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

$$\mathrm{PK}\{(Cert_1, m_1, \ldots, m_{l'_1}) : \mathrm{VerifySign}(Cert_1, m_1, \ldots, m_{l'_1}, m_{l'_1+1}, \ldots, m_{l_1}; VK_1) = 1\} \ . \quad (1)$$

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

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

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 $Cert_1 = \text{Sign}(m_1, \ldots, m_{l_1}; SK_1)$. To this end, the user commits to m_1, i.e., $C = \text{Commit}(m_1, r)$. Then the user (as prover) and issuer (as verifier) execute the following protocol

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

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

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

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

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

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 ($expdate = 2002/05/19$, $age = 55$), the user can decide to prove only that $age > 18$. 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

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

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 identity 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 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, $e : G \times G \to \mathsf{G}$, 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: $(q, G, \mathsf{G}, g, \mathsf{g}, e) \in_R \text{BiLinMapSetup}(\ell_q)$. 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 $\mathrm{PK}\{(\mu, \rho) : C_1 = g^\mu h^\rho \ \wedge \ C_2 = g^\rho\}$ 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

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

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

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

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

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

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.

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

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

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

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 $c_1 = g^{m_1} h^{r_1}, \ldots c_{L'} = g^{m_{L'}} h^{r_{L'}}$, 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 $(a, \{A_i\}, b, \{B_i\}, c, v)$ be a signature on 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.

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}, $\mathsf{V}_{(xy,i)}$, $i = 1, \ldots, L$, and v_s be as follows:

$$\mathsf{v}_x = e(X, \tilde{a}) \ , \quad \mathsf{v}_{xy} = e(X, \tilde{b}) \ , \quad \mathsf{V}_{(xy,i)} = e(X, \tilde{B}_i) \ , \quad \mathsf{v}_s = e(g, \hat{c}) \ .$$

   The prover and verifier compute these values (locally) and then carry out the following zero-knowledge proof protocol:

$$\mathrm{PK}\{(\varepsilon, \mu_1, \ldots, \mu_{L'}, \rho_1, \ldots, \rho_{L'}, \nu, \rho) : c_1 = g^{\mu_1} h^{\rho_1} \ \wedge \ \ldots \ \wedge \ c_{L'} = g^{\mu_{L'}} h^{\rho_{L'}} \ \wedge \ \mathsf{v}_x^{-1} \prod_{i=L'+1}^{L} (\mathsf{V}_{(xy,i)})^{-m_i} = (\mathsf{v}_s)^{-\rho} (\mathsf{v}_{xy})^{\nu} \prod_{i=1}^{L'} (\mathsf{V}_{(xy,i)})^{\mu_i}\} \ .$$

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 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 $v^2 = (\mathrm{abs}(v))^2$ 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

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

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 $\mathrm{abs}(v) = v$ 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 $q < n 2^{-\ell_s - \ell_c - 3}$ 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

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

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.

6 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.