Pseudonym Systems

(Extended Abstract)

Anna Lysyanskaya[1], Ronald L. Rivest[1], Amit Sahai[1], Stefan Wolf[2]

[1] MIT LCS, 545 Technology Square, Cambridge, MA 02139 USA,
{anna, rivest, amits}@theory.lcs.mit.edu

[2] Computer Science Department, ETH Zürich, CH-8092 Zürich, Switzerland,
wolf@inf.ethz.ch

Abstract. Pseudonym systems allow users to interact with multiple organizations anonymously, using pseudonyms. The pseudonyms cannot be linked, but are formed in such a way that a user can prove to one organization a statement about his relationship with another. Such a statement is called a credential. Previous work in this area did not protect the system against dishonest users who collectively use their pseudonyms and credentials, i.e., share an identity. Previous practical schemes also relied very heavily on the involvement of a trusted center. In the present paper we give a formal definition of pseudonym systems where users are motivated not to share their identity, and in which the trusted center's involvement is minimal. We give theoretical constructions for such systems based on any one-way function. We also suggest an efficient and easy-to-implement practical scheme.

Keywords: Anonymity, pseudonyms, nyms, credentials, unlinkability, credential transfer.

1 Introduction

Pseudonym systems were introduced by Chaum [8] in 1985, as a way of allowing a user to work effectively, but anonymously, with multiple organizations. He suggests that each organization may know a user by a different pseudonym, or *nym*. These nyms are *unlinkable*: two organizations cannot combine their databases to build up a dossier on the user. Nonetheless, a user can obtain a credential from one organization using one of his nyms, and demonstrate possession of the credential to another organization, without revealing his first nym to the second organization. For example, Bob may get a credential asserting his good health from his doctor (who knows him by one nym), and show this to his insurance company (who knows him by another nym).

Anonymity and pseudonymity are fascinating and challenging, both technically—can we achieve them?—and socially—do we want them? We focus on technical feasibility, referring the reader in the social question to excellent recent treatments by Brin [4] and Dyson [17].

Chaum and Evertse [10] develop a model for pseudonym systems, and present an RSA-based implementation. While pseudonyms are information-theoretically unlinkable, the scheme relies on a trusted center who must sign all credentials.

Damgård [14] constructs a scheme based on multi-party computations and bit commitments that provably protects organizations from credential forgery by malicious users and the central authority, and protects the secrecy of the users' identities information-theoretically. The central authority's role is limited to ensuring that each pseudonym belongs to some valid user.

Chen [12] presents a discrete-logarithm-based scheme, where a trusted center has to validate all the pseudonyms, but does not participate in the credential transfer. Chen's scheme relies very heavily on the honest behavior of the trusted center, because a malicious trusted center can also transfer credentials between users.

These schemes have a common weakness: there is little to motivate or prevent a user from sharing his pseudonyms or credentials with other users. For example, a user may buy an on-line subscription, obtaining a credential asserting his subscription's validity, and then share that credential with all of his friends. More serious examples (e.g. driver's licenses) are easy to imagine.

We base our proposed scheme on the presumption that each user has a *master public key* whose corresponding secret key the user is highly motivated to keep secret. This master key might be registered as his legal digital signature key, so that disclosure of his master secret key would allow others to forge signatures on important legal or financial documents in his name. Our proposed scheme then has the property that a user can not share a credential with a friend without sharing his master secret key with the friend, that is, without *identity sharing.* Tamper-resistant devices such as smartcards are not considered in this work.

Basing security on the user's motivation to preserve a high-value secret key has been proposed before, such as in Dwork *et al.*'s work on protecting digital content [16] and Goldreich *et al.*'s study of controlled self-delegation [21]. In recent work, Canetti *et al.* [6] incorporated this notion into anonymous credential-granting schemes to prevent credential sharing among users. However, the model considered in their work differs considerably from our own: while we explore a whole system of organizations interacting with pseudonymous users, [6] assume that organizations only grant credentials to users who reveal their identity to them, though the credentials can then be used anonymously. The practical constructions they give, while based on weaker assumptions than ours, are not applicable to our situation since they take crucial advantage of the fact that the credential granting organization knows the identity of the user it grants a credential to.

In our model, a certification authority is needed only to enable a user to prove to an organization that his pseudonym actually corresponds to a master public key of a real user with some stake in the secrecy of the corresponding master secret key, such that the user can only share a credential issued to that pseudonym by sharing his master secret key. As long as the CA does not refuse service, a cheating CA can do no harm other than introduce invalid users into the system, i.e. users who have nothing to lose in the outside world.

In our model, each user must first register with the CA, revealing his true identity and his master public key, and demonstrating possession of the corresponding master secret key. (Sometimes it is not required that a user should be motivated not to share his identity. In those cases, the CA is not needed altogether.) After registration, the user may open accounts with many different organizations using different, unlinkable pseudonyms. However, all pseudonyms are related to each other—there exists an identity extractor that can compute a user's public and secret master keys given a rewindable user who can authenticate himself as the holder of the pseudonym.

An organization may issue a credential to a user known by a pseudonym. A credential may be *single-use* (such as a prescription) or *multiple-use* (such as a driver's license), and may also have an expiration date. Single-use credentials are similar to electronic coins, since they can only be used once in an anonymous transaction. Some electronic coin protocols protect against double-spending by violating the anonymity of double-spenders, but generally do not protect against transfer of the coin. A credential should be usable only by the user to whom it was issued.

In section 2 we formally define our model of a pseudonym system. In section 3 we extend Damgård's result [14], and prove that a pseudonym system can be constructed from any one-way function. In section 4 we give a practical construction of a pseudonym system based on standard number-theoretic assumptions and the hardness of a new Diffie-Hellman-like problem [15, 3] which we prove hard with respect to generic group algorithms. Our construction is easily implementable. Moreover, the secret key that motivates the user not to share his identity is usable in many existing practical encryption and signature schemes [13, 15, 18, 31]. As a result, our system integrates well with existing technology. Finally, we close by discussing some open problems.

2 The Pseudonym Model

2.1 Overview

Informal definitions In a pseudonym system, users and organizations interact using procedures. We begin the discussion of the model by introducing the procedures.

- *Master key generation.* This procedure generates master key pairs for users and organizations. A crucial assumption we make is that users are motivated to keep their master secret key secret. This assumption is justified, because master public/secret key pairs can correspond to those that the users form for signing legal documents or receiving encrypted data. A user, then, is an entity (a person, a group of people, a business, etc.) that holds a master secret key that corresponds to a master public key.
- *Registration with the certification authority.* The certification authority (CA) is a special organization that knows each user's identity, i.e. the master public key of the user. Its role is to guarantee that users

have master public/secret key pairs that will be compromised if they cheat. The user's nym with the CA is his master public key. The CA issues a credential to him that states that he is a valid user.
– *Registration with an organization.* A user contacts the organization and together they compute a nym for the user. There exists an identity extractor which, given a rewindable user that can authenticate himself as the nym holder, extracts this user's master public/secret key pair. Then the user demonstrates to the organization that he possesses a credential from the CA.
– *Issue of credentials.* The user and the organization engage in an interactive protocol by which the user obtains a credential.
– *Transfer of credentials.* A user who has a credential can prove this fact to any organization, without revealing any other information about himself. We call this operation "transfer" of a credential, because a credential is transferred from the user's pseudonym with one organization, to his pseudonym with the other.

We want to protect the system from two main types of attacks:

– *Credential forgery*: Malicious users, possibly in coalition with other organizations including the CA, try to forge a credential for some user.
– *User identity compromise or pseudonym linking:* Malicious organizations form a coalition to try to obtain information about a user's identity, either by getting information about the user's master public/secret key pair, or by identifying a pair of pseudonyms that belong to the same user.

The main difference between our model of a pseudonym system and the previous models is that in our model the notion of a user is well-defined. In the treatment of Damgård, a user is an entity who happens to be able to demonstrate the validity of a credential with the certification authority. Whether this credential was originally issued to the same entity, or to a different one who subsequently shared it, remains unclear and therefore such systems are liable to a credential forgery attack, namely credential forgery by sharing.

2.2 The general definitions

Preliminaries
Let k be the security parameter, and let 1^k denote the unary string of length k. We use the terms such as Turing machine, interactive Turing machine, probabilistic Turing machine, polynomial-time Turing machine, secure interactive procedure, and rewindable access in a standard way defined in the literature [19] and in the full version of the present paper [26].

Procedures
Master key generation:

Definition 1. *Asymmetric key generation G is a probabilistic polynomial-time procedure which, on input 1^k, generates master public/secret key pair (P, S) (notation $(P, S) \in G(1^k)$ means that (P, S) were generated by running G) such that*

1. *The public key P that is produced contains a description (possibly implicit) of a Turing machine V which accepts input S.*
2. *For any family of polynomial-time Turing machines $\{M_i\}$, for all sufficiently large k, for $(P, S) \in G(1^k)$,*

$$\Pr_{P,S}[M_k(P) = s \text{ such that } V(s) = ACCEPT] = neg(k)$$

Each user U generates a master key pair $(P_U, S_U) \in G(1^k)$ and each organization O generates a master public/secret key pair $(P_O, S_O) \in G_U(1^k)$ using asymmetric key generation procedure G_U.

Organization's key generation: For each type C of credential issued by organization O, O generates a key pair $(P_O^C, S_O^C) \in G_O(1^k)$ using asymmetric key generation procedure G_O. In this paper, we assume that each organization only issues one type of credential; our results generalize straightforwardly to handle multiple credential types per organization.

Nym generation: The user U generates a nym N for interacting with organization O by engaging in a secure interactive procedure NG between himself and the organization.

Definition 2. *Nym generation NG is a secure interactive procedure between two parties, a user with master key pair (P_U, S_U), and an organization with master key pair (P_O, S_O). The common input to NG is (P_O), U has private input (P_U, S_U), and O has private input (S_O). We assume that nym generation is done through a secure anonymous communication channel that conceals all information about the user. The common output of the protocol is a nym N for user U with the organization. The private output for the user is some secret information $SI_{U,O}^U$, and for the organization some secret information $SI_{N,O}^O$.*

We let $N(U, O)$ denote the set of nyms that user U has established with organization O. In this paper we assume that there is at most one such nym, although our results can be easily generalized. Similarly, we let $N(U)$ denote the set of nyms the user U has established with any organization, and let $N(O)$ denote the set of nyms that the organization O has established for any user.

Communication between a User and an Organization: After a nym is established, the user can use it to communicate with the organization, using secure nym authentication defined as follows:

Definition 3. *Secure nym authentication is a secure interactive procedure between user U and organization O. Their common input to the procedure is $N \in N(U, O)$. The organization accepts with probability*

$1 - neg(k)$ *if the user can prove that he knows* $(P_U, S_U, SI^U_{U,O})$ *such that* S_U *corresponds to* P_U *and* N *was formed by running* NG *with user's private input* (P_U, S_U) *and private output* $SI^O_{N,O}$. *Otherwise, the organization rejects with probability* $1 - neg(k)$.

Single-use credentials: A single-use credential is a credential that a user may use safely once, but if used more than once may allow organizations to link different nyms of the user. A user who wishes to use such a credential more than once should request instead multiple copies of the credential from the organization.
Multiple-use credentials: A multiple-use credential may be safely transferred to as many organizations as the user wishes without having to interact further with the issuing organization.

Credential issue: To issue a credential to nym $N \in N(U, O)$, the organization first requires that the user proves that he is the owner of N by running nym authentication, and then the organization O and the user U run interactive procedure CI.

Definition 4. *Credential issue procedure* CI *is a secure interactive procedure between the user with master public/secret key pair* (P_U, S_U) *and secret nym generation information* $SI^U_{U,O}$, *and the organization with master public/secret key pair* (P_O, S_O) *and secret nym generation information* $SI^O_{N,O}$, *with the following properties:*

1. *The common input to* CI *is* (N, P_O).
2. *The user's private input to* CI *is* $(P_U, S_U, SI^U_{U,O})$
3. *The organization's private input to* CI *is* $(S_O, SI^O_{N,O})$.
4. *The user's private output is the credential,* $C_{U,O}$.
5. *The organization's private output is secret information,* $CSI^O_{N,O}$.

Note that the output of CI, namely $C_{U,O}$, is not necessarily known to the organization.
Credential transfer: To verify that a user with nym $N \in N(U, O')$ has a credential from organization O, organization O' runs a secure interactive procedure CT with the user U.

Definition 5. *Credential transfer procedure* CT *is a secure interactive procedure between user* U *with master public/secret key pair* (P_U, S_U), *nyms* $N \in N(U, O)$ *and* $N' \in N(U, O')$, *corresponding secret nym generation information* $SI^U_{U,O}$ *and* $SI^U_{U,O'}$), *and credential* $C_{U,O}$; *and organization* O' *that has master public/secret key pair* $(P_{O'}, S_{O'})$ *and secret nym generation information* $SI^O_{N',O'}$. *Their common input to* CT *is* (N', P_O). U*'s private input to* CT *is* $(P_U, S_U, C_{U,O}, N, SI^U_{U,O}, SI^U_{U,O'})$ *(where* N *is* U*'s pseudonym with* O*).* O' *has private input to* CT $SI^O_{N',O'}$. *If the inputs to* CT *are valid, i.e. formed by running the appropriate protocols above, then* O' *accepts, otherwise* O' *rejects with probability* $1 - neg(k)$.

Note that if the credential is single-use, CT does not need to be an interactive procedure. The user needs only reveal $C_{U,O}$ to O', and then O' will perform the necessary computation.

If the credential is multiple-use, this procedure need not be interactive either. The user might only need to compute a function on $C_{U,O}$, P_U and S_U and hand the result over to O' to convince O' that he is a credential holder.

Requirements

All the procedures described above constitute a secure pseudonym system if and only if they satisfy the requirements outlined below. The reader is referred to the full version of the present paper for a more rigorous treatment of these requirements.

Each authenticated pseudonym corresponds to a unique user: Even though the identity of a user who owns a nym must remain unknown, we require that there exists a canonical Turing machine called the *identity extractor* ID, such that for any valid nym N, given rewindable access to a Turing machine M that can successfully authenticate itself as the holder of N with non-negligible probability, $ID(N, M)$ outputs valid master public key/secret key pair with high probability. Moreover, we require that for each nym, this pair be unique.

Security of the user's master secret key: We want to make sure that user U's master secret key S_U is not revealed by his public key P_U or by the user's interaction with the pseudonym system. We require that whatever can be computed about the user's secret key as a result of the user's interaction with the system, can be computed from his public key alone.

Credential sharing implies master secret sharing: User Alice who has a valid credential might want to help her friend Bob to improperly obtain whatever privileges the credential brings. She could do so by revealing her master secret key to Bob, so that Bob could successfully impersonate her in all regards. We cannot prevent this attack, but we do require of a scheme that whenever Alice discloses some information that allows Bob to use her credentials or nyms, she thereby is effectively disclosing her master secret key to him. That is to say that there exists an extractor such that if Bob succeeds in using a credential that was not issued to his pseudonym, then the secret key of another user who does possess a valid credential, can be extracted by having rewindable access to Bob.

Unlinkability of pseudonyms: We don't want the nyms of a user to be linkable at any time better than by random guessing.

Unforgeability of credentials: We require that a credential may not be issued to a user without the organization's cooperation.

Pseudonym as a public key for signatures and encryption: Additionally, there is an optional but desirable feature of a nym system: the ability to sign with one's nym, as well as encrypt and decrypt messages.

2.3 Building a pseudonym system from these procedures

If we are given procedures with the properties as above, we can use them as building blocks for nym systems with various specifications. To ensure that each user uses only one master public/secret key pair, and one that is indeed external to the pseudonym system, we need the certification authority. The certification authority is just an organization that gives out the credential of validity. The user establishes a nym N with the CA, reveals his true identity and then authenticates himself as the valid holder of N. He then proves that $ID(N) = (P_U, S_U)$, where P_U is U's master public key, as the CA may verify. Then the CA issues a credential of validity for N, which the user may subsequently transfer to other organizations, to prove to them that he is a valid user.

In some systems there is no need for a certification authority, because there is no need for a digital identity to correspond to a physical identity. For example, in a banking system it is not a problem if users have more than one account or if groups of individuals open accounts with banks and merchants.

We refer the reader to the full version of the paper for a comprehensive treatment of other useful features a pseudonym system might have.

3 Constructions of pseudonym systems based on any one-way function

This section focuses on demonstrating that the model that we presented in Section 2 is feasible under the assumption that one-way functions exist. Our theoretical constructions use zero-knowledge proofs, and therefore they do not suggest a practical way of implementing a pseudonym system. Rather, their significance is mostly in demonstrating the feasibility of pseudonym systems of various flavors. It is also in demonstrating that the existence of one-way functions is a necessary and sufficient condition for the existence of pseudonym systems as we define them.

3.1 Preliminaries

The definitions for the terms such as one-way functions, zero-knowledge proofs and knowledge extractors, bit commitment schemes [28], and signature schemes [24, 30] can be found in standard treatments [22].

Theorem 1. *The existence of one-way functions is a necessary condition for the existence of pseudonym systems.*

This theorem follows from the way we defined asymmetric key generation. See the final version of this paper [26] for the proof.

In the constructions of a pseudonym systems presented below, we will need to use the fact that existence of one-way functions implies the existence of secure bit commitment schemes [28] and signature schemes [24, 30]; and also of zero-knowledge protocols with knowledge extractors [19].

3.2 Construction of a system with multiple-use credentials

Our theoretical construction of a system with multiple-use credentials is a straightforward extension of the construction by Damgård [14].

Suppose we are given a signature scheme $(G, Sign, Verify)$, where G is the key generation algorithm; $Sign(PK, SK, m)$ is the procedure that, on input key pair $(PK, SK) \in G(1^k)$ and message m produces a signature s, denoted as $s \in \sigma_{PK}(m)$; and $Verify(PK, m, s)$ is the verification algorithm.

Also suppose we are given a bit commitment scheme $(Commit, Check)$ where $Commit(a, r)$ is the commitment algorithm that produces a commitment to a with randomness r; if $c = Commit(a, r)$ then $Check(c, a, r)$ verifies that c is a commitment to a.

A user U runs $G(1^k)$ to create his master public key/secret key pair (P_U, S_U); an organization O creates its master public key pair (P_O, S_O) similarly.

To register with the CA, the user reveals his public key P_U to the CA. The CA outputs $C_{U,CA} \in \sigma_{CA}(P_U)$.

To establish a pseudonym with an organization O, the user U computes $N_{U,O} = Commit((P_U, S_U), R_{U,O})$ where $R_{U,O}$ is a random string that the user has generated for the purposes of computing this pseudonym and which corresponds to his private output $SI^U_{U,O}$.

To prove that his pseudonym $N_{U,O}$ is valid and that he has registered with the CA, the user proves knowledge of P_U, S_U, $R_{U,O}$ and $C_{U,CA}$ such that

1. S_U corresponds to P_U.
2. $N_{U,O} = Commit((P_U, S_U), R_{U,O})$,
3. $Verify_{CA}(P_U, C_{U,CA}) = ACCEPT$.

The identity extractor ID is the knowledge extractor for the above zero-knowledge proof of knowledge that outputs P_U and S_U components.

To issue a credential to a user known to the organization O as N, the organization O outputs a signature $C_{U,O} \in \sigma_O(N)$.

Let the user's nym with organization O' be N'. To prove to O' that he has a credential from O, the user executes a zero-knowledge proof of knowledge of P_U, S_U, R, R', N and $C_{U,O} \in \sigma_O(N)$ such that

1. S_U corresponds to P_U.
2. $N = Commit((P_U, S_U), R)$,
3. $N' = Commit((P_U, S_U), R')$,
4. $Verify_O(N, C_{U,O}) = ACCEPT$.

Theorem 2. *The system described above is a pseudonym system.*

The proof can be found in the full version of the paper.

3.3 Construction of a system with single-use credentials

This is essentially the same construction. The master key and pseudonym generation procedures are identical. The difference is that each credential has a serial number, which is an additional input in the credential issue and transfer procedures.

4 Practical constructions

We will begin this section by describing some well-known constructions based on the discrete logarithm problem. We then show how, using the constructions, to build a scheme that implements our model of a pseudonym system with one-time credentials.

4.1 Preliminaries

Setting We assume that we are working in a group G_q of prime order q, in which the discrete logarithm problem and the Diffie-Hellman problems (computational, decisional, etc.) are believed to be hard. We also rely on the random oracle model.

4.2 Building blocks

Proving equality of discrete logarithms First, we review protocol Π, the protocol of Chaum and Pedersen [11] that is assumed to be a zero knowledge proof of equality of discrete logarithms.

Protocol Π for Proving Equality of Discrete Logarithms:

Common inputs: $g, h, \tilde{g}, \tilde{h} \in G_q$
Prover knows: $x \in \mathbb{Z}_q^*$ such that $h = g^x$ and $\tilde{h} = \tilde{g}^x$

$P \longrightarrow V$: Choose $r \in_R \mathbb{Z}_q^*$; Send $(A = g^r, B = \tilde{g}^r)$.
$V \longrightarrow P$: Choose $c \in_R \mathbb{Z}_q^*$; Send c.
$P \longrightarrow V$: Send $y = r + cx \bmod q$.
V : Check that $g^y = Ah^c$ and $\tilde{g}^y = B\tilde{h}^c$.

Note that to obtain Π_{NI}, the non-interactive version of Π, set $c = \mathcal{H}(A, B)$, where $\mathcal{H}$ is the hash function.

This protocol proves both knowledge of the discrete logarithm x, and the fact that it is the same for (g, h) and $(\tilde{g}, \tilde{h})$. The following summarizes what is known about such a protocol:

Theorem 3. *If, as a result of executing protocol Π, the verifier accepts, then with probability $1 - neg(k)$, the prover knows x such that $g^x = h \bmod p$.*

Theorem 4. *If, as a result of executing protocol Π, the verifier accepts, then with probability $1 - neg(k)$, $x_1 = x_2$, where x_1 is such that $g^{x_1} = h \bmod p$ and x_2 is such that $\tilde{g}^{x_1} = \tilde{h} \bmod p$.*

Conjecture 1. Protocol Π is a secure interactive procedure [11,31].

We note that the knowledge extractor E for protocol Π just needs to ask the prover two different challenges on the same commitment, and then solve the corresponding system of linear equations $y_1 = r + c_1 x$ and $y_2 = r + c_2 x$ to compute the secret x.

Non-interactive proof of equality of DL We note that Π can be made non-interactive (we denote it by Π_{NI}) by using a sufficiently strong hash function $\mathcal{H}$ (for example a random oracle [2]) to select the verifier's challenge based on the prover's first message.

Blind non-interactive proof of equality of DL Clearly, we can obtain a transcript of this non-interactive protocol by executing the interactive protocol. In addition, we can execute the interactive protocol in such a way that the prover's view of it cannot be linked with the resulting transcript. In protocol Γ, if γ is selected at random, the transcript produced by Γ is equally likely to have come from any $\tilde{g}$ and any choice of r and c.

Protocol Γ: Producing a Blinded Transcript of Protocol Π_{NI}:

Common inputs and prover knowledge: same as in protocol Π
Verifier input: $\gamma \in \mathbb{Z}_q^*$.
Verifier wants: use prover of Π to produce valid transcript of protocol Π_{NI} on input $g, h, \tilde{G} = \tilde{g}^\gamma, \tilde{H} = \tilde{h}^\gamma$.
Note: Prover behavior is identical to protocol Π.

$$
\begin{array}{ll}
P \longrightarrow V: & \text{Choose } r \in_R \mathbb{Z}_q^*\text{; Send } (A = g^r, B = \tilde{g}^r). \\
V \longrightarrow P: & \text{Choose } \alpha, \beta, \in_R \mathbb{Z}_q^*\text{. Let } A' = Ag^\alpha h^\beta,\ B' = (B\tilde{g}^\alpha \tilde{h}^\beta)^\gamma. \\
 & \text{Send } c = \mathcal{H}(A', B') + \beta \bmod q. \\
P \longrightarrow V: & \text{Send } y = r + cx \bmod q. \\
V: & \text{Check that } g^y = Ah^c \text{ and } \tilde{g}^y = B\tilde{h}^c. \\
 & \text{Note: } g^{(y+\alpha)} = A'h^{(c-\beta)} \text{ and } \tilde{G}^{(y+\alpha)} = B'\tilde{H}^{(c-\beta)}. \\
V: & \text{Output transcript: } ((A', B'), \mathcal{H}(A', B'), y + \alpha).
\end{array}
$$

The above protocol is blind, that is, if the verifier runs it with the prover several times and then shows one of the outputs to the prover, the prover will not be able to guess correctly which conversation the output refers to, any better than by random guessing. The following theorem is well-known; we refer the reader to the final version of this paper for a proof:

Theorem 5. *The verifier's output in protocol Γ is independent of the prover's view of the conversation.*

4.3 The construction

We are now ready to present our construction based on the building blocks introduced above. Our construction is similar in flavour to that given by Chen [12].

High-level description A user's master public key is g^x, and the corresponding master secret key is x. A user's nym is formed by taking a random base a, such that the user does not know $\log_g a$, and raising it to the power x. As a result, all of the user's nyms are tied to his secret x. When a credential is issued, we want to make sure that it will not be valid for any secret other than x.

A credential in our construction is a non-interactive proof of knowledge of the organization's secret. If the user uses it twice, it can be linked, since he cannot produce another such credential on his own.

Detailed description The pseudonym system protocols are implemented as follows:

User master key generation: The user picks his master secret $x \in \mathbb{Z}_q^*$ and publishes $g^x \bmod p$.

Organization credential key generation: The organization picks two secret exponents, $s_1 \in \mathbb{Z}_q^*$ and $s_2 \in \mathbb{Z}_q^*$, and publishes $g^{s_1} \bmod p$ and $g^{s_2} \bmod p$.

Nym generation: We describe this protocol in the figure below.

Pseudonym Generation:

User U's master public key: g^x
User U's master secret key: x

U :	Choose $\gamma \in_R \mathbb{Z}_q^*$. Set $\tilde{a} = g^\gamma$ and $\tilde{b} = \tilde{a}^x$.
$U \longrightarrow O$:	Send $(\tilde{a}, \tilde{b})$.
O :	Choose $r \in_R \mathbb{Z}_q^*$. Set $a = \tilde{a}^r$.
$O \longrightarrow U$:	Send a.
U :	Compute $b = a^x$.
$U \longleftrightarrow O$:	Execute protocol Π to show $\log_a b = \log_{\tilde{a}} \tilde{b}$
U, O :	Remember U's nym $N = (a, b)$.

Note that in the special case that O is the CA, the user should send (g, g^x) instead of $(\tilde{a}, \tilde{b})$.

Communication between a user and an organization: To authenticate nym (a, b), the user and the organization execute a standard secure protocol that proves user's knowledge of $\log_a b$. (E.g. they can run Π to prove that $\log_a b = \log_a b$.)

Credential issue and transfer: These protocols are described in the figure below.

Issuing a Credential:

User's nym with organization O: (a, b) where $b = a^x$

Organization O's public credential key: (g, h_1, h_2) where $h_1 = g^{s_1}, h_2 = g^{s_2}$
Organization O's secret credential key: (s_1, s_2)

$O \longrightarrow U$:	Send $(A = b^{s_2}, B = (ab^{s_2})^{s_1})$.
U :	Choose $\gamma \in_R \mathbb{Z}_q^*$.
$O \longleftrightarrow U$:	Run Γ to show $\log_b A = \log_g h_2$ with Verifier input γ. Obtain transcript T_1.
$O \longleftrightarrow U$:	Run Γ to show $\log_{(aA)} B = \log_g h_1$ with Verifier input γ. Obtain transcript T_2.
U :	Remember credential $C_{U,O} = (a^\gamma, b^\gamma, A^\gamma, B^\gamma, T_1, T_2)$.

Transferring a Credential to Another Organization:

Organization O's public credential key: (g, h_1, h_2) where $h_1 = g^{s_1}, h_2 = g^{s_2}$
User's nym with organization O': $(\tilde{a}, \tilde{b})$ where $\tilde{b} = \tilde{a}^x$
User's credential from organization O: $C_{U,O} = (a', b', A', B', T_1, T_2)$

O' :	Verify correctness of T_1 and T_2 as transcripts for Π_{NI} for showing $\log_{b'} A' = \log_g h_2$ and $\log_{(a'A')} B' = \log_g h_1$.
$U \longleftrightarrow O'$:	Execute Protocol Π to show $\log_{\tilde{a}} \tilde{b} = \log_{a'} b'$.

The nym as public key for signatures and encryption: There are many encryption and signature schemes based on the discrete logarithm problem that can be used, such as the ElGamal [18] or Schnorr [31] schemes.

Security of the scheme We prove that the scheme presented above satisfies the definition of a pseudonym system given in section 2 in the full version of the present paper [26]. Below we outline the assumptions under which this follows.

Recall the setting – a group G_q of order q; access to a random oracle. The following assumptions are necessary:

1. We rely on the Decisional Diffie-Hellman assumption.

2. We assume that Protocol Π for proving equality of discrete logarithms is secure.
3. We assume that the following problem is hard:

   Problem 1. Let G be a cyclic group with generator g and of order $|G|$. Let g^x and g^y be given. Furthermore, assume that an oracle can be called that answers a query s by a triple (a, a^{sy}, a^{x+sxy}), where $a = g^z$ is a random group element of G. Let this oracle be called for $s_1, s_2, \ldots$. Then, the problem is to generate a quadruple $(t, b, b^{ty}, b^{x+txy})$, where $t \notin \{0, s_1, s_2, \ldots\}$, and where $b \neq e$.

   Theorem 6 shows the hardness of Problem 1 with respect to generic algorithms (as defined by Shoup [32]) unless the group order is divisible by a small prime factor.

   Theorem 6. *Let p be the smallest prime factor of n. The running time of a probabilistic generic algorithm solving Problem 1 for groups of order n is of order $\Omega(\sqrt{p}/(\log n)^{O(1)})$.*

   Proof Idea. The proof is based on the fact that the event $\mathcal{E}$ that two of the computed group elements are equal ($\mathcal{E}$ is called the *collision event*), has the following two properties. First, the event has probability of order $O(T^2/p)$, where T is the number of steps performed by the generic algorithm. Second, given that the event $\mathcal{E}$ does not occur, the algorithm produces a correct 4-tuple only with probability $O(1/p)$. $\square$

   Although for any particular group used, there can exist specific (non-generic) algorithms solving Problem 1, the generic hardness of the problem is strong evidence for the existence of groups for which the problem is hard.

4.4 Multiple-use credentials

We have not been able to construct a system with multiple-use credentials which would completely conform to the specifications of our model. However, with a slight variation on the model and a straightforward modification of the scheme described above, we can get a scheme with multiple-use credentials. Moreover, in this setting we will no longer require the random oracle.

To implement this, our pseudonym generation and credential issue procedure will remain the same. As a result, the user will possess $C_{U,O} = (a, b, A, B)$, where $A = b^{s_2}$, $B = (ab^{s_2})^{s_1}$, and $(a, b) = (a, a^x)$ is the user's nym with the issuing organization. The user can therefore sample, for any γ, the 4-tuples $f_\gamma(C_{U,O}) = (a^\gamma, b^\gamma, A^\gamma, B^\gamma)$. For any 4-tuple formed that way, for any correctly formed pseudonym (a', b'), the user will be able to prove that $\log_a b = \log_{a'} b'$. If the issuing organization is required to cooperate with the receiving organization, it can confirm that $f_\gamma(C_{U,O})$ is a valid credential that corresponds to nym (a^γ, b^γ), or disprove that statement if it is not true. This is as secure as the scheme with one-time credentials.

5 Conclusions and open questions

The present work's contributions are in defining a model for pseudonym systems and proving it feasible, as well as proposing a practical scheme which is a significant improvement over its predecessors. Open problems lie in the area of identifying useful features for a pseudonym system (some features not mentioned in this extended abstract have been introduced and discussed in the full version of the present paper [26]); in removing interactiveness in the theoretical constructions; and in coming up with good practical constructions that conform to our specifications.

Acknowledgements

The first author would like to acknowledge the support of an NSF Graduate Fellowship and the Lucent Technologies GRPW; the third author would like to acknowledge the support of a DOD NDSEG fellowship; the first, second, and third authors would also like to acknowledge DARPA grant DABT63-96-C-0018.

References

1. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology—CRYPTO 98*, pages 26–40. Springer-Verlag, 1998.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Dan Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, pages 48–63. Springer-Verlag, 1998.
4. David Brin. *The Transparent Society: Will Technology Force Us to Choose between Privacy and Freedom?* Perseus Press, 1998.
5. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *Advances in Cryptology—CRYPTO '97*, pages 410–424. Springer-Verlag, 1997.
6. Ran Canetti, Moses Charikar, Ravi Kumar, Sridhar Rajagopalan, Amit Sahai, and Andrew Tomkins. Non-transferable anonymous credentials. *Manuscript, 1998. Revision in submission*, 1999.
7. Ran Canetti, Oded Goldreich, and Shai Halevi. Random oracle methodology, revisited. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 209–218, 1998.
8. David Chaum. Security without identification: transaction systems to make Big Brother obsolete. *Communications of the ACM*, 28(10), 1985.
9. David Chaum. Designated confirmer signatures. In *Advances in Cryptology—EUROCRYPT 94*, pages 86–91. Springer-Verlag, 1994.
10. David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Advances in Cryptology—CRYPTO '86*, pages 118–167. Springer-Verlag, 1986.
11. David Chaum and Torben Pryds Pedersen. Wallet databases with observers (extended abstract). In *Advances in Cryptology—CRYPTO '92*, pages 89–105. Springer-Verlag, 1992.
12. Lidong Chen. Access with pseudonyms. In Ed Dawson and Jovan Golić, editors, *Cryptography: Policy and Algorithms*, pages 232–243. Springer-Verlag, 1995. Lecture Notes in Computer Science No. 1029.

13. R. Cramer and V. Shoup. A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO 98*. Springer-Verlag, 1998.
14. Ivan Bjerre Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals (extended abstract). In *Advances in Cryptology—CRYPTO '88*, pages 328–335. Springer-Verlag, 1988.
15. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
16. C. Dwork, J. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information. In *Proceedings of the 28th STOC*, pages 489–498, 1996.
17. E. Dyson. *Release 2.1: A design for living in the digital age.* Broadway, 1998.
18. T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
19. Oded Goldreich. Secure multi-party computation. *http://theory.lcs.mit.edu/~oded*, 1998.
20. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.
21. Oded Goldreich, Birgit Pfitzmann, and Ronald L. Rivest. Self-delegation with controlled propagation - or - what if you lose your laptop. In *Advances in Cryptology—CRYPTO 98*, pages 153–168. Springer-Verlag, 1998.
22. Shafi Goldwasser and Mihir Bellare. Lecture notes in cryptography. *ftp://theory.lcs.mit.edu/pub/classes/6.875/crypto-notes.ps*, 1996.
23. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.
24. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
25. Joe Kilian and Erez Petrank. Identity escrow. In *Advances in Cryptology—CRYPTO '98*, pages 169–185. Springer-Verlag, 1998.
26. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. *http://theory.lcs.mit.edu/~anna/lrsw99.ps*, 1999.
27. David Mazières and M. Frans Kaashoek. The design, implementation and operation of an email pseudonym server. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, 1998.
28. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
29. Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In *Advances in Cryptology—CRYPTO '94*, pages 61–74. Springer-Verlag, 1994.
30. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.
31. C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
32. V. Shoup. Lower bounds on discrete logarithms and related problems. In *Advances in Cryptology—EUROCRYPT '97*, pages 256–266. Springer-Verlag, 1997.
33. Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.