RZ 3295 (# 93341) 11/27/00
Computer Science 32 pages

Research Report

Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation

J. Camenisch

IBM Research
Zurich Research Laboratory
Säumerstrasse 4
8803 Rüschlikon
Switzerland
jca@zurich.ibm.com

A. Lysyanskaya

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

Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation

J. Camenisch

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland
A. Lysyanskaya

MIT LCS, 545 Technology Square, Cambridge, MA 02139, USA

Abstract

A credential system is a system in which users can obtain credentials from organizations and can demonstrate possession of these credentials. Such a system is anonymous when transactions carried out by the same user cannot be related. 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 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) We suggest more effective means of preventing users from sharing their identity, by introducing *all-or-nothing* sharing: a user who allows a friend to use her identity once, gives him the ability to use all of her credentials. This is implemented by a new cryptographic primitive, called *circular encryption*, which is of independent interest. (3) To prevent misuse of anonymity, our scheme is the first to offer optional anonymity revocation for particular transactions. (4) Our scheme offers separability: all organizations can choose their cryptographic keys independently of each other.

Keywords. Privacy protection, credential system, e-cash, blind-signatures, circular encryption.

1 Introduction

As information gets increasingly accessible, protecting the privacy of individuals becomes an increasingly challenging task. To solve this problem, an application that allows the individual to control the dissemination of information about him is needed. An anonymous credential system (also called pseudonym system), introduced by Chaum [1], is the best known example of such a system. In this paper, we propose a new anonymous credential system, considerably superior to previously proposed ones, that will solve the problem of protecting privacy. Privacy comes essentially for free: the communication and computation costs of our solution are small.

An anonymous credential system [1, 2, 3, 4, 5] consists of users and organizations. Organization know 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 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 the same pseudonym. An example of how such a credential system can be used is given in the appendix.

Desirable properties. Each pseudonym and credential must belong to some well-defined user [5]. 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. A user should be discouraged from sharing his pseudonyms and credentials with other users. The previously known way of discouraging the user from doing this was by *PKI-assured* identity integrity. Identity integrity is PKI-assured when sharing a credential implies also sharing a particular, valuable secret key from *outside* the system (e.g., the secret key that gives the user access to his bank account) [6, 7, 5]. However, such a valuable key does not always exist. Thus we introduce an alternative, novel way of achieving this: the *all-or-nothing* identity integrity property. Identity integrity is all-or-nothing when 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 identity integrity are somewhat orthogonal: 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 his credential (this feature, called *local anonymity revocation*, is also optional). 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. This set of properties guarantees security of credentials and protects organizations against misbehaving organizations and users.

Further desirable properties are those properties that guarantee 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, and the user remains anonymous [8, 9, 1, 2, 4, 3, 5]. These properties guarantee privacy for the users.

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. A one-show credential 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 setting with multiple users who, while remaining anonymous to the organizations, manage to transfer credentials from one organization to another, was first introduced by Chaum [1]. Subsequently, Chaum and Evertse [2] 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 subsequently proposed by Damgård [4] 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 [3] is based on discrete-logarithm-based blind signatures. It is efficient but does not address the problem of colluding users, too. 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 [5] 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 [3]: a user needs to obtain several signatures from the issuing organization in order to unlinkably use a credential several times.

Other related work is that of Brands [8, 9] 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 this scheme need to share the same discrete logarithm group.

The concept of revocable anonymity is found in electronic payment systems (e.g., [10, 11]), group signature schemes (e.g., [12, 13, 14]), and identity escrow [15].

Our solution. This is the first time that a system is proposed that satisfies all the desirable properties listed above. Our solution is based on the decisional Diffie-Hellman and strong RSA assumptions and is provably secure in the random oracle model. Without the random oracle model, we still obtain a practical, PKI-assured credential system. That is (1) our method for achieving all-or-nothing identity integrity will not work; and (2) an organization does not get a receipt of the credential showing.

Our solution is practical. When using an RSA modulus n of 1024 bits, a credential-pseudonym pair is about 4K bits, proving possession of a credential requires about 22 exponentiations in $\mathbb{Z}_n^*$ for both parties and can be done in three rounds. Establishing a pseudonym is somewhat less efficient: it takes about 200 exponentiations in $\mathbb{Z}_n^*$ for both parties, but batch-verification technology could be applied to reduce this, and organizations have to store about 25K bits per user (here computation complexity could be traded against storage).

Prior to our work, the problem of constructing a practical system with multiple-use credentials eluded researchers for some time [8, 9, 3, 4, 5]. We solve it by extending ideas found in the constructions of strong-RSA-based signature schemes [16, 17] and group signature schemes [12].

Discouraging users from sharing credentials by enforcing that sharing just one credential once implies granting access to all of them, is a new idea that was not considered before. This property guarantees that even in the absence of particularly valuable external secrets (which is the case today as there is no universally used public-key infrastructure), a user can still be motivated not to share his identity. The solution is based on a new primitive we call *circular encryption*; 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). 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 access to 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. 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.

Finally, our definitions, although not conceptually new and inspired by the literature on multi-party computation [18, 19], and reactive systems [20] are of interest, since our treatment is more formal than the one usually encountered in the literature on credential and electronic cash systems.

2 Formal Definitions and Requirements

A credential system has the following types of players:

Users: These are entities that receive credentials. The set of users in the system grows over time.

Organizations: These are entities that grant and verify the credentials of the users. Associated with each organization there is a unique (for simplicity of exposition) type of credential it grants. For simplicity, we assume that the set of organizations is fixed in the beginning.

Verifiers: These are entities that verify credentials of the users.

Trusted parties: These are entities that are trusted to perform well specified operations correctly. Specifically, there is a *certification authority (CA)* which is responsible for admitting only valid users into the system: and *anonymity revocation managers* that are responsible for establishing information about a user's identity or pseudonym in case it is needed. There are two revocation managers per organization. R_O is responsible for *global* anonymity revocation of credentials granted by organization O, i.e., if a user shows a credential from O, and then later his identity must be established, R_O can do it. R_O^l is responsible for *local* anonymity revocation, i.e., if a user shows a credential from O, and later his credential must be traced to the pseudonym to which it was issued, R_O^l can do it.

Variations of such a system allow a single organization to issue different types of credentials; to join and leave the system, etc. As the trusted parties perform tasks that are not required frequently, there parties can be implemented in a distributed fashion to weaken the trust assumption. These variations are simple to handle; for simplicity of exposition, however, we do not discuss them here. We note that the solution we propose can be easily adapted to incorporate them.

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

Ideal credential system. For the ideal system, we assume an ideal trusted party T as intermediator that is responsible for the necessary properties of the system. All transactions are made via T. T also ensures anonymity of the users towards the organizations and verifiers. We assume that there are perfectly secret and authenticated channels between T and all parties.

To initialize the system, the organizations create a public file which, for each organization O, describes the type of credential that the organization grants, and its anonymity revocation manager(s) R_O and R_O^l.

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. If the credential system is realized in the PKI model, then all users get their secret keys from the PKI as inputs.

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, transactions ids, 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.

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.

The system supports the following operations by players with T (they are described in more detail in Table 1 which is found in the appendix).

Basic operations: Operations $\mathsf{AddUser}(U, CA)$, $\mathsf{FormNym}(U, O)$, $\mathsf{GrantCred}(U, O)$, $\mathsf{VerifyCred}(U, V)$, and $\mathsf{VerifyCredOnNym}(U, O)$ are basic operations that are necessary for any credential system. They are initiated by a user by sending a request to T who contacts the other involved party and asks it whether it accepts the request or not and then acts accordingly.

Operations for anonymity revocation: $\mathsf{AnonRev}$ (resp., $\mathsf{LocalAnonRev}$) allow organization and verifier to find out the identity (resp., pseudonym) of the user who proved possession of a credential in a given transaction. Upon such a request, T ask the anonymity revocation manager R_O (resp., R_O^l) whether it agrees to revoke or not. An honest revocation manager will only accept when some condition, specified at credential showing time, is satisfied.

Operations for PKI-assured identity integrity: Our model allows the users to give their keys to their friends so that they too will have access to credentials. The reason for it is that any digital information can be duplicated and so any real-world system will have this problem. Existence of operations AddUserWithPKI(U, CA) and FurnishKey(U) motivates the users not to share their keys, assuming an external public-key infrastructure and a CA who is not corrupted.

Operations for all-or-nothing identity integrity: Alternatively, having operation TellAll(U) ensures that either the user shares none of his credentials, or he shares all of them.

Operations for corrupted parties: To model unavoidable misbehavior by corrupted parties, there are IIAnonRev, IILocAnonRev, and IIAddUser. These are queries to T by corrupted revocation managers and organizations. The first two allow a corrupted revocation manager to retrieve the identity or pseudonym of the user involved in some transaction. The last one allows a corrupted organization to establish pseudonyms with and issue credentials to illegal users, i.e., users whose identity is not known to anyone. This models the fact that a corrupted organization can always grant a credential to users that did not join the system properly. (Of course, an honest verifier will detect an illegal user when it asks him for a credential by the CA, assuming that the CA is honest).

One-show credentials: These are just like regular (i.e., multi-show) credentials, except that a user can only use a credential once. If she attempts to use it the second time, her identity is discovered. Achieved by modifying GrantCred(U, O), VerifyCred(U, V), and VerifyCredOnNym(U, O) appropriately.

Expiration dates and other attributes: These are just like regular credentials, except that, when granting such a credential, the issuing organization specifies attributes for it; when the credential is used, the user also requests that T verify an attribute of a credential (e.g., verify that it has not expired yet).

This ideal system captures the intuitive requirements, such as unforgeability of credentials, anonymity of users, unlinkability of credential showings, anonymity revocation with no framing.

Note that this model does not protect the organizations against a corrupted CA who may ask to add a user who should not be part of the system (e.g., a user who has not paid a membership fee, or a user who does not have a public key in the outside public-key infrastructure.) This limitation corresponds to the limitation one expects to encounter in any cryptographic implementation of such a system.

The ideal-world and real-world adversaries. The ideal-world adversary is a probabilistic polynomial-time machine that gets control over the corrupted parties in the ideal world. He receives the following information: the number of users and organizations and the number of users that have valid pseudonyms with an organization. Finally, the adversary can trigger an event as described above.

Similarly, the real-world adversary a probabilistic polynomial-time machine gets full control over the corrupted parties in the real-world, and can trigger events in the real-world.

Security definition. Let an ideal credential system be called ICS, and its cryptographic implementation be denoted CCS. Let $V = \texttt{poly}(k)$ be the number of players in the system with security parameter k. Let Z_i denote the output of the i-th player. 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:

- In the *ICS*, $\mathcal{S}$ controls the players in the ideal-world corresponding to those real world players controlled by $\mathcal{A}$.
- The set of events $\mathcal{S}$ triggers in *ICS* is identical to the set of events that $\mathcal{A}$ triggers in *CCS*.
- We have that

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

  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 The Big Picture of Our Solution

In this section, we give an informal explanation of how our anonymous credential system is realized and why it is provably secure. This section is a preview for the more technical sections that follow.

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 $N_{(U,O)}$, will be tagged with a value $P_{(U,0)}$. 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 that the user and organization contribute randomness to, but only the user knows its value. An appropriate choice of parameters for the length of x_U and $s_{(U,O)}$ guarantees that the resulting $P_{(U,O)}$ is statistically independent of the user's key x_U and of any other pseudonyms formed by the same user with other organizations. In order to ensure the all-or-nothing property, the value $s_{(U,O)}$ must be made available to anyone who knows x_U. This is realized using *key-oblivious verifiable circular encryption* described in Section 4.2.

A credential issued by organization O to a pseudonym $N_{(U,O)}$ is a tuple $(e_{(U,O)}, c_{(U,O)})$ such that $e_{(U,O)}$ is a prime, and $c_{(U,O)}^{e_{(U,O)}} \equiv P_{(U,O)} d_O \pmod{n_O}$. Under the strong RSA assumption, tuples of this form for correctly formed tags cannot be existentially forged even by an adaptive attack (Lemma 8).

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. This 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 users that fully trust each other cannot pool their credentials. In the next section, we will present the machinery needed for asserting such statements.

Optional global or local anonymity revocation is achieved by including an encryption of the user's identity or validating tag under the revocation manager's public key; and a proof that this identity or validating tag corresponds to the validating tag to which ownership of a credential is proved.

4 Preliminaries

Our construction is based on the decisional Diffie-Hellman assumption and the strong RSA assumption (a description of the strong RSA problem is found in Appendix A). It is provably secure in the random oracle model. In this section, we outline some of the less well-known technical points that will be used when we describe our protocols. By $\mathtt{neg}(k)$ we denote any function that vanishes faster than any inverse polynomial.

Proof of knowledge of the discrete logarithm. Let $G = \langle g \rangle$ denote a group of prime order q. A basic protocol we use is the well-known proof of knowledge of the discrete logarithm of some group element $y \in G$ to the base g [21, 22]. This protocol is zero-knowledge in the auxiliary string model of Damgård [23]. Using notation from [13], this protocol is denoted $PK\{(\alpha) : y = g^\alpha\}$, where Greek letters denote the quantity the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, the proof can be described by just pointing out its aim while hiding all details.

In the random oracle model, this protocol can be turned into a signature schemes using Fiat-Shamir heuristic [24]. We use the notation $SPK\{(\alpha) : y = g^\alpha\}(m)$ to denote a signature obtained in this way.

Proof of knowledge of the discrete logarithm modulo a composite. 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 [25]. Second, the largest possible value $2^k - 1$ of the challenge c must be smaller that the smallest factor of G's order. This issue can be addressed by assuring that n is the product of two equal-sized safe primes, i.e., primes p and q such that $p' = (p-1)/2$ and $q' = (q-1)/2$ are prime (such p' and q' are called Sophie Germain primes). Then the order of QR_n will be $p'q'$ and values $k < (\log \sqrt{n}) - 4$ are fine. Third, soundness needs special attention in case the verifier is not equipped with the factorization of n because then deciding membership of 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 [24]. 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 a quadratic residue.

Other proofs in a fixed group. A proof of knowledge of a representation of an element $y \in G$ with respect to several bases $z_1, \ldots, z_v \in G$ [21] is denoted $PK\{(\alpha_1, \ldots, \alpha_v) : y = z_1^{\alpha_1} \cdot \ldots \cdot z_v^{\alpha_v}\}$. A proof of equality of discrete logarithms of two group elements $y_1, y_2 \in G$ to the bases $g \in G$ and $h \in G$, respectively, [26, 27] is denoted $PK\{(\alpha) : y_1 = g^\alpha \wedge y_2 = h^\alpha\}$. Generalizations to prove equalities among representations of the elements $y_1, \ldots, y_w \in G$ to bases $g_1, \ldots, g_v \in G$ are straight forward [13]. A proof of knowledge of a discrete logarithm of $y \in G$ with respect to $g \in G$ such that $\log_g y$ that lies in the integer interval $[a, b]$ is denoted by $PK\{(\alpha) : y = g^\alpha \ \wedge \ \alpha \in [a, b]\}$. Under the strong RSA assumption and if it is assured that the prover is not provided the factorization of the modulus (i.e., is not provided the order of the group) this proof can be done efficiently [28] (it compares to about six ordinary $PK\{(\alpha) : y = g^\alpha\}$.)

Proofs of knowledge or equality in different groups. The previous protocol can also be used to prove that the discrete logarithms of two group elements $y_1 \in G_1, y_2 \in G_1$ to the bases $g_1 \in G_1$ and $g_2 \in G_2$ in *different* groups G_1 and G_2 are equal [29, 30]. Let the order of the groups be q_1 and q_2, respectively. This proof can be realized only if both discrete logarithms lie in the interval $[0, \min\{q_1, q_2\}]$. The idea is that the prover commits to the discrete logarithm in some group, say $G = \langle g \rangle = \langle h \rangle$ the order of which he does not know, and then execute $PK\{(\alpha, \beta) : y_1 \overset{G_1}{=} g_1^{\alpha} \wedge y_2 \overset{G_2}{=} g_2^{\alpha} \wedge C \overset{G}{=} g^{\alpha} h^{\beta} \wedge \alpha \in [0, \min\{q_1, q_2\}]\}$, where C is the commitment. This protocol generalizes to several different groups, to representations, and to arbitrary modular relations.

4.1 Circular Encryption

Our scheme requires each user to encrypt each of her secret keys D_i under one of her public keys E_j, thereby creating "circular encryptions". However, the definition of a semantically secure encryption scheme does not provide security for such encryptions. Moreover, it is not known whether circular security is possible under general assumptions. In this section we introduce a new cryptographic primitive called *circular encryption* which is an encryption scheme that provides security for circular encryptions. We provide a generic construction of such a scheme and prove its security in the random oracle model given any semantically secure encryption scheme.

Definition 1. *A semantically secure encryption scheme $\mathcal{G} = (\mathcal{E}, \mathcal{D})$ on message space $\{0,1\}^{\ell}$, is circular-secure if for all probabilistic polynomial time families of Turing machines $\{A_k\}$, for all sufficiently large k, for all $n = \mathtt{poly}(k)$, for all assignments to $(i_1, \ldots, i_n)$ and $(j_1, \ldots, j_n)$,*

$$|\Pr[A_k(C, E_1, \ldots, E_n) = 0 | (E_1, D_1) \in_R \mathcal{G}, \ldots, (E_n, D_n) \in_R \mathcal{G}; C = (E_{i_1}(0), \ldots, E_{i_n}(0))] - \Pr[A_k(C, E_1, \ldots, E_n) = 0 | (E_1, D_1) \in_R \mathcal{G}, \ldots, (E_n, D_n) \in_R \mathcal{G}; C = (E_{i_1}(D_{j_1}), \ldots, E_{i_n}(D_{j_n}))]| = \mathtt{neg}(k)$$

(I.e., having access to encryptions of the secret keys does not help the adversary in breaking the semantic security of the system.)

Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a random oracle, and let $\oplus$ denote the bitwise XOR operation. Let $\mathcal{G} = (\mathcal{E}, \mathcal{D})$ be a semantically secure cryptosystem. Construct $\mathcal{G}' = (\mathcal{E}', \mathcal{D}')$ as follows: generate (E, D) according to $\mathcal{G}$. To encrypt, a message $m \in \{0,1\}^k$, 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 $\tilde{m} := \mathcal{H}(D(a)) \oplus b$.

Theorem 1. *If $\mathcal{G}$ is semantically secure, $\mathcal{G}'$ is circular-secure.*

The proof of the theorem can be found in the appendix.

As a basis for our circular encryption scheme, we use the ElGamal encryption [31] where the public key is $P = g^x$, where g is a generator of some group. The ElGamal cryptosystem is known to be semantically secure under the decisional Diffie-Hellman assumption. The resulting circular encryption scheme is as follows. Let the order of $G = \langle g \rangle$ be $\approx 2^{\ell}$. 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 $\mathcal{H}\left(\frac{u}{v^x}\right) \oplus z$. We denote this encryption scheme by CEIG.

4.2 Verifiable encryption with a committed public key

In order to guarantee the all-or-nothing property, we must make sure a user encrypts all of her secret keys under some master public key. Then, if she gives away just this one key, she gives away all of her pseudonyms and credentials. Verifiable encryption is a building block that makes sure that a given ciphertext is an encryption of secret information for a given pseudonym.

Verifiable encryption [32, 33], 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 $(s, v) \in \mathcal{R}$. For instance, $\mathcal{R}$ could be the relation $(s, g^s) \subset \mathbb{Z}_q \times G$. More formally,

Definition 2. *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*

- *If $\mathcal{P}$ and $\mathcal{V}$ are honest then $\mathcal{V}^{\mathcal{P}(\mathcal{R},E,s,v)}(\mathcal{R}, E, v) \neq \perp \forall (E, D) \in \mathcal{G}(1^k)$ and for all $x \in (s, v) \in \mathcal{R}$.*
- *There is an efficient algorithm C and c, k_0 such that $\forall \widetilde{\mathcal{P}}$, $\forall k > k_0$, and $\forall (E, D) \in (\mathcal{E}, \mathcal{D})(1^k)$*
$$\Pr[(C(D, e), v) \in \mathcal{R} \mid e = \widetilde{\mathcal{V}}^{\widetilde{\mathcal{P}}(\mathcal{R},E,s,v)}(\mathcal{R}, E, v) \wedge e \neq \perp] = 1 - \mathtt{neg}(k).$$
- *There is a black-box simulator $\mathcal{S}$ such that $\forall\, \widetilde{\mathcal{V}}$, $\forall v$ we have $\widetilde{\mathcal{S}}^{\widetilde{\mathcal{V}}(\mathcal{R},E,v)}(\mathcal{R}, E, v) \stackrel{c}{\approx} \widetilde{\mathcal{V}}^{\mathcal{P}(\mathcal{R},E,s,v)}(\mathcal{R}, E, v)$.*

Generalizing the protocol of Asokan et al. [32], Camenisch and Damgård [33] provide a 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 from the previous section in particular. 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., $e := VE(\mathsf{ElGamal}, (g, y))\{\xi : v = g^\xi\}$ the verifiable encryption protocol for the ElGamal scheme, whereby $\log_g v$ is encrypted in e under public key (y, g). Note that e is not a single encryption, but the verifier's entire transcript of the protocol and contains several encryptions, commitments and responses of the underlying *PK*.

For guaranteeing the all-or-nothing identity integrity property, we need to have each user verifiable encrypt all of his secret information under a public key that corresponds to his 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. Thus a verifiable encryption property must be *key*-oblivious:

Definition 3. *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, whenever $(E, D) \in_R (\mathcal{E}, \mathcal{D})(1^k)$ and another public key E' is chosen at random, for all polynomially bounded $\mathcal{W}$, we have $\mathcal{W}^{\mathcal{P}(s,v)}(v, E) \stackrel{c}{\approx} \mathcal{W}^{\mathcal{P}(s,v)}(v, E')$.*

In case the verifier does not know the encryption public key, previously known constructions do not work, since 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 is works for our construction. Let $P = g^x$ serve as a public key, and x is 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, a *key*-oblivious verifiable encryption scheme is obtained.

In the sequel, we write, e.g., $VE(\mathsf{CEIG}, (\mathcal{H}, a, b, g, C))\{(\xi) : v = g^\xi\}$ for this *key*-oblivious verifiable encryption. Note the following lemma, which is not difficult to prove:

Lemma 2. *Under the decisional Diffie-Hellman assumption, the verifiable encryption scheme described above is key-oblivious.*

5 Anonymous Credential System

This section describes our basic pseudonym system with all-or-nothing and PKI-assured non-transferability. The basic system comprises protocols for a user to join the system, register with an organization, obtain multi-show credentials, and show such credentials. Extensions that allow for one-show credentials as well and for revocability are described in Appendix 6 and the scheme's security is proved in Appendix 7.

Throughout we assume that the users and organizations are connected by perfectly anonymous channels. Furthermore, we assume that for each protocol an organization authenticates self 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 participating parties of this and stops.

In the following we use CA and O_0 as interchangeable names for the pseudonym system's certification authority.

5.1 High-Level Description

System setup. The system parameters are agreed upon and all organizations (including the CA) choose their keys and make the public keys available. It is possible that organizations (apart from the CA) join and leave at any time. Let Γ denote the range from which users pick their secret keys.

Adding user U to the system. To join the system, user U identifies and authenticates himself to the CA. The CA checks that the user is eligible to join the system. Once this is completed, the user chooses a master secret key $x_U \in \Gamma$. The user and the CA run Protocol 1 to establish U's pseudonym $N_{(U,CA)}$ with the CA. This pseudonym must be based on the user's master key x_U. The CA will then grant a credential to $N_{(U,CA)}$, by running Protocol 2, attesting to the fact that the user has completed the registration process.

U's registration with an organization O. In order to register a pseudonym with an organization O, U must approach it and together they execute the following: First, they run Protocol 1

to establish U's pseudonym $N_{(U,O)}$. Now the user must demonstrate to O that she is a valid participant in the system. This is carried out by running Protocol 4 for demonstrating possession of a credential issued by the *CA*. Protocol 4 assures that the *CA*-credential was issued to a user with the same underlying secret key x_U as the key underlying pseudonym $N_{(U,O)}$. Step 1:6 in Protocol 1 guarantees that if a user lets his friend know his secret key x_U, he will thereby enable his friend to use the newly created pseudonym and, should the organization issue it, the corresponding credential.

Granting a credential to U. Once the user has registered with organization O, O might wish to grant a credential to U. To do so, O and U run Protocol 2. (Note that O may require that the user demonstrate possession of credentials from other organizations before it chooses to grant a credential.)

Proving possession of a credential from O_I to organization O_V. In order to demonstrate that she has a credential from O_I to another organization, O_V, the user U and organization O_V run Protocol 4. This protocol demonstrates to O_V that the same key x_U that underlies $N_{(U,O_V)}$, also underlies some pseudonym $N_{(U,O_I)}$ to which organization O_I has issued a credential.

Proving possession of a credential from O_I to verifier V. U and V run just Protocol 3.

Proving possession of a set of credentials to verifier V. In case U is required to hold credential from a set $\mathcal{O}$ of organizations, U must first establish a pseudonym with V, i.e., run Protocol 1, and then run Protocol 4 for each $O_j \in \mathcal{O}$. If it is understood that the established pseudonym is one-time, i.e., if U is not allowed to access the service again just be proving ownership of the established pseudonym, U and V need not execute the step 1:6 in Protocol 1.

Note that it is important that all pseudonyms and credentials of the same user be based on the same secret key x_U. Identity integrity is guaranteed in two ways: (1) PKI-assured way: the *CA* makes sure that the user's secret key x_U is the same as or related to the key used in the PKI; (2) all-or-nothing identity integrity: step 1:6 in Protocol 1 ensures that sharing the key x_U with a friend enables the friend to take full advantage of all of U's pseudonyms and credentials.

5.2 System parameter and key generation.

For simplicity we assume some common system parameter: the length of 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 $\ell_\Delta = \epsilon\ell_\Gamma$ and $\ell_\Gamma = 2\ell_n$, where $\epsilon > 1$ is a security parameter, and $2^{\ell_\Lambda} > 2(2^{2\ell_\Gamma} + 2^{\ell_\Gamma} + 2^{\ell_\Delta})$, and $2(2^{\ell_\Sigma}(2^{2\ell_\Gamma} + 2^{\ell_\Delta}) + 2^{\ell_\Delta}) < 2^{\ell_\Lambda}$.

Each organization O_i (including the *CA*) chooses random $\ell_n/2$-bit primes p'_{O_i}, q'_{O_i} such that $p_{O_i} = 2p' + 1$ and $q_{O_i} = 2q' + 1$ 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 keys 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 together with a proof that n_{O_i} is the product of two safe primes (see [34] for how the latter can be done) and that the elements $a_{O_i}, b_{O_i}, d_{O_i}, g_{O_i}, h_{O_i}$ lie indeed in $QR_{n_{O_i}}$ (this can be done by providing their roots; then, to check that an element s has order at least $p'q'$, one needs only to test whether $\gcd(s \pm 1, n_{O_i}) = 1$).

Finally, 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.

5.3 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_i. Let $x_U \in \Gamma$ be U's master secret. In case O_i is the CA, i.e., U has no yet entered the system, U needs to choose a random $x_U \in_R \Gamma$ before entering the protocol below. The protocol assures that the pseudonym's validating tag is of the right form, i.e., $P_{(U,O_i)} = a_{O_i}^{x_U} b_{O_i}^{s_{(U,O_i)}}$, with $x_U \in \Gamma$ and $s_{(U,O_i)} \in \Delta$. The value $s_{(U,O_i)}$ is chosen jointly by O_i and U without O_i learning anything about either values. Note that this protocol does not assure that U uses the same x_U as with other organizations as well; this is taken care of later in Protocol 4.

Protocol 1.

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 $C_1 := g_{O_i}^{r_1} h_{O_i}^{r_2}$, $C_2 := g_{O_i}^{x_U} h_{O_i}^{r_3}$. U sends N_1, C_1 to O_i.

1:2. To prove that step 1:1 was executed correctly, U serves as the prover to verifier O_i in

$$PK\{(\alpha,\beta,\gamma,\delta) : C_1^2 = (g_{O_i}^2)^\alpha (h_{O_i}^2)^\beta \ \wedge \ C_2^2 = (g_{O_i}^2)^\gamma (h_{O_i}^2)^\delta\}$$

1:3. O_i chooses a random $r \in_R \Delta$ and a value N_2 and sends r,N_2 to U.

1:4. U sets her pseudonym $N_{(U,O)} := N_1 \| N_2$ ($\|$ denotes concatenation). U computes

$$s_{(U,O_i)} = (r_1 + r \bmod (2^{\ell_\Delta+1} + 1)) - 2^{\ell_\Delta} + 1,$$

(essentially, $s_{(U,O)}$ is the sum of r_1 and r, adjusted appropriately so as to fall in the interval Δ) and $\tilde{s} = \left\lfloor \frac{r_1 + r}{2^{\ell_\Delta+1}+1} \right\rfloor$ (essentially, $\tilde{s}$ is the value of the carry resulting from the computation of $s_{(U,O)}$ above). U then sets her validating tag $P_{(U,O_i)} := a_{O_i}^{x_U} b_{O_i}^{s_{(U,O_i)}}$ and sends $P_{(U,O)}$ to O_i.

1:5. Now, U must show that $P_{(U,O)}$ was formed correctly. To that end, she chooses $r_4 \in_R \{0,1\}^{\ell_n}$ and $r_5 \in_R \mathbb{Z}_q$, sets $C_3 := g_{O_i}^{\tilde{s}} h_{O_i}^{r_4}$, and $C := g^{x_U} h^{r_5}$, and sends $P_{(U,O_i)}$, C_3, and C to O_i.

U proves to O_i that the values in step 1:4 were chosen correctly by executing

$$\begin{aligned} PK\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\vartheta,\xi,\varphi) : C_1^2 = (g_{O_i}^2)^\alpha (h_{O_i}^2)^\beta \wedge C_2^2 = (g_{O_i}^2)^\gamma (h_{O_i}^2)^\delta \wedge C_3^2 = (g_{O_i}^2)^\varepsilon (h_{O_i}^2)^\zeta \wedge \\ P_{(U,O_i)}^2 = (a_{O_i}^2)^\gamma (b_{O_i}^2)^\vartheta \ \wedge \ (C_1^2 (g_{O_i}^2)^{(r-2^{\ell_\Delta+1})}) / (C_3^2)^{(2^{\ell_\Delta+1}+1)} = (g_{O_i}^2)^\vartheta (h_{O_i}^2)^\xi \ \wedge \\ C = g^\gamma h^\varphi \ \wedge \ \gamma \in \Gamma \ \wedge \ \vartheta \in \Delta\} \end{aligned}$$

1:6. To guarantee the all-or-nothing property, U and O_i engage in the verifiable encryption protocol

$$w_{P_{(U,O_i)}} = VE(\mathsf{CEIG}, (\mathcal{H}, g, h, C))\{(\alpha,\beta) : P_{(U,O_i)}^2 = (a_{O_i}^2)^\alpha (b_{O_i}^2)^\beta\} \ .$$

1:7. O_i stores $N_{(U,O_i)}$, $P_{(U,O_i)}^2$ and $P_{(U,O_i)}$. Publish $N_{(U,O_i)}$, $P_{(U,O_i)}$, and $w_{(P_{(U,O_i)}}$.

1:8. U stores $N_{(U,O)}$, $P_{(U,O_i)}^2$, $P_{(U,O_i)}$, and $s_{(U,O_i)}$.

In case we want to have PKI-assured non-transferability only, the step 1:6 and 4:2 can be omitted.

5.4 Generation of a credential.

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

Protocol 2.

2:1. U sends $(N_{(U,O_i)}, P_{(U,O_i)})$ to O_i and identifies as its owner by $PK\{(\alpha, \beta) : P^2_{(U,O_i)} = (a^2_{O_i})^\alpha (b^2_{O_i})^\beta\}$.

2:2. O_i makes sure $(N_{(U,O_i)}, P_{(U,O_i)})$ is in its database, chooses a random prime $e_{(U,O_i)} \in_R \Lambda$, computes $c_{(U,O_i)} = (P_{(U,O_i)} d_{O_i})^{1/e_{(U,O_i)}} \mod n_{O_i}$, sends $(c_{(U,O_i)}, e_{(U,O_i)})$ to U, and publishes $(c_{(U,O_i)}, e_{(U,O_i)})$ in its record for $N_{(U,O_i)}$.

2:3. U checks if ${c_{(U,O_i)}}^{e_{(U,O_i)}} \equiv P_{(U,O_i)} d_{O_i} \pmod{n_{O_i}}$ and stores $(c_{(U,O_i)}, e_{(U,O_i)})$ in its record with organization O_i. The tuple $(P_{(U,O_i)}, c_{(U,O_i)}, e_{(U,O_i)})$ is called a *credential record*.

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

5.5 Showing a single credential.

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 a verifier V. Recall that $c_{(U,O_j)}^{e_{(U,O_j)}} = d_{O_j} P_{(U,O_j)}$. U and verifier V engage in the following protocol.

Protocol 3.

3:1. U chooses $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O_i)} h^{r_1}_{O_j}$ and $B = h^{r_1}_{O_j} g^{r_2}_{O_i}$, and sends A, B to V.

3:2. U engages with V in

$$PK\{(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \xi) : \; d^2_{O_j} = (A^2)^\alpha (1/(a^2_{O_j}))^\beta (1/(b^2_{O_j}))^\gamma (1/(h^2_{O_j}))^\delta \; \wedge$$
$$B^2 = (h^2_{O_j})^\varepsilon (g^2_{O_j})^\zeta \; \wedge \; 1 = (B^2)^\alpha (1/(h^2_{O_j}))^\delta (1/(g^2_{O_j}))^\xi \; \wedge \; \beta \in \Gamma \; \wedge \; \gamma \in \Delta \; \wedge \; \alpha \in \Lambda\} \; .$$

The *PK* in step 3:2 proves that U possesses a credential issued by O_j on some pseudonym registered with O_j. We refer to (the proof of) Lemma 5 for more details about this *PK*.

5.6 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 $P_{(U,O_i)}$. Moreover, the protocol also assures that if U gives the secrets of $P_{(U,O_i)}$ to one of her friends, then she would also reveal the secret keys of the pseudonym established with O_j and vice versa, whereby all-or-nothing transferability gets assured. Thus U and O_i engage in the following protocol (in which we assume that O_i has already established that $P_{(U,O_i)} \in QR_{n_{O_i}}$).

Protocol 4.

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 $N_{(U,O_i)}, A, B, C$ to O_i.

4:2. U engages with O_i in

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

The first two steps of this protocol are similar to the ones of Protocol 3, the difference being that here U also proves that the same master secret key is used in $P_{(U,O_i)}$ and 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 the transaction by turning step 3:2 (or step 4:2) into the corresponding *SPK* on the description of the transaction.

5.7 PKI-assured non-transferability.

In this section we show how to ensure that if a user gives away her master secret x_U, then she will also reveal the secret key of an "external" valuable public key PK_U. This is achieved by having the *CA* ask for this public key PK_U and check whether this is the user's public key (e.g., via some external certificate) and then require the user to verifiable encrypt the corresponding secret key. This verifiable encryption is then published.

For the semantically secure scheme underlying the verifiable encryption, we use a variant of ElGamal. The public key is of the form $K = a^x b^y \pmod n$, and the corresponding secret keys are x and y. A ciphertext for m is a tuple $(a^r, b^r, K^r m)$. We will call this variant of the ElGamal cryptosystem ElGamal2.

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.

Protocol 5.

5:1. U sends Y_U, g, and the certificate on Y_U of the external PKI to *CA* who checks their validity.

5:2. U and *CA* engage in $w_{(P_{(U,O_0)},PKI)} = VE(\mathsf{ElGamal2}, (\mathcal{H}, a_{O_0}^2, b_{O_0}^2, P_{(U,O_0)}^2))\{(\alpha) : Y_U = g^{\alpha}\}$.

5:3. *CA* publishes the list $(w_{(P_{(U,O_0)},CA)}, PKI)$.

6 Extensions

6.1 One Show Credentials

The credential 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. 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 [35]. Here, we provide a novel and alternative way to approach this problem, i.e., instead of blinding the signer we blind the verifier. This approach could also be used to implement anonymous e-cash (just consider a credential to be money).

In the following we describe the general idea how it is realized. We do not provide detailed descriptions of the changes to the protocols that need to be made.

- Each organization publishes an additional generator $z_{O_i} \in QR_{n_{O_i}}$.

  A validating tag $P_{(U,O_i)}$ on a user's pseudonym $N_{(U,O_i)}$ is formed slightly differently: $P_{(U,O_i)} = a_{O_i}^{x_U} b_{O_i}^{s_{(U,O_i)}} z_{O_i}^{r_{(U,O_i)}}$, where $r_{(U,O_i)}$ is chosen by O_i and U together in the same way as is $s_{(U,O_i)}$.

  Credential are issued in the same way as before, i.e., U obtains $c_{(U,O_i)}$ and $e_{(U,O_i)}$ such that ${c_{(U,O_i)}}^{e_{(U,O_i)}} \equiv P_{(U,O_i)} d_{O_i} \pmod{n_{O_i}}$ holds.

- When proving possession of a one-show credential issued by O_j (with respect to a pseudonym or not), the user provides to verifier V (which might be an organization) the value $H_{(U,O_j)} = h_{O_j}^{r_{(U,O_j)}}$ and proves that it is formed correctly w.r.t. to the pseudonym U established with O_j. Of course, the various PK's in these protocols have to be adapted to reflect the different form of the pseudonym U holds with O_j. 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 $H_{(U,O_j)}$ 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., [36]).

We describe how the latter can be done such that using a one-show credential twice would expose the user's secret keys connected with 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 reply with $r = cx_U + s_{(U,O_j)}$ and prove correctness of this result. Now, we must have that $\ell_\Delta > \epsilon(\ell_\Gamma + \ell_c)$. Then r hides x_U statistically because $x_U \in \Gamma$ and $s_{(U,O_j)} \in \Delta$. 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 6.

6:1. U chooses $r_1, r_2 \in_R \{0,1\}^{2\ell_n}$, computes $A = c_{(U,O_i)} h_{O_j}^{r_1}$ and $B = h_{O_j}^{r_1} g_{O_i}^{r_2}$, and sends $A, B, H_{(U,O_j)}$ to V.

6:2. V chooses $c \in_R \{0,1\}^{\ell_c}$ and sends c to U.

6:3. U replies with $r = cx_U + s_{(U,O_j)}$ (in $\mathbb{Z}$).

6:4. U engages with V in

$$PK\{(\alpha,\beta,\gamma,\phi,\delta,\varepsilon,\zeta,\xi): \ d^2_{O_j} = (A^2)^{\alpha}(1/(a^2_{O_j}))^{\beta}(1/(b^2_{O_j}))^{\gamma}(1/(z^2_{O_j}))^{\phi}(1/(h^2_{O_j}))^{\delta} \ \wedge \\ B^2 = (h^2_{O_j})^{\varepsilon}(g^2_{O_j})^{\zeta} \ \wedge \ 1 = (B^2)^{\alpha}(1/(h^2_{O_j}))^{\delta}(1/(g^2_{O_j}))^{\xi} \ \wedge \\ H_{(U,O_j)} = h^{\phi}_{O_j} \ \wedge \ g^r_{O_j} = (g^c_{O_j})^{\beta} g^{\gamma}_{O_j} \ \wedge \ \beta \in \Gamma \ \wedge \ \gamma \in \Delta \ \wedge \ \alpha \in \Lambda\} \ .$$

The adaptions for one-show credentials with "built-in" anonymity revocation to Protocol 4 are similar.

6.2 Local and Global Revocation

To enable local and global revocation each organization needs a local and a global revocation manager. Given the transaction of a protocol where some user proved possession of a credential issued by organization O_i, this organization's revocation manager R^l_i will have the task to reveal the pseudonym under which the user is known to O_i and R_i will have the task to prove information that the CA can link to the identity of a user. Each of these managers needs to choose keys of some public key encryption scheme secure against adaptive chosen ciphertext attack. The managers' public keys become part of the respective organizations' public keys. In the following we describe how the protocols for proving possession of a credential must be adapted such that local revocation is possible using Cramer-Shoup encryption [37]. 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.

Setup: Each R^l_i chooses $g_{R^l_i}, h_{R^l_i} \in QR_{R^l_i}n$, and five secret keys $x_{(1,R^l_i)}, \ldots, x_{(5,R^l_i)} \in \Gamma$ and computes $(y_{(1,R^l_i)}, y_{(2,R^l_i)}, y_{(3,R^l_i)}) := (g^{x_{(1,R^l_i)}}_{R^l_i} h^{x_{(2,R^l_i)}}_{R^l_i}, g^{x_{(3,R^l_i)}}_{R^l_i} h^{x_{(4,R^l_i)}}_{R^l_i}, g^{x_{(5,R^l_i)}}_{R^l_i})$ as his public key. This is the Cramer-Shoup cryptosystem modulo a composite.

Each R_i chooses a group and five secret keys $x_{(1,R_i)}, \ldots, x_{(5,R_i)} \in_R \mathbb{Z}_{q_{R_i}}$ and computes $(y_{(1,R_i)}, \ y_{(2,R_i)}, \ y_{(3,R_i)}) := (g^{x_{(1,R_i)}} h^{x_{(2,R_i)}}, g^e x_{(3,R_i)} h^{x_{(4,R_i)}}, g^{x_{(5,R_i)}})$ as his public key. This is Cramer-Shoup cryptosystem in G (cf. setup procedure in Section 5).

Protocol 1 is extended by the following when the user registers with the CA.

1:9. U computes $P_U = g^{x_U}$ otherwise. U sends P_U to CA.

1:10. U engages with CA in

$$PK\{(\alpha,\beta): P^2_{(U,CA)} = (a^2_{CA})^{\alpha}(b^2_{CA})^{\beta} \ \wedge \ P_U = g^{\alpha}\}$$

otherwise.

1:11. Both CA and U store P_U with $P_{(U,CA)}$.

Let m_j be some agreed-upon text describing under which condition R_j is allowed to revoke the anonymity of the transaction of which the current execution of Protocol 3 or 4 is part of, respectively. m_j can include the values already used in those protocols. Below, we provide the steps to be executed after Protocol 3 or 4 in order to get local and/or global revocation.

Protocol 7 (Local Revocation).

7:1. U chooses $r_1 \in_R \mathbb{Z}_n$ and computes the encryption of his validity tag $P_{(U,O)}$: $c_{(1,U,R_j^l)} := g_{R_j^l}^{r_1}$, $c_{(2,U,R_j^l)} := h_{R_j^l}^{r_1}$, $c_{(3,U,R_j^l)} := y_{(3,R_j^l)}^{r_1} P_{(U,O)}$, and $c_{(4,U,R_j^l)} := y_{(1,R_j^l)}^{r_1} y_{(2,R_j^l)}^{r_1 \mathcal{H}(c_{(1,U,R_j^l)}, c_{(2,U,R_j^l)}, c_{(3,U,R_j^l)}, m_j)}$ and sends $(c_{(1,U,R_j^l)}, c_{(2,U,R_j^l)}, c_{(3,U,R_j^l)}, c_{(4,U,R_j^l)})$ to V.

7:2. U and V engage in

$$PK\{(\alpha,\beta,\gamma,\delta,\varepsilon):\ d_{O_j}^2 = (A^2)^\alpha (1/(a_{O_j}^2))^\beta (1/(b_{O_j}^2))^\gamma (1/(h_{O_j}^2))^\delta \ \wedge$$
$$c_{(1,U,R_j^l)} = g_{R_j^l}^\varepsilon \ \wedge\ c_{(2,U,R_j^l)} = h_{R_j^l}^\varepsilon \ \wedge\ c_{(3,U,R_j^l)} = g_{R_j^l}^\beta h_{R_j^l}^\gamma y_{(3,R_j^l)}^\varepsilon \ \wedge$$
$$c_{(4,U,R_j^l)} = \left(y_{(1,R_j^l)} y_{(2,R_j^l)}^{\mathcal{H}(c_{(1,U,R_j^l)}, c_{(2,U,R_j^l)}, c_{(3,U,R_j^l)}, m_j)}\right)^\varepsilon\}$$

Protocol 8 (Global Revocation).

8:1. U chooses $r_2 \in_R \mathbb{Z}_n$ and computes $c_{(1,U,R)} := g^{r_2}$, $c_{(2,U,R)} := h^{r_2}$, $c_{(3,U,R)} := y_{(3,R)}^{r_2} g^{x_U}$, and $c_{(4,U,R_i^l)} := y_{(1,R_i^l)}^{r_2} y_{(2,R_i^l)}^{r_2 \mathcal{H}(c_{(1,U,R_i^l)}, c_{(2,U,R_i^l)}, c_{(3,U,R_i^l)}, m_0)}$ and sends $(c_{(1,U,R_i^l)}, c_{(2,U,R_i^l)}, c_{(3,U,R_i^l)}, c_{(4,U,R_i^l)})$ to V.

8:2. U and V engage in

$$PK\{(\alpha,\beta,\gamma,\delta,\varepsilon):\ d_{O_j}^2 = (A^2)^\alpha (1/(a_{O_j}^2))^\beta (1/(b_{O_j}^2))^\gamma (1/(h_{O_j}^2))^\delta \ \wedge$$
$$c_{(1,U,R)} = g^\varepsilon \ \wedge\ c_{(2,U,R)} = h^\varepsilon \ \wedge\ c_{(3,U,R)} = g^\beta y_{(3,R)}^\varepsilon \ \wedge$$
$$c_{(4,U,R)} = \left(y_{(1,R)} y_{(2,R)}^{\mathcal{H}(c_{(1,U,R)}, c_{(2,U,R)}, c_{(3,U,R)}, m_0)}\right)^\varepsilon\}$$

6.3 Encoding Expiration Dates and Other Personal Attributes

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

7 Security proofs

For each party we have to describe simulator with black-box to the parties that can interact with T in the name of the corresponding ideal-world-party. The parties the adversary controls are subsumed into a single party. We only describe the simulator for the adversary. The other simulators are immediate from that one. In the description, we assume, for ease of exposition, that the certification authority is not corrupted and do not consider single-show credentials and expiration dates. The description of the simulator is slightly more complicated in these cases, but not so different as to merit such a lengthy exposition.

Our interface to the adversary consists of the interfaces between the honest parties and the dishonest parties. Note that the interface from honest organizations and the adversary also contains the channels via which these organizations publish pseudonyms, credentials, and

encryptions. Transactions between dishonest participants happen "inside" the adversary and we need not consider them.

Recall that some events started by honest parties get triggered by the real-world adversary and get "automatically" also triggered in the ideal-world. However, the honest parties from the real-world can also trigger their events started by the corresponding ideal-world party.

Simulator for the adversary $\mathcal{A}$. On input security parameter k, we construct a simulator as follows: First, the adversary specifies the users and organizations controlled by him, and those he does not control. Next, we generate public keys for all organizations and revocation managers and provide them to the adversary.

Simulating events that do not involve parties controlled by $\mathcal{A}$: We have to react to the following events between honest parties when they appear in the public archive of T.

$\mathsf{FormNym}(U, O)$: Get $(O, N_{(U,O)})$ from archive. Choose x_U and $s_{(U,O)}$ and compute $P_{(U,O)}$. Use the simulation property of verifiable encryption (cf. Definition 2) to generate v, i.e., a is a simulation of and w is a simulation of

$$VE(\mathsf{CEIG}, (\mathcal{H}, g, h, C))\{(\alpha, \beta) : P = (a_{O_I}^2)^\alpha (b_{O_I}^2)^\beta\}$$

with random $C \in_R G$. Publish $(N_{(U,O)}, P_{(U,O)}, v)$ as O. Store $N_{(U,O)}, P_{(U,O)}, x_U, s_{(U,O)})$

$\mathsf{GrantCred}(U, O)$: Get $(O, N_{(U,O)})$ from the archive. Get $P_{(U,O)}$ going with $N_{(U,O)}$ from storage, compute credential $(e_{(U,O)}, c_{(U,O)})$, publish $(N_{(U,O)}, e_{(U,O)}, c_{(U,O)})$ as O and store it as well.

$\mathsf{VerifyCred}(U, O_V)$: Here we do not have to do anything.

$\mathsf{VerifyCredOnNym}(U, O_V)$: Here we do not have to do anything.

Simulating requests originating from U controlled by $\mathcal{A}$ to parties not controlled by $\mathcal{A}$:

User controlled by $\mathcal{A}$ requests to enter the system: We are playing the CA towards $\mathcal{A}$ and U towards T. When U identifies herself, we get id-info. If info is ok, run Protocol 1 with U, get $N_{(U,CA)}$ from U and use rewinding to extract x_U and $s_{(U,CA)}$. The protocol will result in $P_{(U,CA)}$. If the protocol is successful continue, otherwise send an abort message to $\mathcal{A}$.

If we do not have PKI-assured non-transferability, request $\mathsf{AddUser}$ with id-info to T and get K_U from T (as the CA is honest, it will accept to add U).

If we have PKI-assured non-transferability, also run Protocol 5. Use rewinding to extract the secret key SK to public key PK of external PKI. If the protocol is successful, request $\mathsf{AddUserWithPKI}(U, CA)$ with $N_{(U,CA)}, PK, SK, id-info)$ from T. Receive K_U from T (as the CA is honest, it will accept to add $\mathcal{A}$).

Finally, request $\mathsf{FormNym}$ with $K_U, N_{(U,CA)}$ (which will be successful as CA is honest). Store $N_{(U,CA)}, K_U, x_U, s_{(U,CA)}, P_{(U,CA)}$.

$\mathsf{FormNym}$ $\mathcal{A}$ *with* O: We are playing the $O \neq CA$ towards $\mathcal{A}$ and U towards T. Run Protocol 1 with $\mathcal{A}$ in the name of O, get $N_{(U,O)}$ from $\mathcal{A}$ and use rewinding to extract x_U and $s_{(U,O)}$. The protocol will result in $P_{(U,O)}$.

If x_U is in storage, retrieve K_U and $(\mathsf{FormNym}(U, O),\ N_{(U,O)}, K_U)$ to T. If T answers "fail", then abort the protocol, otherwise acknowledge to $\mathcal{A}$, store $(N_{(U,O)}, P_{(U,O)}, s_{(U,O)})$ with x_U.

If x_U is not in storage the user has not entered the system correctly. Thus we send T the request IIAddUser containing "Add ?" and obtain K_U back. Then we send (FormNym(U,O) with $N_{(U,O)}, K_U$) to T. If T answers "fail", then abort the protocol, otherwise acknowledge to $\mathcal{A}$, store $(N_{(U,O)}, P_{(U,O)}, s_{(U,O)}, w_{(P_{(U,O)})})$ with x_U.

GrantCred $\mathcal{A}$ *with* O: We are playing O (which could be CA) towards $\mathcal{A}$ and U towards T. Get $N_{(U,O)}, P_{(U,O)}$ from $\mathcal{A}$. If those values are not in the database, reject. If they are, run step 2:1 of Protocol 2 with $\mathcal{A}$. If $\mathcal{A}$ is successful in proving ownership of $N_{(U,O)}, P_{(U,O)}$, extract x_U. If we had assigned $N_{(U,O)}, P_{(U,O)}$ to an honest user, we abort simulation and output `fail`. Use x_U to get K_U from database and perform (GrantCred$(U,O), K_U, N_{(U,O)}$) aus U and O with T. If T replies "Success", continue the protocol as O would, otherwise abort. Store all results and *tid* with x_U.

VerifyCred $\mathcal{A}$ *with* V: We are playing V towards $\mathcal{A}$ and U towards T. $\mathcal{A}$ wants to show ownership of credential by O. Run Protocol 3 with $\mathcal{A}$ in the name of V. Use rewinding to extract $x_U, s_{(U,O)}, e_{(U,O)}$.

If O is dishonest and x_U is not in the database, the user has not entered the system correctly. Thus we send T the request IIAddUser containing "Add ?" and obtain K_U back. Then we send (FormNym(U,O) with (N_1, K_U) to T as U; because O is dishonest, we will get T's request which we reply with "accept" and N_2, where N_1 and N_2 are random. Set $N_{(U,O)}$ to $N_1 \| N_2$ ($\|$ denotes concatenation). We also perform (GrantCred$(U,O), K_U, N_{(U,O)}$) as U and O with T. Continue.

If O is honest, if $((a_O^2)^{x_U}(b_O^2)^{s_{(U,O)}}, e_{(U,O)}) \neq (P_{(x,O)}, e_{(x,O)})$ for all pairs $P_{(x,O)}, e_{(x,O)}$ in the storage, abort simulation and output `fail`. Otherwise continue.

If there has to be local or global anonymity revocation, run also Protocol 7 or 8. If all protocols pass correctly, x_U to get K_U from database. Send (VerifyCred$(U,V), K_U, N_{(U,O)}$) to T. If T replies "Success", acknowledge to $\mathcal{A}$, otherwise abort. Store all results and *tid* with x_U.

VerifyCredOnNym $\mathcal{A}$ *with* O: We are playing O (which could be CA) towards $\mathcal{A}$ and U towards T. Say $\mathcal{A}$ wants to show ownership of credential by O_I. Get $N_{(U,O)}, P_{(U,O)}$ from $\mathcal{A}$. If these values are not in the database, reject. Run Protocol 4 with $\mathcal{A}$ in the name of O. Use rewinding to get x_U, and look up K_U in the database. If x_U is not found, abort simulation and output `fail`.

If O_I is dishonest, query database for $N_{(U,O_I)}$. If it is not found, then we send (FormNym(U,O) with (N_1, K_U) to T as U; because O is dishonest, we will get T's request which we reply with "accept" and N_2, where N_1 and N_2 are random. Set $N_{(U,O_I)}$ to $N_1 \| N_2$ ($\|$ denotes concatenation). Now that we have $N_{(U,O_I)}$, we also perform (GrantCred$(U,O), K_U, N_{(U,O)}$) as U and O with T.

If there has to be local or global anonymity revocation, run also Protocol 7 or 8. If the protocols are successful, send (VerifyCredOnNym$(U,V), K_U, N_{(U,O)}$) to T. If T replies "Success", acknowledge to $\mathcal{A}$, otherwise abort this protocol.

Simulating requests to V *controlled by* $\mathcal{A}$ *originating from* U *not controlled by* $\mathcal{A}$*:* Here we consider

Case I only. For the other case the verifier can be treated as an organization (next item). Hence, there is only Protocol 3 to take care of.

We are playing U towards $\mathcal{A}$ and some V towards T. U wants to show ownership of credential by O.

If the transaction is with anonymity revocation and R_O or R_O^l are controlled by the adversary, revoke anonymity right away, i.e., ask T for the $N_{(U,O)}$ or U. Use these to get $P_{(U,O)}$ and credential on it from storage (note that they must be in our storage as U is honest). Then execute Protocol 3 followed by 7 in case of local revocation and 8 in case of global revocation with $\mathcal{A}$ as that user would. If these execution are successful, send T "accept" and "reject" otherwise.

If the revocation manager is not controlled by the adversary, execute Protocol 3 followed by 7 in case of local revocation and 8 in case of global revocation with $\mathcal{A}$, where in Protocol 3 we send random values $A, B \in_R QR_{n_O}$ and in Protocol 7 and 8 (if we have to run them) we just encrypt a 0. Then we fake all the PK's in the protocol using zero-knowledge simulator for them. If these executions are successful, sent T "accept" and "reject" otherwise.

Simulating requests to O controlled by $\mathcal{A}$ originating from U not controlled by $\mathcal{A}$:

Protocol 1. U with $\mathcal{A}$: We are playing the $O \neq CA$ towards T and U towards $\mathcal{A}$. Obtain $N_{(U,O)}$ from T. Choose a random x_U and Protocol 1 with $\mathcal{A}$ as an honest U would. If the protocol is successful, store $(x_U, N_{(U,O)}, P_{(U,O)}, s_{(U,O)})$ with x_U and send T accept, otherwise send T reject.

Protocol 2. U with $\mathcal{A}$: We are playing U towards $\mathcal{A}$ and O towards T. Get $N_{(U,O)}$ from T and retrieve $x_U, P_{(U,O)}, s_{(U,O)}$ from storage (as the user requesting is honest, these things must already be in the storage).

Run Protocol 2 with $\mathcal{A}$ as an honest U would. If it is successful, store $(e_{(U,O)}, c_{(U,O)})$ with $N_{(U,O)}$ and reply "accept" to T. Otherwise reply T with "reject".

If T replies "Success", continue the protocol as O would, otherwise abort. Store all results with x_U.

Protocol 4. U with $\mathcal{A}$: We are playing U towards $\mathcal{A}$ and some O towards T. U wants to show ownership of credential by O_I.

If the transaction is with anonymity revocation and R_O or R_O^l are controlled by the adversary, revoke anonymity right away, i.e., ask T for the $N_{(U,O)}$ or U (note that they must be in the storage, as U is honest). Then execute Protocol 3 followed by 7 in case of local revocation and 8 in case of global revocation with $\mathcal{A}$ as that user would. If these execution are successful, send to T "accept" and "reject" otherwise.

If the revocation managers are not controlled by $\mathcal{A}$, execute Protocol 3 followed by 7 in case of local revocation and 8 in case of global revocation with $\mathcal{A}$, where in Protocol 3 we send random values $A, B \in_R QR_{n_O}$ and in Protocol 7 and 8 (if we have to run them) we just encrypt a 0. Then we fake all the PK's in the protocol using zero-knowledge simulator for them. If these executions are successful, sent T "accept" and "reject" otherwise.

Simulating requests from honest V or O to corrupted R_O or R_O^l: We get a request from honest V or O to revoke the anonymity for a specific transaction. If the user who was involved in the transaction was honest, then we must have already found out the identity of this user when the transaction took place, so we can translate the request for the adversary. If the user involved is controlled by $\mathcal{A}$, then we have generated the transaction, thus we look up the transaction data in storage and feed it to $\mathcal{A}$. If $\mathcal{A}$, revokes send "accept" to T, otherwise send reject to T.

Simulating requests from corrupted V or O to honest R_O or R_O^l: As R_O or R_O^l is honest, we can decrypt the ciphertext handed over to us by the adversary and check whether the request is correct. If it is not correct, reject (note that the honest revocation manager will not output anything in this case). Otherwise, get the corresponding *tid* from the archive and send request AnonRev (or LocalAnonRev) to T. T answers with "fail", reject request from $\mathcal{A}$, otherwise provide $\mathcal{A}$ with the decryption that points at a specific user U or pseudonym N. Fake the proof that the ciphertext corresponds to that user.

7.1 Proof of successful simulation

Theorem 3. *Under the strong RSA and the discrete logarithm assumptions, for all polynomial-time adversaries, the simulation above will succeed with probability* $1 - \mathtt{neg}(k)$.

Proof. There are only two cases when the simulation fails: when the adversary succeeds in proving possession of a credential that was not issued, which contradicts Lemma 8 and when the adversary succeeds in impersonating an honest user, which contradicts Lemma 7. □

Theorem 4. *If the simulation does not fail, then the adversary cannot distinguish the view furnished by the simulator from that furnished by participating in the real-world system.*

Proof. All the values seen by the adversary in the simulation, except the ciphertexts for the all-or-nothing property and for anonymity revocation, are statistically close to those he sees in the real run of the system. Thus an adversary who distinguishes the two views can distinguish the ciphertexts employed in the simulation from those that correspond to a real system. By a hybrid argument, we can have hybrid simulations in which the first l ciphertexts generated by the simulator are distributed as in the real system, and the rest of the ciphertexts are distributed as in the simulation. It follows that there is an adversary that distinguishes between the lth and the $l+1$st hybrid. It follows that the adversary can break the security of the cryptosystem used for the $l+1$st ciphertext. □

Lemma 5. *The* $PK\{(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \xi) : d_{O_j}^2 = (A^2)^\alpha (1/(a_{O_j}^2))^\beta (1/(b_{O_j}^2))^\gamma (1/(h_{O_j}^2))^\delta \wedge B^2 = (h_{O_j}^2)^\varepsilon (g_{O_j}^2)^\zeta \wedge 1 = (B^2)^\alpha (1/(h_{O_j}^2))^\delta (1/(g_{O_j}^2))^\xi \wedge \beta \in \Gamma \wedge \gamma \in \Delta \wedge \alpha \in \Lambda\}$ *in step 3:2 is a statistical zero-knowledge proof of knowledge of a pseudonym with* O_j *and a credential on that pseudonym.*

Proof. The statistical zero-knowledge part is standard and left to the reader. We must exhibit the knowledge extractor for a pseudonym and a credential. By the properties of the PK protocols and under the strong RSA assumption, the knowledge extractor is can produce values α, β, γ, δ, ε, ζ, and ξ such that the statement after the colon holds. In particular, $B^2 = (h_{O_j}^2)^\varepsilon (g_{O_j}^2)^\zeta$ and $(B^2)^\alpha = (h_{O_j}^2)^\delta (g_{O_j}^2)^\xi$ from which we conclude that $\varepsilon\alpha \equiv \delta \pmod{\mathrm{ord}(h_{O_j})}$. Furthermore, we have $(A^2)^\alpha (1/(h_{O_j}^2))^\delta = (d_{O_j}^2)(a_{O_j}^2)^\beta (b_{O_j}^2)^\gamma = (A^2/(h_{O_j}^2)^\varepsilon)^\alpha$. As $\beta \in \Gamma$, $\gamma \in \Delta$, and $\alpha \in \Lambda$, it follows that $A^2/(h_{O_j}^2)^\varepsilon$ is a valid credential on the pseudonym $(a_{O_j}^2)^\beta (b_{O_j}^2)^\gamma$. Hence the prover must know a pseudonym and a credential from O_j. □

Lemma 6. *The* $PK\{(\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \xi, \eta, \varphi) : d_{O_j}^2 = (A^2)^\alpha (1/(a_{O_j}^2))^\beta (1/(b_{O_j}^2))^\gamma (1/(h_{O_j}^2))^\delta \wedge B^2 = (h_{O_j}^2)^\varepsilon (g_{O_j}^2)^\zeta \wedge 1 = (B^2)^\alpha (1/(h_{O_j}^2))^\delta (1/(g_{O_j}^2))^\xi \wedge P_{(U,O_i)}^2 = (a_{O_i}^2)^\beta (b_{O_i}^2)^\eta \wedge C^2 = (a_{O_j}^2)^\beta (b_{O_j}^2)^\gamma (h_{O_j}^2)^\varphi \wedge$
$\beta \in \Gamma \wedge \gamma \in \Delta \wedge \alpha \in \Lambda\}$ *in step 4:2 is a statistical zero-knowledge proof of knowledge of a pseudonym with* O_j *and a credential from* O_j *such that the pseudonym is based on the same key as* $P_{(U,O_i)}$.

Proof. Similarly as in the proof of Lemma 5, the knowledge extractor gets values α, β, γ, δ, ε, ζ, ξ, η, and φ such that the statement following the colon holds. As in the proof of Lemma 5 is follows that $A^2/(h^2_{O_j})^{\varepsilon}$ is a valid credential on the pseudonym $(a^2_{O_j})^{\beta}(b^2_{O_j})^{\gamma}$. From $C^2 = (a^2_{O_j})^{\beta}(b^2_{O_j})^{\gamma}(h^2_{O_j})^{\varphi}$ it is immediate that C^2 is a commitment to that pseudonym. Finally, from $P^2_{(U,O_i)} = (a^2_{O_i})^{\beta}(b^2_{O_i})^{\eta}$ it follows that both pseudonyms are based on the same master secret, i.e., β. □

Lemma 7. *Assuming factoring is hard, the adversary cannot prove ownership of a pseudonym-credential pair issued to an ICS-controlled user by an ICS-controlled organization.*

Proof. Let $K_{\mathcal{A}}$ be an upper-bound on the number of credentials the adversary requests and K_H be upper-bound on the number of credentials honest users request. Let $K = K_{\mathcal{A}} + K_H$.

Let $\mathcal{A}$ be the adversary that is allowed to adaptively run the Protocols 1, 2, and 4 with O_i and thereby obtain pseudonym-credential pairs $\big(P_j, (c_j, e_j)\big)$, $j = 1, \ldots, K_{\mathcal{A}}$. Assume that at some point $\mathcal{A}$ proves some (honest) verifier V or some (honest) organization O_j possession of a pseudonym-credential pair with values $\big((\tilde{x}, \tilde{s}), (\tilde{c}, \tilde{e})\big)$, with $\tilde{x} \in \Gamma$, $\tilde{s} \in \Delta$, $\tilde{e} \in \Lambda$, and $((a^2_{O_i})^{\tilde{x}}(b^2_{O_i})^{\tilde{s}}, \tilde{e}) = (P^2_j, e_j)$ for some $K_{\mathcal{A}} < j \leq K$. We distinguish the following two cases:

1. $P_j = \pm a^{\tilde{x}}_{O_i} b^{\tilde{s}}_{O_i}$: If $\mathcal{A}$ succeeds in proving possession of such a tuple, then we can play the game below to compute discrete logs modulo a safe prime product.

2. $P_j \neq \pm a^{\tilde{x}}_{O_i} b^{\tilde{s}}_{O_i}$: If $\mathcal{A}$ succeeds in proving possession of such a tuple, then we can play the same to factor a safe prime product.

Game. Let n be the product of two safe primes that we are given to factor.

1. Select primes $e_1, \ldots, e_K \in_R \Lambda$.

2. Set $b = z^{\prod_{1 \leq l \leq K} e_l} \bmod n$.

3. Choose $r_1, r_2 \in_R \Gamma$ and set $a = b^{r_1} \bmod n$ ans $d = b^{r_2} \bmod n$.

4. Select $g, h \in_R QR_n$ and publish (n, a, b, d, g, h) as public key of O_i.

5. Whenever an honest user asks for a credential that O_i grants, choose some $K_{\mathcal{A}} < j \leq K$ that is not used yet, choose $x_j \in_R \Gamma$ and $s_j \in_R]2^{3\ell_\Delta}, 2^{4\ell_\Delta}[$, compute $P_j = b^{x_j r_1 + s_j} \pmod n$ and $c_j = z^{(x_j r_1 + s_j)\prod_{1 \leq l \leq K; l \neq j} e_l} \bmod n$, and publish (P_j, e_j, c_j, v_j), where v_i is a simulated verifiable encryption according to Step 1:6 in Protocol 1.

6. Interact with $\mathcal{A}$ according to the following cases.

   Protocol 1 with O_i: Assume we run this protocol for the j-th time. Run the protocol the same way as O_i would and store $P_{(U,O_I)}$ and P_j.

   Protocol 2 with O_i: Obtain pseudonym P from $\mathcal{A}$ and use rewinding with the PK in step 2:1 to get values $\tilde{x}$ and $\tilde{e}$ such that $P^2 \equiv (a^2)^{\tilde{x}}(b^2)^{\tilde{s}}$. Find j such that $P_j = P$ (if there is no such j abort this protocol). Otherwise set $c_j = y'^{\prod_{1 \leq l \leq K; l \neq j} e_l} z'^{(1+r_2)\prod_{1 \leq l \leq K; l \neq j} e_l}$. Hence $c_j^{e_j} = \pm P_j \pmod n$. If $c_j^{e_j} = P_j \pmod n$ send $\mathcal{A}$ the tuple (c_j, e_j). Otherwise send $(-c_j, e_j)$.

Protocol 3 with V or Protocol 4 with O_l: Act as V or O_l would, but use rewinding with the PK in step 4:2 (resp., in step 3:2) to get values $\tilde{A}$, $\tilde{x} \in \Gamma$, $\tilde{s} \in \Delta$, $\tilde{e} \in \Lambda$, and $\tilde{y}$ such that

$$(\tilde{A}^2/h^{2\tilde{y}})^{\tilde{e}} \equiv (a^2)^{\tilde{x}}(b^2)^{\tilde{s}}d^2 \equiv z^{2(\tilde{x}r_1+\tilde{s}+r_2)} \pmod{n}$$

holds due to Lemma 5 (resp. Lemma 6) and the way we chose a, b, and d.

If $(a^2)^{\tilde{x}}(b^2)^{\tilde{s}} \neq P_j^2$ for some $K_{\mathcal{A}} < j \leq K$, then continue the protocol as V or O_l would. Otherwise distinguish the following two cases:

I. $P_j = \pm a^{\tilde{x}}b^{\tilde{s}}$: In fact we must have $P_j = a^{\tilde{x}}b^{\tilde{s}}$ and thus $a^{\tilde{x}}b^{\tilde{s}} = a^{x_j}b^{s_j}$. From $a^{\tilde{x}-x_j}b^{s_j-\tilde{s}} \equiv 1 \pmod{n}$ and as $a = b^{r_1}$ we have that $r_1(\tilde{x} - x_j) + \tilde{s} - s_j \equiv 0 \pmod{\phi(n)}$ which allows us to factor n, because we choose s_j such that $s_j > |r_1(\tilde{x} - x_j) + \tilde{s}|$. Output the factorization and stop.

II. $P_j \neq \pm a^{\tilde{x}}b^{\tilde{s}}$. In this case $w = \gcd(n, (P_j - a^{\tilde{x}}b^{\tilde{s}}) > 1$ and hence $w|n$. Output $(w, n/w)$ and stop.

All other protocols: All the other protocols we run as the real entities would.

7. If $\mathcal{A}$ stops, output $\perp$ and stop.

It remains to argue that the values we present to the adversary in this game are indistinguishable from those in the real protocol. To this end note that as $\ell_\Delta = \epsilon\ell_\Gamma$ and $\ell_\Gamma = 2\ell_n$, the distribution of the s_i's in Games 1 and 2 are statistically indistinguishable from the uniform distribution in Δ. Also, because $\ell_\Gamma = 2\ell_n$, the distribution of $a = b^r$ with $r \in_R \Gamma$ is statistically indistinguishable from $a \in_R QR_n$, and d.

□

Lemma 8. *Under the strong RSA assumption, if a polynomially bounded adversary succeeds in proving ownership of a valid credential record (P, c, e) from organization O_i, then this credential record was created by running Protocols 1, 2, and 4 with organization O_i.*

The proof of this lemma is closely related to Theorem 1 of Ateniese et al. [12].

Proof. Let $K_{\mathcal{A}}$ be an upper-bound on the number of credentials the adversary requests and K_H be upper-bound on the number of credentials honest users request. Let $K = K_{\mathcal{A}} + K_H$.

Let $\mathcal{A}$ be the adversary that is allowed to adaptively run the Protocols 1, 2, and 4 with O_i and thereby obtain pseudonym-credential pairs $\big(P_j, (c_j, e_j)\big)$, $j = 1, \ldots, K$. Assume that at some point $\mathcal{A}$ proves some (honest) verifier V or some (honest) organization O_j possession of a pseudonym-credential pair with values $\big((\tilde{x}, \tilde{s}), (\tilde{c}, \tilde{e})\big)$, with $\tilde{x} \in \Gamma$, $\tilde{s} \in \Delta$, $\tilde{e} \in \Lambda$, and $((a_{O_i}^2)^{\tilde{x}}(b_{O_i}^2)^{\tilde{s}}, \tilde{e}) \neq (P_j^2, e_j)$ for all $1 \leq j \leq K$. We distinguish the following two cases:

1. $\gcd(e_j, \tilde{e}) = 1$ for all j: If $\mathcal{A}$ succeeds in proving possession of such a tuple, then we can use play Game 1 to solve the flexible RSA problem.

2. There is some j such that $\gcd(e_j, \tilde{e}) = e_j$: If $\mathcal{A}$ succeeds in proving possession of such a tuple, then we can play Game 2 to solve the flexible RSA problem.

Game 1. Let (n, z) be the instance of flexible RSA problem we are given to solve, where n is the product of two safe primes and $z \in QR_n$. We can assume that z has order $p'q'$ as otherwise either $z = 1$ or $\gcd(z-1, n) > 1$ is a non-trivial factor of n.

1. Select $v_1, \ldots, v_K \in_R]-2^{\ell_\Delta} + 2^{2\ell_\Gamma}, 2^{\ell_\Delta}[$ and primes $e_1, \ldots, e_K \in_R \Lambda$.
2. Set $b = z^{\prod_{1 \le l \le K} e_l} \bmod n$.
3. Choose $r_1, r_2 \in_R \Gamma$ and set $a = b^{r_1} \bmod n$ and $d = b^{r_2} \bmod n$.
4. For all $1 \le i \le K$, compute $c_i = z^{(v_i + r_2) \prod_{1 \le l \le K; l \ne i} e_l} \bmod n$.
5. Select $g, h \in_R QR_n$ and publish (n, a, b, d, g, h) as public key of O_i.
6. Whenever an honest user asks for a credential that O_i grants, choose $(P_j = c_j^{e_j}/d, e_j, c_j)$.
7. Interact with $\mathcal{A}$ according to the following cases.

   Protocol 1 with O_i: Assume we run this protocol for the j-th time. Start with the step 1:1 and receive the commitments C_1 and C_2 from $\mathcal{A}$. Use the *PK* in step 1:2 to extract $\tilde{x}_j$, $\tilde{r}_j$, $\tilde{r}'_j$, and $\tilde{r}'_j$ such that $C_1^2 = (g^2)^{\tilde{r}_j}(h^2)^{\tilde{r}'_j}$ and $C_2^2 = (g^2)^{\tilde{x}_j}(h^2)^{\tilde{r}''_i}$ (this involves rewinding of $\mathcal{A}$). In step 1:3, compute $s_j = v_j - \tilde{x}_j r_1$ (in $\mathbb{Z}$). Note that $s_j \in \Delta$. Set $r = s_j + (2^{\ell_\Delta} - 1) - \tilde{r}_j \bmod (2^{\ell_\Delta + 1} - 1)$ and send r to $\mathcal{A}$. Run the remainder of the protocol the same way as O_i would.

   Protocol 2 with O_i: Obtain pseudonym P from $\mathcal{A}$ and verify the *PK*. Find the index j such that $P = a^{v_j}$. If there is no such j, abort this execution of Protocol 2. Otherwise, send (c_j, e_j) to $\mathcal{A}$.

   Protocol 3 with V or Protocol 4 with O_l: Act as V or O_l would, but use rewinding with the *PK* in step 4:2 (resp., in step 3:2) to get values $\tilde{A}$, $\tilde{x} \in \Gamma$, $\tilde{s} \in \Delta$, $\tilde{e} \in \Lambda$, and $\tilde{y}$ such that

   $$(\tilde{A}^2/h^{2\tilde{y}})^{\tilde{e}} \equiv (a^2)^{\tilde{x}}(b^2)^{\tilde{s}} d^2 \equiv z^{2(\tilde{x}r_1 + \tilde{s} + r_2)} \pmod{n}$$

   holds due to Lemma 5 (resp. Lemma 6) and the way we chose a, b, and d.

   If $\gcd(\tilde{e}, e_j) \ne 1$ for some $1 \le j \le K$ continue the protocol as V or O_l would.

   Otherwise, let $\hat{e} := 2(\tilde{x}r_1 + \tilde{s} + r_2) \prod_{1 \le l \le K} e_l$. As $\gcd(\hat{e}, e_j) = 1$ for all $1 \le j \le K$, we have $w := \gcd(\tilde{e}, \hat{e}) = \gcd(\tilde{e}, 2(\tilde{x}r_1 + \tilde{s} + r_2))$. By the extended Euclidean algorithm, there exist $\text{ль}, \text{э} \in \mathbb{Z}$ such that $\text{ль}\,\hat{e} + \text{э}\,\tilde{e} = w$. Set $u := z^{\text{ль}}(\tilde{A}^2/h^{2\tilde{y}})^{\text{э}} \bmod n$ and $e := \tilde{e}/w$. Note that $e > 1$ because $\tilde{e} \in \Lambda$ and $2^{\ell_\Lambda} > 2(2^{2\ell_\Gamma} + 2^{\ell_\Gamma} + 2^{\ell_\Delta}) > |2(\tilde{x}r_1 + \tilde{s} + r_2)|$ and we have $u^e \equiv z \pmod{n}$ and therefore (u, e) is a solution to the flexible RSA problem. Output (u, e) and stop.

   All other protocols: All the other protocols we run as the real entities would.

8. If $\mathcal{A}$ stops, output $\perp$ and stop.

Game 2. Let (n, z) be as in Game 1.

1. Select $v_1, \ldots, v_K \in_R]-2^{\ell_\Delta} + 2^{2\ell_\Gamma}, 2^{\ell_\Delta}[$ and primes $e_1, \ldots, e_K \in_R \Lambda$.
2. Choose $k \in_R \{1, \ldots, K\}$ and set $b = z^{\prod_{1 \le l \le K; j \ne k} e_j} \bmod n$.
3. Choose $r_1, r_2 \in_R \Gamma$ and set $c_k = b^{r_1} \bmod n$, $d = c_k^{e_j}/b^{v_j} \bmod n$, $a := b^{r_2} \bmod n$.

4. For all $1 \leq j \leq K$ $j \neq k$, compute $c_j = z^{(v_j + r_2 + e_k r_1 - v_k)} \prod_{1 \leq j \leq K; l \neq i,k} e_j \bmod n$.

5. Select $g, h \in_R QR_n$ and publish (n, a, b, d, g, h) as public key of O_i.

6. Whenever an honest user asks for a credential that O_i grants, choose $(P_j = c_j^{e_j}/d, e_j, c_j)$.

7. Interact with $\mathcal{A}$ according to the following cases.

   Protocol 1 with O_i: Assume we run this protocol for the j-th time. Start with the step 1:1 and receive the commitments C_1 and C_2 from $\mathcal{A}$. Use the PK in step 1:2 to extract $\tilde{x}_j$, $\tilde{r}_j$, $\tilde{r}'_j$, and $\tilde{r}'_j$ such that $C_1^2 = (g^2)^{\tilde{r}_j}(h^2)^{\tilde{r}'_j}$ and $C_2^2 = (g^2)^{\tilde{x}_j}(h^2)^{\tilde{r}''_i}$ (this involves rewinding of $\mathcal{A}$). In step 1:3, compute $s_j = v_j - \tilde{x}_j r_1$ (in $\mathbb{Z}$). Note that $s_j \in \Delta$. Set $r = s_j + (2^{\ell_\Delta} - 1) - \tilde{r}_j \bmod (2^{\ell_\Delta + 1} - 1)$ and send r to $\mathcal{A}$. Run the remainder of the protocol the same way as O_i would.

   Protocol 2 with O_i: Obtain pseudonym P from $\mathcal{A}$ and verify the PK. Find the index j such that $P = a^{v_j}$. If there is no such j, abort this execution of Protocol 2. Otherwise, send (c_j, e_j) to $\mathcal{A}$.

   Protocol 3 with V or Protocol 4 with O_l: Act as V or O_l would, but use rewinding with the PK in step 4:2 (resp., in step 3:2) to get values $\tilde{A}$, $\tilde{x} \in \Gamma$, $\tilde{s} \in \Delta$, $\tilde{e} \in \Lambda$, and $\tilde{y}$ such that

   $$(\tilde{A}^2/h^{2\tilde{y}})^{\tilde{e}} \equiv (a^2)^{\tilde{x}}(b^2)^{\tilde{s}} d^2 \equiv z^{2(\tilde{x}r_1 + \tilde{s} + r_2)} \pmod{n}$$

   holds due to Lemma 5 (resp. Lemma 6) and the way we chose a, b, and d.

   If $\gcd(\tilde{e}, e_k) \neq e_k$ continue the protocol as V or O_l would.

   Otherwise, we have $\tilde{e} = t\, e_k$ for some $t \geq 1$. Define $C := (\tilde{A}/h^{\tilde{y}})^t c_k \bmod n$ and $\hat{e} := 2(t(\tilde{x}r_1 + \tilde{s}) - v_k) \prod_{1 \leq l \leq K; l \neq k} e_l$. Note that $(C^2)^{e_k} \equiv (b^{2(\tilde{x}r_1 + \tilde{s} + v_k)}) \equiv z^{\hat{e}} \pmod{n}$. As $e_k | \tilde{e}$ and $e_k, \tilde{e} \in \Lambda$ we must have $1 < t < 2^{\ell_\Sigma}$ and $0 \leq 2|t(\tilde{x}r_1 + \tilde{s}) + v_k| < 2(2^{\ell_\Sigma}(2^{2\ell_\Gamma} + 2^{\ell_\Delta}) + 2^{\ell_\Delta}) < 2^{\ell_\Lambda}, e_k$ and hence $\gcd(e_k, \hat{e}) = 1$. Thus there exist $\text{ъ}, \text{э} \in \mathbb{Z}$ such that $\text{ъ}\,\tilde{e} + \text{э}\,\hat{e} = 1$. Finally $u := z^{\text{Ъ}}\,(C^2)^{\text{Э}} \bmod n$ and $e := e_k$ is a solution to the flexible RSA problem. Output (u, e).

   All other protocols: All the other protocols we run as the real entities would.

8. If $\mathcal{A}$ stops, output $\perp$ and stop.

It remains to argue that what the values we present the adversary sees in Games 1 and 2 are indistinguishable from those in the real protocol. This holds for the same reason as for the game in the proof of Lemma 7.

□

References

[1] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, pp. 1030–1044, Oct. 1985.

[2] D. Chaum and J.-H. Evertse, "A secure and privacy-protecting protocol for transmitting personal information between organizations," in *Advances in Cryptology — CRYPTO '86* (M. Odlyzko, ed.), vol. 263 of *Lecture Notes in Computer Science*, pp. 118–167, Springer-Verlag, 1987.

[3] L. Chen, "Access with pseudonyms," in *Cryptography: Policy and Algorithms* (E. D. ann J. Golić, ed.), vol. 1029 of *Lecture Notes in Computer Science*, pp. 232–243, Springer Verlag, 1995.

[4] I. B. Damgård, "Payment systems and credential mechanism with provable security against abuse by individuals," in *Advances in Cryptology — CRYPTO '88* (S. Goldwasser, ed.), vol. 403 of *Lecture Notes in Computer Science*, pp. 328–335, Springer Verlag, 1990.

[5] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf, "Pseudonym systems," in *Selected Areas in Cryptography* (H. Heys and C. Adams, eds.), vol. 1758 of *Lecture Notes in Computer Science*, Springer Verlag, 1999.

[6] C. Dwork, J. Lotspiech, and M. Naor, "Digital signets: Self-enforcing protection of digital information," in *Proc. 28th Annual ACM Symposium on Theory of Computing (STOC)*, 1996.

[7] O. Goldreich, B. Pfitzman, and R. Rivest, "Self-delegation with controlled propagation — or — what if you lose your laptop," in *Advances in Cryptology — CRYPTO '98* (H. Krawczyk, ed.), vol. 1642 of *Lecture Notes in Computer Science*, (Berlin), pp. 153–168, Springer Verlag, 1998.

[8] S. Brands, "Restrictive blinding of secret-key certificates," Tech. Rep. CS-R9509, CWI, Sept. 1995.

[9] S. Brands, "Secret-key certificates," Tech. Rep. CS-R9510, CWI, Sept. 1995.

[10] E. Brickell, P. Gemmel, and D. Kravitz, "Trustee-based tracing extensions to anonymous cash and the making of anonymous change," in *Proceedings of the Sixth Annual ACM-SIAMs*, pp. 457–466, Association for Computing Machinery, Jan. 1995.

[11] M. Stadler, J.-M. Piveteau, and J. Camenisch, "Fair blind signatures," in *Advances in Cryptology — EUROCRYPT '95* (L. C. Guillou and J.-J. Quisquater, eds.), vol. 921 of *Lecture Notes in Computer Science*, pp. 209–219, Springer Verlag, 1995.

[12] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *Advances in Cryptology — CRYPTO 2000* (M. Bellare, ed.), vol. 1880 of *Lecture Notes in Computer Science*, pp. 255–270, Springer Verlag, 2000.

[13] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptology — CRYPTO '97* (B. Kaliski, ed.), vol. 1296 of *Lecture Notes in Computer Science*, pp. 410–424, Springer Verlag, 1997.

[14] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology — EUROCRYPT '91* (D. W. Davies, ed.), vol. 547 of *Lecture Notes in Computer Science*, pp. 257–265, Springer-Verlag, 1991.

[15] J. Kilian and E. Petrank, "Identity escrow," in *Advances in Cryptology — CRYPTO '98* (H. Krawczyk, ed.), vol. 1642 of *Lecture Notes in Computer Science*, (Berlin), pp. 169–185, Springer Verlag, 1998.

[16] R. Cramer and V. Shoup, "Signature schemes based on the strong rsa assumption," in *Proc. 6th ACM Conference on Computer and Communications Security*, pp. 46–52, ACM press, nov 1999.

[17] R. Gennaro, S. Halevi, and T. Rabin, "Secure hash-and-sign signatures without the random oracle," in *Advances in Cryptology — EUROCRYPT '99* (J. Stern, ed.), vol. 1592 of *Lecture Notes in Computer Science*, pp. 123–139, Springer Verlag, 1999.

[18] R. Canetti, *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.

[19] R. Canetti, "Security and composition of multi-party cryptographic protocols," *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

[20] B. Pfitzmann and M. Waidner, "Composition and integrity preservation of secure reactive systems," in *Proc. 7th ACM Conference on Computer and Communications Security*, ACM press, November 2000.

[21] D. Chaum, J.-H. Evertse, and J. van de Graaf, "An improved protocol for demonstrating possession of discrete logarithms and some generalizations," in *Advances in Cryptology — EUROCRYPT '87* (D. Chaum and W. L. Price, eds.), vol. 304 of *Lecture Notes in Computer Science*, pp. 127–141, Springer-Verlag, 1988.

[22] C. P. Schnorr, "Efficient signature generation for smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 239–252, 1991.

[23] I. Damgård, "Efficient concurrent zero-knowledge in the auxiliary string model," in *Advances in Cryptology — EUROCRYPT 2000* (B. Preneel, ed.), vol. 1807 of *Lecture Notes in Computer Science*, pp. 431–444, Springer Verlag, 2000.

[24] A. Fiat and A. Shamir, "How to prove yourself: Practical solution to identification and signature problems," in *Advances in Cryptology — CRYPTO '86* (A. M. Odlyzko, ed.), vol. 263 of *Lecture Notes in Computer Science*, pp. 186–194, Springer Verlag, 1987.

[25] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations," in *Advances in Cryptology — CRYPTO '97* (B. Kaliski, ed.), vol. 1294 of *Lecture Notes in Computer Science*, pp. 16–30, Springer Verlag, 1997.

[26] D. Chaum, "Zero-knowledge undeniable signatures," in *Advances in Cryptology — EUROCRYPT '90* (I. B. Damgård, ed.), vol. 473 of *Lecture Notes in Computer Science*, pp. 458–464, Springer-Verlag, 1991.

[27] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology — CRYPTO '92* (E. F. Brickell, ed.), vol. 740 of *Lecture Notes in Computer Science*, pp. 89–105, Springer-Verlag, 1993.

[28] F. Boudot, "Efficient proofs that a committed number lies in an interval," in *Advances in Cryptology — EUROCRYPT 2000* (B. Preneel, ed.), vol. 1807 of *Lecture Notes in Computer Science*, pp. 431–444, Springer Verlag, 2000.

[29] E. F. Brickell, D. Chaum, I. B. Damgård, and J. van de Graaf, "Gradual and verifiable release of a secret," in *Advances in Cryptology — CRYPTO '87* (C. Pomerance, ed.), vol. 293 of *Lecture Notes in Computer Science*, pp. 156–166, Springer-Verlag, 1988.

[30] J. Camenisch and M. Michels, "Separability and efficiency for generic group signature schemes," in *Advances in Cryptology — CRYPTO '99* (M. Wiener, ed.), vol. 1666 of *Lecture Notes in Computer Science*, pp. 413–430, Springer Verlag, 1999.

[31] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology — CRYPTO '84* (G. R. Blakley and D. Chaum, eds.), vol. 196 of *Lecture Notes in Computer Science*, pp. 10–18, Springer Verlag, 1985.

[32] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 591–610, Apr. 2000.

[33] J. Camenisch and I. Damgård, "Verifiable encryption and applications to group signatures and signature sharing," Tech. Rep. RS-98-32, BRICS, Departement of Computer Science, University of Aarhus, Dec. 1998.

[34] J. Camenisch and M. Michels, "Proving in zero-knowledge that a number n is the product of two safe primes," in *Advances in Cryptology — EUROCRYPT '99* (J. Stern, ed.), vol. 1592 of *Lecture Notes in Computer Science*, pp. 107–122, Springer Verlag, 1999.

[35] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology — Proceedings of CRYPTO '82* (D. Chaum, R. L. Rivest, and A. T. Sherman, eds.), pp. 199–203, Plenum Press, 1983.

[36] S. Brands, "Electronic cash systems based on the representation problem in groups of prime order," in *Preproceedings of Advances in Cryptology — CRYPTO '93*, pp. 26.1–26.15, 1993.

[37] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Advances in Cryptology — CRYPTO '98* (H. Krawczyk, ed.), vol. 1642 of *Lecture Notes in Computer Science*, (Berlin), pp. 13–25, Springer Verlag, 1998.

[38] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Advances in Cryptology — EUROCRYPT '97* (W. Fumy, ed.), vol. 1233 of *Lecture Notes in Computer Science*, pp. 480–494, Springer Verlag, 1997.

[39] J. Camenisch and M. Michels, "A group signature scheme with improved efficiency," in *Advances in Cryptology — ASIACRYPT '98* (K. Ohta and D. Pei, eds.), vol. 1514 of *Lecture Notes in Computer Science*, pp. 160–174, Springer Verlag, 1998.

[40] E. Fujisaki and T. Okamoto, "A practical and provably secure scheme for publicly verifiable secret sharing and its applications," in *Advances in Cryptology — EUROCRYPT '98* (K. Nyberg, ed.), vol. 1403 of *Lecture Notes in Computer Science*, pp. 32–46, Springer Verlag, 1998.

A The Strong RSA Assumption

The *flexible RSA problem* is the following. Let $n = pq$ be a randomly generated RSA modulus. Let a random element z from $\mathbb{Z}_n^*$ be given. Find an element $u \in \mathbb{Z}_n^*$ and a number $e \in \mathbb{Z}_{>1}$ such that $z \equiv u^e \pmod{n}$. The *strong RSA assumption (SRSA)* is that this problem is hard to solve.

It is stronger than the traditional *RSA assumption* which states that given a modulus n and an exponent e it is hard to find $u, z \in \mathbb{Z}_n^*$ such that $z \equiv u^e \pmod{n}$. Although both assumptions are stronger than assuming integer factorization to be hard, the only known way of solving the respective problems involves factoring the modulus.

In fact our scheme is based on a slightly different but weaker assumption, the strong QR-RSA assumption(cf. [16]). It states that the flexible RSA problem is hard in case $z \in QR_n$. It is easy to see that this flexible QR-RSA problem is reducable to the flexible RSA problem.

The strong RSA assumption was independently introduced by Barić and Pfitzmann [38] and by Fujisaki and Okamoto [25] and has subsequently proved instrumental for constructing practical existentially unforgeable signature schemes secure against adaptive chosen message attacks [16, 17], and for constructing other important primitives such as group signatures [12, 39] and verifiable secret sharing [40].

B Proof of Theorem 1

Proof. We reduce an attack on $\mathcal{G}$ to an attack on $\mathcal{G}'$. Our input is $\tilde{E}$, which is a public key for $\mathcal{G}$. We choose two challenge messages, m_0 and m_1, at random from the message space. In response, we receive the challenge ciphertext $e = E(m_?)$.

Now we must use the circular encryption adversary $\mathcal{A}$. The adversary breaks the circular security of the scheme for some value n which is polynomial in k, and for some assignment to $(i_1, \ldots, i_n)$ and $(j_1, \ldots, j_n)$. Pick c at random from $1, \ldots n$. Produce n public keys $E_1, \ldots, E_n$ as follows: $E_l := \tilde{E}$, while the rest of them are chosen at random by running $\mathcal{G}$.

Choose n random values $r_1, \ldots, r_n$. Choose n random values $R_1, \ldots, R_n$.

For some L such that $i_L = c$, set $e_L = e$. Set $e_l = E_{i_l}(r_l)$, $l \neq L$. Give the n 3-tuples (E_{i_l}, e_l, R_l) to the adversary as a challenge. Note that until the adversary made any queries to $\mathcal{H}$, this ciphertext looks distributed exactly correctly.

Now have to answer the adversary's random oracle queries.

While the adversary asks for $H(u)$, $u \neq r_i$, $u \neq m_0, m_1$, give a random response. If the adversary halts, flip a coin \$, declare "Challenge was an encryption of $m_\$$." Exit.

If the adversary asks for $H(m_i)$, declare "Challenge was an encryption of m_i." Exit.

If the adversary asks for $H(r_i)$, flip a coin \$ and declare "Challenge was an encryption of m\$." Exit.

Now we must show that given an adversary that succeeds with the circular encryption scheme, this simulator will produce the right guess for ? with probability $1/2 + 1/\mathtt{poly}$. First, note that

$$\Pr[A \text{ asks } \mathcal{H}(m_i) | e_L \text{ is encryption of } m_{i-1}] = \mathtt{neg}$$

because m_0,m_1 were chosen independently of everything else. Then,

$$\begin{aligned} 1/\texttt{poly} &= \Pr[\mathcal{A} \text{ succeeds }] \\ &= \Pr[\mathcal{A} \text{ asks } \mathcal{H}(r_i) \text{ or } H(m_i)] \cdot \Pr[\mathcal{A} \text{ succeeds } | \mathcal{A} \text{ asks } \mathcal{H}(r_i) \text{ or } \mathcal{H}(m_i)] \\ &+ \Pr[\mathcal{A} \text{ does not ask } \mathcal{H}(r_i) \text{ or } H(m_i)] \cdot \Pr[A \text{ succeeds } | \text{ does not ask } \mathcal{H}(r_i) \text{ or } H(m_i)] \end{aligned}$$

Note that $\Pr[\mathcal{A} \text{ succeeds } | \mathcal{A}\text{does not ask}\mathcal{H}(r_i) \text{ or } H(m_i)] = \texttt{neg}$, because the circular ciphertext is, by construction, independent of the secret keys encrypted.

Therefore, $Pr[\mathcal{A} \text{ asks } \mathcal{H}(r_i) \text{ or } H(m_i)]|Asucceeds] = 1/\texttt{poly}$.

Recall: e_l's are all just encryptions of random elements, and the challenge ciphertext was inserted in a random place. So with probability $1/n$ the first question that A asks that is related to e_i's is $H(mi)$. Using the first equation above:

$$\Pr[\mathcal{A} \text{ asks } m_i | \mathcal{A} \text{ succeeds }] = 1/\texttt{poly}$$

Therefore, $\Pr[\mathcal{A} \text{ asks } m_i] = 1/\texttt{poly}$, and so, using the first equation again,

$$\Pr[\mathcal{A} \text{ asks } m_i \text{ and } e = \tilde{E}(m_{i-1})] = \texttt{neg}$$

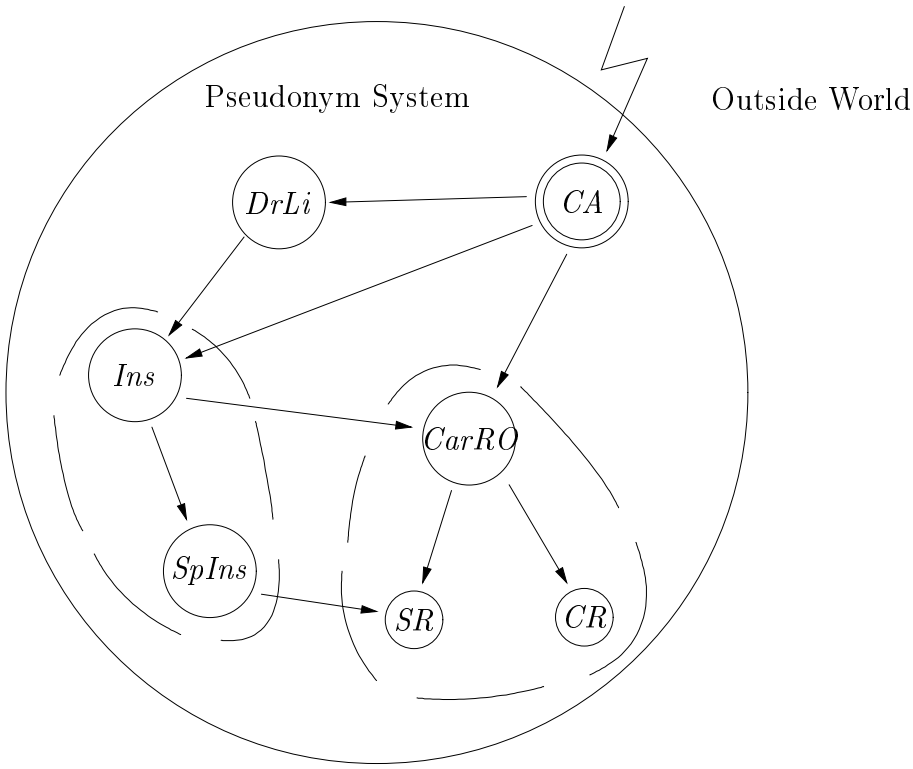
and so, putting everything together,

$$\begin{aligned} \Pr[\text{ correctly guess challenge bit}] &= \\ 1/2 \Pr[\mathcal{A} \text{ does not ask } m_i] &+ \Pr[\mathcal{A} \text{ asks } m_i] \\ &= 1/2 + 1/\texttt{poly} \end{aligned}$$

□

C Example of an Anonymous Credential System with Revocation

Figure C displays an example of a pseudonym system. The nodes of the graph represent the *CA* (the double circle), the organizations (the big circles), and the verifiers (the small circles). The arrows represent an example of a given user's credential showings. An arrow from X to Y means that the user showed to entity Y a credential issued to him by entity X. The dashed line groups organizations that trust each other to check various credentials properly. This example of a pseudonym system that allows a user to obtain a driver's license through organization *DL*, a car insurance through organization *Ins*, a sports car insurance through *SpIns*, and access to a car rental through the car rental organization *CarRO*. The access to car rental is as follows. The user first registers with the car rental organization *CarRO* who verifies that he is a valid user (has a *CA*-credential) and has car-insurance (has an *Ins*-credential). The car rental organization needs not to worry whether the user has a driver's license as the insurance is responsible and liable for that. Now, if the user want to rent a ordinary car, he goes to verifier/ car rental agency *CR*, shows that he owns a credential from *CarRO*, and get a car. However, if the user would like a sports car, he goes to verifier/sports car agency *SR*, and has to show that he own not only a credential from *CarRO* but also from *SpIns*, i.e., that he has a special insurance for sports car. While none of the credentials reveal any information about the user's real identity or pseudonym, the showing of credentials can be carried out such



that a designated revocation manager can later find the user's identity and/or pseudonyms. For instance, in case our user has a non-criminal car accident, the revocation manager reveals the pseudonym the user has with the corresponding insurance company and the cost of his insurance will go up. Whereas if he has a criminal accident, then the revocation manager also reveals his real identity and he goes to jail.

Operation name	*Request* *from*	*to Org*	*Content of* request to T	*T's test procedure*	*T sends to Org test result and*	*Org sends to T if accept*	*T sends to requester if* *Org accept*	*Org reject*	*Add to* *p. arch.*	*s. arch.*
AddUser	U	CA	U	None	U		random K_U	fail		(U, K_U)
FormNym	U	O	K_U, N_1	K_U exists?	N_1	N_2	$N_{(U,O)}$ 1)	fail	$(O, N_{(U,O)})$	$(U, N_{(U,O)})$
GrantCred	U	O	$N_{(U,O)}, K_U$	K_U valid ?, $N_{(U,O)}$ U's nym with O?	$N_{(U,O)}$		Success	fail	$(O, N_{(U,O)})$	(O, U)
VerifyCred	U	V	K_U, $N_{(U,O_I)}$, O_I Condition A Condition B	K_U valid? $N_{(U,O_I)}$ U's nym with O_I? U has cred from O_I?	Condition A Condition B		Success		(tid, V)	
VerifyCred OnNym	U	O_V	K_U, $N_{(U,O_I)}$, O_I $N_{(U,O_V)}$,O_V Conditions A,B	K_U valid? $N_{(U,O_I)}$ U's nym with O_I? U has cred from O_I?	$N_{(U,O_V)}$, O_I Conditions A,B		Success		(tid, O_I, O_V)	
AddUser WithPKI	U	CA	U, SK_U, PK_U with PK_U	SK_U corresp. to PK_U?	U, PK_U,		random K_U	fail		(U, PK_U, SK_U)
FurnishKey	U	—	K_U	K_U exists?			SK_U	fail	*tid*	
TellAll	U	—	K_U	K_U exists?			All s.arch. about K_U	fail		
AnonRev	V or O_V	R_O	"*tid*"	"*tid*" exists?, "*tid*" with O? "*tid*" with revocation? Condition A ?	V or O_V Condition A		U	fail		
Local AnonRev	V or O_V	R_O^l	"*tid*"	"*tid* exists?', "*tid*" with O? "*tid*" with revocation? Condition B?	V or O_V Condition B		$N_{(U,O)}$	fail		
IIAddUser	U	O		None	"?"		random K_U	fail		("?",K_U)
IIAnonRev	R_O	—	"*tid*"	"*tid*" exists?, "*tid*" with O? "*tid*" with revocation?			U	N.A.		
IILocAnonRev	R_O^l	—	"*tid*"	"*tid*" exists? ,"*tid*" with O? "*tid*" with revocation?			$N(U, O)$	N.A.		

Table 1: Ideal-world specifications. Every transaction has a unique transaction identifier *tid* that describes the type of transaction and its serial number. We assume some mechanism that ensure uniqueness of *tid*'s. This *tid* is included in the message from the requesting player. Then T performs the the test T as listed in the table and send the *Org* to whom the request is addressed, the test result, the *tid*, and what is specified in the table (but not providing the O any other information about , in particular not U's id). The *Org* replies with either *reject* or *accept*. In the latter case *Org* also sends T what is specified in the table. Finally, T forwards the *Org*'s full reply, the *tid*, and some extra information as specified in the table. Every time a new random identifier is produced (e.g., a key K_U or a pseudonym $N_{(U,O)}$) it is just a random binary string of length k, where k is the security parameter. The last two column show T's update to it public and secret archive. Variations such as one-show credentials and credentials with additional attributes, e.g., expiration dates, are not shown. 1) $N_{(U,O)}$ is computed as $N_1 \| N_2$.