

A General Certification Framework with Applications to Privacy-Enhancing Certificate Infrastructures

Jan Camenisch, Dieter Sommer, Roger Zimmermann
IBM Research, Zurich Research Laboratory

Abstract

Interactions in electronic media require mutual trust to be established, preferably through the release of certified information. Disclosing certificates for provisioning the required information often leads to the disclosure of additional information not required for the purpose of the interaction. For instance ordinary certificates unnecessarily reveal their binary representation.

We propose a certificate-based framework comprising protocol definitions and API specifications for controlled, i.e., well-specified, release of data. This includes controlled release during the certification of data and controlled release of certified data. The protocols are based on proofs of knowledge of certificates and relations over the attributes, ensuring that no side information but only the specified data is revealed. Furthermore, the protocols allow for releasing certified data in plain or encrypted form and allow one to prove general expressions over the data items. Our framework can be seen as a generalization of anonymous credential systems, group signature, traceable signature, and e-cash schemes. The framework encompasses a specification language that allows one to precisely specify what data to release and how to release them in the protocols.

We show how our framework can be implemented cryptographically and how a privacy-enhanced PKI that integrates into today's PKI on the Internet can be built using the framework. We consider our framework a central building block to achieve privacy on the Internet.

1 Introduction

In today's interactions in electronic media users are frequently required to release personally identifying information (PII). A basic principle fostered by consumer protection agencies and privacy advocates is the idea of data minimization stating that the amount of data being provided should be minimal for a given purpose. Another principle is that users should be in control of their data.

The digital world makes it easy for organizations to build extensive profiles of users based on the data they obtain in interactions. It also provides new ways of protecting the privacy of users by employing cryptography to limit the amount of data being released in an interaction following the data minimization principle.

Today's infrastructures within electronic media have often not been designed to take privacy into consideration. Scalable privacy support within such infrastructures requires the availability of a privacy architecture which allows for integrating privacy principles into the infrastructures. Research efforts involving privacy architectures are currently ongoing within the European PRIME project [42] or the NSF-funded PORTIA project [41]. Such architectures include privacy-enhancing authorization mechanisms allowing decisions to be based on a requester's (certified) attributes rather than on her identity, and languages for requesting general (certified) statements from a party. It is likely that such privacy architectures will emerge within the next few years.

In this paper, we define a general framework for data minimization allowing for precise specifications of what (certified) data to release to whom in a transaction and describe its implementation. We then show how our framework can be used to implement a privacy-enhancing PKI. In addition we suggest a possible migration path from today's PKI towards the privacy-enhanced PKI.

The framework we define allows a party to obtain certificates while the issuer does not necessarily learn all data items being certified and it allows the party to prove properties over certified data without revealing any other information, even not the certificate itself. These proofs encompass proofs of polynomial relations over certified data items and the computation of commitments and encryptions of polynomials over certified data items. The proofs allow for logical AND and OR expressions combining such relations. The framework includes protocols with API definitions for obtaining and proving knowledge and properties of certificates while being able to control precisely what data to release and to whom. We present a specification language for specifying this release of data. Another contribution of this paper is an approach for implementing the cryptographic protocols required to release data either in clear or encrypted form and to prove that data is certified using on-the-fly generation of the actual protocols for each protocol instance.

Finally, we show how to realize a privacy-enhancing public key infrastructure (pPKI). Our approach complements the current X.509-based PKI and integrates smoothly into it. Our pPKI provides for a PKI that can be seen as a generalized pseudonym system with extensions such as credential attributes. It allows one to make use of the full capabilities of our certification framework.

Related Work. The research area our framework is positioned in was pioneered by Chaum who defined the concepts of credential systems [18, 20, 21], group signature schemes [23], and electronic cash systems [19]. Our framework can be seen as a generalization of these systems, as well as anonymous attestation schemes [7], traceable signature schemes [32] and identity escrow schemes [33]. Indeed, our framework can be instantiated to obtain a generalized anonymous credential system, a group signature scheme, traceable signature scheme, or an e-cash scheme.

The pseudonym system of Brands [6] also provides efficient techniques for proving relations among committed values, but his overall construction fell short of supporting multi-show unlinkability thus restricting its applicability.

A cryptographic framework for releasing certified data has been proposed by Bangerter et al. [2]. The framework provides an initial step towards a general framework as put forth in our paper. The basic idea of using signature protocols for obtaining and proving signatures remains the same, but our approach is more general with respect to several aspects. It provides much more powerful proof capabilities; Bangerter et al.'s only allows AND-linked and less powerful predicates. In particular, what was provided in the context of verifiable encryption does not match the functionality of our framework. Their framework is thus not capable of handling many of the scenarios we can easily account for. Furthermore, our framework is much more concrete, i.e., we provide a specification language and concrete protocol interfaces close to an API-level making it ready for implementations and integration into privacy architectures and infrastructures. This was missing in the prior work. The high level of abstraction of our specification language allows for an easy integration into privacy architectures. The implementation of our framework is readily employable for pseudonymous authentication.

Regarding implementation, we are not aware of anyone having followed an approach as ours of using a compiler for generating zero-knowledge proof protocols from a given protocol specification input. Previous work in the field of compiler generated protocols includes the work of Tsoukalidis and Weis [45] for applying compilers to protocol generation for cryptographic protocols and the work of Millen and Muller [36] focusing on the automatic generation of authentication protocols from high-level protocol specifications.

The current X.509-based Internet certificate infrastructure as defined by the IETF is mainly being used for authentic binding of public keys to identities. The standards include algorithms such as signature schemes [3], protocols for certificate management [1], and definition of certificate structures for public key certificates and certificate revocation lists [30] and for attribute certificates [27]. However, privacy is not considered here.

Paper Outline. We first present our generalized framework for controlled data release in Section 2 including an interface specification for the protocols and the definition of a specification language for data release. Section 3 describes the approach we took for implementing our framework using a compiler for on-the-fly generation of the involved zero-knowledge proof protocols. In Section 4 we apply our framework to obtain a privacy-enhanced PKI extensions for today's Internet PKI. Section 5 concludes our work.

2 A General Certification Framework Supporting Controlled Release of Data

We define a general framework for the controlled release of certified data and controlled release of data for certification. Controlled release of certified data means that the releasing party can specify which information on data items of certificates to release and to whom. Controlled release of data for certification means that the issuer of a certificate learns only a specified subset of the data items being certified.

Controlled release of certified data encompasses the release of (partial) information on the data items of certificates or polynomials over data items, either in clear or encrypted form. This involves computation of commitments and encryptions of certified data items or multivariate polynomials thereof.

Encryptions that contain a value that is provably a multivariate polynomial relation between certified data items of multiple certificates, other commitments, and encryptions have a wide field of applications, such as conditional show of data, or escrow of values that provably fulfill some relation w.r.t. certified data items.

Our framework is based on Bangerter et al.'s framework [2], but offers more evolved functionality, in particular w.r.t. general proofs of OR-related predicates and a concrete specification language for specifying what information to release.

We first present the building blocks, and then the protocols for certificate issuance and proof of certificates.

2.1 Cryptographic Building Blocks

The basic cryptographic building blocks of our framework are particular signature schemes, commitment schemes, encryption schemes, and zero-knowledge proofs of knowledge. We informally describe the building blocks and how they interact to form our framework.

Commitment Schemes. A *commitment scheme* allows a party to commit to a tuple (x_1, \dots, x_u) of (secret) values to another party. A commitment does not reveal any (computational) information on the tuple to the other party (hiding property) and prevents the committing party to change the values being committed to at a later stage (binding property). Either one of the properties can be information theoretic, but not both at the same time.

The *Commit* operation requires a public key PK and a type specifier. In many schemes a random group element r is created by the algorithm and returned as part of o_i .

$$(c_i, o_i) := \text{Commit}(x_{i,1}, \dots, x_{i,l}, PK) \quad (1)$$

The object c_i contains the information to be sent to the other party, namely the cryptographic commitment, and the public key. The object o_i contains x_1, \dots, x_u and randomness r and is retained by the committing party. If the commitment is used in a later proof, o_i provides the secret information knowledge of which is proved and c_i provides the appropriate information for the verifying party.

Signature Schemes and Protocols. In our framework we make use of *signature protocols* for obtaining and proving knowledge of signatures on tuples of messages. Each message can either be known to the signer, he can know a commitment of it or have no information on it. Obtaining a signature is a protocol between the receiver of the signature and the issuer where the issuer learns only a subset of the messages to be signed. Proving knowledge of a signature on a tuple of messages is a protocol between a prover and a verifier where information on the messages m_i is revealed selectively, i.e., allowing for proving relations on commitments to the messages m_i . Theoretically, such protocols can be obtained for any signature scheme, but these constructions would not be practical due to the required computational effort.

Two signature schemes with practical protocols, the SRSA-CL scheme [11] and the BL-CL scheme [12] are schemes that we use within our framework. These schemes allow proofs to be performed on the messages m_i being signed. Furthermore, the schemes indeed allow that a signature be issued on messages that are not known to the issuer, but to which the issuer only knows a commitment. These schemes fit ideally into our framework as they allow the efficient discrete logarithm based proofs of knowledge and relations between discrete logarithms of different certificates, commitments, and encryptions.

In addition to the privacy-enhancing signature protocols where the signer learns only a subset of the messages, our framework also supports conventional signature schemes such as RSA [43], Schnorr [44], or DSA [37]. For these schemes the protocols become trivial in that they resemble classical signature creation and verification. However, they do not allow one to hide messages from the issuer or the verifier of a signature.

Verifiable Encryption. Our framework makes use of *encryption schemes* to encrypt values within the protocol for proving knowledge of certificates. The encryption gets a tuple of messages (m_1, \dots, m_k) , a label L , and a public key PKE as input and outputs the encryption of the tuple.

The decryption algorithm requires a ciphertext, a label L and a decryption key SKE as input and outputs a tuple of messages (m_1, \dots, m_k) . The label L must be the same one as used for the encryption. The label encodes a condition under which decryption may be performed. The key SKE is private to a party trustworthy for a particular application. The party performs a decryption only in case the condition that is presented is fulfilled.

Within the certificate protocols properties over the messages m_i of an encryption that has been computed can be proved, in particular polynomial relations to data items of certificates, commitments, and other encryptions.

The verifiable encryption schemes of Camenisch and Damgård [9] and of Camenisch and Shoup [15] are applicable to our framework. The latter are particularly suitable as they integrate smoothly into the discrete logarithm based zero-knowledge proof protocols.

Zero-knowledge Proofs of Knowledge. In a *zero-knowledge proof* of knowledge a prover convinces a verifier that she knows a witness w such that a predicate P is fulfilled without releasing any further information on w .

We employ *zero-knowledge proofs* for obtaining signatures and for proving knowledge of signatures. Within proof protocols, zero-knowledge proofs are used for proving knowledge of certificates, and proving relations between data items of certificates, commitments and encryptions.

Concepts. A *certificate* on a tuple (d_1, \dots, d_k) of data items is a tuple $(d_1, \dots, d_k, \sigma)$, where σ is a signature on the data items. A certificate is obtained via an instance of the *CertificateIssuance* protocol. Knowledge of a certificate can be proved via the *CertificateProof* protocol. A certificate as defined in our framework allows one to treat data items individually in case the advanced signature protocols are being employed.

A *certificate image* or *image* is the result of a protocol involving one or multiple certificates. An image contains the assertion over the certificates that has been proved. We note that the image does not always contain the signature of the issuer, but a transcript of a proof of knowledge of the signature performed by the owner.

During the issuing protocol of a certificate, the signature on the data items is computed by the issuer without all data items being known to the issuer in clear and sent to the receiver. The proof protocol of a certificate proves to the verifier the prover's knowledge of a signature on data items while selectively revealing them or proving statements about them, and making commitments and encryptions of polynomials over data items. Multi-show unlinkability follows immediately from the fact the a certificate is not disclosed, but knowledge of it is proved.

We note that for the protocols defined below, certificates, commitments, and encryptions are defined on tuples of data items rather than on single items.

2.2 Certificate Issuance Protocol

The issuance of a certificate is a protocol *CertificateIssuance* between a receiver and an issuer. The outcome for the receiver is a certificate for the data items d_1, \dots, d_k of which the issuer only learns a subset. A proof involving the certificate can later be done w.r.t. the issuer's public key for signature verification PKS_I .

A data item to be signed can either be known to the issuer in plaintext form, be known as a commitment to the data item, the data item can be jointly randomly generated, or it can be unknown to the issuer, i.e., known to the receiver only.

Known data items are used for all items that are contributed by the issuer or that the issuer has to know from a policy point of view. Committed data items are useful for the case where the receiver has proved before that the committed value is a particular data item appearing in another certificate and where this data item is to be included into the new certificate without the issuer learning it. An example for this is when the name and other personal data from a passport certificate should be included into the certificate to be issued. Jointly randomly generated data items are useful for limited-show certificates such as e-coins. Data items unknown to the issuer can be used to realize e-coins, as well.

During the execution of the protocol the data items to be jointly randomly generated are computed and the issuer computes a value being a signature on the data items d_1, \dots, d_k and sends it to the receiver. The receiver obtains the certificate by copying the data items and signature into a certificate data structure.

2.2.1 Protocol Interface Specification

The protocol *CertificateIssuance* for issuing a certificate is a protocol between a receiver and an issuer. Common input is presented to both parties and each party additionally receives a private input. Both parties obtain commitments as output, the receiver in addition gets opening information on the commitments and the newly issued certificate as output. The certificate the receiver obtains is comprised of a tuple $(d_1, \dots, d_k, \sigma)$ of data items d_i and the signature σ on the data items. The way the signature has been obtained for each individual data item is not reflected in the certificate.

Let d_1, \dots, d_k be the data items on which to obtain a certificate. Some of the items may be unknown as of the start of the protocol. Let $D = \{1, \dots, k\}$ be the set of indices for the data items. Let the sets D_{known} , $D_{\text{committed}}$, D_{random} , and D_{unknown} partition D . Thus, $D = D_{\text{known}} \cup D_{\text{committed}} \cup D_{\text{random}} \cup D_{\text{unknown}}$ and the subsets are mutually disjoint. A data item d_i with $i \in D_{\text{known}}$ is known to both parties, a data item d_i with $i \in D_{\text{committed}}$ is known by the receiver only, the issuer has a commitment thereof. A d_i with $i \in D_{\text{random}}$ is generated within the protocol, and a d_i with $i \in D_{\text{unknown}}$ is known by the receiver, the issuer has no data regarding it. The interface of the protocol is specified as follows:

<i>CertificateIssuance</i>	
Common input	$\{d_i : i \in D_{\text{known}}\}, \{c_i : i \in D_{\text{committed}} \cup D_{\text{random}}\}, PKS_I, \chi$
Receiver private input	$\{o_i : i \in D_{\text{committed}} \cup D_{\text{random}}\}, \{d_i : i \in D_{\text{unknown}}\}$
Issuer private input	SKS_I
Receiver output	$\{c_i, o_i : i \in D_{\text{random}}\}, cert$
Issuer output	$\{c_i : i \in D_{\text{random}}\}$

The commitments and openings $\{c_i, o_i : i \in D_{\text{random}}\}$ are uninstantiated when presented as protocol input and are generated during the protocol execution. The o_i s contain the randomly generated data items for retrieval by the receiver and the c_i s the corresponding commitments.

Any commitment c_i contains—regardless of its instantiation state—information on the commitment public key, the algorithm, and the number of values in the tuple it will be a commitment of. After the protocol execution, the previously uninstantiated commitments and corresponding openings have been generated and the commitments can be retrieved by both parties, the opening information, however on the prover side.

2.2.2 Issuance Specification Language

The issue specification χ passed in both party's inputs is expressed using a simple language specifying for each data item how it is contributed to the protocol and what information the issuer learns about it. This is defined by a triple for each data item: The first element is the index of the data item being specified where $cert[i]$ refers to d_i of the certificate, the second element is a specifier of the issuance mode for the item being either of known, committed, random, or unknown. The last element is—depending on the second element—either a reference d_i to a data item of the input, an index $c_i[j]$ to an instantiated commitment, an index to an uninstantiated commitment, or a reference d_i to a data item depending on whether the second element specifies known, commitment, random, or unknown, respectively.

$$\chi = \{(cert[1], \text{known}, d_1), (cert[2], \text{committed}, c_1[1]), (cert[3], \text{random}, c_2[1]), (cert[4], \text{unknown}, d_2)\} \quad (2)$$

The specification in Example (2) is for issuing a certificate on (d_1, \dots, d_4) where $D_{\text{known}} = \{1\}$, $D_{\text{committed}} = \{2\}$, $D_{\text{random}} = \{3\}$, and $D_{\text{unknown}} = \{4\}$. After successful protocol execution $o_3[1]$ contains the randomly generated data item d_3 and $c_3[1]$ contains a commitment of d_3 .

2.3 Certificate Proof protocol

The protocol *CertificateProof* is a protocol between a prover and a verifier that allows the prover to prove to the verifier knowledge of valid certificates and general statements over the data items of multiple certificates, commitments, and encryptions. It further allows one to compute commitments and encryptions specified through multivariate polynomials over data items and encryptions.

The underlying key principle is that certificates are not sent, but statements regarding them are proved using zero-knowledge proofs of knowledge such that no more information than what is specified is revealed. From the applied cryp-

topographic mechanisms it follows immediately that multiple proofs over the same certificate are mutually unlinkable and unlinkable to the *CertificateIssuance* protocol instance it was issued in.

2.3.1 Protocol Interface Specification

The common inputs to the prover and verifier are the public keys for signature verification $PKS_{I_1}, \dots, PKS_{I_k}$, (possibly uninstantiated) commitments c_1, \dots, c_u , and (possibly uninstantiated) encryptions e_1, \dots, e_v . Let the sets $\hat{C} \subseteq C = \{1, \dots, u\}$ and $\hat{E} \subseteq E = \{1, \dots, v\}$ contain the indices of uninstantiated commitments and uninstantiated encryptions, respectively. The prover additionally gets as input a list of certificates $cert_1, \dots, cert_k$ having been obtained from issuers I_1, \dots, I_k , (uninstantiated) opening information o_1, \dots, o_u corresponding to the commitments, and a prover-side proof specification ξ_P . The verifier gets a verifier-side proof specification ξ_V as input.

The verifier's output are the commitments $\{c_i : i \in \hat{C}\}$ and encryptions $\{e_i : i \in \hat{E}\}$ having been computed during the protocol execution and a certificate image Π containing the statement conveyed through the proof together with the proof transcript. The prover's output are the commitments and openings $\{c_i, o_i : i \in \hat{C}\}$ and the encryptions and corresponding plaintexts $\{e_i, p_i : i \in \hat{E}\}$.

<i>CertificateProof</i>	
Common input	$PKS_{I_1}, \dots, PKS_{I_k}, c_1, \dots, c_u, e_1, \dots, e_v$
Prover private input	$cert_1, \dots, cert_k, o_1, \dots, o_u, p_1, \dots, p_v, \xi_P$
Verifier private input	ξ_V
Prover output	$\{c_i, o_i : i \in \hat{C}\}, \{e_i, p_i : i \in \hat{E}\}$
Verifier output	$\{c_i : i \in \hat{C}\}, \{e_i : i \in \hat{E}\}, \Pi$

Commitments. The set $C = \{1, \dots, u\}$ defines the indices i of the commitments and openings: $\{c_i, o_i : i \in C\}$. The subset of the commitments and corresponding openings $\{c_i, o_i : i \in \hat{C} \subseteq C\}$ are uninstantiated, i.e., the commitment contains only information on the commitment public key and further parameters, but not yet a cryptographic commitment and the openings contain not yet the opening information for the commitments. The objects $\{c_i, o_i : i \in C \setminus \hat{C}\}$ are instantiated commitments and openings.

Uninstantiated commitments in the proof protocol allow one to create commitments during the protocol execution that can later be used to issue a signature on the value being committed to or to use it within other proofs. An application example for this is if an organization issues a certificate that shall contain the user's identity from another certificate. To achieve this, the other certificate is first proved and the identity data item is being committed to. Then the new certificate is issued using that commitment and other data items.

Encryptions. Analogously, the set $E = \{1, \dots, v\}$ defines the indices for the encryptions e_i and corresponding plaintexts p_i . The subset $\hat{E} \subseteq E$ defines again the set of uninstantiated encryptions and plaintexts. An uninstantiated encryption contains an algorithm identifier, a label L , and an encryption public key PKE_T of a party T , and the number of elements in the plaintext tuple of the encryption. The label L defines a condition under which party T will perform a decryption. The same label that has been used for encryption and a private key SKE_T known to T has to be passed to the decryption algorithm in order for the algorithm to decrypt correctly. Considering these properties, the encryptions realize a conditional show of the underlying values under condition L .

Uninstantiated encryptions will be computed during the protocol execution. After the protocol has terminated the encryptions and corresponding plaintexts can be retrieved, plaintexts only by the prover.

Proof Specification. The proof specification defines what statement to prove over the list of certificates, commitments, and encryptions thereby also defining what values to commit to and what values to encrypt. The proof specification ξ_P for the prover and ξ_V for the verifier are very similar, the only difference being that the prover's specification in addition allows the definition of what is to be proved. The specification language will be further elaborated on below.

2.3.2 Specification Language

A specification is a logical formula over predicates. The specification language for expressing a proof specification ξ_P or ξ_V uses variables to reference data items, committed values, and encrypted values of the input. We stress that a variable always references the secret input of the prover and is used for expressing statement over this secret input. Secret input are certificates, commitment opening information, and plaintexts to encryptions. The verifier only has corresponding data objects that do not allow to infer the secrets. Such objects are commitments or encryptions.

Referenced commitments and encryptions and their corresponding opening and plaintext objects can be either uninstantiated or instantiated. Data items are references to the (secret) data items of certificates expressed with variables of the form $cert_i[j]$ where i is the index of the certificate the data item is contained in and j is the index within the data items of the certificate. A variable $c_i[j]$ refers to the j -th value of the tuple committed to with commitment i . A variable $e_i[j]$ refers to the j -th plaintext value corresponding to the tuple being the plaintext for the i -th encrypted tuple. If a c or e refers to an uninstantiated instance, restrictions are imposed on the usage of the variable.

Predicates. The language for specifying the disclosure of certified data uses predicates as basic building blocks. A predicate defines a relation between multiple data items (possibly of different certificates), committed values, numeric constants, and encrypted values. The arithmetic relation being proven is over \mathbb{Z} with operator notation being $+$ and \cdot . The relational

operator in a predicate can be any of $=$, \neq , $>$, $<$, \geq , or \leq with their usual semantics.¹ Additive and multiplicative numeric constants and exponents can be the same size as data items.² A predicate expresses the relation between the involved prover-known secret values.

$$cert_1 \quad (3) \quad cert_1[1] \geq c_1[2] \quad (5)$$

$$cert_1[1] \quad (4) \quad 10 \cdot cert_1[1] + 20 \cdot cert_2[1]^4 = 4 \cdot e_1[1] + 40 \quad (6)$$

Example (3) expresses knowledge of the signature of certificate $cert_1$ without making any statement on the data items of the certificate. Example (4) states knowledge of a signature of to data item d_1 of $cert_1$ without any further statement. Predicates (3) and (4), each for itself, express the same. In Example (5) a greater-than-or-equal relation is stated between the data item d_1 in $cert_1$ and the second element of the tuple committed to with c_1 . Example (6) states a more general polynomial relation.

Specification Formula. Multiple predicates are combined to a logical formula. Arbitrary formulas can be constructed by interconnecting predicates with the logical \wedge and \vee operators. The usual precedence and semantics of the operators applies. Paratheses can be applied in their usual semantics.

A logical formula can allow that multiple different clauses in its disjunctive normal form (DNF) be fulfillable by the certificates provided as input, for example both predicates in the formula $cert_1[1] = 10 \vee cert_1[2] = 20$ can be fulfilled with one certificate $cert_1$ with appropriate data items. Thus it is required that the prover specify what predicate(s) of the formula to fulfill. This is particularly important in case uninstantiated commitments and encryptions are to be instantiated. The construct of the proof specification language used for defining this is a pair of $\langle \rangle$ elements placed around the predicates of the formula that the prover intends to satisfy with the provided secrets. The secrets must fulfill the specified predicates in order that the proof be successful. One pair of $\langle \rangle$ elements can span multiple \wedge -connected predicates. It is important for the semantics of instantiation of commitments and encryptions that the $\langle \rangle$ -marked predicates make exactly one clause in the DNF of the formula be fulfilled.

The $\langle \rangle$ notation is, of course, only applied to the prover's specification. The remainder of the proof specification is equal for the prover and verifier in the protocol.

$$\langle cert_1[1] > 21 \rangle \vee cert_2[1] \quad (7)$$

$$\langle cert_1[1] > 21 \wedge cert_2[1] \rangle \quad (8)$$

Example (7) proves that the certified data item $cert_1[1]$ is greater than 21 or that the prover knows a signature on data item $cert_2[1]$. It evaluates to true if either of the predicates is fulfilled. The $\langle \rangle$ notation defines that the prover provides $cert_1$ as input. Example (8) requires both certificates $cert_1$ and $cert_2$ to be provided as input by the prover, also expressed with the same notation.

Uninstantiated Variables. Instantiated variables are instantiated through the presented certificates, commitments, and encryptions. Uninstantiated commitments are instantiated with values as a first step in the protocol and uninstantiated encryptions are computed. Both of this is done according to the prover's setting using the $\langle \rangle$ notation in the proof specification. Depending on the specification, a commitment or encryption is instantiated either to a predicate-defined or to a random value. Thus, after the instantiation step, all committed and encrypted values are fixed.

An uninstantiated variable is instantiated depending on the predicate it appears in. If an uninstantiated variable (commitment or encryption) appears in a predicate in the fulfilled clause in the DNF it is being assigned as specified by the predicate it is defined through. (It can also appear in other DNF clauses) The value of the variable is derived from the polynomial relation being stated in the predicate. If it appears only in DNF clauses that are non-marked, it is instantiated with a random value. It is not possible to have a uninstantiated variables appear in predicates in one DNF clause that do not unambiguously define all of them or that are contradicting.

The actual instantiation of the variable and its underlying commitment or encryption is governed by the prover by the $\langle \rangle$ language element. Thus one can think of it being correctly instantiated when referring to it in predicates.

$$c_1[1] = cert_1[1] + cert_2[1]^2 \quad (9) \quad c_1[2] = e_1[1] + c_1[4] \quad (11)$$

$$e_1[1] = cert_1[1] + cert_2[1]^2 \quad (10) \quad cert_1[1] + cert_1[2] = cert_1[3] + c_1[1] \quad (12)$$

Assume that in the example predicates (9) to (12) variables $c_1[1]$, $e_1[1]$, $c_1[2]$, and $c_1[1]$ are uninstantiated. In (9) $c_1[1]$ gets assigned the expression on the right side of the equality sign, the same is true in (10) for $e_1[1]$. Predicate (11) is similar in flavour. Example (11) assigns $c_1[1]$ with the value defined by the predicate.

$$\langle cert_1[1] > 10 \wedge e_1[1] = cert_1[1] \rangle \vee \langle (cert_1[1] \leq 10 \wedge e_1[1] = cert_1[2]) \rangle \quad (13)$$

$$\langle (cert_1[1] = 10) \rangle \vee \langle cert_1[1] = 20 \wedge e_1[1] = cert_1[2] \rangle \quad (14)$$

In Example (13) above, $e_1[1]$ is instantiated with $cert_1[2]$. In Example (14) $e_1[1]$ is instantiated with a random number.

An initially uninstantiated commitment or encryption can be used like any other variable with the additional restriction that it be unambiguously defined through the predicates it appears in.

¹One application of polynomial relations over multiple attributes of a certificate are single-show and general k -show certificates.

²Processing of exponents requires additional runtime linear in the length of an efficiently computable addition chain for the exponent.

2.4 Semantics of the Specification

The scope of variables appearing in the formula is the whole formula, i.e., they reference the same data item, committed value, or encrypted value throughout the formula. The semantics of the formula in terms of what is being proved over the prover's secret input is precisely defined by the logical formula in the semantics as defined in logics. Only a subset of the referenced certificates is required as input in order that the formula be fulfillable.

A particular variable $cert_i[j]$ appearing in a predicate specifies that knowledge of the signature of the tuple of data items containing the data item is proved.

The formula evaluates to true if the prover inputs a set of certificates and commitments such that the formula is fulfilled. This is the case if the formula evaluates to true considering the boolean value of each of the predicates. It is important that at least one clause in the DNF of the formula is fulfilled.

A predicate evaluates to true, iff the prover-known secrets fulfill the polynomial relationship defined by the predicate.

Certificate Images. Depending on the way the proof is performed cryptographically, the certificate image can be either convincing to third parties or does not contain any information on whether the proof was actually executed. We denote the further a *transferable image* and the latter a *non-transferable image*. The transferable image has much the same properties as a conventional certificate, i.e., it is a statement about its owner endorsed by an issuing party. In fact, a transferable image can be endorsed by multiple parties in case certificates of different issuers were involved in the proof. This concept of a certificate containing \wedge and \vee related statements over data items of different issuers is a new concept. It is interesting to note that this certificate is created within a proof protocol which did not involve any issuers and nevertheless is a statement endorsed by all these issuers. Thus, a certificate image of a certificate in our framework much resembles the classical idea of a certificate.

Example for Using our Framework. When considering current directions of legislation towards preventing unconditional anonymity, use cases based on conditional anonymity become increasingly important. Consider, for example, a service provider being required to provide—under well-defined conditions such as a court order—an identity-equivalent item of the service requester for each transaction. Assume existence of a US authority and a European Union authority each providing a public key for encryption. Consider furthermore that users have electronic passport certificates issued either by the US or the European Union. Let the certificate $cert_1$ in the example be a US passport, $cert_2$ a European Union passport. Assume a user has only a US passport. Let the data item $d_1[1]$ be the passport number in the passports which, if known to the respective authority, easily allows to find the associated holder.

$$\langle (e_1[1] = cert_1[1] \wedge e_1[2] = 1 \wedge e_2[2] = 0) \rangle \vee (e_2[1] = cert_2[1] \wedge e_2[2] = 1 \wedge e_1[2] = 0) \quad (15)$$

This proof states either of the following: *i*) The prover has a United States passport and its serial number is encrypted in $e_1[1]$, the constant 1 is encrypted in $e_1[2]$ indicating that e_1 is the encryption containing a valid passport number. For encryption the public key of the US authority and an appropriate condition L_{US} (both provided in the input) is used. Additionally 0 is encrypted in $e_2[2]$ indicating that e_2 contains no passport number. For the latter, the EU public key and EU condition L_{EU} is used. No statement is made regarding $e_2[1]$. *ii*) The user has a European Union passport and this passport's serial number's encryption is $e_2[1]$ and the constant 1 is encrypted in $e_2[2]$ indicating that e_2 contains the user's passport number. This is done using the encryption key of the EU agency and a condition L_{EU} . A 0 is encrypted under the US key and condition in $e_1[2]$. No statement is made regarding $e_1[1]$.

By using such a proof of a disjunction, no information on which passport the user has is conveyed. Only in the rare case when the encryptions have to be decrypted by the respective government agency under the defined condition, will it also be clear what country the user is a citizen of. Scenarios of the above type will become increasingly important if legislation requires that anonymity must not be unconditional, i.e., be revocable. For such a legislative environment our framework allows one to perform minimum disclosure proofs as above.

2.5 Applications of the Framework

Our framework has numerous concrete applications. A very promising application is to construct a generalized certificate system being a generalized pseudonym and credential system on top of it. We do this in Section 4. Also a group signature scheme can be easily constructed out of it by turning a proof protocol into a signature protocol by applying the Fiat-Shamir heuristic [28, 40]. The certificates issued by an issuer are the signature keys, the group of parties holding a certificate from the issuer are the group. The group manager is a dedicated party that provides an encryption key. The key is used to verifiably encrypt a data item of the certificate that identifies the holder. Electronic cash, even allowing for k-show coins, can be realized by using the capability of proving polynomial relations over the data items of a certificate.

3 Implementation

In this section we first present the basic ideas of how the cryptographic mechanisms interplay in the proof protocols. We use the high complexity of the protocols as one motivation for our compiler-based approach for the implementation.

Cryptographic Mechanisms. In the common parameters model, we build upon several known protocols for proving statements about discrete logarithms, such as (1) proof of knowledge of a discrete logarithm modulo a prime [44] or a composite [29, 26], (2) proof of knowledge of equality of representation modulo two (possibly different) prime [22] or composite [14]

moduli, (3) proof that a commitment opens to the product of two other committed values [13, 16, 5], (4) proof that a committed value lies in a given integer interval [17, 13, 13, 4], and also (5) proof of the disjunction or conjunction of any two of the previous [24].

That is, we can use these protocols to efficiently prove the statements we defined among messages that are 1) committed using the Pedersen commitment scheme [39, 26], 2) encrypted using the Camenisch-Shoup encryption scheme, or 3) signed using the Camenisch-Lysyanskaya signature schemes [11, 12] without revealing the messages themselves.

For obtaining concurrent zero-knowledge proofs, Damgård's construction [25] is applied. Non-interactive zero-knowledge proofs are obtained by using the Fiat-Shamir heuristic [28]. The latter in addition allows to create a group signature scheme of the protocol for proving certificates.

Zero-Knowledge Proofs. The mechanisms mentioned above allow one to construct the zero-knowledge proofs required for the *CertificateProof* protocol. The input to the protocol on the prover side provides secrets such as certificates, values the prover is committed to, and values that are encrypted. The secret values are discrete logarithm representations, i.e., tuples of exponents. The secrets are preimages to multi-exponentiation homomorphisms. The homomorphism image of a tuple of secrets is known to the verifier such as the homomorphism itself.

A certificate issued using the SRSA-CL signature protocols [11] is a tuple $(d_1, \dots, d_k, (e, A, v))$ where A is an element of the group defined by the public verification key PKS_I . The other elements of the tuple are exponents. Proving knowledge of such a signature involves creating a new multi-exponentiation homomorphism involving generators from the public key and a value \hat{A} obtained by randomizing the signature. The exponents are the remaining values from the certificate. A proof of knowledge of the signature is a proof of knowledge of the preimage of the new homomorphism (i.e., proof of knowledge of the exponents).

The cryptographic opening information of a commitment contains the preimage of a multi-exponentiation homomorphism defined by the commitment public key and algorithm type. A proof of knowledge of the commitment is a proof of knowledge of the preimage of the homomorphism w.r.t. the verifier-known function image.

The plaintext of an encryption is similarly a discrete logarithm representation. A proof that the plaintext is actually contained in the verifier-known encryption amounts to the construction of multiple homomorphisms defined by the encryption public key and algorithm and a proof of relations among elements of the preimage tuples (i.e., among the exponents).

The proof of algebraic relations such as the greater-than relation again requires the application of preimage proofs of knowledge w.r.t. multiple homomorphisms and proofs of relations among the preimage elements.

The aforementioned individual proofs are combined into one zero-knowledge proof for one instance of the *CertificateProof* protocol depending on the proof specifications ξ_P or ξ_V . The high complexity of the resulting proof motivates a compiler-based approach for automatically generating the zero-knowledge proofs on-the-fly for each protocol instance.

Approach. The input to the framework layer, that is the protocol specification, determines the zero-knowledge proofs to be used for proving the statements as outlined above. The framework layer generates a specification for a zero-knowledge proof of knowledge from this proof specification for each protocol instance. The generality of our proof specification leads to complex zero-knowledge proofs that are derived from the specification. Thus, an additional software component—a compiler for zero-knowledge proofs of knowledge—has been developed and implemented that takes a zero-knowledge proof specification as input and outputs a zero-knowledge proof implementing the specification. The proof is then executed by an interpreter.

There are multiple motivations for the compiler-based approach. As the proofs to be generated are very complex, it is advisable from an engineering point of view to migrate the proof generation to a separate software component that can be separately maintained and verified. Implementing the proofs manually without using a compiler approach would be highly error-prone and hardly possible to be done. Next, the specification input for the zero-knowledge proof compiler allows for easier verifiability of the correctness of the implementation of the framework layer.

Technical Characteristics and Performance. Data items of certificates are integer numbers of a certain bit length. The length can be chosen when parameterizing an implementation. Our implementation supports data items to be encoded as 256 bit integers. This allows enough flexibility for the data items, e.g., the use of SHA-256 [38], and is practicable in today's computing environments.

To emphasize the real-world viability of our protocols, we provide some performance figures of proofs of certificates using the SRSA-CL signature protocol and a group with 2048 bit SRSA modulus. A *CertificateProof* protocol proving knowledge of a certificate with 1 attribute takes approximately 350 ms, with 4 attributes 380 ms, with 20 attributes 560 ms, and with 40 attributes 790 ms on a 1.8 GHz Pentium M machine using a version 1.4 Java runtime environment. The figures do not include overhead for communication over a network. A C/C++ implementation would be faster by approximately a factor of 4. The computational effort is distributed between a prover and a verifier. These performance figures have been obtained by using methods using similar ideas to the ones put forth by Brickell et al. [8] and Lim and Lee [34] requiring reasonable memory for maintaining precomputations for speeding up exponentiation.

4 Constructing a Privacy-Enhancing Certificate Infrastructure

This section introduces a construction of a privacy-enhancing PKI (pPKI) based on the framework presented in Section 2. Our pPKI adds privacy-enhancements to the functionality of an X.509 certificate infrastructure as used in today's Internet. A gradual move from PKI towards pPKI is possible which makes pPKI feasible for being deployed on the Internet in the near future.

The basic idea of a certificate infrastructure is to provide the authentic binding of attributes to entities. A certificate asserts such a binding through the signature of the certificate issuer. A special class of certificates are public key certificates (PKCs) that bind public keys to entities.

Current approaches are mainly based on binding attributes (and in particular keys) to an entity via the identity of the entity. Public key certificates as defined by X.509 bind public keys to identities. Such certificates are widely used in practice today for establishing secure connections between entities. Attribute certificates bind attributes to an entity by binding it—by reference to the PKC’s unique certificate number—to a PKC of the entity.³ A proof of ownership of the attribute certificate requires it and the PKC it was issued on to be released implying releasing identity information and establishing linkability. This privacy-invasive property renders the approach inapplicable.

We extend the current Internet infrastructure using certificates based on our framework to account for the current lack of privacy support. The extension complements the functionality of PKI with regard to privacy-enhancing mechanisms useful for end users and integrates into the existing infrastructure. It uses current PKC functionality for validating public keys used for certificate proofs.

We define attribute certificates and PKCs in our extension. A certificate can be based on any signature scheme, either conventional schemes such as RSA [43], Schnorr [44], or DSA [37] or schemes and protocols such as SRSA-CL [11] or BM-CL [12]. In the further case, no more in terms of privacy protection than in traditional PKI can be achieved, in the latter case the full potential of our framework for controlled release of certified data can be delivered to the Internet PKI.

We next describe functionality that must be provided for integrating our extensions into the Internet PKI, define our PKCs and attribute certificates, show how they are used, and provide an idea of how to gradually deploy our extensions.

4.1 Adaptation Layer

The application of our framework for X.509-type certificates requires an adaptation layer to be built on top of the framework. This layer adds functionality specific to the aspects of X.509 certificates and their mapping to certificates of our framework. Adaptation layers can be constructed for any kind of usage, such as group signature schemes, traceable signature schemes, or electronic cash systems.

Mapping of Attributes. A property of X.509 certificates is that they consist of a hierarchic structure of typed fields, the types being defined in ASN.1 [31]. Through object identifiers further (arbitrary) attribute types can be defined. A certificate is represented using a tag-length-value structure. The X.509 profile definitions define names for the fields which we use for referring to the fields.

The adaptation layer provides functionality for handling X.509-represented certificates. It is able to map X.509 attributes contained within the field hierarchy of a certificate to size-limited integer data items of certificates of our framework.

The mapping handles integer-like attributes such as ASN.1 *INTEGER* or *GeneralizedTime* types such that they are mapped to integers allowing the proofs of algebraic relations of our framework to be applied.

Non integer-like attributes are hashed to an integer data item limiting the applicable proofs on the attribute to proofing equality and inequality. This approach allows for an arbitrarily large attribute to be compressed into one data item that can be treated as individual data item.

As an example, consider the mapping of the *GeneralizedTime* type used in X.509 which is a restricted ASN.1 *GeneralizedTime* format. An attribute of this type has a format defining year, day, hour, minute, and second followed by a specifier “Z” defining a zero offset to UTC time. We map such an attribute to an integer representing the date and time in seconds. This allows for proving time-wise validity of a certificate without revealing the validity interval.

Providing the Certificate Structure. In PKI-like use of certificates, they are sent thus also conveying the full field structure of the certificate. This involves the data type for an attribute and handling of multi-valued attributes. Our certificate framework that the extensions are based on only supports a flat attribute model of integer attributes without further structural information being contained in a certificate.

Our approach is to encode the full structural information of a certificate as an additional data item in the underlying certificate. The structure need not be stored as an X.509 field as it can be derived from the X.509 certificate representation. In a proof protocol the derived structure is communicated automatically to the verifier and it is proved that the data item this information is mapped to is a hash of this information conforming to the mapping.

We require that all certificates issued with a particular key have the same structure, i.e., individual certificates should not have attributes others don’t have. This is required in order to not reduce the anonymity set.

Despite this, it can be required that an attribute be optionally certified. In case of a non-integer attribute, this can be easily accomplished by an appropriate encoding, in case of an integer attribute an additional data item must be provided for specifying whether the attribute has been certified. Structural information regarding this must be provided in an additional field in the X.509 certificate.

Specification Language. As X.509 certificates allow for attributes that are typed and possibly multi-valued, we also provide issue and proof specification languages supporting this. For referring to fields and attributes, we use the names specified in the syntax definitions in the certificate profiles [30, 27]. We only present the adaptations of the proof specification language, the issuance specification language is adapted similarly. The reference to an attribute is to be changed from an index-based to a name-based notation. For example, the reference $cert_i[j]$ for some j becomes $cert_i[subject.country]$. An element in an ASN.1 SEQUENCE is addressed with an index $[i]$ into the sequence. An element of an ASN.1 SET is also referred to with

³Uniqueness is guaranteed among the certificates issued by a particular entity.

an index $[j]$.⁴ As an example, consider the *attributes* field of an attribute certificate. The specifier $cert_2.attributes[i][j]$ addresses the j -th attribute in the multi value set of the i -th multi-valued attribute in the sequence of attributes of $cert_2$.

In order to support optionally certified integer-like attributes, a language construct has to be provided for conveying to the verifier whether an attribute has been certified.

When executing a proof (or issuing a certificate) a proof (issue) specification on the X.509 layer must be mapped to a specification at the framework layer. This mapping is based on the certificate’s structural definition.

Certificate Revocation. Traditionally, certificate revocation is either avoided by issuing only short-lived certificates or is implemented via so-called certificate revocation lists (CRLs) made available by the issuer of the certificate [30]. The first method can clearly also be applied to anonymous credentials while the second is not directly applicable, as a naive implementation would allow one to link different *CertificateProof* transactions as the prover would provide some piece of information that the verifier can look-up in the CRL.

Luckily, the literature proposes ways to overcome this. The first one applies so-called cryptographic accumulators [10] that compress all valid certificates into a single value v and then allows a user to efficiently prove that her certificate is contained in that value. However, this approach also requires the user to update her certificate whenever a certificate is revoked and thus removed from the accumulated value. Another approach, and the one we adopt for our framework, is to include some unique serial number, say s , into a certificate that can be put on a CRL when it is revoked. Now, one requires the user to provide the commitment $u = g^s$, where g is a randomly chosen element of some suitable group and use it in the *CertificateProof* protocol to prove its correctness. Thus, one can figure out whether the (anonymous) certificate is revoked by checking whether $u = g^{s_i}$ for all s_i in the revocation list. This process can be considerably sped-up using batch-verification techniques. As concerns the issuance and proof protocols, this can be realized with our specification language. Moreover, on the X.509 protocol interface level, this specification is transparent.

4.2 Types of Certificates

We define public key certificates (PKCs) and attribute certificates analogously to the X.509 definitions, but enhance privacy when obtaining and using these certificates. In our presentation, we will abstract many of the fields and show only the ones that are of particular interest regarding a privacy-enhancing use of the certificates. We employ a simple notation for certificates similar to the one in Section 2, but without restricting attributes to integers. Attributes not interesting for the understanding of the concept are omitted.

Public Key Certificates. A public key certificate (PKC) based on a certificate of our framework contains the same attributes as an X.509 PKC. We do not use fully qualified names of attributes, but abbreviate them. For example, *sub* refers to the subject of a PKC, i.e., an ASN.1 distinguished name, *sno* to the certificate *serialNumber*, *PK* refers to the *subjectPublicKeyInfo.subjectPublicKey*, and *algo* to the algorithm identifier *subjectPublicKeyInfo.algorithm*. For simplicity of presentation, we aggregate the identity data which is a multi-valued attribute, into one attribute *sub*. The simplified structure of a PKC is presented in (16). The field *snoRefM* is explained further below.

$$PKCert(sub, sno, snoRefM, PK, algo, \dots) \quad (16)$$

An entity can obtain public key certificates (PKCs) on their civil identity or on pseudonymous identities. Such a PKC binds a public key of any type to the identity of the person or a pseudonym of her. Conceptually it is irrelevant whether the pseudonym is bound to the civil identity or to a pseudonym of the entity as it is only a binding of a public key to a name.

It is important that there be one certificate for an entity—the master PKC—that binds the master public key of the entity to her real identity. A master PKC is a PKC on a civil identity sub_U that involves the master public key PK_U of the holder. Any other certificate of a user can be bound to her master certificate and thus to the person. A proof of knowledge of the master certificate requires knowledge of the master private key. The user is highly motivated not to share the master private key with others as sharing would allow for certificate sharing. This master private key could, for example, be the user’s signature key. This approach for preventing certificate sharing follows the one put forth by Lysyanskaya et al. [35] for pseudonym systems. In addition to preventing certificate sharing, the master key is used in certificate proofs to link together PKCs and attribute certificates.

Example (17) describes a master PKC of a user containing concrete attributes. The first occurrence of $\#_U$ is the certificate number of the PKC, the next occurrence the reference *snoRefM* to the master PKC of the user, in this case to itself.

$$PKCert_{PK_U} = PKCert(sub_U, \#_U, \#_U, PK_U, \dots) \quad (17)$$

Other PKCs than the master PKC on the identity or on pseudonyms reference the user’s master PKC’s certificate number $\#_U$. This allows to link any PKC to its holder. This attribute requires an extension of the X.509 PKC profile as the concept of having PKCs on pseudonyms that are bound to the master PKC is not supported in the current X.509 proposals.

$$PKCert_{PK_{U,1}} = PKCert(sub_{U,1}, \#_{U,1}, \#_U, PK_{U,1}, \dots) \quad (18)$$

Example (18) shows a PKC on a pseudonymous identity $sub_{U,1}$. Such a PKC can be used similarly to a pseudonym in a pseudonym system. The attribute $sub_{U,1}$ is the pseudonymous identity of the user on which the PKC is issued, $\#_{U,1}$ is the certificate number, $\#_U$ the link to the master PKC, and $PK_{U,1}$ the public key of the pseudonym.

⁴A set can be indexed due to the canonical representation imposed by ASN.1 DER (distinguished encoding rules) encoding.

Attribute Certificates. The attribute certificates we define conform to the profile for X.509 attribute certificates [27] except for a minor extension requiring an additional attribute for including the certificate number $\#_U$ of the master PKC of the certificate holder.

An attribute certificate can be obtained on any either identity-bound or pseudonym-bound PKC. An attribute certificate is linked to the PKC it was obtained on by inclusion of the PKC’s serial number. This is comparable to the X.509 approach, but the information released to the issuer can be limited compared to classical X.509. A further field includes the reference $snoRefM$ to the master PKC. The fields $attr_1, \dots, attr_n$ are the attributes useable for application purposes. The general structure of an attribute certificate of our PKI extension is presented in (19).

$$AttrCert(sno, snoRef, snoRefM, \dots, attr_1, \dots, attr_n) \quad (19)$$

A concrete example of an attribute certificate issued on a PKC with serial number $\#_{U,1}$ and a master PKC with serial number $\#_U$ is given in 20. The index notation $U, 1, 1$ immediately shows that the certificate was issued on $PKC_{U,1}$.

$$AttrCert_{U,1,1} = AttrCert(\#_{U,1,1}, \#_{U,1}, \#_U, \dots, a_1, \dots, a_n) \quad (20)$$

In the case an attribute certificate is issued on the master PKC, the second and third fields both are the serial number $\#_U$ of the master PKC. This shows that all attribute certificates have the very same form regardless of whether they are issued on the master PKC or any other PKC of the entity.

Clearly, an arbitrary number of attribute certificates can be obtained on each PKC, be it an identity or pseudonym PKC.

4.3 Usage of the Certificates

Every user has to obtain a master PKC PKC_U on her identity, issued by a certification authority. Such a certificate binds the user’s master public key to her identity. Proving knowledge of the master PKC requires knowledge of the master private key. Thus we realize the binding of proofs of certificates to the holder of the certificates (by the master private key) thus preventing certificate sharing of non-identity-based certificates. This is a key aspect for non-identity certificates as for identity certificates the problem is tackled by the identity binding. A user can obtain further PKCs on her civil identity or on pseudonymous identities of her. On any of these PKCs multiple attribute certificates can be obtained. Each attribute certificate is linked to the PKC it was obtained on and in addition to the user’s master PKC as explained further above.

Our pPKI approach allows—depending on the privacy and compatibility requirements of a particular situation—different modes of operation. What is supported for a certificate in terms of privacy-enhancing capabilities is governed by whether conventional signature schemes or the advanced signature protocols are used for a certificate. In the most privacy-preserving mode the system is used much like a generalized pseudonym and credential system. In compatibility mode to the classical X.509 PKI it can be used much like the current approach within the Internet PKI.

Obtaining a Non-Master PKC. The process of obtaining a PKC that is not a master PKC requires knowledge of the master PKC PKC_U to be proved and a commitment being made to the number $\#_U$. The issuing of the new PKC includes $\#_U$ via the commitment. This does not reveal any information on $\#_U$ to the issuer, but still allows to integrate the link to the master PKC into the PKC to be issued.

For PKCs it can be useful that an issuer requires certain attributes to be proved by the receiver before issuing the PKC. This makes pseudonymous PKCs applicable for being used with multiple parties which is an extension to pseudonym systems.

Obtaining an Attribute Certificate. When obtaining an attribute certificate on any PKC_i , the certificate numbers $\#_U$ of the master PKC and $\#_{U,i}$ of PKC_i are included in the attribute certificate using commitments. We note that this is done using the same ideas as for obtaining a non-master PKC.

Performing Proofs over Certificates. In classical Internet PKI, a PKC and an attribute certificate issued on it are revealed together to proof ownership of the associated attributes. In our approach certificates can be used in a proof without sending them, but with just proving knowledge of them and selectively disclosing certified data in plain or encrypted form. The proofs are based on the general constructions allowed by our framework of Section 2.

When knowledge of an attribute certificate is to be proven, partial information of it can be conveyed in the proof. A proof of an attribute certificate is performed together with a proof of the master PKC in order to bind the proof to the owner of the certificates. The difference to conventional PKI is that no information on the identity of the user has to be disclosed in order to reveal certified attributes. Multiple proofs involving the same certificates are unlinkable, so is the proof to the issuing of any of the involved certificates.

When performing proofs over multiple certificates, an appropriately specified proof is executed. In particular, the certificates have to be linked to the master certificate via the serial numbers and references.

The example proof specification in (21) is the analogous example of the example in Section 2 using pPKI. The certificate $AttrCert_{U,1}$ is a US passport that is used as input, $AttrCert'_{U,1}$ is not provided as input.

$$\begin{aligned} \langle PKCert_U[sno] = AttrCert_{U,1}[snoRef] \wedge enc_1[1] = AttrCert_{U,1}[sno] \wedge enc_1[2] = \text{“valid”} \wedge enc_2[2] = \text{“invalid”} \rangle \vee \\ PKCert'_{U,1}[sno] = AttrCert'_{U,1}[snoRef] \wedge enc_2[1] = AttrCert'_{U,1}[sno] \wedge enc_2[2] = \text{“valid”} \wedge enc_1[2] = \text{“invalid”} \end{aligned} \quad (21)$$

Applications as the one in the example above show the power and privacy-enhancing capabilities of certificates based on our pPKI certificates.

Further Applications of Pseudonymous PKCs. A PKC on a pseudonymous identity can be viewed as a generalized concept of a pseudonym as defined for pseudonym systems. The PKC acts as a pseudonym that allows to authenticate the holder as the owner and it binds a public key to the pseudonym. Thus, a PKC can—in addition to the use as pseudonym with the party it was obtained from—be used for a pseudonym with multiple parties. A pseudonym that is useable with multiple parties allows new use cases such as establishing reputation with respect to the pseudonym. The binding of a public key to the pseudonym allows for signature operations. Liability can be limited to the pseudonym or can be propagated to the person by use of verifiable encryption.

Complementary Usage with the Current Internet PKI. PKCs used for binding the key to an identity as in X.509 are applicable as usual. This will be used for PKCs for certificate issuers and service providers. Certificate chain validation will in addition be required for the public keys that are used for verifying the signature on certificates. For these use cases identifiability by identity is not an issue, but necessary.

When a PKC or attribute certificate conforming to the usage in today’s deployed X.509 infrastructure is required to be provided for a particular purpose, a transferable image of the certificate can be created by the prover. This image can be used equivalently to a classical certificate, i.e., sent using already specified protocols. Only the new signature protocols need to be available for this. We note that if more evolved proofs are contained in the image such as algebraic relations over attributes of multiple certificates, such statements are not representable any more in X.509 certificates. This way of using the certificates compromise the privacy-enhancing mechanisms and are for compatibility only.

4.4 Deployment into existing X.509

Required Changes for an Adoption. In order to deploy our privacy-enhancing extensions to the Internet PKI, some main issues have to be standardized. One item involves the SRSA-CL and BM-CL signature schemes. The structures of PKCs, certificate and attribute certificates have to be extended. Protocols for obtaining and proving certificates based on the protocols for signing committed values have to be standardized.

Systems that make use of the new privacy-enhancing functionality of proving attributes must be adopted to support the general specification of release of certified data. This can be achieved by deploying parts of new privacy architectures.

The certificate revocation list profile [30] can remain unchanged and immediately support revocation for certificates of our framework. Furthermore, large parts of the infrastructure for registration and certification of users can remain unchanged as the certificate and attribute format remains mainly as it is. The relevance for the registration mechanisms to remain the same should not be underestimated.

Migration Path from PKI to pPKI. We think that the greatest chances to move from PKI to pPKI are given by a gradual approach. The phased approach below gives a rough idea on how this could be performed in practice.

At a first stage it would suffice to standardize only the SRSA-CL and/or BM-CL signature schemes. Certificates would be obtained as usual using the new signature schemes without protocols. Proofs would only involve single certificates and could support simple selective disclosure of attributes. Instead of executing *CertificateProof* protocol, a certificate image would be created and sent to the verifier. This stage would require only minor extensions to today’s infrastructure including code at the user’s side. The application of this could be access control to electronic content. This stage can be used for paving the way to more enhanced functionality in the further stages.

The next stage would involve the protocols for obtaining and proofing certificates in a privacy-enhanced way, and the extensions to the certificates. Due to a lack of a surrounding privacy architecture, the full functionality would not be widely used, but major privacy-protection would be offered compared to today’s situation.

In a third stage the deployment of components of a standard privacy architecture would provide the necessary framework for using the full functionality of the certificate extensions as presented in the paper.

Considering the generality of our approach we think that a privacy architecture and infrastructure, e.g., similar to what is being done in the European PRIME project, is required to fully support the certificate framework in an open environment.

5 Conclusion

We defined a general certificate framework for obtaining certificates and (partially) releasing certified data either in plain or encrypted form. In particular, our framework is a step to realize proactive privacy. The framework includes cryptographic primitives for realizing the functionality, definition of protocol interfaces for the *CertificateIssuance* and *CertificateProof* protocols and a powerful specification language with well-defined semantics that allows to define what data to release in a transaction. We briefly described our implementation of the framework that makes use of a zero-knowledge protocol compiler to handle the complexity of the proofs required for our protocols. Finally, we applied our framework to define an extension to the Internet certificate infrastructure that adds privacy functionality thus being interesting for end users.

Our framework goes far beyond what has previously been available in terms of comprehensiveness and generality of the proofs. We allow for multi-show unlinkability and predicates defining polynomial relations between data items of multiple certificates, committed values, and encryptions. Proof specifications can link predicates in boolean formulas connected with logical ANDs and ORs. We allow for verifiable encryptions to be contained in the proofs in a way that opens up the opportunity for completely new applications as presented in our examples by proving OR-linked statements over encryptions. The powerful proof mechanisms are made easily accessible by an intuitive specification language. The language allows to precisely express what to release in a particular protocol instance. Our protocol interface definition and specification language are close to an API level. Considering our achievements, the framework allows to perform highly privacy-protecting release

of data that is interesting for practical application in electronic media such as the Internet in the near future. We have implemented a subset of our framework with restricted proof functionality. Work is ongoing and the final version will allow for the full power of proofs as defined in this paper. We implemented on-the-fly generating of the required zero-knowledge proof protocols.

Finally, we construct a privacy-enhancing PKI from our framework that integrates smoothly with the Internet PKI. The proposed approach of gradually deploying our framework within existing X.509 infrastructures is the most feasible way of deploying privacy-enhancing techniques on the Internet within the next few years.

Our contributions can pave the way to deploying long-needed privacy technology within existing Internet infrastructures, thus increasing the chance of an adoption through the community.

References

- [1] ADAMS, C., AND FARRELL, S. Internet X.509 Public Key Infrastructure Certificate Management Protocols. RFC 2510 (Proposed Standard), Mar. 1999.
- [2] BANGERTER, E., CAMENISCH, J., AND LYSYANSKAYA, A. A cryptographic framework for the controlled release of certified data. In *Twelfth International Workshop on Security Protocols 2004* (2004), LNCS, Springer Verlag.
- [3] BASSHAM, L., POLK, W., AND HOUSLEY, R. Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3279 (Proposed Standard), Apr. 2002.
- [4] BOUDOT, F. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology — EUROCRYPT 2000* (2000), B. Preneel, Ed., vol. 1807 of LNCS, Springer Verlag, pp. 431–444.
- [5] BRANDS, S. Rapid demonstration of linear relations connected by boolean operators. In *Advances in Cryptology — EUROCRYPT '97* (1997), W. Fumy, Ed., vol. 1233 of LNCS, Springer Verlag, pp. 318–333.
- [6] BRANDS, S. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.
- [7] BRICKELL, E., CAMENISCH, J., AND CHEN, L. Direct anonymous attestation. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security* (New York, NY, USA, 2004), ACM Press, pp. 132–145.
- [8] BRICKELL, E. F., GORDON, D. M., MCCURLEY, K. S., AND WILSON, D. B. Fast exponentiation with precomputation (extended abstract). In *EUROCRYPT* (1993), R. A. Rueppel, Ed., vol. 658 of *Lecture Notes in Computer Science*, Springer, pp. 200–207.
- [9] CAMENISCH, J., AND DAMGÅRD, I. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In *Advances in Cryptology — ASIACRYPT 2000* (2000), T. Okamoto, Ed., vol. 1976 of LNCS, Springer Verlag, pp. 331–345.
- [10] CAMENISCH, J., AND LYSYANSKAYA, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — CRYPTO 2002* (2002), M. Yung, Ed., vol. 2442 of LNCS, Springer Verlag, pp. 61–76.
- [11] CAMENISCH, J., AND LYSYANSKAYA, A. A signature scheme with efficient protocols. In *Third Conference on Security in Communication Networks* (2002), G. Persiano, Ed., vol. 2576 of LNCS, Springer Verlag, pp. 274–295.
- [12] CAMENISCH, J., AND LYSYANSKAYA, A. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO 2004 (to appear)* (2004), LNCS, Springer Verlag.
- [13] CAMENISCH, J., AND MICHELS, M. Proving in zero-knowledge that a number n is the product of two safe primes. In *Advances in Cryptology — EUROCRYPT '99* (1999), J. Stern, Ed., vol. 1592 of LNCS, Springer Verlag, pp. 107–122.
- [14] CAMENISCH, J., AND MICHELS, M. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology — CRYPTO '99* (1999), M. Wiener, Ed., vol. 1666 of LNCS, Springer Verlag, pp. 413–430.
- [15] CAMENISCH, J., AND SHOUP, V. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology — CRYPTO 2003* (2003), D. Boneh, Ed., LNCS.
- [16] CAMENISCH, J. L. *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.
- [17] CHAN, A., FRANKEL, Y., AND TSIOUNIS, Y. Easy come – easy go divisible cash. In *Advances in Cryptology — EUROCRYPT '98* (1998), K. Nyberg, Ed., vol. 1403 of LNCS, Springer Verlag, pp. 561–575.
- [18] CHAUM, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24, 2 (Feb. 1981), 84–88.
- [19] CHAUM, D. Blind signature systems. In *Advances in Cryptology — CRYPTO '83* (1984), D. Chaum, Ed., Plenum Press, p. 153.

- [20] CHAUM, D. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM* 28, 10 (Oct. 1985), 1030–1044.
- [21] CHAUM, D., AND EVERTSE, J.-H. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Advances in Cryptology — CRYPTO '86* (1987), M. Odlyzko, Ed., vol. 263 of *LNCS*, Springer-Verlag, pp. 118–167.
- [22] CHAUM, D., AND PEDERSEN, T. P. Wallet databases with observers. In *Advances in Cryptology — CRYPTO '92* (1993), E. F. Brickell, Ed., vol. 740 of *LNCS*, Springer-Verlag, pp. 89–105.
- [23] CHAUM, D., AND VAN HEYST, E. Group signatures. In *Advances in Cryptology — EUROCRYPT '91* (1991), D. W. Davies, Ed., vol. 547 of *LNCS*, Springer-Verlag, pp. 257–265.
- [24] CRAMER, R., DAMGÅRD, I., AND SCHOENMAKERS, B. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — CRYPTO '94* (1994), Y. G. Desmedt, Ed., vol. 839 of *LNCS*, Springer Verlag, pp. 174–187.
- [25] DAMGÅRD, I. Efficient concurrent zero-knowledge in the auxiliary string model. In *Advances in Cryptology — EUROCRYPT 2000* (2000), B. Preneel, Ed., vol. 1807 of *LNCS*, Springer Verlag, pp. 431–444.
- [26] DAMGÅRD, I., AND FUJISAKI, E. An integer commitment scheme based on groups with hidden order. In *Advances in Cryptology — ASIACRYPT 2002* (2002), vol. 2501 of *LNCS*, Springer.
- [27] FARRELL, S., AND HOUSLEY, R. An Internet Attribute Certificate Profile for Authorization. RFC 3281 (Proposed Standard), Apr. 2002.
- [28] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — CRYPTO '86* (1987), A. M. Odlyzko, Ed., vol. 263 of *LNCS*, Springer Verlag, pp. 186–194.
- [29] FUJISAKI, E., AND OKAMOTO, T. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology — CRYPTO '97* (1997), B. Kaliski, Ed., vol. 1294 of *LNCS*, Springer Verlag, pp. 16–30.
- [30] HOUSLEY, R., POLK, W., FORD, W., AND SOLO, D. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280 (Proposed Standard), Apr. 2002.
- [31] ITU-T. Itu-t recommendation x.660 information technology - asn.1 encoding rules: Specification of basic encoding rules (ber), canonical encoding rules (cer) and distinguished encoding rules (der), 1997.
- [32] KIAYIAS, A., TSIOUNIS, Y., AND YUNG, M. Traceable signatures. In *EUROCRYPT* (2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 571–589.
- [33] KILIAN, J., AND PETRANK, E. Identity escrow. Theory of Cryptography Library, Record Nr. 97-11, <http://theory.lcs.mit.edu/~tcrypt01>, Aug. 1997.
- [34] LIM, C. H., AND LEE, P. J. More flexible exponentiation with precomputation. In *CRYPTO* (1994), Y. Desmedt, Ed., vol. 839 of *Lecture Notes in Computer Science*, Springer, pp. 95–107.
- [35] LYSYANSKAYA, A., RIVEST, R., SAHAI, A., AND WOLF, S. Pseudonym systems. In *Selected Areas in Cryptography* (1999), H. Heys and C. Adams, Eds., vol. 1758 of *LNCS*, Springer Verlag.
- [36] MILLEN, J., AND MULLER, F. Cryptographic protocol generation from CAPSL. Tech. Rep. SRI-CSL-01-07, SRI International, December 2001.
- [37] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186, May 1994.
- [38] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. Secure Hash Standard. Federal Information Processing Standards Publication 180-2, August 2002.
- [39] PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — CRYPTO '91* (1992), J. Feigenbaum, Ed., vol. 576 of *LNCS*, Springer Verlag, pp. 129–140.
- [40] POINTCHEVAL, D., AND STERN, J. Security proofs for signature schemes. In *Advances in Cryptology — EUROCRYPT '96* (1996), U. Maurer, Ed., vol. 1070 of *LNCS*, Springer Verlag, pp. 387–398.
- [41] PORTIA project. crypto.stanford.edu/portia.
- [42] PRIME project. www.prime-project.eu.org.
- [43] RIVEST, R., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 2 (Feb. 1978), 120–126.
- [44] SCHNORR, C. P. Efficient signature generation for smart cards. *Journal of Cryptology* 4, 3 (1991), 239–252.
- [45] TSOUKALIDIS, I., AND WEIS, S. Cryptographic protocol language. <http://theory.lcs.mit.edu/~cis/cpl/index.html>, 2004.