An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation

Jan Camenisch
IBM Research
Zurich Research Laboratory
CH–8803 Rüschlikon
jca@zurich.ibm.com

Anna Lysyanskaya*
MIT LCS
545 Technology Square
Cambridge, MA 02139 USA
anna@theory.lcs.mit.edu

Abstract. A credential system is a system in which users can obtain credentials from organizations and demonstrate possession of these credentials. Such a system is anonymous when transactions carried out by the same user cannot be linked. An anonymous credential system is of significant practical relevance because it is the best means of providing privacy for users. In this paper we propose a practical anonymous credential system that is based on the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a safe prime product and is considerably superior to existing ones: (1) We give the first practical solution that allows a user to unlinkably demonstrate possession of a credential as many times as necessary without involving the issuing organization. (2) To prevent misuse of anonymity, our scheme is the first to offer optional anonymity revocation for particular transactions. (3) Our scheme offers separability: all organizations can choose their cryptographic keys independently of each other. Moreover, we suggest more effective means of preventing users from sharing their credentials, by introducing *all-or-nothing* sharing: a user who allows a friend to use one of her credentials once, gives him the ability to use all of her credentials, i.e., taking over her identity. This is implemented by a new primitive, called *circular encryption*, which is of independent interest, and can be realized from any semantically secure cryptosystem in the random oracle model.
Keywords. Privacy protection, credential system, pseudonym system, e-cash, blind signatures, circular encryption, key-oblivious encryption.

1 Introduction

As information becomes increasingly accessible, protecting the privacy of individuals becomes a more challenging task. To solve this problem, an application that allows the individual to control the dissemination of personal information is needed. An anonymous credential system (also called pseudonym system), introduced by Chaum [18], is the best known idea for such a system. In this paper, we propose a new efficient anonymous credential system, considerably superior to previously proposed ones. The communication and computation costs of our solution are small, thus introducing almost no overhead to realizing privacy in a credential system.

* This research was carried out while the author was visiting IBM Zürich Research Laboratory.

An anonymous credential system [18, 19, 21, 25, 34] consists of users and organizations. Organizations know the users only by pseudonyms. Different pseudonyms of the same user cannot be linked. Yet, an organization can issue a credential to a pseudonym, and the corresponding user can prove possession of this credential to another organization (who knows her by a different pseudonym), without revealing anything more than the fact that she owns such a credential. Credentials can be for unlimited use (these are called *multiple-show* credentials) and for one-time use (these are called *one-show* credentials). Possession of a multi-show credential can be demonstrated an arbitrary number of times; these demonstrations cannot be linked to each other.

Basic desirable properties. It should be impossible to forge a credential for a user, even if users and other organizations team up and launch an adaptive attack on the organization. Each pseudonym and credential must belong to some well-defined user [34]. In particular, it should not be possible for different users to team up and show some of their credentials to an organization and obtain a credential for one of them that that user alone would not have gotten. Systems where this is not possible are said to have *consistency of credentials*. As organizations are autonomous entities, it is desirable that they be separable, i.e., be able to choose their keys themselves and independently of other entities, so as to ensure security of these keys and facilitate the system's key management.

The scheme should also provide user privacy. An organization cannot find out anything about a user, apart from the fact of the user's ownership of some set of credentials, even if it cooperates with other organizations. In particular, two pseudonyms belonging to the same user cannot be linked [8, 18, 19, 21, 25, 34].

Finally, it is desirable that the system be efficient. Besides requiring that it be based on efficient protocols, we also require that each interaction involve as few entities as possible, and the rounds and amount of communication be minimal. In particular, if a user has a multiple-show credential from some organization, she ought to be able to demonstrate it without getting the organization to reissue credentials each time.

Additional desirable properties. It is an important additional requirement that the users should be discouraged from sharing their pseudonyms and credentials with other users. The previously known way of discouraging the user from doing this was by *PKI-assured non-transferability*. That is, sharing a credential implies also sharing a particular, valuable secret key from *outside* the system (e.g., the secret key that gives access to the user's bank account) [26, 32, 34]. However, such a valuable key does not always exist. Thus we introduce an alternative, novel way of achieving this: *all-or-nothing* non-transferability. Here, sharing just one pseudonym or credential implies sharing all of the user's other credentials and pseudonyms in the system, i.e., sharing *all* of the user's secret keys *inside* the system. These two methods of guaranteeing non-transferability are different: neither implies the other, and both are desirable and can in fact be combined.

In addition, it may be desirable to have a mechanism for discovering the identity of a user whose transactions are illegal (this feature, called *global anonymity revocation*, is optional); or reveal a user's pseudonym with an issuing organization in case the user misuses her credential (this feature, called *local anonymity revocation*, is also optional). It can also be beneficial to allow *one-show* credentials, i.e., credentials that should only be usable once and should incorporate an *off-line* double-spending test. It should be possible to encode attributes, such as expiration dates, into a credential.

Related work. The scenario with multiple users who, while remaining anonymous to the organizations, manage to transfer credentials from one organization to another, was first introduced by Chaum [18]. Subsequently, Chaum and Evertse [19] proposed a solution that is based on the existence of a semi-trusted third party who is involved in all transactions. However, the involvement of a semi-trusted third party is undesirable.

The scheme later proposed by Damgård [25] employs general complexity-theoretic primitives (one-way functions and zero-knowledge proofs) and is therefore not applicable for practical use. Moreover, it does not protect organizations against colluding users. The scheme proposed by Chen [21] is based on discrete-logarithm-based blind signatures. It is efficient but does not address the problem of colluding users. Another drawback of her scheme and the other practical schemes previously proposed is that to use a credential several times, a user needs to obtain several signatures from the issuing organization.

Lysyanskaya, Rivest, Sahai, and Wolf [34] propose a general credential system. While their general solution captures many of the desirable properties, it is not usable in practice because their constructions are based on one-way functions and general zero-knowledge proofs. Their practical construction, based on a non-standard discrete-logarithm-based assumption, has the same problem as the one due to Chen [21]: a user needs to obtain several signatures from the issuing organization in order to use unlinkably a credential several times.

Other related work is that of Brands [8] who provides a certificate system in which a user has control over what is known about the attributes of a pseudonym. Although a credential system with one-show credentials can be inferred from his framework, obtaining a credential system with multi-show credentials is not immediate and may in fact be impossible in practice. Another inconvenience of these and the other discrete-logarithm-based schemes mentioned above is that all the users and the certification authorities in these schemes need to share the same discrete logarithm group.

The concept of revocable anonymity is found in electronic payment systems (e.g., [9, 37]) and group signature and identity escrow (e.g., [2, 14, 20, 33] schemes.

Prior to our work, the problem of constructing a practical system with multiple-use credentials eluded researchers for some time [8, 21, 25, 34]. We solve it by extending ideas found in the constructions of strong-RSA-based signature schemes [23, 30] and group signature schemes [2].

Our contribution. In Section 2 we present our definitions for a credential system with the basic properties. Although not conceptually new and inspired by the literature on multi-party computation [15, 16] and reactive systems [36], these definitions are of interest, as our treatment is more formal than the one usually encountered in the literature on credential and electronic cash systems. We omit formal definitions for a credential system satisfying the additional desirable properties and instead refer the reader to the full version of this paper [12].

Our basic credential system, presented in Section 4, provably satisfies the basic properties listed above under the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a strong prime product. Our basic solution is practical. When using an RSA modulus n of 1024 bits, a credential-pseudonym pair is about 4K bits, and the most expensive operation of proving possession of a credential requires about 22 exponentiations in $\mathbb{Z}_n^*$ for both parties and can be done in three rounds.

Our extended credential system, presented in Sections 6 and 7, describes how to incorporate additional desirable properties into the basic credential system. These are also efficient, except the one with all-or-nothing non-transferability: when using RSA moduli of length 1024 bits, establishing a pseudonym is somewhat less efficient: it takes about 200 exponentiations in $\mathbb{Z}_n^*$ for both parties, but batch-verification techniques [4] could be applied to reduce this, and organizations have to store about 25K bits per user (here computation complexity could be traded against storage).

All-or-nothing non-transferability is based on a new primitive we call *circular encryption*, discussed in Section 6.1; we implement this primitive in the random oracle model; it is a challenging open problem whether this primitive can be realized outside the random oracle model.

This work is the first to introduce one-show credentials with an off-line double-spending test, similar to known e-cash schemes (in fact, our one-show credentials can be used as anonymous coins). These one-show credentials are described in Section 7.1. More precisely, our double-spending test mechanism together with the all-or-nothing property ensures that, if a user presents such a credential more than once, then the verifying entity gets the ability to demonstrate possession of all the pseudonyms and credentials of that user — a strong incentive to users not to double-spend. From a technical point of view, it might be interesting that the anonymity of our one-show credentials is not obtained by blind signatures but by an alternative mechanism.

Another innovation of this work is the possibility of anonymity revocation, described in Section 7.2. We stress that this feature is entirely optional. Moreover, for each transaction, the user has the freedom of specifying under which conditions the anonymity can be revoked (maybe subject to conditions of the other parties involved in the transaction). The user may also choose unconditional anonymity, and then his identity will not be retrievable under any circumstances. Yet another innovation is separability for organizations.

2 Formal Definitions and Requirements

A basic credential system has *users*, *organizations*, and *verifiers* as types of players. Users are entities that receive credentials. The set of users in the system may grow over time. Organizations are entities that grant and verify the credentials of the users. Each organization grants a unique (for simplicity of exposition) type of credential. Finally, verifiers are entities that verify credentials of the users.

Variations of such a system allow a single organization to issue different types of credentials. For the purposes of non-transferability, we can add a CA to the model who verifies that the users entering the system possess an external public and secret key. This CA will be trusted to do his job properly. For PKI-assured non-transferability, this will make sure that access to a user's pseudonym or credential is sufficient to obtain this user's secret key from an external PKI. For all-or-nothing non-transferability, this will make sure that access to one pseudonym or credential of a user is sufficient to obtain access to all of them. To allow revocable anonymity, an anonymity revocation manager can be added. This entity will be trusted not to use his ability to find out a user's identity or pseudonym unless dictated to do so. As the trusted parties perform tasks that are not required frequently, these parties can be implemented in a distributed fashion to weaken the trust assumptions. Finally, a credential may include an attribute, such as an expiration date. These variations are simple to handle in the model; for simplicity of exposition of the model, however, we do not discuss them here. The extended solution we propose already incorporates some of them, and can be easily adapted to incorporate others.

We first give a specification for an ideal credential system that relies on a trusted party T as an intermediator; we then explain what it means for a cryptographic system to conform to this specification.

Initialization: To initialize the system, the organizations create a public file which, for each organization O, describes the type of credential that the organization grants.

Ideal communication: All communication is routed through T. If the sender of a message wishes to be anonymous, he requests T not to reveal his identity to the recipient. Also, a sender of a message may request that a session be established between him and the recipient. This session then gets a session id *sid*.

Events in the system: Each transaction between players is an event in the system. Events in the system can be triggered through external processes, some of which may be controlled by an adversary. An external process can trigger some particular event between a particular user and organization; or may trigger a set of events; or may cause some probability distribution on the events.

Input of the players: The players are interactive Turing machines. Initially, they have no input; then as transactions are triggered, they obtain inputs and act accordingly.

Output of the players: In the end of the system's lifetime, each user outputs a list of the transactions she participated in, complete with the pseudonym used in each transaction, session ids of the transactions, and transaction outcomes. Organizations and verifiers output a list of transaction identifiers for transactions in which they participated, the pseudonym involved (in case of organizations), and the outcome of the transaction.

The system supports the following transactions:

$\mathsf{FormNym}(U, O)$: This protocol is a session between a user U and an organization O. The user U contacts T with a request to establish a pseudonym between herself and organization O. She further specifies the login name L_U by which T knows her and the corresponding authenticating key K_U. If she does not have an account with T yet, she first establishes it by providing to T a login name L_U and obtaining K_U in return. She further specifies N_1. Then T verifies the validity of (L_U, K_U) and, if K_U is the authenticating key corresponding to login name L_U, contacts O and tells it that some user wants to establish a pseudonym with it with prefix N_1. The organization either accepts or rejects. If it accepts, it sends a pseudonym suffix N_2, so the pseudonym becomes $N_{(U,O)} := N_1 \| N_2$ ($\|$ denotes concatenation). T forwards the resulting $N_{(U,O)}$ to U in case of acceptance, or notifies U of rejection.

$\mathsf{GrantCred}(N, O)$: This protocol is a session between a user U and an organization O. U approaches T, and submits her login name L_U, her authenticating key K_U, the pseudonym N, and the name of organization O. If K_U is not a valid authenticating key for L_U, or if N is not U's pseudonym with O, then T replies with a "Fail" message. Otherwise, T contacts O. If O accepts, then T notifies the user that a credential has been granted, otherwise it replies with "Reject."

$\mathsf{VerifyCred}(V, N, O)$: This protocol is a session between a user U and a verifier V. A user approaches T and gives it her login L_U, her authenticating key K_U, a name of the verifier V, a pseudonym N and a name of a credential-granting organization O. If K_U is a valid authenticating key for L_U, and N is U's pseudonym with organization O, and a credential has been granted by O to N, then T notifies V that the user talking to V in the current session *sid* has a credential from O. Otherwise, T replies with a "Fail" message.

$\mathsf{VerifyCredOnNym}(V, N_V, N_O, O)$: This protocol is a session between a user U and an verifier V. U approaches T and gives it her login name L_U, her authenticating key K_U, a name of the verifier V, pseudonyms N_V and N_O, and a name of a credential-granting organization O. If K_U is a valid authenticating key for L_U, and N_V is U's pseudonym with V while N_O is U's pseudonym with organization O, and a credential has been granted by O to N_O, then T notifies V that the user with pseudonym N_V has a credential from O.

This ideal system captures the intuitive requirements, such as unforgeability of credentials, anonymity of users, unlinkability of credential showings, and consistency of credentials. Ideal operations that allow additional desirable features can be implemented as well.

Let us briefly illustrate the use of the credential system by a typical example. Consider a user U who wants to get a credential from organization O.

Organization O requires the possession of credentials from organizations O_1 and O_2 as a prerequisite to get a credential from O. Assume that U possesses such credentials. Then U can get a credential from O as follows: she first establishes a pseudonym with O by executing $\mathsf{FormNym}(U, O)$ and then shows O her credentials from O_1 and O_2 be executing $\mathsf{VerifyCredOnNym}(O, N_O, N_{O_1}, O_1)$ and $\mathsf{VerifyCredOnNym}(O, N_O, N_{O_2}, O_2)$. Now O knows that the user it knows under N_O possesses credentials from O_1 and O_2 and will grant U a credential, i.e., U can execute $\mathsf{GrantCred}(N_O, O)$. We remark that the operation $\mathsf{VerifyCred}(V, N, O)$ exists for efficiency reasons. This operation can be used by U if she wants to show a party only a single credential, e.g., to access a subscription-based service.

The ideal-world (resp., real-world) adversary. The ideal-world (resp., real-world) adversary is a probabilistic polynomial-time machine that gets control over the corrupted parties in the ideal world (resp., real world). He receives, as input, the number of honest users and organizations, as well as all the public information of the system. The adversary can trigger an event as described above.

Definition 1. *Let the ideal credential system described above be denoted ICS. Let a cryptographic credential system without T be denoted CCS. Let $V = \texttt{poly}(k)$ be the number of players in the system with security parameter k. By ICS$(1^k, E)$ (resp., CCS$(1^k, E)$) we denote a credential system with security parameter k and event scheduler E for the events that take place in this system. As events are scheduled adversarially, E schedules them according to the adversary's wishes, therefore we will write $E^{\mathcal{A}}$. By $Z_i(1^k)$, we denote the output of party i in the credential system. If $\{A_1(1^k), \ldots, A_V(1^k)\}$ is a list of the players' outputs, then we denote these players' outputs by $\{A_1(1^k), \ldots, A_l(1^k)\}^{CS(1^k,E)}$ when all of them, together, exist within a credential system CS. CCS is secure if there exists a simulator $\mathcal{S}$ (ideal-world adversary) such that the following holds, for all interactive probabilistic polynomial-time machines $\mathcal{A}$ (real-world adversary), for all sufficiently large k:*

1. In the ICS, $\mathcal{S}$ controls the ideal-world players corresponding to the real-world players controlled by $\mathcal{A}$.

2. For all event schedulers $E^{\mathcal{A}}$

$$\{\{Z_i(1^k)\}_{i=1}^{V}, \mathcal{A}(1^k)\}^{CCS(1^k,E)} \stackrel{c}{\approx} \{\{Z_i(1^k)\}_{i=1}^{V}, \mathcal{S}^{\mathcal{A}}(1^k)\}^{ICS(1^k,E)} ,$$

where $\mathcal{S}$ is given black-box access to $\mathcal{A}$. ("$D_1(1^k) \stackrel{c}{\approx} D_2(1^k)$" denotes computational indistinguishability of the distributions D_1 and D_2.)

3 Protocol Notation

By $\texttt{neg}(k)$ we denote any function that vanishes faster than any inverse polynomial in k. By $\texttt{poly}(k)$ we denote a function bounded by a polynomial in k.

In the description of our scheme, we use the notation introduced by Camenisch and Stadler[14] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

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

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* $v < \alpha < u$," 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.

In the random oracle model, such protocols can be turned into signature schemes using the Fiat-Shamir heuristic [28]. We use the notation $SPK\{(\alpha) : y = g^{\alpha}\}(m)$ to denote a signature obtained in this way.

It is important that we use protocols that are *concurrent* zero-knowledge. They are characterized by remaining zero-knowledge even if several instances of the same protocol are run arbitrarily interleaved. In the public key model, Damgård [24] shows a general technique for making the so-called Σ-protocols (these include all the proofs of knowledge used here) composable under concurrent composition without incurring a penalty in communication or round complexity. All the proofs of knowledge we use in this paper incorporate this technique.

In this paper we apply such PK's and SPK's to the group of quadratic residues modulo a composite n, i.e., $G = QR_n$. This choice for the underlying group has some consequences. First, the protocols are proofs of knowledge under the strong RSA assumption [29]. Second, the largest possible value of the challenge c must be smaller that the smallest factor of G's order. Third, soundness needs special attention in the case that the verifier is not equipped with the factorization of n because then deciding membership in QR_n is believed to be hard. Thus the prover needs to convince the verifier that the elements he presents are indeed quadratic residues, i.e., that the square roots of the presented elements exist. This can in principle be done with a protocol by Fiat and Shamir [28]. However, often it is sufficient to simply execute $PK\{(\alpha) : y^2 = (g^2)^{\alpha}\}$ instead of $PK\{(\alpha) : y = g^{\alpha}\}$. The quantity α is defined as $\log_{g^2} y^2$, which is the same as $\log_g y$ in case y is in QR_n.

For the an explanation of how the PK's used in the paper can be realized efficiently, we refer to the full version of this paper [12].

4 The Basic Anonymous Credential System

The basic system comprises protocols for a user to join the system, register with an organization, obtain multi-show credentials, and show such credentials.

Throughout we assume that the users and organizations are connected by perfectly anonymous channels. Furthermore, we assume that for each protocol an organization authenticates itself to the user and that they establish a secure channel between them for each session. For any protocol we describe, we implicitly assume that if some check or sub-protocol (e.g., some proof of knowledge PK) fails for some party, it informs the other participants of this and stops.

4.1 High-Level Description

In our system, each organization O will have, in its public key PK_O, an RSA modulus n_O, and five elements of QR_{n_O}: $(a_O, b_O, d_O, g_O, h_O)$. Each user U will have her own master secret key x_U. A pseudonym of user U with organization O, denoted $N_{(U,O)}$, is just a name by which the user is known to the organization, and consists of a user-generated part N_1 and an organization-generated part N_2. The pseudonym $N_{(U,O)} = N_1 \| N_2$ will be tagged with a value $P_{(U,O)}$. This *validating tag* is of the form $P_{(U,O)} = a_O^{x_U} b_O^{s_{(U,O)}}$, where $s_{(U,O)}$ is a short random string to which the user and organization contribute randomness, but of which only the user knows its value. An appropriate choice of parameters for the length of x_U and $s_{(U,O)}$ ensures that the resulting $P_{(U,O)}$ is statistically independent of the user's key x_U and of any other validating tags formed by the same user with other organizations.

A credential issued by O to a pseudonym $N_{(U,O)}$ is a tuple $(e_{(U,O)}, c_{(U,O)})$ where $e_{(U,O)}$ is a sufficiently long prime and $c_{(U,O)}{}^{e_{(U,O)}} \equiv P_{(U,O)} d_O$. Under the strong RSA assumption, such tuples cannot be existentially forged for correctly formed tags even by an adaptive attack (Theorem 2).

To protect the user's privacy in our system, proof of possession of a credential is realized by a proof of knowledge of a correctly formed tag $P_{(U,O)}$ and a credential on it. This is done by publishing statistically secure commitments to both the validating tag and the credential, and proving relationships between these commitments. It can also include a proof that the underlying secret key is the same in both the committed validating tag (corresponding to the pseudonym formed with the issuing organization) and the validating tag with the verifying organization. This ensures consistency of credentials, e.g., guarantees that even users that fully trust each other cannot pool their credentials.

4.2 System Parameter and Key Generation.

We name some common system parameters: the length of all the RSA moduli ℓ_n, the integer intervals $\Gamma =]-2^{\ell_\Gamma}, 2^{\ell_\Gamma}[$, $\Delta =]-2^{\ell_\Delta}, 2^{\ell_\Delta}[$, $\Lambda =]2^{\ell_\Lambda}, 2^{\ell_\Lambda+\ell_\Sigma}[$ such that $\ell_\Delta = \epsilon(\ell_\Lambda+\ell_n)+1$, where $\epsilon > 1$ is a security parameter, and $\ell_\Lambda > \ell_\Sigma+\ell_\Delta+4$.

Each organization O_i chooses random $\ell_n/2$-bit primes p'_{O_i}, q'_{O_i} such that $p_{O_i} = 2p'_{O_i} + 1$ and $q_{O_i} = 2q'_{O_i} + 1$ are prime and sets modulus $n_{O_i} = p_{O_i} q_{O_i}$. It also chooses random elements $a_{O_i}, b_{O_i}, d_{O_i}, g_{O_i}, h_{O_i} \in QR_{n_{O_i}}$. It stores $SK_{O_i} := (p_{O_i}, q_{O_i})$ as its secret key and publishes $PK_{O_i} := (n_{O_i}, a_{O_i}, b_{O_i}, d_{O_i}, g_{O_i}, h_{O_i})$ as its public key. In the public-key model, we assume that there is a special entity that verifies, through a zero-knowledge protocol with O_i, that n_{O_i} is the product of two safe primes (see [13] for how this can be done efficiently) and that the elements $a_{O_i}, b_{O_i}, d_{O_i}, g_{O_i}, h_{O_i}$ are indeed in $QR_{n_{O_i}}$ (see, for example, Goldwasser et al. [31]). Alternatively, this can be carried out in the random oracle model using the Fiat-Shamir heuristic [28]. The parameter ℓ_Λ should be chosen such that computing discrete logarithms in $QR_{n_{O_i}}$ with ℓ_Λ-bits exponents is hard.

4.3 Generation of a Pseudonym

We now describe how a user U establishes a pseudonym $N_{(U,O)}$ and its validating tag $P_{(U,O)}$ with organization O. Let $x_U \in \Gamma$ be U's master secret. The protocol below assures that the pseudonym's validating tag is of the right form, i.e., $P_{(U,O)} = a_O^{x_U} b_O^{s_{(U,O)}}$, with $x_U \in \Gamma$ and $s_{(U,O)} \in \Delta$. The value $s_{(U,O)}$ is chosen jointly by O and U without O learning anything about either x_U or $s_{U,O}$. Note that this protocol does not force U to use the same x_U as with other organizations; this is taken care of later in Protocol 4.

Protocol 1

1. *U chooses a value $N_1 \in \{0,1\}^k$, and values $r_1 \in_R \Delta$ and $r_2, r_3 \in_R \{0,1\}^{2\ell_n}$. U sets $C_1 := g_O^{r_1} h_O^{r_2}$, $C_2 := g_O^{x_U} h_O^{r_3}$. U sends N_1, C_1, and C_2 to O.*
2. *To prove that C_1 and C_2 are formed correctly, U serves as the prover to verifier O in*
$$PK\{(\alpha,\beta,\gamma,\delta) : C_1^2 = (g_O^2)^\alpha (h_O^2)^\beta \ \wedge\ C_2^2 = (g_O^2)^\gamma (h_O^2)^\delta\} \ .$$
3. *O chooses a random $r \in_R \Delta$ and a value N_2 and sends r, N_2 to U.*
4. *U sets her pseudonym $N_{(U,O)} := N_1 \| N_2$. U computes $s_{(U,O)} = (r_1 + r \bmod (2^{\ell_\Delta+1}-1)) - 2^{\ell_\Delta} + 1$, ($s_{(U,O)}$ is the sum of r_1 and r, adjusted appropriately so as to fall in the interval Δ). U then sets her validating tag $P_{(U,O)} := a_O^{x_U} b_O^{s_{(U,O)}}$ and sends $P_{(U,O)}$ to O.*
5. *Now, U must show that $P_{(U,O)}$ was formed correctly. To that end, she computes $\tilde{s} = \left\lfloor \frac{r_1+r}{2^{\ell_\Delta+1}-1} \right\rfloor$ ($\tilde{s}$ is the value of the carry resulting from the computation of $s_{(U,O)}$ above) and chooses $r_4 \in_R \{0,1\}^{\ell_n}$, sets $C_3 := g_O^{\tilde{s}} h_O^{r_4}$, and sends C_3 to O. Furthermore, U proves to O that the values in step 4 were chosen correctly by executing*
$$\begin{aligned} PK\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\vartheta,\xi) :\ & C_1^2 = (g_O^2)^\alpha (h_O^2)^\beta \ \wedge\ C_2^2 = (g_O^2)^\gamma (h_O^2)^\delta \ \wedge \\ & C_3^2 = (g_O^2)^\varepsilon (h_O^2)^\zeta \wedge \frac{C_1^2 (g_O^2)^{(r-2^{\ell_\Delta}+1)}}{(C_3^2)^{(2^{\ell_\Delta+1}-1)}} = (g_O^2)^\vartheta (h_O^2)^\xi \wedge \\ & P_{(U,O)}^2 = (a_O^2)^\gamma (b_O^2)^\vartheta \ \wedge\ \gamma \in \Gamma \ \wedge\ \vartheta \in \Delta\} \ . \end{aligned}$$
6. *O stores $N_{(U,O)}$, $P_{(U,O)}^2$ and $P_{(U,O)}$.*
7. *U stores $N_{(U,O)}$, $P_{(U,O)}^2$, $P_{(U,O)}$, and $s_{(U,O)}$.*

4.4 Generation of a Credential

A credential on (N, P) issued by O is a pair $(c, e) \in \mathbb{Z}_{n_O}^* \times \Lambda$ such that $P_{(U,O)} d_O = c^e$. To generate a credential on a previously established pseudonym $N_{(U,O)}$ with validity tag $P_{(U,O)}$, organization O and user U carry out the following protocol:

Protocol 2

1. *U sends $(N_{(U,O)}, P_{(U,O)})$ to O and authenticates herself as its owner by executing*
$$PK\{(\alpha,\beta) : P^2_{(U,O)} = (a^2_O)^\alpha (b^2_O)^\beta\} \ .$$
2. *O makes sure $(N_{(U,O)}, P_{(U,O)})$ is in its database, chooses a random prime $e_{(U,O)} \in_R \Lambda$, computes $c_{(U,O)} = (P_{(U,O)} d_O)^{1/e_{(U,O)}} \mod n_O$, sends $c_{(U,O)}$ and $e_{(U,O)}$ to U and stores $(c_{(U,O)}, e_{(U,O)})$ in its record for $N_{(U,O)}$.*
3. *U checks if ${c_{(U,O)}}^{e_{(U,O)}} \equiv P_{(U,O)} d_O \pmod{n_O}$ and stores $(c_{(U,O)}, e_{(U,O)})$ in its record with organization O. The tuple $(P_{(U,O)}, c_{(U,O)}, e_{(U,O)})$ is called a* credential record.

Step 1 can be omitted if Protocol 2 takes place in the same session as some other protocol where U already proved ownership of $N_{(U,O)}$.

4.5 Showing a Single Credential

Assume a user U wants to prove to a verifier V the possession of a credential issued by O, i.e., possession of values $(P_{(U,O)} = a_O^{x_U} b_O^{s_{(U,O)}}, c_{(U,O)}, e_{(U,O)})$, where $c_{(U,O)}^{e_{(U,O)}} = d_O P_{(U,O)}$. U and verifier V engage in the following protocol:

Protocol 3

1. *U chooses $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O)} h_O^{r_1}$ and $B = h_O^{r_1} g_O^{r_2}$, and sends A, B to V.*
2. *U engages with V in*

$$PK\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi) : \ d_O^2 = (A^2)^\alpha (\frac{1}{a_O^2})^\beta (\frac{1}{b_O^2})^\gamma (\frac{1}{h_O^2})^\delta \ \wedge$$
$$B^2 = (h_O^2)^\varepsilon (g_O^2)^\zeta \ \wedge \ 1 = (B^2)^\alpha (\frac{1}{h_O^2})^\delta (\frac{1}{g_O^2})^\xi \ \wedge$$
$$\beta \in \Gamma \ \wedge \ \gamma \in \Delta \ \wedge \ \alpha \in \Lambda\} \ .$$

The *PK* in step 2 proves that U possesses a credential issued by O on some pseudonym registered with O. We refer to the proof of Lemma 2 in the full version of this paper [12] for more details about this *PK*.

4.6 Showing a Credential with Respect to a Pseudonym

Assume a user U wants to prove possession of a credential record $(P_{(U,O_j)} = a^{x_U} b^{s_{(U,O_j)}}, c_{(U,O_j)}, e_{(U,O_j)})$ to organization O_i with whom U has established a pseudonym $(N_{(U,O_i)}, P_{(U,O_i)})$. That means O_i not only wants to be assured that U owns a credential by O_j but also that the pseudonym connected with this credential is based on the same master secret key as $P_{(U,O_i)}$.

Protocol 4

1. *U chooses random $r_1, r_2, r_3 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O_j)} h_{O_j}^{r_1}$ and $B = h_{O_j}^{r_1} g_{O_j}^{r_2}$, and sends $N_{(U,O_i)}, A, B$ to O_i.*
2. *U engages with O_i in*

$$PK\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\xi,\eta):\ d_{O_j}^2 = (A^2)^\alpha (\frac{1}{a_{O_j}^2})^\beta (\frac{1}{b_{O_j}^2})^\gamma (\frac{1}{h_{O_j}^2})^\delta \ \wedge$$
$$B^2 = (h_{O_j}^2)^\varepsilon (g_{O_j}^2)^\zeta \ \wedge\ 1 = (B^2)^\alpha (\frac{1}{h_{O_j}^2})^\delta (\frac{1}{g_{O_j}^2})^\xi \ \wedge$$
$$P_{(U,O_i)}^2 = (a_{O_i}^2)^\beta (b_{O_i}^2)^\eta \ \wedge\ \beta \in \Gamma \ \wedge\ \gamma \in \Delta \ \wedge\ \alpha \in \Lambda\} \ .$$

The first three equations of this proof of knowledge are the same as Protocol 3. The fourth equation proves that the same master secret key is used in $P_{(U,O_i)}$ and in the validating tag to the pseudonym established with O_j.

In the random oracle model, the verifier (or verifying organization) can obtain the receipt from a showing transaction by turning step 2 of Protocol 4 (or Protocol 3, respectively) into the corresponding *SPK* on the description of the transaction. This step will add efficiency and also will enable a user to sign an agreement with a verifier using her credential as a signature public key. This could, for instance, be useful if possessing a credential means being allowed to sign on behalf of the issuing organization (cf. group signatures).

5 Proof of Security for the Basic Credential System

The following technical lemmas about the protocols described above are stated here without proof; their proofs can be found in the full version of this paper [12].

Lemma 1. *Under the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a safe prime product, step 5 of Protocol 1 (the protocol for establishing a pseudonym) is a statistical zero-knowledge proof of knowledge of the correctly formed values x_U, $s_{(U,O)}$ that correspond to a pseudonym validating tag $P_{(U,O)}$.*

Lemma 2. *Under the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a safe prime product, step 2 of Protocol 3 (the protocol for showing a single credential) is a statistical zero-knowledge proof of knowledge of the values $x \in \Gamma$, $s \in \Delta$, $e \in \Lambda$, and c such that x, s correspond to a pseudonym validating tag $P = a_O^x b_O^s$, and $c^e = P d_O \bmod n_O$.*

Lemma 3. *Under the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a safe prime product, step 2 of Protocol 4 (the protocol for showing a credential corresponding to a given validating tag $P_{(U,O_i)}$) is a statistical zero-knowledge proof of knowledge of the values $x \in \Gamma$, $s_1, s_2 \in \Delta$, $e \in \Lambda$, and c such that $P_{(U,O_i)} = a_{O_i}^x b_{O_i}^{s_1} \bmod n_{O_i}$, x, s_2 correspond to a validating tag $P = a_{O_j}^x b_{O_j}^{s_2}$ and $c^e = P d_{O_j} \bmod n_{O_j}$ holds.*

5.1 Description of the Simulator

We now describe the simulator $\mathcal{S}$ for our scheme and then in Section 5.2 show that it satisfies Definition 1.

Setup. For the organizations not controlled by the adversary, the simulator sets up their secret and public keys as dictated by the protocol. For each organization, the simulator creates an archive where it will record the credentials issued by this organization to the users controlled by the adversary. It also initializes a list of the users controlled by the adversary.

Generation of a pseudonym. If a user controlled by the adversary establishes a pseudonym from an honest organization, the simulator uses the knowledge extractor of Lemma 1 to discover the user's underlying key x and the value s. If no user with key x is present in the list of dishonest users, $\mathcal{S}$ creates a new user U with login name L_U, and runs $\mathsf{FormNym}(U, O)$ to create a pseudonym $N_{(U,O)}$ for this user, and to obtain a key K_U for further interactions of this user with T. The simulator stores the record $(U,\ L_U,\ x,\ K_U,\ N_{(U,O)},\ s)$ in its list of users controlled by the adversary. If some user U with key x is already present, the simulator runs $\mathsf{FormNym}(U, O)$ to create a pseudonym $N_{(U,O)}$ for this user, and adds $(N_{(U,O)}, s)$ to U's record.

If an honest user, through T, establishes a pseudonym with an organization controlled by the adversary, our simulator will use the zero-knowledge simulator from Lemma 1 to furnish the adversary's view of the protocol.

Generate a credential. If a user controlled by the adversary requests a credential from an honest organization O, then, upon receiving a message from T to that effect, the simulator runs the knowledge extractor for the proof of knowledge of step 1 of Protocol 2. It determines the values x and s. The simulator looks at its list of the pseudonyms of users controlled by the adversary. If it does not find a record with x and s, then it refuses to grant a credential (as an organization would). If it finds that there is a record containing these x and s and pseudonym N, then the simulator runs $\mathsf{GrantCred}(N, O)$ with T. Upon hearing from T that the user may have a credential, the simulator runs the organization's side of the rest of the Protocol 2, and issues the correct e and c. It stores the values (x,s,e,c) in the archive for organization O.

If an honest user, through T, requests a credential from an organization controlled by the adversary, then the simulator will run the zero-knowledge simulator for step 1 of Protocol 2, and execute the rest of the user's side of it. If the user accepts, then the simulator informs T that the credential was granted.

Showing a single credential. This part of the simulator can easily be inferred from the part for *Showing a credential with respect to a pseudonym* that follows.

Showing a credential with respect to a pseudonym. If a user controlled by the adversary wants to show a credential from an honest organization O_j to an honest organization O_i with whom it has pseudonym $N_{(U,O_i)}$, then the simulator runs O_i's part of Protocol 4, and extracts the user's values $(x, s_{(U,O_i)}, s_{(U,O_j)}, e, c)$ with the knowledge extractor of Lemma 3. If O_i's side of Protocol 4 accepts, while $(x, s_{(U,O_j)}, e, c)$ is not in the archive of O_j, then $\mathcal{S}$ rejects. Otherwise, it finds the user U with key x, the user's corresponding key K and pseudonym $N_{(U,O_j)}$ and runs $\mathsf{VerifyCred}(O_i, N_{(U,O_i)}, N_{(U,O_j)}, O_j)$.

If a dishonest user wants to prove to an honest organization O_i that he has a credential from a dishonest organization O_j, then the simulator runs O_i's side of Protocol 4, with the knowledge extractor of Lemma 3 to obtain the values $(x, s_{(U,O_i)}, s, e, c)$. If O_i's side of the protocol rejects, it does nothing. Otherwise: (1) It checks if there exists a user with key x in O_j's archive. If so, denote this user by U. If not, let U be the user with key x. Next it runs $\mathsf{FormNym}(U, O_j)$ to get $N_{(U,O)}$. (2) It checks if U has a credential record in O_j's archive. If not, it runs $\mathsf{GrantCred}(N_{(U,O_j)}, O_j)$. (3) It runs $\mathsf{VerifyCredOnNym}(O_i, N_{(U,O_i)}, N_{(U,O_j)}, O_j)$.

If an honest user (through T) wants to prove to organization O_i controlled by the adversary, that he has a credential from an honest organization O_j, then the simulator runs the zero-knowledge simulator of Lemma 3 to do that.

5.2 Proof of Successful Simulation

We show that our simulator fails with negligible probability only. As a first step, we show in Theorem 2 that a tuple (x, s, e, c) the knowledge of which is essential for proving possession of a credential, is unforgeable even under an adaptive attack. For this we rely on the following theorem due to Ateniese et al. [2]:

Theorem 1. *Suppose an ℓ_n-bit RSA modulus $n = pq = (2p' + 1)(2q' + 1)$ is given, where p, q, p', and q' are primes. Let $\Delta' =]-2^{\ell_{\Delta'}}, 2^{\ell_{\Delta'}}[$, $\Pi' =]-T, T[$ and $\Lambda' =]2^{\ell_{\Lambda'}}, 2^{\ell_{\Lambda'}+\ell_{\Sigma'}}[$ with $2^{\ell_{\Delta'}} \geq T > 2^{2\ell_n}$ and $\ell_{\Lambda'} > \ell_{\Sigma'} + \ell_{\Delta'} + 3$. Suppose random $b, d \in_R QR_n$ are given. Further, suppose we have access to an oracle which, on the i-th query outputs tuples (y_i, e_i, c_i) such that $y_i \in_R \Pi'$, $e_i \in_R \Lambda'$ is a prime, and $c_i^{e_i} = b^{y_i} d \bmod n$. Under the strong RSA assumption, it is hard, upon seeing the oracle output for $1 \leq i \leq K$, K polynomial in ℓ_n, to produce a tuple (y, e, c) such that for all $1 \leq i \leq K$, $(y, e) \neq (y_i, e_i)$, and $y \in \Delta'$, $e \in \Lambda'$, and $c^{2e} = (b^y d)^2$.*

Theorem 2. *Suppose an ℓ_n-bit RSA modulus $n = pq = (2p' + 1)(2q' + 1)$ is given, where p, q, p', and q' are primes. Suppose random $a, b, d \in_R QR_n$ are given. Further, suppose we have access to an oracle $\mathcal{O}$ which, on the i-th query with a value $x_i \in \Gamma$, outputs a tuple (s_i, e_i, c_i) such that that $s_i \in_R \Delta$, $e_i \in_R \Lambda$ is a prime, and $c_i^{e_i} = a^{x_i} b^{s_i} d$. Under the strong RSA assumption and the discrete logarithm assumption modulo a safe prime product, it is hard, upon seeing the oracle output for $1 \leq i \leq K$, K polynomial in ℓ_n, to produce a tuple (x, s, c, e) such that for all $1 \leq i \leq K$, $(x, s, e, c) \neq (x_i, s_i, c_i, e_i)$, and $x \in \Gamma$, $s \in \Delta$, $e \in \Lambda$, and $c^{2e} = (a^x b^s d)^2 \bmod n$.*

Proof. We will prove our theorem by exhibiting a reduction to Theorem 1. The reduction has access to a forger $\mathcal{A}$ that forges a tuple (x, s, e, c) under conditions stated in the theorem. Using $\mathcal{A}$, the reduction will forge a tuple (y, e, u)

under conditions stated in Theorem 1. This, in turn, contradicts the strong RSA assumption.

The reduction will take, as input, the public parameters (n, b, d) as in Theorem 1. Then it will define the public parameters for the setting of the theorem: b and d are as given, and to form a, pick $\alpha \in_R [0, n/4]$ and set $a := b^\alpha$.

Then, the reduction makes K queries and obtains a set of tuples $\{(y_i, e_i, c_i)\}$. Now the reduction proceeds as follows: upon receiving a query $x_i \in \Gamma$, set $s_i := y_i - \alpha x_i$. Setting the parameter T of Theorem 1 to $2^{\ell_\Delta} - 2^{\ell_\Lambda + \ell_n}$ will assure that $s_i \in \Delta$. Setting $\ell_\Delta = \epsilon(\ell_\Lambda + \ell_n) + 1$ with $\epsilon > 1$ (cf. Section 4.2) assures that $T > 2^{\ell_n}$ and also that s_i will be distributed statistically close to uniformly from Δ. Further, note that $a^{x_i} b^{s_i} d = b^{\alpha x_i + s_i} d = a^{y_i} d = c_i^{e_i}$. Setting $\ell_{\Sigma'} = \ell_\Sigma$ and $\ell_{\Lambda'} = \ell_\Lambda$, the tuple (s_i, e_i, c_i) is distributed statistically close to the distribution induced by the actual oracle $\mathcal{O}$.

After answering the K queries, the reduction receives from the forger a tuple (x, s, e, c) such that for all $1 \le i \le K$, $(x, s, e, c) \neq (x_i, s_i, e_i, c_i)$, $x \in \Gamma$, $s \in \Delta$, $e \in \Lambda$, and $c^{2e} = (a^x b^s d)^2 \bmod n$. Compute $y = s + \alpha x$. Setting ℓ'_Δ of Theorem 1 to $\ell_\Delta + 1$ gives us the condition $\ell_\Lambda = \ell_{\Lambda'} > \ell_{\Sigma'} + \ell_{\Delta'} + 3 = \ell_\Sigma + \ell_\Delta + 4$ (cf. Section 4.2). With these settings, the triple (y, e, c) constitutes a forgery for Theorem 1, provided that $(y, e) \neq (y_i, e_i)$ for all i.

Suppose the probability that $(y, e) = (y_i, e_i)$ for some i is non-negligible. Then we can use $\mathcal{A}$ to break discrete logarithm modulo n. Suppose $(g, h) \in QR_n$ are given. It is known that finding (α_1, β_1) and a distinct (α_2, β_2) such that $g^{\alpha_1} h^{\beta_1} = g^{\alpha_2} h^{\beta_2}$ is hard if factoring is hard and computing discrete logarithms modulo a safe prime product is hard.

The reduction takes, as input, the modulus n, and the values (g, h). Then it selects K random primes $\{e_i \in_R \Lambda\}_{i=1}^K$, chooses a random $v \in_R QR_n$, and sets $d = v^{\prod_{i=1}^k e_i}$, $a = g^{\prod_{i=1}^k e_i}$, $b = h^{\prod_{i=1}^k e_i}$.

On input (x_i, V_i), do the following: select an $s_i \in_R \Delta$. Compute the value $E_i = \prod_{j=1, j\neq i}^K e_j$. Set $u_i := (g^{x_i} h^{s_i} v)^{E_i}$. Note that by construction, $c_i^{e_i} = a^{x_i} b^{s_i} d$. Then output (s_i, e_i, c_i). With non-negligible probability, obtain a forgery (x, s, e, c) from the forger such that $(x, s, e, c) \neq (x_i, s_i, e_i, c_i)$ for all i, and yet for some i, $(a^{x_i} b^{s_i} d)^2 = (a^x b^s d)^2$. Because a,b, and d are quadratic residues, it follows that $a^{x_i} b^{s_i} d = a^x b^s d$. From here, we either break the discrete logarithm problem or factor n.

Lemma 4. *Under the strong RSA assumption and the decisional Diffie-Hellman assumption modulo a safe prime product, the simulator rejects with only negligible probability.*

Proof. (Sketch) Note that the only time when the simulator rejects is when a dishonest user makes the verifier accept in Protocol 3 or in Protocol 4, and yet the tuple (x, s, e, c) extracted by the simulator was not given to the adversary by the simulator itself. Under the appropriate assumptions, by Lemmas 2 and 3 knowledge extraction succeeds with probability $1 - \mathtt{neg}(k)$. Then if we are given an adversary that can make the simulator reject non-negligibly often, we can use this adversary to create a forgery to contradict Theorem 2.

The statistical zero-knowledge property of the underlying protocols gives us Lemma 5 which in turn implies Theorem 3.

Lemma 5. *The view of the adversary in the real protocol is statistically close to his view in the simulation.*

Theorem 3. *Under the strong RSA assumption, the decisional Diffie-Hellman assumption modulo a safe prime product, and the assumption that factoring is hard, the credential system described above is secure.*

6 All-or-nothing and PKI-based Non-transferability

The protocols described in Section 4 ensure consistency of credentials, i.e., credential pooling is not possible. However, credential (or pseudonym) lending is still possible. More precisely, revealing to a friend the secrets x_U and $s_{(U,O_i)}$ attached to some credential does not mean that the friend obtains some other valuable secret of the user or can use any of the user's other credentials. This section provides protocols to obtain PKI-based non-transferability and all-or-nothing non-transferability to discourage users from credential lending.

The idea of the former is that the user provides the *CA* with a (verifiable) encryption of some valuable external secret that can be decrypted with x_U.

The idea for achieving the latter is similar, i.e., the user provides each organization with a (verifiable) encryption of the secrets underlying her validating tag. This approach raises some technical problems:

First, the approach requires that each user encrypts each of her secret keys D_i under one of her public keys E_j, thereby creating "circular encryptions". However, the canonical definitions [35] of secure encryption do not provide security for such encryptions. Moreover, it is not known whether circular security is possible under general assumptions. Nevertheless, we introduce in this section a new cryptographic primitive called *circular encryption* which is an encryption scheme that provides security for circular encryptions. Given any semantically secure encryption scheme, we provide a generic construction of such a scheme and prove its security in the random oracle model

Second, the encryptions made by a user must not reveal the public key this encryption was made with, i.e., we require that the encryption scheme be *key-oblivious*. We provide a formal definition of this and show that our circular encryption scheme satisfies it.

Third, the encryption must be verifiable. To this end we review the verifiable encryption protocol due to Camenisch and Damgård [10] and adapt it to suit our needs. Specifically, we want to enable verification without revealing the public key. We provide a verification method involving a *committed* public key, so that by inspecting this verifiable encryption, an adversary would not be able to discover the underlying public key.

Independently of and concurrently with our work, Black et al. [5] proposed symmetric encryption schemes for key-dependent messages (which is what we call circular symmetric encryption) and Bellare et al. [3] studied key-private encryption (which is what we call key-oblivious encryption).

6.1 Circular Encryption

Definition 2. *Let $n, m \in \texttt{poly}(k)$. A semantically secure encryption scheme $\mathcal{G} = (\mathcal{E}, \mathcal{D})$ is circular-secure if*

1. *There exists a message, denoted by 0, such that for all $E \in \mathcal{E}(1^k)$, 0 is in the message space of E.*
2. *For all $E_1 \in \mathcal{E}(1^k)$, $D_2 \in \mathcal{D}(1^k)$, the message space of E_1 includes D_2.*
3. *For all n-node directed graphs G with m edges, given n randomly chosen public keys, $\{E_i\}_{i=1}^n$, we have: $\{E_i(D_j)\}_{(i,j)\in E(G)} \overset{c}{\approx} \{E_i(0)\}_{(i,j)\in E(G)}$.*

The idea here is that having access to encryptions of the secret keys does not help the adversary in breaking the security of the system. Note that if, in the definition above, we had limited our attention to acyclic graphs, then any semantically secure cryptosystem would be enough to satisfy such a definition. As the definition can only be more powerful if we include graphs that have cycles, we call this notion of security "circular security."

Let us present a cryptosystem that satisfies this definition in the random oracle model. Suppose the length of a secret key is $p(k)$. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{p(k)}$ be a random oracle, and let $\oplus$ denote the bitwise XOR operation. Let $\mathcal{G} = (\mathcal{E}, \mathcal{D})$ be a semantically secure cryptosystem with a sufficiently large message space. Construct $\mathcal{G}' = (\mathcal{E}', \mathcal{D}')$ as follows: generate (E, D) according to $\mathcal{G}$. To encrypt a message $m \in \{0,1\}^{p(k)}$, E' picks a random $r \in_R \{0,1\}^\ell$ and sets $E'(m) := (E(r), \mathcal{H}(r) \oplus m)$. To decrypt a tuple (a, b), D' computes $\tilde{m} := \mathcal{H}(D(a)) \oplus b$. For this construction, the following theorem holds (the proof can be found in the full version of this paper [12]).

Theorem 4. *If $\mathcal{G}$ is semantically secure, $\mathcal{G}'$ is circular-secure.*

As a basis for our circular encryption scheme, we use the ElGamal encryption [27] in some $G = \langle g \rangle$. It is easy to see that the ElGamal cryptosystem is semantically secure under the decisional Diffie-Hellman assumption. Let $P = g^x$ be a public key. The resulting circular encryption scheme is as follows. To encrypt a message $m \in \{0,1\}^k$, choose a random element $r_1 \in G$ and a random integer $r_2 \in \{0,1\}^{2\ell}$, and compute the encryption $(u, v, z) := (P^{r_2} r_1, g^{r_2}, \mathcal{H}(r_1) \oplus m)$. Decryption works by computing $\mathcal{H}(u/v^x) \oplus z$. We denote this encryption scheme by CEIG.

6.2 Verifiable Encryption with a Committed Public Key

Verifiable encryption [1, 10], is a protocol between a prover and a verifier such that as a result of the protocol, on input public key E and value v, the verifier obtains an encryption e of some value s under E such that $(s, v) \in \mathcal{R}$. Here $\mathcal{R}$ is a relation such as, e.g., $\{(s, g^s) | s \in \mathbb{Z}_q\} \subset \mathbb{Z}_q \times G$. More formally,

Definition 3. *Let $(\mathcal{E}, \mathcal{D})$ be a semantically secure encryption scheme. A two-party protocol between a prover $\mathcal{P}(\mathcal{R}, E, s, v)$ and a verifier $\mathcal{V}(\mathcal{R}, E, v)$ is a verifiable encryption protocol with respect to public keys $\mathcal{E}$ for a polynomial-time verifiable relation $\mathcal{R}$ if*

– *For all* $(E, D) \in \mathcal{G}(1^k)$ *and for all* $(s, v) \in \mathcal{R}$*, if* $\mathcal{P}$ *and* $\mathcal{V}$ *are honest then* $\mathcal{V}_{\mathcal{P}(\mathcal{R},E,s,v)}(\mathcal{R}, E, v) \neq \perp$.
– *There is an efficient extractor algorithm* C *such that for all sufficiently large* k*, and* $\forall(E, D) \in (\mathcal{E}, \mathcal{D})(1^k)$

$$\Pr[(C(D, e), v) \in \mathcal{R} \mid e = \mathcal{V}_{\widetilde{\mathcal{P}}(\mathcal{R},E,s,v)}(\mathcal{R}, E, v) \wedge e \neq \perp] = 1 - \mathtt{neg}(k) \ .$$

– *There is a black-box simulator* $\mathcal{S}$ *such that* $\forall\ \widetilde{\mathcal{V}}$*,* $\forall(s, v) \in \mathcal{R}$ *we have* $\mathcal{S}^{\widetilde{\mathcal{V}}(\mathcal{R},E,v)}(\mathcal{R}, E, v) \overset{c}{\approx} \widetilde{\mathcal{V}}_{\mathcal{P}(\mathcal{R},E,s,v)}(\mathcal{R}, E, v)$*, where the probability "hidden" in the* $\overset{c}{\approx}$ *notation is over the choice of* E *and the random cointosses of* $\widetilde{\mathcal{V}}$.

Note that e is not a single message from the prover, but the verifier's entire transcript of the protocol. Furthermore, C does not necessarily extract the same s that was the additional input to the prover. It could extract some other value $s' \neq s$, but only if $(s', v) \in \mathcal{R}$.

It is clear that an (inefficient) way of implementing verifiable encryption would be for the prover to encrypt s under the public key E, and then carry out a zero-knowledge proof that the encrypted value satisfies relation $\mathcal{R}$ with respect to v. But this is not satisfactory, because it is important that verifiably encryption be executed efficiently enough to be useful in practice. Generalizing the protocol of Asokan et al. [1], Camenisch and Damgård [10] provide a practical verifiable encryption scheme for all relations $\mathcal{R}$ that have an honest-verifier zero-knowledge three-move proof of knowledge where the second message is a random challenge and the witness can be computed from two transcripts with the same first message but different challenges. This includes most known proofs of knowledge, and all proofs about discrete logarithms considered in this paper. Their construction is secure with respect to any public key for a semantically secure cryptosystem.

We use similar notation for verifiable encryption as for the PK's and denote by, e.g., $e := VE(\mathsf{ElGamal}, (g, y))\{(\xi) : a = b^{\xi}\}$ the verifiable encryption protocol for the ElGamal scheme, whereby $\log_b a$ is encrypted in e under public key (y, g).

For guaranteeing the all-or-nothing non-transferability, we need to have each user verifiable encrypt all of her secret information under a public key that corresponds to her secret key. However, revealing this public key will leak information about the user. Therefore, we need to realize verifiable encryption in such a manner that the public key corresponding to the resulting ciphertext cannot be linked to the verifier's view, i.e., a verifiable encryption scheme must be key-oblivious:

Definition 4. *Let* $(\mathcal{P}, \mathcal{V})$ *be a verifiable encryption scheme with respect to public keys* $\mathcal{E}$*, for a polynomial-time verifiable relation* $\mathcal{R}$*. We say that this scheme is* key-oblivious *if for all polynomially bounded* $\tilde{\mathcal{V}}$*, for all* $E, E' \in \mathcal{E}(1^k)$ *and* $\forall(s, v) \in \mathcal{R}$ *we have* $\tilde{\mathcal{V}}_{\mathcal{P}(\mathcal{R},E,s,v)}(\mathcal{R}, v, E, E') \overset{c}{\approx} \tilde{\mathcal{V}}_{\mathcal{P}(\mathcal{R},E',s,v)}(\mathcal{R}, v, E, E')$*, where the probability "hidden" in the* $\overset{c}{\approx}$ *notation is over the random cointosses of* $\widetilde{\mathcal{V}}$.

In case the verifier does not know the public key under which the encryption is carried out, previously known constructions do not work, as they require that the verifier be able to check that a given ciphertext is an encryption of a given value. Thus we propose a new construction, based on the circularly secure variant of the ElGamal cryptosystem described above. Here we assume that the prover $\mathcal{P}$ knows the secret key of the encryption; this is not the general case, but it works for our construction. Let $P = g^x$ serve as a public key, and x as the corresponding secret key. Let $C = Ph^r$ be a commitment to P, where h is another generator of $G = \langle g \rangle$, and let $(u, v, z) = (P^{r_2} r_1, g^{r_2}, \mathcal{H}(r_1) \oplus m)$ be an encryption of m as above. To convince the verifier that (u, v, z) is an encryption of m under the public key committed to by C, the prover reveals r_1 and engages with the verifier in $PK\{(\alpha, \beta, \gamma) : C = g^\alpha h^\beta \ \wedge \ v = g^\gamma \ \wedge \ u/r_1 = v^\alpha\}$. The verifier further needs to check if $z = \mathcal{H}(r_1) \oplus m$. By using techniques developed by Camenisch and Damgård [10], a key-oblivious verifiable encryption scheme is obtained.

In the sequel, we write, e.g., $\mathit{Com\text{-}VE}(\mathsf{CElG}, (\mathcal{H}, g, h, C))\{(\xi) : a = b^\xi\}$ for this key-oblivious verifiable encryption with respect to a committed public key. The proof of the following lemma uses standard techniques and is given in the full version of this paper [12]:

Lemma 6. *Under the decisional Diffie-Hellman assumption, the verifiable encryption scheme described above is key-oblivious.*

6.3 All-or-nothing Non-transferability

As already mentioned, all-or-nothing non-transferability is achieved by ensuring that if a user U gives away her master secret x_U, then she will also reveal the secret keys underlying her validating tag with O. More precisely, U has to supply O a verifiable encryption of these secrets w.r.t. the secret key x_U. This is done in the following protocol, which U and O should carry out as part of Protocol 1. A prerequisite of the protocol is that during the setup of the system, a group $G = \langle g \rangle = \langle h \rangle$ of prime order $q > 2^{\ell_\Gamma}$ is chosen such that $\log_g h$ is unknown.

Protocol 5

1. *U chooses $r \in_R \mathbb{Z}_q$, sets $C := g^{x_U} h^r$, and sends C to O. U proves to O that C is a commitment to her public key by carrying out*
$$PK\{(\gamma, \vartheta, \varphi) : \ P^2_{(U,O)} = (a^2_O)^\gamma (b^2_O)^\vartheta \ \wedge \ C = g^\gamma h^\varphi\} \ .$$
2. *U and O engage in the verifiable encryption protocol*
$$w_{P_{(U,O)}} = \mathit{Com\text{-}VE}(\mathit{CElG}, (\mathcal{H}, g, h, C))\{(\alpha, \beta) : P^2_{(U,O)} = (a^2_O)^\alpha (b^2_O)^\beta\} \ .$$
3. *O publishes $N_{(U,O)}$ and $w_{P_{(U,O)}}$.*

However, publishing $N_{(U,O)}$ and $w_{P_{(U,O)}}$ is not sufficient for using U's credential with O even when knowing x_U. Therefore, the organizations must publish all related information together with the verifiable encryption. Hence, at the end of Protocol 2, O must publish $(c_{(U,O)}, e_{(U,O)})$ together with $N_{(U,O)}$. Thus, we obtain all-or-nothing transferability: whenever a user's friend gets to know x_U, he can look at the organizations' public records to obtain all information needed to use all the user's credentials.

6.4 PKI-assured Non-transferability.

We assume that the user possesses some external valuable public key PK_U. Then PKI-assured non-transferability is achieved by having the CA ask for this public key, check whether it is indeed the user's public key (e.g., via some external certificate), and require the user to verifiably encrypt the corresponding secret key SK_U with respect to x_U. This verifiable encryption is then published by the CA. Now, if the user ever gives x_U away to her friend, then her friend, by reading the CA's public records, will recover the verifiable encryption of SK_U, and will succeed in decrypting it.

The technical realization is similar to the one for all-or-nothing non-transferability. The main difference is that we do not need circular encryption and thus can use regular ElGamal. We give an example for what this protocol looks like when U's external public key Y_U is discrete-logarithm based, i.e., $Y_U = g^x$ for some generator g in some group G. Other cases are similar. A prerequisite of the protocol is that during the setup of the system, a group $G = \langle g \rangle = \langle h \rangle$ of prime order $q > 2^{\ell_\Gamma}$ is chosen such that $\log_g h$ is unknown.

Protocol 6

1. *U sends Y_U, g, and the certificate on Y_U of the external PKI to CA who checks their validity.*
2. *U chooses $r \in_R \mathbb{Z}_q$, sets $C := g^{x_U} h^r$, and sends C to CA. U proves to CA that C is a commitment to her public key by carrying out*

$$PK\{(\gamma, \vartheta, \varphi) : \; P^2_{(U,O_0)} = (a^2_{O_0})^\gamma (b^2_{O_0})^\vartheta \;\wedge\; C = g^\gamma h^\varphi\} \enspace .$$

3. *U and CA engage in*

$$w_{PKI} = Com\text{-}VE(\mathsf{ElGamal}, (\mathcal{H}, g, h, C))\{(\alpha) : Y_U = g^\alpha\} \enspace .$$

4. *CA publishes (w_{PKI}, PKI).*

7 One-Show Credentials and Revocation

This section describes how the basic credential scheme can be extended to allow for global and local revocation as well as to enable organizations to issue one-show credentials.

7.1 One-Show Credentials

The credentials we considered so far can be shown an unlimited number of times. However, for some services it might be required that a credential can only be used once (e.g., when it represents money). Of course, one possibility would be that a user just reveals the credential to the verifier. This, however, would mean that the user is not fully anonymous any more as the verifier and the organization then both know the credential and thus can link the transaction to the user's pseudonym. Traditionally, this problem has been solved using so-called blind signatures [17]. Here, we provide a novel and alternative way to approach this problem, i.e., instead of blinding the signer we blind the verifier. In the sequel we describe the general idea, the changes to the protocols that need to be made, and provide a protocol for showing one-show credentials.

Addition to key generation. Each organization O publishes an additional generator $z_O \in QR_{n_O}$.

Changes to Protocol 1. The validating tag $P_{(U,O)}$ on a user's pseudonym $N_{(U,O)}$ is formed slightly differently: $P_{(U,O)} = a_O^{x_U} b_O^{s_{(U,O)}} z_O^{r_{(U,O)}}$, where $r_{(U,O)}$ is chosen by O and U together in the same way as $s_{(U,O)}$ is. (Credentials, however, are issued in the same way as before, i.e., U obtains $c_{(U,O)}$ and $e_{(U,O)}$ such that ${c_{(U,O)}}^{e_{(U,O)}} \equiv P_{(U,O)} d_O \pmod{n_O}$ holds.)

Showing a one-show credential. When proving possession of a one-show credential issued by O (with respect to a pseudonym or not), the user provides to verifier V (which might be an organization) the value $H_{(U,O)} = h_O^{r_{(U,O)}}$ and proves that it is formed correctly w.r.t. to the pseudonym U established with O. Of course, the various proofs of knowledge in the respective protocols have to be adapted to reflect the different form of the pseudonym U holds with O. These adaptions, however, are immediate and we do not describe them here.

Now, different usages of the same credential can be linked to each other but not to the user's pseudonym with the issuing organization. This allows to prevent users from using the same credential several times, if the verifier checks with the issuing organization whether $H_{(U,O)}$ was already used or not, similar as it is done for anonymous on-line e-cash.

Off-line checking could be done as well. As here double usage can only be detected but not prevented, a mechanism for identifying double-users is required. This could for instance be achieved using revocation as described in the previous section, or using similar techniques that are used in for anonymous off-line e-cash (e.g., [7]).

We now describe how the latter can be done such that using a one-show credential twice would expose the user's secret keys connected with the corresponding pseudonym. Together with (any kind of) non-transferability this would be quite a strong incentive for the users not to use one-show credentials twice. The main idea is that the verifying entity chooses some random challenge c from a suitably large set, say $\{0,1\}^{\ell_c}$ with $\ell_c = 60$, and the user replies with $r = cx_U + s_{(U,O)}$ and proves correctness of this result. To assure that r hides x_U statistically, we must have that $\ell_\Delta > \epsilon(\ell_\Gamma + \ell_c)$ because $x_U \in \Gamma$ and $s_{(U,O)} \in \Delta$. However, when a user uses the same credential twice, one can compute x_U from the the different replies the user provides. We present the resulting protocol for showing a single credential (cf. Protocol 3).

Protocol 7

1. *U chooses $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O_i)} h_O^{r_1}$ and $B = h_O^{r_1} g_O^{r_2}$, and sends $A, B, H_{(U,O)}$ to V.*

2. V chooses $c \in_R \{0,1\}^{\ell_c}$ and sends c to U.
3. U replies with $r = cx_U + s_{(U,O)}$ (computed in $\mathbb{Z}$).
4. U engages with V in

$$PK\{(\alpha,\beta,\gamma,\varphi,\delta,\varepsilon,\zeta,\xi): \ d_O^2 = (A^2)^\alpha(\frac{1}{a_O^2})^\beta(\frac{1}{b_O^2})^\gamma(\frac{1}{z_O^2})^\varphi(\frac{1}{h_O^2})^\delta \ \wedge$$
$$B^2 = (h_O^2)^\varepsilon(g_O^2)^\zeta \ \wedge \ 1 = (B^2)^\alpha(\frac{1}{h_O^2})^\delta(\frac{1}{g_O^2})^\xi \ \wedge$$
$$H_{(U,O)} = h_O^\varphi \ \wedge \ g_O^r = (g_O^c)^\beta g_O^\gamma \ \wedge \ \beta \in \varGamma \ \wedge \ \gamma \in \varDelta \ \wedge \ \varphi \in \varDelta \ \wedge \ \alpha \in \varLambda\} \ .$$

The adaption of Protocol 4 to implement one-show credentials with built-in anonymity revocation is similar.

7.2 Local and Global Revocation

For simplicity we assume a single revocation manager R who is responsible for local and global revocation (extending the scheme to one revocation manager per organization is easy). Given the transcript of a protocol where some user proved possession of a credential from organization O_i, R will have the task of providing information that allows the organization to identify the pseudonym of the user in case of local revocation, or allows the CA to retrieve the identity of the user.

In the sequel we describe how the protocols for proving possession of a credential must be adapted such that local revocation is possible using Cramer-Shoup encryption [22]. We then discuss global revocation. We remark that it can be decided at the time when the possession of a credential is proved whether local and/or global revocation shall be possible for the transaction at hand.

Additions to key generation. The revocation manager R chooses a group $G = \langle g \rangle = \langle h \rangle$ of prime order $q > 2^{\ell_\Gamma}$. The he chooses five secret keys $x_1, \ldots, x_5 \in_R \mathbb{Z}_q$ and computes $(y_1,\ y_2,\ y_3) := (g^{x_1}h^{x_2}, g^{x_3}h^{x_4}, g^{x_5})$ as his public key. Each organization O publishes an additional generator $v_O \in QR_{n_O}$.

Changes to Protocol 1. A validating tag $P_{(U,O)}$ on a user's pseudonym $N_{(U,O)}$ is formed slightly differently: $P_{(U,O)} = a_O^{x_U} b_O^{s_{(U,O)}} v_O^{x_{(U,O)}}$, where $x_{(U,O)}$ is chosen from $\varGamma$ by U. However, credentials are issued in the same way as before, i.e., U obtains $c_{(U,O)}$ and $e_{(U,O)}$ such that ${c_{(U,O)}}^{e_{(U,O)}} \equiv P_{(U,O)} d_O \pmod{n_O}$ holds.

If Protocol 1 is carried out with the CA, it is extended by the following steps.

8. U computes $Y_U = g^{x_U}$ and sends Y_U to CA.
9. U engages with CA in

$$PK\{(\alpha,\beta,\gamma): P_{(U,CA)}^2 = (a_{CA}^2)^\alpha(b_{CA}^2)^\beta(v_O^2)^\gamma \ \wedge \ Y_U = g^\alpha \ \wedge \ \gamma \in \varGamma\} \ .$$

10. Both CA and U store Y_U with $P_{(U,CA)}$.

In case Protocol 1 is carried out with an organization O different from the CA, it is extended by the following steps.

8. U computes $Y_{(U,O)} = g^{x_{(U,O)}}$ and sends $Y_{(U,O)}$ to O.
9. U engages with O in

$$PK\{(\alpha,\beta,\gamma) : P^2_{(U,O)} = (a^2_O)^\alpha (b^2_O)^\beta (v^2_O)^\gamma \ \wedge \ Y_{(U,O)} = g^\gamma \ \wedge \ \gamma \in \Gamma\} \ .$$

10. Both O and U store $Y_{(U,O)}$ with $P_{(U,CA)}$.

Changes to Protocols 3 and 4. Suppose Protocol 3 (resp., Protocol 4) is being executed. Suppose the user U and the verifying organization V agree upon text m that describes under what conditions V can find out U's identifying information. Specifically, m describes the conditions under which V may find out U's pseudonym with the issuing organization O, as well as the conditions under which V may find out U's identity. The text of m can also include part of the communication transcript of the current protocol. The former mode of anonymity revocation is called *local* revocation, while the latter is called *global* revocation. We provide the two protocols to be executed as sub-routines of Protocol 3 (resp., Protocol 4) in order to get local and/or global revocation, respectively, where the user proves possession of a credential issued by organization O.

Protocol 8 (Global Revocation)

1. *U chooses $r_2 \in_R \mathbb{Z}_n$ and computes $w_1 := g^{r_2}$, $w_2 := h^{r_2}$, $w_3 := y_3^{r_2} Y_U$, and $w_4 := y_1^{r_2} y_2^{r_2 \mathcal{H}(w_1,w_2,w_3,m_0)}$ and sends $w_{(U,R)} = (w_1,w_2,w_3,w_4)$ to V.*
2. *U and V engage in*

$$PK\big\{(\alpha,\beta,\gamma,\delta,\varepsilon,\xi) : \ d_O^2 = (A^2)^\alpha (\frac{1}{a_O^2})^\beta (\frac{1}{b_O^2})^\gamma (\frac{1}{v_O^2})^\xi \frac{1}{h_O^2})^\delta \ \wedge \ w_1 = g^\varepsilon \ \wedge \\ w_2 = h^\varepsilon \ \wedge \ w_3 = g^\beta y_3^\varepsilon \ \wedge \ w_4 = \big(y_1 y_2^{\mathcal{H}(w_1,w_2,w_3,m_0)}\big)^\varepsilon\big\} \ .$$

Protocol 9 (Local Revocation)

1. *U chooses $r_1 \in_R \mathbb{Z}_q$ and computes $w_1 := g^{r_1}$, $w_2 := h^{r_1}$, $w_3 := y_3^{r_1} Y_{(U,O)}$, and $w_4 = y_1^{r_1} y_2^{r_1 \mathcal{H}(w_1,w_2,w_3,m_j)}$ and sends $w_{(U,R_j^l)} = (w_1,w_2,w_3,w_4)$ to V.*
2. *U and V engage in*

$$PK\big\{(\alpha,\beta,\gamma,\delta,\varepsilon,\xi) : \ d_O^2 = (A^2)^\alpha (\frac{1}{a_O^2})^\beta (\frac{1}{b_O^2})^\gamma (\frac{1}{v_O^2})^\xi (\frac{1}{h_O^2})^\delta \ \wedge \ w_1 = g^\varepsilon \ \wedge \\ w_2 = h^\varepsilon \ \wedge \ w_3 = g^\xi y_3^\varepsilon \ \wedge \ w_4 = \big(y_1 y_2^{\mathcal{H}(w_1,w_2,w_3,m_j)}\big)^\varepsilon\big\} \ .$$

Revocation. Upon presentation of an encryption $w = (w_1,w_2,w_3,w_4)$ and a revocation condition m, stemming from Protocol 8 or 9, the revocation manager checks whether $w_4 = w_1^{x_1 + x_3 \mathcal{H}(w_1 \| w_2 \| w_3 \| m)} w_2^{x_2 + x_4 \mathcal{H}(w_1 \| w_2 \| w_3 \| m)}$ and whether m is satisfied. If these checks succeed, he returns $\hat{Y} := w_3 / w_1^{x_5}$. In case of local revocation, $\hat{Y}$ will allow retrieval of the user's pseudonym with the organization that issued the credential of which that user proved possession. In case of global revocation, $\hat{Y}$ will allow the CA to retrieve the identity of the user.

7.3 Encoding Expiration Dates and Other Personal Attributes

Expiration dates and other attributes of credentials can be encoded in the exponent $e_{(U,O)}$ as this is the organization's choice. We need to divide the interval Λ into subintervals. Then, if a user is required to prove certain attributes of her credential, she proves that the exponent lies in the subinterval instead of proving that it lies in Λ.

Acknowledgements

The authors are grateful to Ron Rivest for fruitful discussions and comments. We thank the anonymous referees for their helpful and detailed remarks. The second author acknowledges the support of an NSF graduate fellowship and of the Lucent Technologies GRPW program.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, 2000.
2. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, vol. 1880 of *LNCS*, pp. 255–270. Springer Verlag, 2000.
3. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. Manuscript, 2001.
4. M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 236–250. Springer Verlag, 1998.
5. J. Black, P. Rogaway, and T. Shrimpton. Encryption scheme security in the presence of key-dependent messages. Manuscript, 2001.
6. F. Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 431–444. Springer Verlag, 2000.
7. S. Brands. Untraceable Off-line Cash in Wallets With Observers. In *CRYPTO '93*, vol. of *LNCS*. pp. 302–318. Springer Verlag, 1993.
8. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy* . PhD thesis, Eindhoven Institute of Technology, the Netherlands, 1999.
9. E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *Proc. ACM-SIAMs*, pp. 457–466. ACM press, 1995.
10. J. Camenisch and I. Damgård. Verifiable encryption and applications to group signatures and signature sharing. Technical Report RS-98-32, BRICS, Departement of Computer Science, University of Aarhus, December 1998.
11. J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. Technical Report Research Report RZ 3295, IBM Research Division, 2000.
12. J. Camenisch and A. Lysyanskaya. An Efficient Non-transferable Anonymous Credential System with Optional Anonymity Revocation. http://eprint.iacr.org/2001.
13. J. Camenisch and M. Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 107–122.

14. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424. Springer Verlag, 1997.
15. R. Canetti. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, 1995.
16. R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
17. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, pp. 199–203. Plenum Press, 1983.
18. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.
19. D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO '86*, vol. 263 of *LNCS*, pp. 118–167. Springer-Verlag, 1987.
20. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265. Springer-Verlag, 1991.
21. L. Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, vol. 1029 of *LNCS*, pp. 232–243. Springer Verlag, 1995.
22. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 13–25, 1998, Springer Verlag.
23. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proc. 6th ACM CCS*, pp. 46–52. ACM press, 1999.
24. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 431–444. Springer Verlag, 2000.
25. I. Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In *CRYPTO '88*, vol. 403 of *LNCS*, pp. 328–335.
26. C. Dwork, J. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information. In *Proc. 28th STOC*, 1996.
27. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO '84*, vol. 196 of *LNCS*, pp. 10–18. Springer Verlag, 1985.
28. A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194, 1987.
29. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, vol. 1294 of *LNCS*, pp. 16–30, 1997.
30. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 123–139, 1999.
31. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proc. 27th FOCS*, pages 291–304, 1985.
32. O. Goldreich, B. Pfitzman, and R. Rivest. Self-delegation with controlled propagation—or—what if you lose your laptop. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 153–168, 1998.
33. J. Kilian and E. Petrank. Identity escrow. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 169–185, Springer Verlag, 1998.
34. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, vol. 1758 of *LNCS*. Springer Verlag, 1999.
35. S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, 1988.
36. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM CCS*, pp. 245–254. ACM press, 2000.
37. M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In *EUROCRYPT '95*, vol. 921 of *LNCS*, pp. 209–219. Springer Verlag, 1995.