of that control, is an all or nothing event. You get the dialup box that says, "there is new Active X control, do you want to run it?" In fact there's a little extra check box which nobody in their right minds would tick, which is to trust all subsequent content from the same service provider.

But what does this approval mean? This goes back to Fabio's comment. Approval essentially says, this control can do absolutely anything it wants to because it's got full permissions for the user, and the user has no opportunity to negotiate with the service provider to say, well I'm happy for it to process this

[1] Fabio Massacci, these proceedings.

type of data, but I'm not happy if it can overwrite my hard disc. There's no real negotiation, it's all or nothing.

I suggest that when we're considering what a control should be doing, in other words what I'm approving it to do, what we're really concerned about is the side effects that that control has. If I run a process on my computer and it has absolutely no side effects then it's very hard for me to say it's misbehaving. It might degrade the performance of my computer, but it's not going to do anything really serious to damage it. So what I would like to say is, misbehaviour is doing something that I didn't permit, whereas essentially the way Microsoft validates a control, it means doing something they didn't expect; and it's very hard to fault somebody through a contract that says, you did something I didn't expect; it's a lot easier to say, you did something I didn't permit.

Frank Stajano: You say in practice it currently means doing something I didn't expect?

Reply: In practice on the grounds that Microsoft have given me no option but to do it that way. I'm not saying it should be that way.

Mike Roe: The only contract that it allows you to agree to is to give the server permission to do absolutely anything they like to your computer.

Frank Stajano: But the difference between the two simply depends on how pessimistic you are because if you think this is going to screw things up, then you were expecting it... [Laughter]

Pasi Eronen: The problem is that there is also this gulf between what the user thought he was permitting, and what the software, or the operating system, interpreted was permitted; these are probably not same.

Richard Clayton: I was going to raise that point by drawing your attention to something that the mass mailing worms use, which is to bring up an end user licence, that you can click on and say, I give you permission to send out lots of unsolicited email from my machine. They argued, and the anti-virus vendors for a while accepted the argument that, since the user had given permission, this was OK. Actually the vendors eventually decided to block those sort of things, as a form of mal-ware, because users don't expect there to be such a term buried in the middle of an end user licence agreement.

Reply: This is where I hope some people will disagree with me but I'm going to consider side effects as being what I would call destruction, which is either deleting or corrupting data, leakage of information, or denial of service. I'd invite comments as to whether I've missed anything there, because they're very broad categories.

Richard Clayton: I can blow up your monitor if I want to. You also missed addition of data, because if it puts an illegal image of an underage child on your machine, you'd be very upset about that, but he hasn't actually corrupted any of your data.

Reply: In a sense I would say that was corruption of data, but I think you're probably right, it should be in a category.

I'm really not going to look at the third of these categories but I'll be happy to discuss it afterwards. In what I'm going to propose, I'm going to look at the first two. This is not at all a new idea, I'm essentially going to tell you about a sandbox, but I think it is actually worth trying again. There have been various attempts but they don't seem to have been very successful so far. I'm going to suggest a very minor variant. I would like to order possible side effects into a hierarchy; from what I would call benign side effects up to serious side effects. Again I will be very broad in my categories, because I think one of the failures people had in previous sandboxes is to have such fine grain control that the user didn't understand what he was actually permitting. As an explicit design role, I wanted to minimise the number of queries that the user actually gets.

The sort of things I consider to be benign are what I'd call a constrained user interface, something like: you can open a window to interact with the user, but you're not allow to interfere with other things that are running on the computer. The user is always in control, if he closes that window that interface is gone, the application can't force it to remain in there.

Slightly more controversial, what I call damage to self, is if an application decides to corrupt itself. I'm regarding that as being a benign operation; you've sent me something which changed itself, you might as well have sent me the changed version, so as far I'm concerned you haven't actually caused damage. There are circumstances under which that is a wrong assumption, but for general processing applications I think that's reasonable.

The other one which may be slightly controversial is persistent storage, I don't mind if the application chooses to write things to my hard disc provided it's not damaging information that I've stored; I know that's in contradiction to Richard's earlier comment.

Here is where it becomes a little more interesting, what I call reasonable activity. If an application is constrained to only the things I put down as benign, you wouldn't be able to do anything useful with it. Essentially it could display a picture for you, it could store some data on a disc, maybe do some fancy slideshow, but it can't actually process any of your existing data. I'm going to be slightly provocative by referring to damage, because what I mean by that is, giving write access, or delete access, which may or may not actually cause damage. I'm putting it in that way to emphasise what a misbehaving application could do with the privileges. In practice it's saying write access. If I'd explicitly said, look this application can overwrite that particular file, then there's a definite contract between me and the service provider. If it does it I can't claim that I didn't permit it. I might also permit damage to certain classes of file, say, this application is intended to modify Excel spreadsheets; I might want to have a slightly more complicated way of describing it rather than say, this can overwrite any Excel spreadsheet, I might want to constrain it by saying, it can overwrite any Excel spreadsheet within this particular subdirectory. And for reading specified files, it might be that I've given explicit permission for you to read a particular file, or permission for reading classes of files, again I would describe it in the same way as giving write access.

If we're not concerned with privacy we might treat the reading operation as being benign, which simplifies the user interface, but I would say that should be a configuration option, as some people are concerned with privacy.

Vaclav Matyas: How do we categorise security related content for some files? I may not be concerned with privacy, but I might be concerned with security related information.

Reply: Again I'm being quite broad with the terms. By privacy I'm saying anything you regard as sensitive, e.g. passwords.

Vaclav Matyas: Well maybe not sensitive to the user, but to the machine.

Reply: What would be sensitive to the machine and that the user doesn't care about? You might not be aware of it, but I think if you were you could explain to the user.

Vaclav Matyas: Setting up the period or the place where I download my updates for an IDS or Virus Checker.

Reply: In which case you probably do want to say that's a reasonable and sensible thing to explicitly permit.

And finally I'm going to say serious side effects. This is actually where I dislike Active X; unconstrained damage is what I would call giving blanket write access, which is essentially what Active X does at the moment, and configuration changes, which are permanent changes to the environment like, you know, mucking up my screen settings, which it really is not the business of a transient Active X control to be mucking around with.

Ross Anderson: But that's what people want to do in Active X controls, they want to make your browsers full screen size so they can sell you dancing pigs!

Reply: Well if that's a requirement then you'll have to move that down into the reasonable and have it as an explicit permission. It would complicate the user interface, but I understand that some people might want it.

Ross Anderson: But if we go down that route whereby an application can modify its files but not modify other people's files, then you end up in essence doing a somewhat lower-grade implementation of Trusted Computing with the implications for lock in, etc.

Reply: I'm not saying that this is perfect; really I'm trying to see whether I can do something useful with it. There have been previous attempts at this: Java, when it was released, came out with a sandbox that had quite a complicated set of access permissions. The problem was that nobody actually understood them.

George Danezis: The Java sandbox also considered the network as a resource, and by default only allowed you to talk to the originating machine.

Reply: I haven't done that because that's one of the things that people have been confused about, trying to configure Java systems. They never actually understand them, so I can see why it was designed in, but on the other hand it is very difficult to explain to end users. So recent implementations of Java tend to be implemented with all or nothing controls: if you want to run this Java applet, then it gets all the permissions there are. I suggest that a simpler user interface might change things.

Frank Stajano: Is it really your assertion that the Java sandbox was replaced by all or nothing because it was too complicated to explain? I believe that their sandboxes were replaced by all or nothing because people like Microsoft like to give the all. For example, the Windows update feature depends on your Active X control being able to completely replace components of your operating system, and of course you couldn't do that with a sandbox, that's why everything became all or nothing.

Reply: The sandbox was sufficiently flexible to allow you to define your own machine.

Frank Stajano: If it allows you to re-write the operating system, then it's fairly new standards.

Reply: Well it depends. Assuming that we had the original version of Java within the access permissions, and I knew that one of the applications was an update feature which is explicitly designed for changing system files, then I can make a risk assessment as to whether I'm going to permit that knowing that the other applets don't have that permission. So it was sufficiently flexible to do what they want, which is why I think it was more that in order to run those sorts of application you had to tell a user, you can't just tick OK, you have to open up this complicated dialup box, go down, click on the system permissions to this applet.

So I'm suggesting that the user interface should essentially hide all what I'd previously called serious side-effects, rather than confuse the user by telling them. Hide the user interfaces, it may have a system configuration option which the administrator can turn on, but it shouldn't be something the end user gets to see. I'm going to suggest that all the things I classified as benign be subject to all or nothing, so essentially if you permit the applet to run, you cannot turn off permission to run options. And so my user interface is going to concentrate on the reasonable side effects.

My suggestion is that the user interface will only show the reasonable side effects, so which of these do you wish to permit, and then I would hope the user would have an idea of what the application is able to do.

Mike Roe: The yes and no that we currently have is already too complicated, given that as when you're browsing you can get hundreds or thousands of these an hour. The user suffers fatigue in understanding what they are being asked. Asking the user whether they want to permit this random website to do some random thing to their machine just seems to be the wrong model.

Ross Anderson: Which of the obvious things do you impose on the marketplace, do you compel marketers to pay three cents to Microsoft, with two of these cents go to the user every time you do a pop up window. Now that might align corporate incentives, and privacy incentives.

Pasi Eronen: I very much agree with that, and even the yes/no boxes are too much. Go to some website and get this yes/no box, whatever the warning reads, the user interprets this as, do you want to go to this website, yes or no, and if he wanted to go there he always clicks yes, regardless of what the dialogue box says.

Bruce Christianson: Do you wish to continue or to abort?

Tuomas Aura: It's not just that. Any user interface, or any access control system, that would make it safe for a user to go on to an arbitrary website and download software from there and execute that on his machine, with some level of access to the file system.

Richard Clayton: The problem with yes/no is the questions that they ask. I use a system that asks yes or no all the time, because I've got a little personal firewall on my machine not so much to protect me, but it's amusing to watch. It pops a question on a regular basis, it names some implausible part of the operating system, and it asks whether or not I wish to allow this access to the network. I have no idea what the question means. I assign the value of yes or no on the basis of whether or not it looked vaguely plausible, which means that if you manage to introduce something onto my machine which says, I am the printer spool widget accessor, please give me permission to access the Internet, I'll probably decide it is a good idea. So that's where the problem is, it's not in the question or permissions, it's in who is asking.

David Wheeler: Do you envisage a restore mechanism, or commit mechanism, so that if you have an undesirable thing you can ban it?

Reply: I think that would be a very good idea, but it complicates the user interface yet more.

David Wheeler: It complicates the programming. I don't think it complicates the user interface.

Fabio Massacci: Why don't we formulate the question to the user in the name of an access commander application? Say, "if you allow these things, this other application like Excel may have problems sometimes," and so on, rather than files, because files are a sense of issue concept.

Reply: Well actually that would be sensible because a lot of applications have a large class of files that users don't understand.

Frank Stajano: Speaking from experience and as someone who usually says no on the basis of general paranoia, the thing to consider is that if you offer this interface it should be clear to the developers that people could answer either yes or no. Usually when you do answer 'no', which is not what the developers experience when they test the system, nothing works. People say, please visit this website, but you can't even get in because the front page is based on Macromedia Shockwave or something. I don't even see the thing that gives you the menu options to get past that; I don't even see it if I open the source, because I can only accept after the thing shows and unless I accept I don't see it.

Tuomas Aura: How much did you lose by not seeing this website?

Frank Stajano: I don't know. [Laughter]

Bogdan Popescu: Some people have some sort of garbage collection. They basically accept more or less everything within certain limits and then occasionally just get rid of all this stuff, and start over.

Reply: You can only do that if either you've got the sort of unwinding mechanism David was suggesting, or you've got rigid segregation in which the side-effects don't make a difference to requirements.

Bogdan Popescu: Or have some specific account where you only do this more dangerous stuff – like sandboxing but on a higher level, not just at the application level. An all-environment solution.

Frank Stajano: That's like having another hat[2].

Chris Mitchell: It does seem that most of time no sensible person would want to run these things, but just occasionally they're extremely useful, like Windows update. I don't have any realistic hope of this, but it might be nice to persuade Microsoft to word the box rather differently: say, "this is something you wouldn't normally want to do, but just occasionally it might be very useful." [Laughter] A kind of user education, this sort of thing is really not something you can do lightly. At the moment there is no sense that this is a dangerous thing, the question is asked in a very neutral way, and I would suggest that that in itself is dangerous.

Matt Blaze: I think we're casual about this side-effect, I mean, essentially we've encouraged people into the habit of doing frankly dangerous things for very small amounts of value, and what we're trying to do is make those things less dangerous rather than figuring out other ways of achieving that small amount of value. I think we're trying to solve the hardest version of this problem, when in fact there are much simpler versions that are considerably more practical. Essentially there is a model that says, anybody who is providing you with any kind of service whatsoever owns your entire computing environment for the purpose of providing that service to you. This seems insane when most of the time that service is simply manipulating a few bits in a window, and receiving some mouse clicks from you. I think really we should develop a less intrusive model for providing these useful, but small, services rather than try to solve the much harder problem of how to allow arbitrary people to run arbitrary code on my machine without actually having that code be arbitrarily powerful.

[2] See Frank Stajano, these proceedings.

Reply: What you've just described is what I call benign operations. I'm perfectly happy to have the website mucked around with inside a constrained window, that doesn't bother me. It's the durable write access to absolutely everything in my file system that bothers me.

Matt Blaze: But I think most of this is just display and simple interaction. Because it is possible to do things that make the keyboard vibrate, and display dancing frogs on the screen, providers of services do that, and now you've no choice but to turn it on. We could provide a narrower interface that meets the needs, and I think that would take a lot of these other problems all away.

Bruce Christianson: Text-only frogs[3]. [Laughter]

Virgil Gligor: Also one might add something about sandboxing and being able to put controls on other people's programs and unallowed machines. The biggest problem with that is that we tend to put too stringent controls on these programs. In other words, there is an incompatibility between what a program needs and what we think it should do. So no matter how much explanation the writer of that program is going to give us about the effects of the program on our machine, we always need the last recourse, namely pressing the red button and changing our mind, and undoing what was done so far. In other words, if I click the wrong one out of these choices, I'll always need a recourse, so I agree with David that that's a fundamentally necessary feature, and not easy to provide.

[3] See LNCS 3364, p350.

Anonymous Authentication

Partha Das Chowdhury, Bruce Christianson, and James Malcolm

Computer Science Department, University of Hertfordshire, UK
{P.Das-Chowdhury, B.Christianson, J.A.Malcolm}@herts.ac.uk

Abstract. The contribution of this paper is a mechanism which links authentication to audit using weak identities and takes identity out of the trust management envelope. Although our protocol supports weaker versions of anonymity it is still useful even if anonymity is not required, due to the ability to reduce trust assumptions which it provides. We illustrate the protocol with an example of authorization in a role based access mechanism.

1 Introduction

Authentication, authorisation and audit are three traditional concerns in building a privilege management infrastructure (PMI); the purpose of authentication is to identify a particular user and verify that a user is who he/she is claiming to be; the goal of authorisation is to provide access for certain users to certain resources based on predefined business rules; and an audit trail links actions to principals retrospectively. Traditionally, authentication is based on permanent credentials linked to a fixed long-term identity and authorisation is linked to audit via the authentication mechanism explicitly using the same permanent credential and identity. Privacy is not an explicit goal of traditional authentication/authorisation mechanisms [11,10,1]. This we believe is not quite right even if privacy is not a requirement; such approaches compel users to enter into unneccesary trust relationship with parts of the system infrastructure. For example services must trust Kerberos' judgement about the identity of the requestor.

In this paper we present an authorisation mechanism which takes identity out of the trust management envelope. Although our protocol supports weaker versions of anonymity, it is useful even if anonymity is not required at all, because of the ability to weaken trust assumptions; the approach we present here allows users to control the risks they are exposed to rather than forcing them to enter into an unnecessary compulsive trust relationship with the system infrastructure. Similarly, services also do not need to trust the authentication mechanism of a third party.

Role based access control (RBAC) is one mechanism for building a PMI. The main idea behind RBAC is that the permissions are associated with roles instead of directly with users. The usual approach for role activation is authentication effected by role members using fixed credentials like a key certificate or a role certificate. It is well known that repeated use of fixed credentials enables an adversary to correlate the transactions of any particular user of a system. The problems of using fixed credentials have been well understood in the world of commercial transactions, and this has led to the advent of several anonymous payment mechanisms. We focus on privilege management infrastructures where key certificates are extensively used for making authorisation decisions. The certificates when used repeatedly also enable an adversary to correlate all the access requests of its victims. Our intention is to show some alternative ways to link authentication, Role Based Access Control, and Trust Management using transient identities (surrogates) in an auditable but unlinkable way. What we are suggesting here is creating some conceptually neat layers in trust management approaches using explicit labels and policies. Our approach can coexist with other traditional non-anonymous approaches.

This paper is organized as follows: Section 2 gives an outline of what we mean by transient identities. In section 3 we present a brief overview of trust management systems, which is followed by our design goals and assumptions in section 4. Section 5 sets out the protocol for anonymous authentication which is the main contribution of this paper, followed in section 6 by conclusions.

2 Surrogates

Surrogates [7] are transient numbers generated from a parent number (such as a bank account number, credit card number, or social security number) by some mathematical function. They are used to preserve the anonymity of the legitimate owner of the parent number. The party who accepts the surrogates in exchange of goods or services bears the risk of fake surrogates [4]. There is a difference between blinded credentials and the way we generate surrogates here. Blinded credentials need to be certified by some authority [5,14], and can be reused [5], or else the user needs to get a new credential issued after every single transaction [14]. Surrogates are for single use, doesn't need to be certified and we use and throw them away after every single transaction. Blinded credentials can be verified by the acceptor if the acceptor knows the public key of the issuer. Surrogates are verified by proving knowledge of a secret that was used to generate the surrogate without revealing the secret.

The surrogates in our protocol are generated by the user, and although the issuer can verify them the issuer cannot masquerade as the user. The surrogates for a particular user are generated from the parent public key of that user. Each surrogate has a corresponding secret (ie private) key. These secret values are generated from the secret key corresponding to the parent public key, and the public surrogate together with its corresponding secret value forms the surrogate pair. The surrogate pair can only be used by the legitimate owner of the corresponding parent public key. The requirements for surrogates can be summarised as:

- Un-correlatability - Different transactions initiated by a particular principal cannot be linked to each other or back to the initiator of the transaction.
- Unforgeability - It should be hard for an adversary to generate and use credentials belonging to another user.

– Verifiability - The verifier who might be the owner of the resource or an access granting service should be able to able to unambiguously verify that the requestor is who he/she is claiming to be.

2.1 Generation and Use of Surrogates

Our key generation method is similar to the traditional one described by Diffie and Hellman in [9]. Cathy selects a secret $s_c \in 1 \ldots (P-1)$ and generates her public key as

$$X = g^{s_c} \bmod P \tag{1}$$

The surrogates are generated by modular exponentiation of X using an exponent τ where $\tau \in 1 \ldots (P-1)$. The secret value corresponding to a surrogate is generated by modular multiplication of the exponent τ with the secret s that was used to generate X. The initial value (τ_0) of the exponent τ is supplied by a (partially) trusted third party such as the user's bank. The subsequent values (τ_i) of the exponent τ, used to generate the surrogates and their corresponding secret, are generated by the user using the linear congruence equation,

$$\tau_i = (A * \tau_{i-1} + O) \bmod P \tag{2}$$

where A and O are selected by a third party and are fixed. Then, using τ_i, surrogate K_i^+ and its corresponding secret K_i^- for the i^{th} transaction are generated as:

$$K_i^- = \tau_i * s_c \bmod (P-1) \tag{3}$$

$$K_i^+ = g^{K_i^-} \bmod P = X^{\tau_i} \bmod P \tag{4}$$

To generate the surrogates corresponding to a user, first the exponent is calculated by equation 2 and then the secret is calculated by equation 3, finally the surrogate is generated from the secret by equation 4. The third party sends the user A, O, and the initial value τ_0 of the exponent so that the user can generate his/her surrogates.

To use a surrogate K_i^+ for transaction i a user proves knowledge of the corresponding K_i^-. Only the legitimate owner of X and s_c can both generate and use surrogates corresponding to X. The third party cannot masquerade as the legitimate owner of X as the corresponding s_c is secret but can resolve disputes and can correlate transactions conducted with surrogates generated from X, thus facilitating auditing. However various transactions conducted by the same user cannot be correlated with each other by an adversary at the point of use of the surrogates. Learning one set of K_i^+, K_i^- values does not help the attacker generate future K_i^+, K_i^- values nor can the attacker get the secret value s_c.

3 Trust Management Systems

Trust management systems help applications answer the question "Can I trust this public key for this purpose?" Keynote [2] is the latest version of a set of trust management approaches that came from Matt Blaze, Joan Feigenbaum and John Ioannidis. Keynote works more or less like a database query engine. It can function as a stand alone service interfacing with other parts of the system and helping them in making decisions. Let's lump these other parts of the system together and call them by a common name application. Whenever any application faces the question "Should we carry out this dangerous action" then it refers to Keynote for an opinion and based on that opinion it decides its future course of action. The application presents the Keynote trust management engine with a set of local policies that should be taken into account while taking a decision on this particular request along with the credentials of the requestor and details about the proposed action. If the proposed action conforms to the local policy then keynote advises the requestor to proceed otherwise Keynote advises it not to perform this action as it is against the local policy. Keynote acts as a compliance checker for the application. The policies are specified in the form of assertions and the actions are specified which are evaluated against these assertions.

4 Design Goals and Assumptions

4.1 Design Goals

1. Cathy is a subscriber of an electronic newspaper. The newspaper uses a role based authorisation model where Cathy is assigned the role of subscriber. The process is same for every subscriber.
2. The subscription department (SA) prepares and sends the information Cathy needs to generate her surrogates. At regular intervals the subscription department also sends the web server (WS) the information needed to authenticate Cathy and other subscribers.
3. Every time Cathy reads the newspaper she first authenticates herself to the web server (WS) using her current surrogate and upon a successful authentication her role is activated.
4. Cathy generates her own surrogate for the current session.
5. The web server authenticates locally whether or not Cathy is a valid subscriber. The role server consults the local trust management engine to ascertain whether or not Cathy is allowed to carry out the actions she requested.
6. The next time Cathy reads this newspaper she uses a different surrogate.
7. The SA cannot masquerade as Cathy.

4.2 Cryptographic and Infrastructural Assumptions

We assume the existence of a secure authenticated communication channel between Cathy and the subscription department of the electronic newspaper. This can be implemented by digital signatures where every communication between Cathy and SA is digitally signed. This provides authentication. The communication link can be secured by for example SSL/TLS. These communications between Cathy and the SA are denoted by $\longrightarrow$ in the protocol.

An anonymous communication channel between Cathy and WS is also assumed for our purposes. This can be implemented by Chaum's mix nets [6] or Mixminion [8]. The mix network makes it harder for an adversary observing the network to gain any additional information about the communicating partners beyond its a priori belief. Communications over the mix nets are denoted by $\longrightarrow_m$ in our protocol.

Our protocol depends on the difficulty of computing discrete logarithms in the multiplicative group Z_P^* where P is a large prime. P should be chosen such that (P-1) has at least one large prime factor, since if (P-1) has only small prime factors then computing discrete logarithm is easy [12]. We also assume that principals are capable of keeping their secrets secret.

5 The Protocol

The protocol can be described under two phases namely

1. Preparation phase
2. Transaction phase

5.1 Preparation Phase

Cathy sends the subscription department (SA) her public key.

1. $Cathy \longrightarrow SA : X = g^{s_c} \pmod{P}$

where g and P are public and g is the generator modulo P. SA sends Cathy the information she needs to generate her surrogates which is :

2. $SA \longrightarrow Cathy : \delta = \prec \tau_0, O_\tau, A \succ$

SA selects $\tau_0, A_\sigma, O_\tau \in Z_P^*$. We use the Linear Congruential Method to generate exponent τ_i where the offset O_τ and the modulus P are co-prime to each other and A is the constant used in the linear congruence method. τ_i is the exponent used for generating the surrogate. For each subscriber SA sends the web server (WS) m surrogates by repeating the following for each of the next m values of i:

$$\tau_i = (A * \tau_{i-1} + O_\tau) \bmod P - 1$$
$$X_i = X^{\tau_i} \bmod P$$

3. $SA \longrightarrow WS : \prec X_1, ..., X_m \succ$

In step 3, the surrogates of various customers are sent mixed up together in a batch. At the end of this phase SA only retains $\prec X, A, \tau_i, O_\tau \succ$ for each subscriber.

5.2 Transaction Phase

For transaction i Cathy calculates her i^{th} surrogate K_i^+ . The corresponding private exponent K_i^- of the current surrogate is also generated by Cathy, as described in section 2.1, equations (2)(3)(4).

To activate her role for her current transaction Cathy sends the web server her current surrogate K_i^+ and proves knowledge of the corresponding secret key K_i^-. If the web server finds a similar surrogate sent to it by the SA then it grants Cathy access. To be specific the WS activates relevant permissions for Cathy. To ascertain the validity of Cathy's request the WS consults the local trust management engine. An implementation incorporating trust management engines can be found in [13].

4. $Cathy \longrightarrow_m WS : K_i^+$

The next time Cathy reads the electronic newspaper she uses a different surrogate to authenticate herself. The role server can authenticate Cathy anonymously without the issuing authority being online. The issuing authority can periodically send Cathy's new surrogates to the web server. SA cannot masquerade as Cathy to the web server.

6 Conclusion

The surrogates can be verified locally by the web server without the cooperation of the issuing authority (SA). The various transactions of any particular user cannot be correlated with each other without the active cooperation of the issuing authority. For purposes of dispute resolution the issuing authority SA in our protocol can correlate the actions of the subscribers.

Our protocol supports weaker versions of anonymity, and it is still useful even if anonymity is not required at all, because of the ability to weaken trust assumptions. The web server where principals request access does not need to trust the authentication mechanism of a third party. From the point of individual privacy users are no longer required to entrust web servers with credentials linked to their long term fixed identity, using which an adversary could uniquely identify the user.

Our protocol can also be integrated with web servers which currently use trust management systems [2,3] to make decisions based on fixed policies and credentials. The trust management systems can be placed in the web server and make decisions based on a surrogate supplied by the requestor and role-specific policies specified by the appropriate authority. The trust policies can be communicated to the web server by the appropriate authority.

Our work also has implications in other areas like electronic payment, price discrimination, licensing enforcement. If Cathy avails herself of services using her surrogates and wishes to pay for the service then there should be ways of tying the surrogates to the payment for auditing purposes while preserving Cathy's anonymity. Similarly for price discrimination if a buyer can prove with his/her surrogate which price band he/she belongs to and the buyer is the legitimate owner of the surrogate, then price discrimination can be done with online transient identities, thus protecting offline identities.

References

1. Yolanta Beresnevichiene. A role and context based security model. Technical Report 558, University of Cambridge, 2003.
2. Matt Blaze, Joan Feigenbaum, J. Ioannides, and Angelos Keromytis. The Keynote Trust Management System. *Request For Comments Series*, (2704), 1999.
3. Matt Blaze, Joan Feigenbaum, and M. Strauss. Compliance Checking In The Policymaker Trust Management System. *Proceedings of the* 2nd *Conference on Financial Cryptography: Lecture Notes in Computer Science Series*, 1465:251–265, 1998.
4. Nicholas Bohm, Ian Brown, and Brian Gladman. Who Carries The Risk Of Fraud In Ecommerce. *Journal of Information Law and Technology*, 2000(3):173–199, 2000.
5. Jan Camenisch and Els Van Herreweghen. Design And Implementation Of The *idemix* Anonymous Credential System. *Proceedings of the* 9th *ACM conference on Computer and Communications Security*, pages 21–30, 2002.
6. David Chaum. Untraceable Electronic Mail, Return Addresses And Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
7. B. Crispo. *Delegation of Responsibility.* PhD thesis, University of Cambridge, 1999.
8. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design Of A type III Anonymous Remailer. *Proceedings of the* 24th *IEEE Symposium on Security and Privacy*, pages 2–15, 2003.
9. W. Diffie and M. Hellman. New Directions In Cryptography. *IEEE Transactions on Information Theory*, 22:472–492, 1976.
10. David Ferraiolo, Ravi Sandhu, Serban Gavrilla, Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard For Role Based Access Control. *ACM Transactions on Information and Systems Security*, 4(3):224–274.
11. B. Clifford Neuman and Theodore Tso's. Kerberos: An Authentication Service For Computer Networks. *IEEE Communications*, 32(9):33–38.
12. S. Pohlig and M. Hellman. An Improved Algorithm For Computing Logarithms And Its Cryptographic Significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.
13. Bogdan Popescu, Martin Van Steen, and Andrew Tanenbaum. A Security Architecture for Object Based Distributed Systems. *Proceedings of the* 18th *Annual Computer Security Applications Conference*, 2002.
14. Paul Syverson and David Goldshlag. Unlinkable Serial Transactions: Protocols And Applications. *ACM Transactions on Information and Systems Security*, 2(4):354–389, 2000.

Anonymous Authentication (Transcript of Discussion)

Bruce Christianson

University of Hertfordshire, UK

I'm going to start with the point which Dieter finished with yesterday,[1] that Kerberos was originally intended to have three heads. The first was supposed to be an authentication mechanism, next there was supposed to be an authorisation or access control stage, and then there was supposed to be accounting, auditing and that kind of thing. My perception is that we did the first one, and we were half-way through doing the second one when public key cryptography came along. Then we all disappeared down a rabbit hole for twenty years, and we've just emerged now. The effect of public key was that we went back and did authentication again, but we never re-did authorisation, or did audit at all. Traditionally the authentication that's associated with access control is a strong authentication, and it's linked directly to the authorisation mechanism. The link between access and audit is indirect, and they're linked via the identity that was used to do the authentication. There's something not quite right about this. Even if (partial) anonymity isn't a requirement, when you sit back and look at it dispassionately, it's not the right way to do it. The fact that this isn't quite right has some implications for system infrastructure that I want to raise.

The intention of this brisk ramble is to show some alternative ways to link authorisation (in this particular example a rather *derigiste*, very 1960's role-based access control mechanism; absolutely traditional, no surprises here) and a trust management system (think KeyNote here, again, deliberately very traditional) using surrogates (or transient identities, I don't want to get hung up on what word we use) in an uncorrelatable but auditable way. And the question is, what kind of infrastructure do you need to do that?

This is a talk about mechanisms. I'm repeatedly going to say: I don't care what your trust policy is, just tell me what it is and we'll enforce it. The particular example that I'm giving is based on linear congruences as the basic protocol mechanism. There are plenty of other mechanisms; ring signature is another good one, with slightly different properties. If you happen not to like linear congruences, don't worry, just use something else.

Here is a plain vanilla scenario, and the players are Cathy the naive but well meaning user, the authentication service which decides which roles people can act in, and the role server that, when you activate the role, decides what you're allowed to do, and which permissions you're allowed to have. It's the role server which has the trust management link. So the role server is responsible for implementing the trust management policy, but may also be granting tickets, managing files, or doing things like that. In the plain vanilla scenario we just

[1] Dieter Gollman, these proceedings.

lump all that together. We have a large prime of form 2q+1 where q is a prime, g is a generator, and these are publicly known. The bitstring s is Cathy's secret, and it's not publicly known, in fact it's not known to anyone except Cathy.

In the beginning the authorisation service is periodically on-line. At some point when it's on-line, Cathy interacts with it via a secure service with properties that we'll come back to later on. Essentially Cathy says, here's a kind of Diffe-Hellman public key thing, and the authorisation server says to Cathy, here's a thing that's a bit like a cheque book for surrogates. In fact, it's the linear congruence generator: an initial value, and the offset and the base for linear congruence generation, and that's Cathy's cheque book. From these values, Cathy can generate all her surrogates, and the authorisation service can generate all Cathy's surrogates as well[2]. The surrogates have this form: the i-th surrogate X_i is X to the power of the i-th exponent, where X is that public key thing, and the i-th exponent is generated from the previous exponent by the linear congruence.

At the end of this phase, in the simplest vanilla model where everything is driven by steam, the authorisation service simply communicates to the role service explicitly which surrogates are currently potentially valid, and the only requirement is that obviously they have to be communicated all mixed up, so that you don't get a big batch of Cathy's surrogates just after Cathy has been to the bank[3]. So that imposes some constraints on the public visibility of the kind of communication used, but on the other hand this is a kind of clearing process where at the end of each business day a massive amount of this information changes hands, and so it's easy to ensure it's all mixed up with every potential user.

Now, if I am the only person in the staff common room who reads the Guardian, no amount of 0s and 1s is going to help. Every time somebody picks up the Guardian, you know it's me. So we're assuming that all the services have the property that Cathy isn't the only person who uses them, so the surrogates are sufficiently mixed that it's impossible watching this traffic from outside to pick the ones that correspond to Cathy's cheque book. The authentication service retains the last cheque (which corresponds to the predecessor of the first cheque in the next cheque book), so if Cathy and the authorisation service have interacted once they don't need to interact again for a new supply of cheques to be sent, and there's no point at which all three parties need to be online together.

The trick is that we assume that Cathy has some form of anonymous communication – Mixnet, or one of these things – which allows her to communicate anonymously with the role server so she can activate the surrogate X_i essentially by displaying in some way knowledge of the corresponding discrete log. If Cathy can prove that she knows $log_g X_i$, then clearly this is a communication from the person who controls a surrogate that has the right to activate a particular role. The role server is then justified in activating that role, and they have proof that they are justified that they can put in the audit trail.

[2] Although the authorisation server can't use Cathy's surrogates; knowledge of s is needed to sign the cheques.

[3] See the discussion in Chris Mitchell's transcript, these proceedings.

In this simple case the role is pre-defined, but subject to the trust management policy. For example, the role manager might whiz along and say, this role now has this permission, or doesn't have that permission anymore. If you're bothered about revocation you can make the surrogate do a ring signature just before you allow access. That's quite a nice thing because you don't have to keep a list, you just have to keep a multiplication control. If someone can produce the corresponding signature then you know they're on the list, but you don't have any idea which one on the list they are, in fact you don't even have the list. The disadvantage of ring signature is that it really, really *is* anonymous which is why it's usually to better use it as part of a two level process.

Anyway, in some cases you do want actions to be correlated; if you're doing an atomic transaction which involves taking money out of one account, and putting it into another account, you want the two halves of the transaction to be correlated for when you do your reconciliation, so there you want different tickets which are related to the same surrogate to be linkable. In other cases actions done by Cathy shouldn't be linkable, and so you want actions using a different surrogate not to be linkable; that's OK too, even when the surrogates are acting in the same role.

Bruno Crispo: Here the role server audit trail tracks how roles are distributed or assigned, not how these roles are used.

Reply: I'm assuming at the moment that controlling how the roles are used is done in the traditional way: you have a role based access control system, you presumably have a policy that says how the roles are used.

Bruno Crispo: When you exercise the right role you are acting as a role. Does the role server correlate that or will it just assign the role?

Partha Das Chowdhury: When you activate a role you go to the role server, present your credentials and activate the role, but the rights that go with the roles are assigned in the conventional way.

Bruno Crispo: If I activate a role that I didn't have before, what does the role server do?

Reply: I go to the role server and say, I now want this surrogate to be able to start acting in this role. The role server might respond by saying, here is a permit for the car park, the tickets for the cinema, a voucher for some popcorn, and whatever else. Now you are back in the familiar world except that no-one in the cinema complex knows who you are.

Mike Roe: Is the access control also done at the role server, or at the authentication service?

Reply: I'm hedging my bets on that, in the vanilla scenario I'm just trying to copy Kerberos.

Mike Roe: The analogy there is that the role server is prepared to give you a ticket for anybody, but sends it in a role, and the provider you go to is in charge of working out whether or not that role is allowed access.

Reply: Yes, and then I have to go to somebody else and convince them that this ticket allows me in, and there might be separate outposts of the trust management policy that do that. There might be some sort of rapid response revocation mechanism that says, oh no, you're not coming in here. But the real reason I'm hedging my bets on that is that I don't really care. I don't want to nail my colours to any particular way of doing this, if you've got a favourite way of doing that then I want to support it too.

The authentication service can produce more surrogates at any point, cheques that correspond to Cathy's key, but it can't use them. It doesn't have the secrets that would allow it to use the surrogates to activate a role to get a ticket to do anything. The role server can keep an audit trail of which roles it's activated and which surrogates it's activated them in, but it can't fake entries in that audit trail because unless Cathy really does reveal the information that's needed to activate the role, the role server has no possible way of getting that confirmation.

Although the authentication server can fabricate surrogates, the role server can't. The authentication server has potentially the power to link the surrogates to Cathy, but the conditions under which it can do that depend not only on the policy but also on the details of that initial interaction that the authentication server had with Cathy. You can put protocol mechanisms in there that will limit the circumstances under which the surrogates can be linked back to Cathy.

What I've described is the very simplest, plain vanilla, just plug it in where you currently have your Kerberos-engine, type layer. More exciting flavours are possible. An access right could be the right to activate another role potentially with a different surrogate. So you can get into the situation where you're going repeatedly back and forward between the authentication mechanism and the role server, the ticket granting ticket mechanism, to use the capabilities you've now got in order to activate different roles under different aliases. The authentication service needn't be a monolithic service, it needn't be a single server, it needn't even be a single brand service, there can be lots of different varieties with very heavy fire breaks in between. By doing that you can also recursively address some of the problems about under what circumstances unlinkability can be broken. You can actually have a non first-order interaction between the trust management, the role based access control policy itself, and the authentication mechanism. The next logical thing to do is to carve off the binding of surrogates to roles, and say, that's a separate service, with a separate set of mechanisms to do that.

What are the implications of the fact that we can do things like this? Doing things this new way can co-exist perfectly well with doing things the conventional way, we're not saying everybody has to move to being anonymous. This means it's logical to have a look at some of the protocols that are currently used to support the infrastructure for access control. In particular things like electronic payment protocols: if you were using a service in this way, how might you pay for it? If you get a cheque from Cathy for services rendered then you've blown the anonymity, but equally you need to be able to tie a payment to the particular payee that the payment relates to via the surrogate somehow. So the desire to do this has implications for how you build up payment mechanisms, they would look a little bit different to the way they look at the moment. The point that I am trying to make is that (when you think about it for a bit) actually some of the changes you might make for these reasons would be sensible changes to make anyway. The only reason we haven't done it that way in the first place was because we got hung up on Kerberos when we started.

Now we consider protocols for pricing, and all this marvellous market segmentation stuff. Again, I'm not taking a view on how you should do differential pricing but, if you have a mechanism for doing differential pricing, then you need to know which price band the surrogate is in. You can't answer that question by saying Cathy is a student, you've got to have some other way of getting at that information that doesn't leak information about who is using the surrogate. Provided that for any particular role, or for any particular price band, there is a reasonable number of people who are potentially in it, there is no problem. If you have a particular price band that there's only one person in then clearly anonymity is a non-goal, unless that person is prepared to pay a different price to preserve their anonymity.

Licensing enforcement protocols for enforcing the conditions of licence use are a nice example of where the possibility of doing this kind of authentication does I think significantly change the game. We want to charge different people different prices for doing the same things. When we license a piece of software we charge users different prices depending on which thing they're doing, or how often they're doing it. Users very often don't want information about what they're using the software for to be sent back to the people who licensed them the software because then they'll get targeted marketing. This approach to authentication has the potential to keep everybody happy. It would appear that a lot of things could usefully be redesigned under these slightly different assumptions, and the purpose of this talk is really to suggest that it's worthwhile looking at the implications of exactly where you have the separation between the authentication mechanism and the authorisation of use that comes afterwards.

Matt Blaze: First of all, if I understand it, essentially what you're advocating in this design is creating a layer that would be precisely left out of trust management, which is provision for anonymity. When we designed, for example, KeyNote, we deliberately included all sorts of complex access control policies, but deliberately chose not to think about how you would do things anonymously. We didn't like explicit labels, and these have explicit policies for threshold security. Now what you're advocating is to have a different layer that essentially provides those labels anonymously, in some sense. That has the nice property that it separates these two things into conceptually neat layers, and that appeals to one side of me, but another side of me is horrified because now we have policy, not only specified by multiple places, but being evaluated by multiple layers of the system. We have no idea how those two things are going to interact, and what the implications of having them be out of sync with each other might be.

Reply: Yes, that's a good point. You still have a notion of identity, you still have the notion of users with identities acting in roles, but the identities are weak identities, not strong identities, and therefore your pre-conceptions about how much weight you can place on linking traceability through user identity has to change.

Matt Blaze: Right, but there might also be implications about the policies themselves.

Reply: Yes, certainly. It would be a useful exercise to work through some of these policies and see whether the assumption has simply been made because it's convenient to make, or whether in fact we could do without the assumption of strong identity most of the time. I think my argument is that for a lot of applications we can do without the assumption of strong identities, and in some cases doing without will actually make the application better, for example it would actually allow us to do certain types of licensing enforcement without being intrusive in the way that they currently are.

Matt Blaze: Yes, I agree. It would be a useful exercise to go through some real applications.

Fabio Massacci: Could you not achieve the same thing by using a credential translation service? It seems that you could just have this package of X.509 public keys and ask the authentication server to translate these and then have a key each side of my attribute certificate. Provided it's not my name in the second bunch of certificates, would it be any different, I mean, beside the mathematics.

Reply: There are lots of other ways of doing the mechanism. They all make slightly different assumptions about who trusts what, and for which purpose. In this particular case one of the key points is that Cathy doesn't want to trust anybody very much, including the authentication server.

What you're advocating is that there are just an awful lot of public keys out there and some of them are correlated to other public keys, but that information is concealed. The thing that's a slight nuisance there is that Cathy has to carry all the certificates around with her; she can't generate them from each other. The concrete advantage of using linear congruence exponents is that Cathy, like the authentication server, only has to remember one thing and then she can generate all the others. But conceptually you can think of it as having something like anonymous certificates with attributes. In fact, you don't really even need to remember all the public keys, but that's a separate issue.

Towards a Mechanism for Discretionary Overriding of Access Control

Erik Rissanen[1], Babak Sadighi Firozabadi[1], and Marek Sergot[2]

[1] Swedish Institute of Computer Science
[2] Department of Computing, Imperial College of Science, Technology and Medicine, University of London

Abstract. Because it is difficult to predict access needs in advance and the limitations of formal policy languages it is difficult to completely define an access control policy ahead of the actual use. We suggest the use of an policy language which allows for override of denied access in some cases for increased flexibility. The overrides should be audited and we suggest that the access control policy can be used for finding the people who should perform the audit.

1 Introduction

Traditional access control models either permit access or deny it completely. There is an implicit assumption that all access needs are known in advance and that the conditions of those needs can be expressed in machine readable form. However, reality is dynamic and unanticipated situations occur all the time. Most manual administrative procedures are flexible enough to allow people to stretch the rules. In contrast, most access control models for electronic systems do not allow for similar flexibility. A pedantic access control system can cause frustration, or worse, can prevent people from doing their job. In [7], for instance, it is suggested that if a secretary cannot forge the signature of his/her boss in an electronic system, the electronic system is not going to work. An alternative would be to delegate the required power to the secretary explicitly, but this is just another form of planning in advance. Recent works in distributed access control models [2,6,3] suggests mechanisms to allow delegation and grant of authorisations when they are needed. We might also imagine a solution in the form of a system for requesting access rights as needed.

However, all these approaches require that some authorisations are created in advance. Either the explicit permission or a delegation right that allows the permission to be created when it is needed. None of these approaches address the issue that it is not always possible to plan ahead, and that the authority empowered to create an authorisation may not be available at the time access is needed and authorisation for it is missing.

What we propose instead is a system that would routinely allow access, under specified constraints, even in the case when it is not explicitly permitted. Instead the system depends on audit and sanctioning for enforcement. This mirrors manual organisations. Established organisation theory recognizes such dependency on rule bending, [5].

Of course such procedures are not appropriate in all situations. For instance, some information has to be secret and trusted only to a select group. This is also the case in manual organisations: some rooms are locked and require special key cards to allow people in. Our proposed model makes the same distinctions.

We propose to explicitly, in the security policy, distinguish between what a principal *can* do, what it is *permitted* to do, and what it is *forbidden* to do. The importance of these distinctions in computer security is argued in [4]. The intersection of *can* and *not permitted* is what we refer to as *possibility-with-override* or (sometimes) *ability to override.*

We also propose the concept of *authority resolution* which is a process which automatically finds an access control policy authority who can approve a specific override.

We believe that by including override and authority resolution in the access control policy directly, management of overrides can be improved.

1.1 Related Work

The idea of being able to override denied access is not new. Here we mention just some of the recent work.

Dean Povey, [8,9], has done work that is similar to ours. Povey recognizes the problem with legitimate demand for access in unanticipated situations. His main focus is on guaranteeing system integrity by means of transactions that can be rolled back. We believe that automatic recovery in general is not possible and in many cases costly to implement, so we put our emphasis on audit and manual recovery. However, the issues are orthogonal: automatic recovery, as Povey demonstrates, may be possible in some cases, and here it can be used in conjunction with our methods.

Gunnar Stevens and Volker Wolf [10] have performed a case study at a steel mill. They differentiate between *ex-ante*, *uno-tempore* and *ex-post* access control, depending on whether the permission is granted before, during or after access. Stevens and Wolf noticed that the manual processes used by the steel mill was a mixture of all three types of access control. This compelled them to design a computerized access control system that would allow such a mixture. In relation to their work, this paper deals with ex-post access control.

Jaeger et. al. [11] introduce a concept called *access control spaces.* This concept is used primarily for analyzing conflicts in access control policy or to analyze whether a set of assigned permissions and constraints on possible assignments completely cover all possible assignments. The relation to our work is that access control spaces, which present a partition of permissions similar to which we present, can be used to eliminate any ‘forgotten’ access possibilities. However, there is nothing access control spaces can do for those cases where the desired policy cannot be expressed in the given policy language. Jaeger et. al. in fact suggest the use of access override and audit in some cases.

XACML [12] includes the concept of obligation which can be used for instance for specifying different access levels and that an access should be logged.

We have not been able to find any previous work on automatic authority resolution.

2 Motivation

Existing access control systems implicitly assume that all knowledge about the access needs of users are known, and defining an access control policy is 'simply' a matter of specifying that knowledge in a suitable formal language. To leave that assumption, we need to think about what kind of knowledge we have and what we cannot have.

Assume that we are to draft an access control policy for an organisation. In order to do so, we can study how the organisation is supposed to function, and define the conditions and subjects of access to specified objects. From this we can gather a set of access rules, which we can encode in the machine readable security policy.

However this list of access needs is going to be incomplete. There are two sources of incompleteness. The first is that not all situations for an access permission will be expressible in the machine readable language in use; the second is that there are many needs we simply cannot predict.

This incompleteness in the possibility of drafting a security policy will present itself as a conflict between confidentiality/integrity on the one hand and availability on the other. For instance, we want to protect the confidentiality of medical data, but we are also afraid of emergencies where unanticipated access might be needed.

To solve this conflict, we want to extend our access control model such that we can leave the decision to access at the discretion of the user. We let the user have the possibility to override a denial of access, within specific bounds. The user is better capable of judging the situation in an unanticipated event than the access control system is. We do not place the whole burden of enforcing the security policy on the access control mechanism alone. Instead, to combat abuse, we log all uses of such access overrides in a special way. Note that this is not the same as logging any access for a post review, as it is usually done.

We note that, while designing an access control policy, we will likely find situations where we can say with certainty that access should not be permitted. The access control model should support this so that access override would not always be possible.

3 Overrides

We can divide all possible future access scenarios into the following categories:

1. *Anticipated, allowed and machine encodable*: Situations for which we can say ahead of time that access should be allowed and for which we can express the conditions in machine readable form. Ex. "All employees can read the company newsletter."

2. *Anticipated, denied and machine encodable*: Situations for which we can say ahead of time that access should be denied and for which we can express the conditions in machine readable form. Ex. "Non-medical personnel may not read patient records."
3. *Anticipated, allowed and not machine encodable*: Situations for which we can say ahead of time that access should be permitted but we cannot express the conditions in machine readable form. Ex. "In case of an emergency, any doctor may read the patient's records." (We cannot always formally define "an emergency".)
4. *Anticipated, denied and not machine encodable*: Situations for which we can say ahead of time that access should be denied but we cannot express the conditions in machine readable form.
5. *Unanticipated*: Situations that we have forgotten to consider or cannot predict.

When defining the security policy, we need to partition the "situation space" into these categories, and then map the categories into the security policy. To support the ambiguous cases, we allow three possible outcomes to an access request: *permitted* access, *denied* access and access *possible with override.*

Those situations that we can describe will be easy to include in the *permitted* and *denied* access categories. The situations where we have only a partial description of the situation are placed in the *possible* category. How the forgotten and unanticipated situations are classified depends on how we describe the security policy. There are three alternatives:

1. Define the *permitted* accesses and the *possible* accesses. By default everything else is *denied.*
2. Define the *permitted* accesses and the *denied* accesses. By default everything else is *possible.*
3. Define the *denied* and *possible.* By default everything else is *permitted.*

For instance, in the first case we could say that a doctor is *permitted* to read only the records of his own patients, so we include a permission for that only. However, we can imagine that in case of an emergency a doctor may need to access the records of other patients as well, but there is no way we can define all emergencies formally and in advance, so we include a possibility-with-override for doctors to read any patient record. By default anyone else is denied access and cannot override.

There is a difference between the cases. We cannot describe the scenarios that we do not know in advance, so they will fall in the default category. The most sensible default depends on the organisation. In some cases it may be sensible to protect confidentiality by making denied access the default. In other cases availability may be more important, so possible-with-override is the better default. The third case, where everything is permitted by default, may seem counterintuitive, but might have value in cases where availability is critical.

Jaeger et. al. [11] suggest a categorization similar to the second type, where they allow an user to override some denied accesses.

3.1 Access Control Policies

In this paper, we will focus on the first case listed above, that is when the access control policy defines permitted and possible accesses.

To include override in an access control model we introduce the *possibility-with-override* concept, which is analoguos to a permission with two differences. The first difference is that using a possibility-with-override to gain access should be different for the user than using a regular permission. Some kind of warning message should be presented. The other difference is that uses of possibilities should be logged and thorougly audited.

Most access control policy languages can probably be easily extended to support both and languages already support more than a yes or no answer to an request. For instance, in XACML [12] an obligation could be used to implement both the warning and the logging.

However, in order to assist the audit, we would also like each override to be automatically reported to an authority of that particular permission. This leads to *authority resolution.*

4 Authority Resolution

We call the mechanism by which we from an access control policy can find a set of authorities for a given permission *authority resolution.* The permission does not necessarily have to exist in the policy, in which case we want to find all principals who would be authorised to create the permission. Note that we do not assume that all authorisations are created by a single administrator, but instead we imagine that administration of authorisations is decentralised, with different administrators/managers controlling different parts of the policy.

The authority resolution mechanism can be used in conjunction with possibilities to assist in the audit of overrides. We (informally) suggest the following approach:

1. The user tries to access an object.
2. The application uses an access control decision function to check whether access should be permitted.
3. The access control decision function responds that the user is not permitted to access, but can override that to gain access.
4. The application asks for a confirmation from the user and the user confirms.
5. The application performs the access.
6. The application uses the authority resolution function to get a set of authorities whom to report the override to.
7. One or more of the authorities audit the override and do any sanctioning ex-post.

To answer the question of who is a *legitimate authority* we note that an approval of an override is in effect a retroactive issuing of a permission that would have permitted the access. Thus *the authorities who should be able to approve an* *override are precisely those who can create a permission for the access that was overridden.* How to calculate who they are depends on the access control policy language. We will need a language that contains meta level authorisations, that is permissions for changing the access control policy itself.

In some access control policies it is possible that there are multiple legitimate authorities for approving a given override. This leads us to ask whom among these we should notify and in case we notify multiple authorities, in what order and what happens in case some of them approve and some of them disapprove.

From an organisational point of view we can notice that most organisations are more or less decentralised in that there are vertical and horisontal separations of authority. By horisontal separation we mean that different managers have different areas of authority. For instance the manager of the sales department has different authorisations than the manager of the engineering department. By vertical separation we mean that there typically are different levels of management and higher level management commonly delegate authority to lower levels of management. Even though high level managers in principle have power over low level managers, they may not be familiar with the details of day to day operations. (On a side note, a superior person in an organisational hierarchy does not necessarely have all the authorizations of her subordinates, which somewhat confuses the "vertical" and "horisontal" analogies.) We therefore find two desirable properties for the authority resolution algorithm.

4.1 Desirable Properties

We desire the authority resolution and override approval mechanisms to have

- *Safeness:* Only legitimate authorities should be included.
- *Unobtrusiveness:* Among the legitimate authorities, we should start the audit by notifying those who are most likely to understand the override and least likely to be bothered unnecessarily.

The first property should be easily achieved given an access control model.

For the second property there are more considerations. Mainly we need to find some order in which the authorities should be notified to minimize hassle for high level managers. But we also need to consider what we should do in case several authorities have differences in opinion. Let us look at the order of notification first.

4.2 Order of Notification

First we note that the person who created the possibility-with-override that made an override possible is a prime candidate to be notified first, as long as he is a legitimate authority. The rationale is that whoever made the override possible is best placed to judge whether to approve the override.

Now, depending on the particular access control framework, there may be additional information available. If the access control framework has some kind of hierarchy of authorisations, the hierarchy can be used to order the authorities. One example of such a framework is the calculus of privileges, [3,1], in which all authorisations are issued in the form of delegations, and the delegations form a hierarchy with higher level managers at the top. Such high level authorities should be notified after lower level authorities. Another example would be a directory tree in which authorisations can be inherited along the nodes. Someone authorised for a more specific subtree should be notified before others, since we as a heuristic may assume that he is more familiar with the details of that particular subtree.

It should be noted that depending on the access control framework the ordering of the authorities may not be complete.

4.3 Conflict Resolution

Now, assuming we have some (perhaps incomplete) ordering of the authorities, what would we do in case authorities we notify do not take action or differ in their opinions?

Again, this depends on the particular access control framework. We would like to mimic the semantics of the particular underlaying framework in use as much as possible. So since we can view the approval as a retroactive creation of a permission, we have to ask the question, what would happen if the authorities would have created different permissions for an access corresponding to the override? This will depend on the conflict resolution policy of the framework.

Consider the case in which the access control framework does not have negative permissions. In case some authorities approve and some disapprove, the approvals should have precedence, since there would be no way to create a negative permission in the first place. If all of them disapprove (or do not care), we view the override as disapproved. This means that the order in which we notify the authorities is not critical as long as we notify all of them until someone approves.

In case there are negative permissions in the access control framework, depending on the conflict resolution algorithm, an authority who also could issue a negative permission blocking the overridden access, would be able to disapprove an override and stop it from being sent to higher level authorities.

Thus, how many of all the authorities should be notified depends on the semantics of the particular access control framework in use.

5 Future Work

We have started work on incorporating these ideas in the “calculus of privileges” access control framework, which supports the management stucture of authorisations in the form of a hierarchy of delegations. We will present an algorithm for authority resolution based on that framework and prove some properties of that algorithm.

References

1. Bandmann O., Dam M., Sadighi Firozabadi B.: Constrained Delegations. Proceedings of 2002 IEEE Symposium on Security and Privacy (2002)
2. Blaze M., Feigenbaum J., Lacy J.: Decentralised Trust Management. Proceedings of the 17th Symposium on Security and Privacy (1996) 164 – 173
3. Sadighi Firozabadi B., Sergot M., O. Bandmann: Using Authority Certificates to Create Management Structures. Proceedings of Security Protocols 9th International Workshop (2001) 134 – 145
4. Sadighi Firozabadi B., Sergot M.: Power and Permission in Security Systems. Proceedings of Security Protocols 7th International Workshop (1999) 48 – 53
5. Jaffee D.: Organization Theory: Tension and Change. (2001) 99
6. Li, Grosof, Feigenbaum: A Logic-based Knowledge Representation for Authorization with Delegation. Proceedings of The 12th Computer Security Foundations Workshop (1999)
7. Odlyzko A. M.: Economics, psychology, and sociology of security. Financial Cryptography: 7th International Conference (2003) 182 – 189
8. Povey D.: Optimistic security: a new access control paradigm. Proceedings of the 1999 workshop on New security paradigms (2000) 40 – 45
9. Povey D.: Enforcing Well-Formed and Partially Formed Transactions for UNIX. Proceedings of the 8th USENIX Security Symposium (1999) 47 – 62
10. Stevens G., Wulf V.: A new dimension in access control: studying maintenance engineering across organizational boundaries. Proceedings of the 2002 ACM conference on Computer supported cooperative work (2002) 196 – 205
11. Jaeger T., Edwards A., Zhang X.: Managing access control policies using access control spaces. Proceedings of the seventh ACM symposium on Access control models and technologies (2002) 3 – 12
12. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

Towards a Mechanism for Discretionary Overriding of Access Control (Transcript of Discussion)

Erik Rissanen

Swedish Institute of Computer Science

Last year, the Swedish Prime Minister was stabbed to death in a shopping mall in Stockholm, and of course the police thoroughly investigated it. They had some privacy problems during the investigation: many policemen just looked at the case, because there was no access control on the police system. They didn't have a whole system on-line, because they cannot really predict the needs of individual policemen, and they cannot really audit the whole thing either because there were so many accesses. In the case of the prime minister we suspect that something was going on because he was a famous person, and they know from experience that this tends to happen with famous people, but in the case of a policemen accessing his neighbour's data, or something like that, then there is little reason to notice that something is going on.

There was another case I read about in the newspaper, where a policeman had been selling information from the police databases to private security companies, and this had been going on for years. So there is this inherent conflict between availability and privacy, and integrity and secrecy of data, and that's what we have been looking at in this work.

It is difficult in some cases to describe an access control policy completely, it has a conflict with availability, and more importantly with privacy. So we may need to have some kind of mechanism to override that, and then we audit those overrides. This is very similar to what a number of organisations do. Here we try to optimise this audit to some kind of open systems, in such a way that you don't end up with this big pile of logs that we don't really know what to do with.

When you think about an access control policy there are two reasons why it's difficult to define in advance. First of all it's difficult to know all the needs, because you might forget something, or the world might change. Secondly, even when you know all the needs in advance, you cannot always express your conditions in a formal language because we don't always know how to express them for a computer. So let's consider an access case. Basically there are three dimensions. Firstly, whether this case was anticipated in advance or not. Secondly, whether access should be allowed or not. Thirdly, whether the decision can be encoded in machine computable form. For example, in a hospital a doctor may read all the records of his own patients; the janitor may not read any records at all; a doctor may read any record in case of an emergency, but you can't really express an emergency for a computer.

We have two ideas about how to optimise this audit. The first one is to have some kind of middle ground, between denial and permitted, suspicious cases that we would audit more heavily. This is not a new idea by any means, but usually, this depends a lot on the organisation structure, and how the manual processes in the organisation work and so on. And then we have the idea of a big organisation where you might not be sure who knows about this access. In this case, we would like to look at the policy information and private use, and who would be able to explain or approve the override, to make the audit more efficient.

When you have these three cases for the access policies here, depending on your needs, what is more important? Is it privacy or is it availability? You would choose one of these as a default because these are anticipated and machine encodable. There is a limit on how far you can go; there will always be something that you cannot control. This offers a solution that maintains just a couple of difficulties because there could be several people who have the authority to approve. For instance, in the military, the chief of command may have the authority to approve anything, but you don't want to go to him every time something happens, because he would be completely overloaded, and he wouldn't really know about the things going in the organisation. You would want to start somewhere down the bottom and ask people there, and if you cannot reach them then you would move up. You want to find some kind of structuring policy. So if you think about it, approving is in effect granting permission, but you do it afterwards. You just find the people from the policy who would be authorised to grant this permission in the first place so you can ask those whether it's OK or not. And to be able to do this you need a policy language that contains authorisation. I can just show this last thing; basically it does trace-up, and you get a list of people who, according to some criteria, should know more about the access.

Ross Anderson: If I might make a comment here, your position paper essentially encapsulates the whole of organisational theory, right? There's a large discipline in business schools of how you set up organisations to resolve these problems. As a matter of practical fact you canvass all disputes by an appeal to the lowest common manager, companies need staff departments that set up policies, policies have exceptions (all persons shall do X except with the authority of the head of department who may do X-prime), and so on. What are the audit implications of this? There's a lot of macroeconomics, and people worrying agency defence, and so on. In effect you're trying to reinvent organisational theory, and it might be more effective if you were to go and read through the relevant two or three shelf-feet in a business school.

Reply: That's what I'm doing actually.

Ross Anderson: The context is not simply a matter of arrow diagrams.

Reply: This work is not meant to replace organisational theory completely. This is not a direct mapping of an organisation, this is a tool for the organisation to use. Right now we are going to evaluate this within the military in Sweden; the idea is not to replace all management of companies.

George Danezis: I think that what you explain is really interesting. When I teach undergraduates the most basic model, the LaPadula model, I explain to them that the soldier gives information to the field commander, and the field commander gives information to the general, so information flows up, as it should, and then the general can't actually talk to the field commander because that would involve information going back down. So this idea of declassification which is very often presented as an exception to the model, is actually the rule; it is the interesting case that is happening everyday, whether in the military or in other organisations. Maybe we should move towards a concept that security policy models, and also the actual security policies implemented using them, describe the things that are by default safe, but this doesn't mean that we can't do things, just that we require other authorisation, or accountability.

Reply: Wolfs did a study as a student in Germany, and he found that in practice you got all kinds of different permissions you don't want. So you could just ask for a permission whenever you're doing that access and so on.

Matt Blaze: In organisations, there is a big difference between where authority is flowing from, and who actually has the ability to properly do things. A CEO doesn't have the ability to write in the organisation if the CEO doesn't know how to use a computer. In fact, this is true in organisations in general; the president of the United States theoretically has authority over all the executive branch agencies, but can't just walk into any post office and start delivering mail to people. Even though there is some theoretical flow of authority that goes back in that direction, almost certainly that would represent a security breach because that's not an ordinary exercise that he can do.

Reply: There are two different kinds of authorisation. One is that he has the authority to tell the post office what they should do, but he doesn't have the authority to give himself permission to work in the post office. In this model we differentiate between these, but this is not a lecture on our access control model. A node may not have the authority to approve even though it was in the chain of command, but we would like to keep to the structure of this graph as much as possible.

Pasi Eronen: In organisations there is a saying that any kind of formal chain of command doesn't really explain how organisations actually work. The formal chain of command on paper has nothing to do with how decisions and permissions get decided in practice.

Reply: Yes, and we are attempting to model how they are assigned in practice, not how it is on paper. The whole background to this research on the authorisation model is dynamic military systems. It's not meant to be a general giving orders. It's meant to be like a soldier giving access to his friend, and so on. Now to map this through an organisation is something that you need to basically model the pieces for.

Tuomas Aura: Even in the military I think commands don't go along the lowest common boss; if you send a fighter command to the artillery, you're completely in a different part of the organisation, and you may have never ever communicated with that artillery unit before.

Reply: This is exactly the kind of thing we're looking at in our model. This is very early work and I'm not sure whether it's going to lead to anything, but I think it's very interesting that we haven't been able to find this anywhere else.

Last Orders

David Wheeler: Of course, *conceptually* any problem in Computer Science can be solved by adding another level of indirection. But then there is performance to consider ...

Author Index