Pseudonym Systems

by

Anna Lysyanskaya

A.B., Computer Science and Mathematics (1997) Smith College

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

Author .....
Department of Electrical Engineering and Computer Science
Spring, 1999

Certified by .....
Ronald L. Rivest
Webster Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Pseudonym Systems

by

Anna Lysyanskaya

Submitted to the Department of Electrical Engineering and Computer Science
on Spring, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

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 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. This is joint work with Ronald L. Rivest and Amit Sahai.

Thesis Supervisor: Ronald L. Rivest
Title: Webster Professor of Electrical Engineering and Computer Science

Acknowledgments

In the first place, I would like to thank Ron Rivest for getting me interested in Cryptography, suggesting pseudonym systems as a research topic, working on this topic with Amit Sahai and myself with infinite patience and professionalism, and helping me write this thesis. I would also like to thank Amit Sahai for working with us on this project.

I am also grateful to Alice Wang, David Mazières and Sofya Raskhodnikova for reading drafts of this thesis and suggesting improvements.

I acknowledge the support of an NSF graduate fellowship (any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation), DARPA grant DABT63-96-C-0018, and Lucent Technologies Graduate Program for Women grant.

Contents

Chapter 1

Introduction

One of the many advantages of living in the computerized world is the ease with which we can find information we are looking for. In just a matter of seconds, computers search through gigabytes of data and extract precisely the answer. But hand in hand with this convenience, comes the side effect that information is becoming increasingly difficult to keep private. Once a piece of one's private information is incorporated into someone else's data bank, one has no control over who will be able to look at it. Fortunately, in many cases such effects can be eliminated by applying appropriate cryptographic tools.

In the world without computers, it is possible to send someone an anonymous letter. In some sense, we can call it one-way anonymity: a letter can be delivered without any trace of who sent it, but how does one reply to it? This same service is available in the digital world through mix-masters and such services.

Going one step further, we have two-way anonymity, or pseudonymity: not only can a letter from Bob be delivered anonymously to Alice, but two of them can correspond with each other, while Bob remains anonymous to Alice (and she knows him under pseudonym Charlie). These kinds of services are also available through anonymous re-mailers[29].

But what if Bob wants to prove to someone else something about his relationship with Alice?

In this thesis we discuss the cryptographic design of a system that achieves this ability. We call such systems pseudonym systems (they could also be called "pseudonym and credential systems"). In a pseudonym system, there are two main features. First of all, individuals interact with different organizations using different pseudonyms. As a result, even if organizations record all their data, and several organizations compare records, they will not be able to establish which pseudonyms come from which individuals, or even which pseudonyms belong to the same individual. Secondly, an individual in a pseudonym system can prove a statement to an organization about his relationship with another organization

remaining anonoymous to both. By proving such a statement, no information other than the statement itself is revealed to the receiving organization.

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[5] and Dyson[17].

We illustrate the notion of a pseudonym system with examples.

1.1 Motivating example

Our main motivating example is a medical one. While medical records do constitute sensitive information and the desire to restrict access to health data is understandable, we use this example primarily to illustrate our technical points. Health systems have also served as motivation for studies in other areas of cryptography, such as trust management[3].

Consider the components of a health care system. A person Alice receives health care guarantee from her employer. The employer knows her name and social security number and is able to withhold a certain amount from Alice's paycheck for her medical insurance. A health care provider is an insurance company that is paid by the employer to process Alice's medical insurance claims. Doctors (hospitals) diagnose and treat patients. Pharmacists prepare medicine. Research labs study patients mostly for the benefit of science at large. Suppose all these organizations know the patient's name and social security number, and share all information they have about the patient with each other.

There are obvious problems with this scenario. For example, if the research lab discovers that Alice has a high risk of getting cancer, her health care provider will raise the cost of her insurance. Or, if the employer finds out that Alice wants to have a baby, the employer might try to fire her before she applies for maternity leave. Thus one can see that this health care system's main weakness is that it allows disclosure of information of extremely private nature. The problem is that the patients' names are not secret and that two organizations can establish which files refer to the same patient. To eliminate these weaknesses, three goals must be simultaneously met: the anonymity of the patients, their ability to, for example, get reimbursed for their medical expenses through the insurance company, and the guarantee that only patients who do in fact have medical insurance should be able to do so. Let us elaborate on these three goals.

On the one hand, we need to make patients pseudonymous to the organizations. A patient approaches an organization and establishes a pseudonym with it. The organization has no access to other information about the user, and only refers to him by this pseudonym.

On the other hand, this has to be done in such a way that a patient is still able to convince one organization about the status of his relationship with another. A patient who

has money withheld from his paycheck certainly wants to prove to his health care provider that he has indeed paid for his health care. After visiting a doctor and paying his bill, the patient will want to prove to his health care provider that he has done so, so that he will be reimbursed appropriately. Also, the patient may choose to prove to his doctor that the research lab has discovered that he is likely to get cancer. Therefore, it is important that while the patient's transactions with different organizations in the system are unlinkable, he should be able to prove to one organization a statement about his relationship with another.

Finally, the system will break down if there is no guarantee that several users cannot share the same pseudonym: for example, for the cost of one medical insurance policy, a whole group of friends will be covered. Thus patients have to be motivated to be honest and not to enable their friends to use their identity.

A system like this may be useful in other settings as well, especially in settings which only arise in the digital world, such as pseudonymous access to data and other electronic services.

1.2 Discussion of previous work

Pseudonym systems were introduced by Chaum[9] 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 can not 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. A certification authority (CA) plays the important role of guaranteeing that the users in the system can be trusted to behave responsibly and that the credentials are transferred properly.

Shortly after the notion of pseudonym systems was introduced, Chaum and Evertse[11] developed a model for pseudonym systems, and presented an RSA-based implementation. The scenario they propose models the one in our motivating example, with one important difference. In their scenario, a trusted center conducts the operation of transferring a user's credential from one organization to another. The pseudonyms in their model are information-theoretically unlinkable. The heavy reliance on a trusted center is the main weakness of this work.

Damgård[15] constructed a scheme that implements a pseudonym system that does not rely on a trusted third party as much. His scheme is based on multi-party computations and bit commitments. The advantages of his scheme is that it provably protects organizations from credential forgery by malicious users and the central authority, and protects the

secrecy of users' identities information-theoretically. The central authority's role is limited to ensuring that each pseudonym belongs to some valid user. The drawback, however, is that Damgård's result is not meant to be implemented in practice: it is based on zero-knowledge proof constructions, which are inefficient. We will describe Damgård's scheme in more detail in the chapter dedicated for theoretical constructions.

Chen[13] presents a practical discrete-logarithm based scheme for Damgård's model. In her scheme, a trusted center has to validate all the pseudonyms, but does not participate in the credential transfer. One of this scheme's main disadvantages is that it requires honest behavior of the CA: A malicious CA has all it needs to transfer credentials between users. In the present thesis, we present a practical scheme that does not have this problem.

Aside from various disadvantages we have mentioned above, these schemes have a major 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. In the setting from our medical motivating example, a patient may share his medical insurance with all of his friends. Other serious examples (e.g. drivers' licenses) are easy to imagine.

This happens because the notion of a user is not very well defined, the CA is still the one responsible for making sure that users who enter the system can be trusted to behave responsibly. It is important to conceptualize what constitutes a user's digital identity. In existing schemes, this question is more or less avoided. The existing schemes assume that a certification authority which interacts with all the players in the system, will grant to someone the privileges of a user based on its own judgement. The fact that a certification authority may be deceived in this judgement, and by doing so enable groups of individuals to share an identity and all privileges that this identity brings, is overlooked.

This serious problem is solved in the present thesis.

1.3 Overview of our results

The contribution of this thesis is two-fold. On the one hand, we present a pseudonym system model that corresponds to the settings we would like to obtain in real life. In our model, the identity of a user is a well-defined concept, and users have a stake in behaving responsibly. At the same time, their anonymity is protected at all times. We discuss variations and extensions of the model to make it suitable to many different real-life scenarios. We also show, by our theoretical constructions, that our basic model, as well as its numerous extensions, are realizable. On the other hand, we present a practical and easily implementable construction of the basic model under reasonable number-theoretic assumptions. The results

of this thesis have been obtained in joined unpublished work with Ronald Rivest and Amit Sahai[28].

1.3.1 Discussion of the model

The main distinction of our model from its predecessors [11, 13, 15] is that the notion of a user is well-defined. We base our proposed scheme on the presumption that each user has a *master public key* whose corresponding master secret key the user is highly motivated to keep secret. This master public 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 model requires 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.* Thus in our model a user is an entity that possesses this master secret key.

Basing security on the user's motivation to preserve a high-value secret key has been used before, such as in Goldreich *et al.*'s study of controlled self-delegation[21]. In a recent unpublished paper, Canetti *et al.*[7], incorporate this notion into anonymous credential-granting schemes. However, their organizations only grant credentials to users whose identity they know, as opposed to pseudonymous users.

In this work, we incorporate protection from identity-sharing into a pseudonym model where not only can credentials be shown anonymously, they can be granted to parties based on unlinkable pseudonyms. The user opens accounts with many different organizations using different, unlinkable pseudonyms. However, all pseudonyms are related to each other: a user can authenticate a valid pseudonym only if he possesses a master secret key that was used to create this 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). 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 revealing the identity 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. This can be used in such settings as voting, and also in situations such as drug prescriptions: a person may be entitled to buy a drug for himself, but he cannot sell his prescription to someone else.

In our model, a certification authority (CA) is just one of the organizations in the system. It 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. The system works in such a way 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.

For some systems, there is no such thing as an "invalid user." In those cases, a certification authority is not needed. These cases are precisely the ones that are covered by the Chaum-Evertse[11], Damgård[15], and Chen[13] model: the pseudonyms are indeed formed in the correct way, but whether several users are sharing identity, or a user has more than one identity, remains up to the users. Thus, using our techniques, the already studied model can be implemented without the need for the certification authority altogether.

We show that schemes that implement various flavors of our model exist if one-way functions exist. This extends the result of Damgård.

1.3.2 The practical scheme

Our practical scheme meets the specifications of our model of single-use credentials and is easy to implement. It is based on an assumption which is related to the decisional Diffie-Hellman assumption[16, 4]. The secret key that motivates the user not to share his identity is usable in many existing practical encryption and signature schemes[14, 16, 18, 33]. As a result, our system integrates well with existing technology.

1.4 Organization of this thesis

In Chapter 2 we formally define our model of a pseudonym system. We also discuss variations of our model and various settings for which we consider them suitable. In Chapter 3 we outline Damgård's theoretical construction[15], and extend it to suit our pseudonym system model. In Chapter 4 we give a practical construction of a pseudonym system with single-use credentials. Finally, we close by discussing some open problems.

Chapter 2

The Pseudonym Model

2.1 Overview

2.1.1 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, if a user 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 can 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

2.2.1 Preliminaries

Let k be the security parameter, and let 1^k denote the unary string of length k.

We assume that the computational device that is being used by all the parties is a one-tape Turing machine[34]. By an interactive Turing machine, we mean a machine which, besides an input and computation tape, also has two more tapes: one tape for reading intermediate input from, and one tape for writing intermediate output on. By a probabilistic Turing machine we mean a machine which, besides its input and computation tape, also has a tape which contains independent bits drawn uniformly at random from $\{0,1\}$ (this tape is called the machine's "random tape" throughout). By a polynomial-time Turing machine we mean a machine which runs in polynomial number of steps. By a non-uniform family of Turing machines, $\{M_k\}$, we mean a device which contains a Turing machine M and a collection of strings $\{a_k\}$, such that the length of string a_k is polynomial in k. On input w of length k, the device will run M on (w, a_k).

A *negligible function* is a function $neg(k)$ such that $neg(k) < 1/(p(k))$ for all polynomials p for sufficiently large k. A *non-negligible function* $nonneg(k)$ satisfies $nonneg(k) > 1/p(k)$

for some polynomial $p(k)$. A *hard* function is one not computable by a non-uniform family of probabilistic polynomial-time Turing machines for all valid inputs but a negligible fraction for sufficiently large k; an *easy* function is one computable in probabilistic polynomial-time for all valid inputs.

A *secure interactive procedure* is a secure multi-party computation as defined by Oded Goldreich [19]. We omit Goldreich's level of formalism, and state the definition less formally:

Definition 1 *An interactive procedure with communication transcript T for common input X between probabilistic polynomial-time Turing machine A with input a, random tape R_A and output a' and probabilistic polynomial-time Turing machine B with input b, random tape R_B and output b' is a secure two-party interactive procedure if the following conditions hold:*

- *There exists a simulator S_A such that, on input (X, A, a, a'), it produces a random tape R_S and transcript T_S such that for any function $f(X, A, a, a', B, b, b')$, any non-uniform family of probabilistic polynomial-size Turing machines C, if T is produced with B following the protocol, and A behaving arbitrarily,*

$$\begin{aligned} |\Pr[C(X, A, a, a', R_A, T) &= f(X, A, a, a', B, b, b')] - \\ \Pr[C(X, A, a, a', R_S, T_S) &= f(X, A, a, a', B, b, b')]| = neg(k) \end{aligned}$$

- *There exists a simulator S_B such that, on input (X, B, b, b'), it produces a transcript T_S and a random tape R_S such that for any function $f(X, A, a, a', B, b, b')$, any non-uniform family of probabilistic polynomial-size Turing machines C, if T is produced with A following the protocol, and B behaving arbitrarily*

$$\begin{aligned} |\Pr[C(X, B, b, b', R_B, T) &= f(X, A, a, a', B, b, b')] - \\ \Pr[C(X, B, b, b', R_S, T_S) &= f(X, A, a, a', B, b, b')]| = neg(k) \end{aligned}$$

Most importantly, the definition above captures the property that whatever the players can efficiently compute as a result of their interaction, they could compute in the setting with a trusted third party. In the trusted third party setting, the third party takes their inputs and computes the outputs for them. Then they run a simulator that, upon inputing the input and output of one of the players, comes up with a transcript and a random tape for that player. Then from this simulated conversation, that player can compute the function on the other player's inputs and outputs just as effectively as though he participated in the interactive protocol.

We now define the notion of rewindable access by Turing machine M to an interactive Turing machine A.

Definition 2 *A Turing machine M is said to have rewindable access to Turing machine A if*

1. *A is provided with a special tape T which it may access only per M's orders; and which M cannot access directly.*
2. *At any step s of A's computation, M can order A to save its state on the tape T.*
3. *At any step of A's computation, M can order A to return to any state previously saved on T.*

The intuition for the definition of rewindable access is that, if we have rewindable access to an interactive machine, we can see how it will behave depending on the inputs we give it, and will be able to infer some information we would otherwise not have access to.

2.2.2 Procedures

Master key generation:

Definition 3 *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 non-uniform family of polynomial-time Turing machines $\{M_i\}$, for 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 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 public key/secret key pair $(P_O^C, S_O^C) \in G_O(1^k)$ using asymmetric key generation procedure G_O. In this thesis, 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 4 *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 $SI^U_{U,O}$, and for the organization some secret information $SI^O_{N,O}$.*

We let $N(U, O)$ denote the set of nyms that user U has established with organization O. In this thesis 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 5 *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 $1 - neg(k)$.*

Credentials: There are two kinds of credentials.

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 6 *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 credential 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 7 *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.

2.2.3 Requirements

All the procedures described above constitute a secure pseudonym system if and only if they satisfy the requirements below. Throughout we assume that there is an interactive probabilistic polynomial-time Turing machine A that may serve as the adversary in our definitions.

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.

We now formalize the requirement that the identity underlying a nym must not be alterable. We let A specify the number of users and organizations, and let A corrupt all the users and all except one of organizations. We build an interactive probabilistic Turing machine M to simulate the uncorrupted organization O. Adversary A and simulator M initialize the parties they control. Each user (controlled by A) registers with O using as many different nyms as A specifies. Every time A registers a new nym, we use the identity extractor to extract a corresponding master public key/secret key pair. Let $\{(N_i, P_i, S_i)\}$ be the set of nyms A successfully registers for with O along with the master public key/secret key pairs that ID extracts from A for each one. Organization O issues as many credentials to each nym as A specifies. Now A attempts to authenticate one of the nyms N_j to O. We use ID to extract a master public key/secret key pair P', S' from A as it authenticates N_j to O. We say that A succeeds if he has a non-negligible probability of authenticating N_j and yet $(P', S') \neq (P_j, S_j)$. The system satisfies the unique user for every nym requirement if for all interactive probabilistic polynomial-time Turing machines A, the probability that A succeeds in the attack described above is at most $neg(k)$, where the probability is taken over the coin tosses of A and M.

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. Let machine A be the adversary that wants to compute a function of a user's master secret key. Let user U's public key P_U be known. We let A specify the number of users and organizations in the system, and let A control all users but user U, and all organizations. We then build an interactive probabilistic Turing machine M that will control U and that will simulate the life of the system together with A. A initializes all the players it controls by running the master key generation procedure for each user and for each organization. M initializes U by generating a master public/secret key pair for U, (P_U, S_U). The user U establishes a nym with each organization. Then each organization issues U a corresponding credential which he transfers to organizations as specified by A. This last step can be repeated as

many times as A specifies. Then in the end A outputs its guess for the value of the function it wants to compute. A succeeds if the value it outputs is correct. The system satisfies the security of master key requirement if for all interactive probabilistic polynomial-time Turing machines A, there exists a simulator probabilistic polynomial-time Turing machine S such that Pr[A succeeds after the experiment] $\leq$ Pr[S succeeds on input P_U] + $neg(k)$.

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.

Let user Alice be interactive probabilistic polynomial-time Turing machine A, user Bob be interactive probabilistic polynomial time Turing machine B, and the rest of the system be simulated by interactive probabilistic Turing machine M. First, M initializes the system and A initializes user Alice with master public key/secret key pair (P_A, S_A). Then A and B have a conversation, and together make requests to M to set up Alice's nyms with various organizations and to issue credentials to those nyms. Recall that every time a credential is issued, the user to whom it is issued must authenticate that he or she is the holder of the nym. That way, the identity extractor can verify that for all Alice's nyms N_i, $ID(N_i) = (P_A, S_A)$. Then A halts. B chooses an organization O from the ones that Alice has set up an account with and has a credential from. Then B chooses organization O' from the ones that Alice has an account with. At this point, we consider B running from its current state as B_T. B_T runs the CT protocol with O' to transfer O's credential for Alice to Alice's nym with O'. A and B have succeeded if O' accepts. The system satisfies the "credential sharing implies master secret sharing" property if for all interactive probabilistic polynomial-time Turing machines A and B such that the probability of success is $nonneg(k)$, there exists a probabilistic polynomial-time Turing machine S such that, $S(B_T)$ outputs S_A with probability $nonneg(k)$.

Unlinkability of pseudonyms: We don't want the nyms of a user to be linkable at any time better than by random guessing. We let the adversary A specify the number of users $u \geq 2$ in the system, the number of organizations $o \geq 2$, and corrupt $o_c \leq o$ organizations and $u_c \leq u - 2$ users. Then we construct another interactive probabilistic Turing machine M that will control all the players that are not controlled by A and cooperate with A to simulate the life of a system. The machines A and M each initialize all the players they control by running the master key generation procedure for each user and the key generation procedure for each organization. For each user U a nym is established with each organization O, and each user authenticates himself as the valid holder of that nym.

For each nym established we use ID to extract a public key/private key pair (P, S). We thus (mentally) define a function id which maps each established nym to the corresponding master public key/private key pair extracted by ID. Then each organization issues a credential to each user, and each user transfers his credentials to his nyms with all other organizations. (If it is a single-use credential, the users first get a corresponding number of copies of each credential.) This last step may be repeated a number of times specified by A. Finally, A outputs a pair of nyms (N_1, N_2) that are nyms of user(s) controlled by M with organizations controlled by A. In addition, a simulator S chooses uniformly at random a pair of distinct nyms (N_1', N_2') that are also nyms of user(s) controlled by M with different organizations controlled by A. The system satisfies the unlinkability requirement if for all interactive probabilistic polynomial-time Turing machines A and sufficiently large security parameter k, $\Pr[id(N_1) = id(N_2)] - \Pr[id(N_1') = id(N_2')] \leq neg(k)$. (Note that the simulator may do much better than the adversary, because the simulator never outputs a pair of nyms with the same organization. The point of this definition is that an adversary can only do negligibly better than such a simulator.)

Unforgeability of credentials: We require that a credential may not be issued to a user without the organization's cooperation. We let adversary A specify the number of users, the number of organizations, and let A corrupt all the users and all except two organizations. We build an interactive probabilistic Turing machine M to simulate the uncorrupted organizations O and O'. The machines A and M initialize the parties they control. Each user (controlled by A) registers with O and O' using as many different nyms as A specifies. For each nym established we require that the user authenticates that he is the valid holder of that nym and we use ID to extract a corresponding master public key/private key pair (P, S). We thus (mentally) define a function id which maps each established nym to the corresponding public key/private key pair extracted by ID. Organization O issues as many credentials to each nym as A specifies. Let $\{N_i\}$ be the set of nyms A registers with O and obtains credentials for. Now A transfers the credentials to users' nyms with O' by running the CT procedure. A succeeds if O' accepts a credential of a user known by N_U such that $id(N_U) \notin \{id(N_i)\}$. The system satisfies the unforgeability of credentials requirement if for all interactive probabilistic polynomial-time Turing machines A, for sufficiently large k, the probability that A succeeds is at most $neg(k)$, where the probability is taken over the coin tosses of A and M.

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.

2.4 Other possible features

2.4.1 Distributed CA and other organizations

Since in real life we cannot assume that there is an organization that is trusted by all other organization, it is a reasonable goal that the duties of the CA should be distributed in some way. The most obvious one would be to have a group of organizations all participate in being the CA and issuing the CA credential, and verifying the identity of a user. This step requires two extensions to the model:

1. The organization's master key generation becomes distributed and robust, i.e. protected from incorrect behavior of a minority.

2. Procedure CI becomes distributed and robust.

Two new parameters are introduced: n is the number of organizations that represent the certification authority, t is the number of organizations that need to get together in order to issue a credential (t for "threshold"), and, as usual, no coalition of $t-1$ players should be able to make any progress towards issuing a credential or discovering the master secret key of the organization they are distributively representing.

2.4.2 Expiration date

In some systems there may be need to incorporate an expiration date into a credential. This feature can be easily added to our construction of a pseudonym system from any one-way

function below; incorporating it into a practical system with a method that is more clever than updating the public key of the issuing organization, remains an open question.

Credential issue with expiration date: 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, and then the organization O and the user U run interactive procedure CI. The common input to CI is $(N, P_O^C, date)$. The user's private input to CI is $(P_U, S_U, SI_{U,O}^U)$ The organization's private input to CI is $(S_O^C, SI_{N,O}^O)$. The user's private output is the credential, $C_{U,O}$. Note that O does not see this output.

Credential transfer: To verify that a user with nym $N' \in N(U, O')$ has a credential from organization O that has not expired at time *date*, organization O' runs a secure interactive procedure CT with the user U. Their common input to CT is $(N', P_O, date)$. U's private input to CT is $(P_U, S_U, C_{U,O}, N, SI_{U,O}^U, SI_{U,O'}^U)$ (where N is U's pseudonym with O). O' has private input to CT $SI_{N',O'}^O$. If the inputs to CT are valid, i.e. formed by running the appropriate protocols above, then O' accepts, otherwise O' rejects with probability $1 - neg(k)$.

To accommodate the expiration date option, the unforgeability requirement has an obvious addition.

2.4.3 No-more-than-once credentials

Another variation on the setting is when an organization issuing a credential only wants the user to use it once, while an organization verifying the credential does not care that it is a one-time credential and is satisfied even if it has already been used (and the user does not care about linking). Our model can accommodate this setting if we have the credential issuing procedure take the public key of the receiving organization as an extra input, and employ the techniques introduced by Jakobsson *et al.* in their work on designated verifier signatures[25].

2.4.4 Use-l-times credentials

A clear extension on the single-use credential is a credential that is used a certain number of times. This can be achieved trivially by generating l copies of the credential. It would be interesting to have the length of the credential itself be the same whether this is a use l times credential or a use once credential.

We have to modify the credential issue procedure such that it takes l as input. We also need to add the obvious *spending limit* requirement: if a credential is used more than l times, then two or more instances of credential transfer can be linked.

2.4.5 Identity escrow

Another optional feature of the system is anonymity-revocation ability by a very trusted passive third party. This is a topic discussed by Kilian and Petrank[27]. Such a party may be distributed over a large number of organization, so the user will be assured of anonymity if he trusts a big number of them.

To make sure that the trusted third party will actually be able to revoke anonymity, the main procedures will now have the public key of the trusted third party as an extra input.

We will also have an extra procedure that discovers the identity of a pseudonym holder taking the secret key of the trusted third party as input. This procedure will coincide with the identity extractor.

In Chapter 3 we will present some theoretical constructions on how this can be implemented.

2.4.6 Credential revocation

Another nice feature, and a challenging question from the point of view of implementation, is the one of credential revocation. Considering that the credentials, as well as the pseudonyms, are unlinkable, is it even possible to revoke a credential, other than by changing the public key and reissuing all credentials that have been issued so far? It is clear that if the issuing organizations could create a database of revoked credentials and a receiving organization could check an unspent credential against that database, this would violate the unlinkability requirement of the basic model. However, in the presence of a trusted third party, this functionality may be achievable. We discuss constructions for it in Chapter 3.

2.4.7 Credentials from a group of organizations

A number of organizations may be issuing a credential that is of the same type. For example, if one possesses a Massachusetts driver's license, it does not matter in which town it was issued. However, the fact that a person has a driver's license from a specific town may disclose too much information about that person. Therefore, the notion of credentials from a group may be a useful one. This notion is similar to that of group signatures. If an organization is a member of a group, it can issue a credential from the whole group. Computationally, this credential does not reveal any information about which organization in the group issued it. However, there is a group manager who has a secret key that would allow him to determine the identity of the issuing organization, if needed. In Chapter 3, we will demonstrate a theoretical construction that implements this feature.

Chapter 3

Constructions of pseudonym systems based on any one-way function

This chapter focuses on demonstrating that the model that we presented in Chapter 2 is feasible under the assumption that one-way functions exist. Our theoretical constructions use zero-knowledge proofs very heavily, 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. Whether one-way functions exist is still an open problem in cryptography and complexity theory, and the present work is yet another reason why we want them to exist and why we need research in complexity theory.

We will begin by introducing basic cryptographic notions, such as one-way functions, bit commitments, signature schemes and zero-knowledge proofs. Then we will show how, using bit commitments, digital signatures and zero-knowledge proofs, Damgård constructs a credential mechanism which is different from our model. Then we show how to extend his result to suit the model for a pseudonym system we defined in the preceding chapter, as well as its extensions.

3.1 Preliminaries

All of the preliminary definitions below can be found in standard treatments[22].

Definition 8 *A function* $f : \{0,1\}^* \to \{0,1\}^*$ *is one-way if*

- *There exists a probabilistic polynomial-time algorithm* M *such that* $M(x) = f(x)$.

- *For any non-uniform family of probabilistic polynomial-time Turing machines* $\{A_k\}$, *for sufficiently large* $|y|$,

$$Pr[A_{|y|}(y) = x \text{ such that } f(x) = y] = neg(|y|)$$

where the probability is taken over the random bits of A.

From the way we defined our master key generation protocol, it follows that the existence of pseudonym systems implies the existence of one-way functions. Therefore, the existence of one-way functions is a necessary condition for building pseudonym systems. We prove this fact in the following theorem:

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

Proof: Suppose we are given a pseudonym system as defined in Chapter 2. We show how to construct a one-way function from the asymmetric key generation procedure G which is required for the construction of our pseudonym system.

Recall that G is a probabilistic polynomial-time procedure that takes as input the unary string 1^k, and has some random bits on its random tape. Denote these bits by $R \in \{0,1\}^{p(k)}$, where $p(k)$ is a polynomial that is the maximum length of a random string that G can possibly require (since G runs in polynomial time, R cannot possibly have a longer than polynomial length). By $G_R(1^k)$ we denote the output of G when the contents of its random tape is R. By $P(G_R(1^k))$, we denote the first, public, part of $G_R(1^k)$. By $P(G(1^k))$ we denote the set of all possible values of $P(G_R(1^k))$.

The one-way function we propose is

$$f : \{0,1\}^{p(k)} \rightarrow \{P(G(1^k))\}$$
$$f(R) = P(G_R(1^k))$$

We need to show that this function is one-way. First of all, there exists an efficient algorithm that computes it. Second, suppose there existed a (non-uniform) polynomial-time adversary that, on input P, could with non-negligible probability produce a string R such that $P = P(G_R(1^k))$. Then, by running $G(1^k)$ on the random string R, we obtain S such that the verifier V encoded in P accepts. This contradicts the security of the asymmetric key generation G. Therefore, a (non-uniform) polynomial-time adversary is unable invert this function. Therefore, this is a one-way function. ■

It turns out that using one-way functions, we can construct a bit commitment scheme[30] and a signature scheme that is existentially unforgeable under adaptive chosen plaintext attacks[24, 32]. (Actually, just as for the existence of our pseudonym system, the existence of one-way function is a necessary condition for the existence of these constructions.) Both are defined below.

A bit commitment scheme is a scheme that allows a user to demonstrate that he is committed to a value without disclosing it right away. The analogy in the real world is writing a commitment on a piece of paper, and depositing it into a safe in the bank. Obviously, once it's there, its contents cannot be changed, however no one except the person who deposited it has access to it.

Definition 9 *A bit commitment scheme is a polynomial-time algorithm that computes function Commit, with the following properties:*

1. *Commit takes two inputs: R and b, such that R is a random string of length $p(k)$, where p is a polynomial, and b is a string of length l.*

2. **Hard to invert:** *For all non-uniform families of probabilistic polynomial-time Turing machines $\{M_k\}$, $\Pr_R[M_{|(b,R)|}(Commit(b,R)) = b] \leq 2^{-l} + neg(k)$.*

3. **Information hiding:** *For all functions $f : \{0,1\}^* \rightarrow \{0,1\}^*$, for all non-uniform families of probabilistic polynomial-time Turing machines $\{M_k\}$, there exists a simulator probabilistic polynomial-time Turing machine S such that*

$$\Pr_R[A(Commit(b,R)) = f(b)] \leq \Pr[S(1^k, l) = f(b)] + neg(k)$$

4. **Commitment:** *Let $X_{M(x)}$ be the event that on input x machine M found two pairs, (b_1, R_1) and (b_2, R_2) such that $b_1 \neq b_2$ and $Commit(b_1, R_1) = Commit(b_2, R_2)$. For all non-uniform families of probabilistic polynomial-time Turing machines $\{M_k\}$, $\Pr[X_{M_k(1^k)}] = neg(k)$, where the probability is taken over the random bits of M_k.*

To commit to a string, one has to compute the function *Commit*. To open a commitment and prove that the value to which one was committed is b, it is sufficient to provide (b, R). A bit commitment scheme can be constructed from a one-way function[30].

We now give a definition of a signature scheme that is existentially secure against adaptive chosen message attack. That is to say that even after an adversary obtains signatures on some messages it adaptively picked, it still cannot produce a single valid signature on its own. It has been shown that such a signature scheme can be constructed from a one-way function[32].

Definition 10 *A signature scheme consists of three algorithms:*

Key generation G**:** *G is a probabilistic polynomial-time algorithm that on input 1^k produces pairs (P, S), where P is the* public key *of a signature scheme and S is its* secret key*. The notation $(P, S) \in G(1^k)$ denotes that (P, S) was obtained by running G on input 1^k. The notation $P \in G(1^k)$ denotes that P is a public key obtained by running $G(1^k)$.*

Sign σ**:** *σ is a probabilistic polynomial-time algorithm that, on input message $m \in \{0,1\}^{p(k)}$, $(P, S) \in G(1^k)$, produces a string s. The notation $s \in \sigma_P(m)$ denotes that s was obtained in this way.*

Verify V**:** *V is a probabilistic polynomial-time algorithm such that on input message m, public key P and string s accepts if and only if $s \in \sigma_P(m)$.*

Of course, one doesn't want to use a signature scheme just with this definition, because there is nothing said about its security. The following definition incorporates this notion:

Definition 11 *A signature scheme is existentially secure if for all non-uniform families of probabilistic polynomial Turing machines $\{M_k\}$, for sufficiently large k,*

$$\Pr[M_k(P \in G(1^k)) \text{ outputs } (m, s) \text{ s.t. } V(m, s, P) = ACCEPT] = neg(k)$$

where the probability is taken over the random bits of G and M_k.

This is still not enough, because this definition tells us only that when the adversary has access to a public key, it cannot forge a signature. But what if it has access to other information? The definition below captures the strongest kind of security for digital signatures: the property that an attacker, even if he has obtained signatures on messages of his own choice, cannot come up with a valid signature on a single other message.

Definition 12 *Consider a of probabilistic polynomial-time interactive algorithm A. Let A have the property that, on input $P \in G(1^k)$, A sets $i = 1$ does the following for $1 \leq i \leq q(k)$ (q is a polynomial):*

- *Produce ith query m_i based on all the input available so far.*
- *Receive $s_1 \in \sigma_P(m_i)$ from the signing oracle.*
- *Increment i.*

Then, A produces some output T, denoted as $T \in A(P)$.

Consider a probabilistic polynomial-time algorithm B, which on input $(P \in G(1^k), T \in A(P))$*, outputs a pair* (m, s)*. Let* $X(A, B)$ *be the event that B outputs* (m, s) *such that* $\forall 1 \leq i \leq q(k), m \neq m_i$ *and* $V(m, s, P) = ACCEPT$.

A signature scheme is existentially secure against adaptive chosen message attack if for all families of probabilistic polynomial-time interactive Turing machines $\{A_k\}$*, and for all families of probabilistic polynomial-time Turing machines* $\{B_k\}$*, for sufficiently large k,* $\Pr[X(A_k, B_k)] = neg(k)$*, where the probability is taken over the random bits of G,* A_k*,* B_k.

Recall the definition of a secure interactive procedure in Chapter 2. If we assume that one-way functions exist, we can talk about efficient zero-knowledge proofs that a string s belongs to language L in NP. By $witness(L, s)$ we denote the set of strings such that a non-deterministic Turing machine M that decides L accepts s if it makes the guesses encoded by $a \in witness(L, s)$.

Definition 13 *A zero-knowledge proof is a secure interactive procedure such that*

1. *Common input is string s and the description of a non-deterministic Turing machine that decides language L.*
2. *Player A has no input. (A is the* Prover*.)*
3. *Player B has no input. (B is the* Verifier*.)*
4. *Player A produces no output.*
5. *Player B's output is the bit* b' *such that:*
   - $s \in L \Rightarrow \Pr[b' = 1] = 1$
   - $s \notin L \Rightarrow \Pr[b' = 0] \geq 1 - neg(k)$

We now define the notion of a knowledge extractor E. E is an interactive polynomial-time machine, which is given rewindable access to prover P that can prove the statement $s \in L$.

Definition 14 *Suppose we are given a pair of interactive probabilistic polynomial-time machines* (P, V) *which correctly perform a zero-knowledge proof that* $s \in L$*. A knowledge extractor E for a prover P is a probabilistic polynomial-time machine which, if it is given rewindable access to P, outputs* $a \in witness(L, s)$.

We state the following well-known theorem without proof[19].

Theorem 2 *If one-way functions exist, then for any language $L \in NP$, there exists a polynomial-time verifier V and probabilistic polynomial-time knowledge extractor E such that for all strings s, $s \in L$ if and only if there exists a probabilistic polynomial-time prover P such that P and V correctly perform the zero-knowledge proof that $s \in L$ and, given rewindable access to P, E outputs $a \in witness(L, s)$ with high probability.*

A zero-knowledge proof system (P, V) for which a knowledge extractor exists, is called a *strong* zero-knowledge proof system. From this point on, when we refer to any zero-knowledge proof, we mean "strong zero-knowledge."

We are now ready to present Damgård's construction for a credential system based on any one-way function.

3.2 Damgård's construction

Since Damgård's model is different from our own and is not the focus of this paper, we will not present his model, but will just sketch the construction.

In Damgård's model there is no notion of a user's master secret key and master public key, but each user has an ID string. For example, a user's ID string is his social security number. Each organization, including the CA, has a public key/secret key pair obtained by running $G(1^k)$, where G is the key generation procedure for signatures, as defined above.

To register with the CA, the user U reveals his ID_U to the CA, and obtains $C_{U,CA} \in \sigma_{CA}(ID_U)$.

To establish a pseudonym with an organization O, the user U computes

$$N_{U,O} = Commit(ID_U, R_{U,O})$$

where $R_{U,O}$ is the random string that the user generates in order to establish this pseudonym. To prove that it is valid and that he has registered with the CA, the user proves knowledge of ID_U, $R_{U,O}$ and $C_{U,CA}$ such that

1. $N_{U,O} = Commit(ID_U, R_{U,O})$
2. $Verify_{CA}(ID_U, C_{U,CA}) = ACCEPT$

A credential from organization O is just a signature $C_{U,O} \in \sigma_O(N_{U,O})$. To transfer a credential from $N_{U,O}$ to $N_{U,O'}$, the user proves in zero knowledge that he knows ID_U, $R_{U,O}$, $N_{U,O}$, $R_{U,O'}$ and $C_{U,O}$ such that:

1. $N_{U,O} = Commit(ID_U, R_{U,O})$
2. $N_{U,O'} = Commit(ID_U, R_{U,O'})$

3. $C_{U,O} \in \sigma_O(N_{U,O})$

The security of this construction follows from the security of the bit commitment scheme and the security of the signature scheme.

3.3 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 presented in the previous section.

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 3 *The system described above is a pseudonym system.*

Proof: It is straightforward to verify that the input-output specifications for all the procedures described above satisfy the definition of a pseudonym system. Therefore, the only things that remain to be shown are the following:

1. Show that the nym generation, credential generation, and credential transfer procedures are secure.

2. Show that the system satisfies all of the additional requirements.

Security of the nym generation procedure: Since the nym generation is a one-round procedure (i.e. the user generates a nym and hands it over to the organization), its security follows trivially.

Security of the credential issue procedure: Since the credential generation is also a one-round procedure, its security follows trivially.

Security of the credential transfer procedure: The user has a credential $C_{U,O}$ issued to him by organization O to his pseudonym N. He transfers it to organization O which knows him by N'. Since zero-knowledge proofs are used to prove knowledge of P_U, S_U, R, R', N and $C_{U,O} \in \sigma_O(N)$, nothing except the following is revealed to the organization:

1. S_U corresponds to P_U.

2. $N = Commit((P_U, S_U), R)$,

3. $N' = Commit((P_U, S_U), R')$,

4. $Verify_O(N, C_{U,O}) = ACCEPT$.

The above statements can be true if and only if the user known to O' by N' possesses a credential from O. Therefore, the simulator for the zero-knowledge proof of these statements, is a simulator for the zero-knowledge proof of the fact that U possesses $C_{U,O}$, and we have satisfied the definition of a secure interactive procedure.

Each authenticated pseudonym corresponds to a unique user: It follows from the definition of a bit commitment scheme that a polynomial-time user can only know one way to open a commitment. For authentication, we use proofs of knowledge of how to open a commitment. A knowledge extractor for this proof will output the (P, S) used in the commitment. Therefore the identity is unique.

Security of the user master secret key: This follows from the security of the bit commitment to the secret key, and the zero-knowledge properties of the proofs used in the protocols.

Credential sharing implies master secret key sharing: The proof used in credential transfer is a proof of knowledge, and therefore, given rewindable access to the prover, one can recover the secret key corresponding to the nym on which the credential was issued. Hence, if a user U is able to use a credential belonging to user U', then the secret key of U' can be extracted from U.

Unlinkability of pseudonyms: It follows from the zero-knowledge properties of the proofs used in the protocols that no organization learns more than what the nym itself reveals. A nym is a bit commitment to a user's secret key, and so by the security properties of the bit commitment, a nym of one user cannot be distinguished from a nym of another user.

Unforgeability of credentials: In order to forge a credential, a user must succeed in transferring a credential he does not possess. By the existential unforgeability under chosen message attack of the signature scheme used, no user can forge a signature on his nym. If a user were able to give a proof of knowledge of such a signature with non-negligible probability, which he must in order to transfer it, one could extract the forged signature from the user in polynomial time. This is a contradiction, and hence credentials cannot be forged. ■

3.4 Construction of a system with single-use credentials

This is essentially the same construction. The master key and pseudonym generation procedures are identical. The differences are in the credential issue and transfer procedures.

Before requesting a credential, the user generated a random serial number for his credential, w, and a random string R_w and computes $W = Commit(w, R_w)$. He then hands S over to the issuing organization.

To issue a one time credential to a user known to the organization O as N, the organization O obtains input W from the user and outputs a signature $C^W_{U,O} \in \sigma_O(W, N)$.

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

1. $W = Commit(w, R_w)$
2. S_U corresponds to P_U.
3. $N = Commit((P_U, S_U), R)$,
4. $N' = Commit((P_U, S_U), R')$,
5. $Verify_O((W, N), C^W_{U,O}) = ACCEPT$.

The security of this system follows from the security of the multiple-use credential system described above, as well as from the security of the bit commitment scheme.

3.5 Other variations

The ability to prove a statement in zero-knowledge is a very powerful tool for constructing pseudonym systems of various flavors. It is therefore not surprising that, in addition to the basic pseudonym system construction, we could also come up with theoretical constructions that implement some other possibly desirable features of a pseudonym system.

3.5.1 Distributed CA and other organization

Given the system we have constructed, and the results in multi-party computation[19], we know that if trapdoor permutations exist, each party in the scheme we constructed can be replaced by a group of parties, such that the functionality of the system is preserved.

3.5.2 Expiration date

To incorporate an expiration date into a credential, we have the following modification to the credential issue and transfer protocol:

To issue a credential with expiration date D to a user known to the organization O as N, the organization O outputs a signature $C^D_{U,O} \in \sigma_O(D, N)$.

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

1. Current time is not D yet.
2. S_U corresponds to P_U.
3. $N = Commit((P_U, S_U), R)$,
4. $N' = Commit((P_U, S_U), R')$,
5. $Verify_O((D, N,)C^D_{U,O}) = ACCEPT$.

The security of this extension follows from everything else we have seen.

3.5.3 No more than once credentials

Here we replace the signature scheme we are using for credential issue with designated verifier signature scheme[25]. This will require the existence of trapdoor permutations, which is a stronger assumption than that of the existence of one-way functions.

3.5.4 Identity escrow

We can construct pseudonym systems with identity escrow if public-key cryptosystems semantically secure against adaptive chosen ciphertext attacks exist[1, 14].

To generate a nym, instead of the *Commit* function, the user U encrypts (P_U, S_U) under the public key E of the trusted third party. The security of the resulting pseudonym system follows from the security of the original pseudonym system and the security of the cryptosystem. To revoke anonymity of a pseudonym, the trusted third party decrypts the pseudonym in question.

Alternatively, the user can just append $E(P_U)$ to $N \in Commit(P_U, S_U)$, and prove that the public key P underlying them is the same. That way the trusted third party cannot obtain the user's secret key just by revoking his anonymity. This additional step will not compromise the security of the pseudonym system because the encryption scheme that will be used is secure against adaptive chosen ciphertext attack.

However, the assumption we are using here about the existence of such a cryptosystem is rather strong. There are no general-assumption-based constructions of cryptosystems that are semantically secure against adaptive chosen ciphertext attack.

3.5.5 Credential revocation

As we have previously mentioned, credential revocation can only work if there is a trusted third party. If we allowed it without a trusted third party, user's pseudonyms could become linkable as a result of credential transfer.

Since we do not want to involve the trusted third party in confirming that a credential has not been revoked, credential revocation is a procedure that may cause some identity compromise. In case of single-use credentials, this identity compromise is not very threatening: a credential can only be revoked before it is used, and the user holding it is notified, and therefore this user can prevent his identity from being compromised. However, revoking a multiple-use credential will compromise the identity of its holder to a larger extent, because all organizations to which this credential has been transferred in the past will now be able to link the user by this credential.

If we assume that there is a trusted third party, and there exists a cryptosystem semantically secure against adaptive chosen ciphertext attack, we can construct pseudonym systems with credential revocation. We do this differently for single- and multiple-use credentials.

For single-use credentials, just as before, the user generated a random serial number for his credential, w. He then encrypts it under trusted third party encryption algorithm E using randomness R_w to get $W \in E(w, R_w)$. (We index the randomness by w to denote that this randomness is secret information pertaining to credential with serial number w).

As before, W is handed over to the issuing organization.

The credential transfer protocol remains essentially the same.

To revoke a credential with encrypted serial number W, the issuing organization hands it over to the trusted third party. The trusted third party decrypts and gets w. It checks that w has not been spent yet. It then appends w to a list of revoked credentials.

For multiple-use credentials, this will be done in a completely different fashion. The trusted third party publishes its public encryption algorithm, E_{TTP}.

The user U picks a random serial number w and a random public-key encryption and decryption functions (E_w, D_w) (we index them by w to denote that they are associated with the credential with serial number w). He then computes $W = (W_1, W_2)$ where $W_1 \in E_{TTP}(E_w, D_w)$ and $W_2 \in E_w(w)$ and hands W over to the issuing organization.

The credential issue protocol is essentially the same.

To transfer a credential, the user produces a string $W' \in E_w(w)$, and hands it over to the receiving organization O'. He then performs all the necessary zero-knowledge proofs of correctness, as before.

To revoke a credential, the issuing organization contacts the trusted third party and produces W. The trusted third party decrypts $W_1 \in E_{TTP}(E_w, D_w)$ and obtains the credential decryption algorithm D_w. This allows it to decrypt $W_2 \in E_w(w)$ and recover w. Now the trusted third party publishes w, D_w in the database for revoked credentials. Any organization receiving a credential from a user can now run D_w on the string W' that it receives from a user and check if it decrypts to w, and, if so, reject the credential.

The problem with this is that if a user's credential is revoked, his pseudonyms can be linked. There are other constructions that prevent that, but their drawback is that they take as input to a zero-knowledge proof the entire database of revoked credentials.

3.5.6 Credentials from a group of organizations

Let us assume the existence of a public-key cryptosystem secure against adaptive chosen ciphertext attack. If we allow the public key of each group of organizations to be linear in the size of the number of the organizations in the group, this extension is trivial. The group public key is just a list of public keys of organizations in the group $\{P_{O_i}\}$ and the encryption algorithm of the group manager E_{GM}. Everything remains the same as in the main construction, except that in the credential transfer procedure, instead of proving that $Verify_O(N, C_{U,O}) = ACCEPT$, the user proves that there exists $P_O \in \{P_{O_i}\}$ such that $Verify_O(N, C_{U,O}) = ACCEPT$ and discloses $E_{GM}(s)$, proving that $s = P_O$.

It remains an open question how to do this in a more efficient way, i.e. without the increase in the size of the public key of the issuing group of organizations[6].

Chapter 4

Practical constructions

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

4.1 Preliminaries

Our construction is based on number-theoretic assumptions that certain functions are hard.

4.1.1 The discrete logarithm problem

The hardness of computing discrete logarithms (DL for short) in a group of prime order, is a widely believed conjecture. Suppose we are given an k-bit prime number q and a prime number p such that $q|(p-1)$. By $\mathbb{Z}_p^*$ we denote the multiplicative group modulo p. Let $g \in \mathbb{Z}_p^*$ be of order q. Then g is a generator of a group of order q, let's call it G_q.

Conjecture 1 *Given $g, h \in G_q$, such that h was selected from G_q uniformly at random, it is hard to compute an integer x such that*

$$g^x = h \ mod \ p$$

For the ease of notation, we will sometimes drop the " mod p" part of the arithmetic expressions in G_q.

4.1.2 The Diffie-Hellman problem

The setting for the computational Diffie-Hellman problem[16, 4] is the same as for the discrete logarithm problem: we have a group G_q with a generator g just as before.

Conjecture 2 *Given $g, h_1, h_2 \in G_q$, where $h_1 = g^x \bmod p$ and $h_2 = g^y \bmod p$ are selected uniformly at random, it is hard to compute h_3 such that $h_3 = g^{xy} \bmod p$.*

Besides the standard, computational, Diffie-Hellman conjecture, there is also a conjecture that it is hard to distinguish the distribution D of the form (g, g^x, g^y, g^{xy}) from the distribution R of the form (g, g^x, g^y, g^z), where x, y, z are chosen independently.

Conjecture 3 *For any sequence of primes $\{q_i\}$, where $|q_i| = i$, for all non-uniform families of oracle Turing machines $\{M_k^?\}$, for sufficiently large k, if the a-priori probability of selecting D, as opposed to R as the oracle is $1/2$, then*

$$Pr[M_k^? \text{ guesses the oracle correctly}] \leq 1/2 + neg(k)$$

4.1.3 The random oracle model

The random oracle model[2] is a widespread way of arguing about the security of a cryptographic scheme. Suppose there existed a universally trusted and always available trusted third party. Then most of cryptographic problems would go away. The random oracle is a way of relaxing this desire for a trusted third party, by the desire that there is a trusted source of random function: bits that everyone trusts not to be precomputed, but to have been obtained through a query to an oracle that outputs random bits.

If a scheme is secure in the random oracle model, it follows that breaking it implies solving a computationally hard problem if a truly random oracle existed. But since a random oracle does not exist, and is replaced by a collision-resistant hash function, a proof of security in the random oracle model is a weaker proof of security than a traditional one[8]. However, it is still considered good enough in practice if it is used with a scheme that is sensibly designed (i.e. with a scheme that does not output its secret key upon receiving some low probability input).

4.2 Building blocks

The protocols that follow are the main building blocks of our construction.

4.2.1 Proving equality of discrete logarithms

First, we review protocol Π, the protocol of Chaum and Pedersen[12] 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 $(\tilde{g}, \tilde{h})$.

Theorem 4 *If, as a result of executing protocol* Π*, the verifier accepts, then with probability* $1 - neg(k)$*, the prover knows* x *such that* $g^x = h \bmod p$.

Proof: Suppose we are given rewindable access to the prover, and have him respond to different challenges c_1 and c_2, we obtain x by solving a system of linear equations $y_1 = r + c_1 x$ and $y_2 = r + c_2 x$. Then the prover who can respond to more than one challenge for a fixed commitment r must know x.

Suppose the prover can respond to only one challenge. The probability that the verifier picks that challenge is negligible. Therefore, in this case, the verifier will reject with probability $1 - neg(k)$. ■

Theorem 5 *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* $\tilde{g}^{x_1} = \tilde{h} \bmod p$.

Proof: Suppose the verifier accepts and $x_1 \neq x_2$. Then we have: $y = r_1 + cx_1 = r_2 + cx_2$. It follows that for any choice of r_1, r_2, there is a unique challenge c such that this equation is satisfied. The probability that the verifier will pick this challenge is negligible. ■

Finally, we conjecture that this protocol does not reveal anything about the value of the secret x:

Conjecture 4 *Protocol* Π *is a secure interactive procedure[12, 33].*

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.

4.2.2 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.

4.2.3 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.

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

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.

Theorem 6 *The verifier's output in protocol Γ is independent of the prover's view of the conversation.*

Proof: Suppose that γ and r are fixed, as they are in the beginning of a conversation between the prover and the verifier.

Then observe that for any choice of α, β, there is a unique choice for the verifier's output. And, for each choice of verifier's output, there is a unique choice of α, β.

It follows that, if α and β are always picked uniformly at random, this will induce the uniform distribution on the output of the verifier. Therefore, the prover will get no information as to which conversation a given output corresponds to. ■

4.3 The construction

We are now ready to present our construction based on the building blocks introduced above. We will not present the construction due to Lidong Chen[13] but we would like to acknowledge our construction uses these same building blocks and, as a consequence, many of the same ideas.

4.3.1 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.

4.3.2 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[33] schemes.

4.3.3 The assumptions used

We use the random oracle model. In addition, the hardness assumptions below are necessary to prove security of our scheme for pseudonym systems. Recall the setting – publicly known are: p, a large prime number; q, a large prime divisor of $p-1$; and g a generator of a multiplicative subgroup of $\mathbb{Z}_p^*$ of order q. We denote this group by G_q.

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 introduce a new assumption. Given are:
   (a) $g, g^{s_1}, g^{s_2} \in G_q$
   (b) oracle access to a machine M that does the following: on input x, M produces $(a, a^{xs_2}, a^{s_1 + xs_1s_2})$, where $a \in_R G_q$

   The assumption is that it is hard, after querying M on inputs $\{x_i\}$, to produce a 3-tuple of the form $(a, a^{ys_2}, a^{s_1+ys_1s_2})$ such that $a \neq 1$ and $y \notin \{x_i\}$. While this last assumption appears to be new, and not implied by other standard number-theoretic assumptions, we believe that it is reasonable.

4.3.4 Proof of security granted the assumptions

Under the assumptions above, the proposed practical scheme satisfies the definition of a pseudonym system given in Chapter 2.

It is clear that all the procedures satisfy the input/output specifications.

Key generation: the security of the key generation procedure follows from the assumption about the hardness of computing discrete logarithms.

Nym generation: nym generation is a secure procedure because, if we assume that procedure Π is simulatable, the nym generation is also simulatable, for both parties.

Credential issue: in the random oracle model, credential issue is a secure procedure. This follows because, in the random oracle model, from the security of Γ, we get that there is a simulator that can simulate either party's view of the protocol given this party's input and output.

Credential transfer: credential transfer is secure if Π is secure.

Each pseudonym corresponds to a unique user: The nyms in our system are of the form (a, a^x). The master public/secret key pairs are of the form (g^x, x). Thus, the ID function is the knowledge-extractor for protocol Π that produces x, and thus g^x, x can be formed. Thus, for every nym there is exactly one public key/secret key pair that can be extracted.

Security of the user master secret key: We assume that the equality of discrete logarithms proof is secure. From that, and the hardness of the discrete logarithm problem, we get that pseudonym generation, credential issue and credential transfer are also secure procedures. Therefore, there exists a simulator for any interaction in the system. Therefore everything that can be inferred about the user master secret key throughout the lifetime of the system, can be extracted by simulating the lifetime of the system without the user's participation. That is, x can be extracted from g^x. This contradicts the hardness of discrete logarithm assumption.

Credential sharing implies master secret key sharing: If a party in our scheme can prove knowledge of the secret key through protocol Π, then this party knows x since we could extract x from it by running the knowledge extractor E with it.

Unlinkability of pseudonyms: We show that the pseudonyms are unlinkable by induction. We will show that if two nyms can be linked at moment t, then they could be linked at moment $t-1$. Thus, they can be linked at the time when they were just created. This violates the decisional Diffie-Hellman assumption.

Suppose we are at time t and two nyms are linked for the first time in the lifetime of the system. By "linked" we mean that there exists a function that can now be computed on these pseudonyms that could not be computed before, and could not be computed if a trusted third party were carrying out all credential transfers.

Suppose at time t we just finished an operation not involving the owner of these pseudonyms. Then whatever can be known about them at time t must have been known at time $t-1$.

Suppose at time t we just finished an operation involving the nym holder, but not

involving any other of his pseudonyms. If, as a result of this operation, we can link the two pseudonyms, then we have also linked some other of his pseudonyms, and so we consider these other nyms as being linked for the first time. So without loss of generality assume that at time t we just performed an operation that involved one of the pseudonyms.

Suppose we just finished credential transfer between these two pseudonyms. But credential transfer is a secure interactive procedure. Therefore, whatever can be computed as a result of credential transfer, can be computed just from its output. But if we could link nyms as a result of the output of credential transfer, then we violate the decisional Diffie-Hellman assumption. Therefore, if our last operation at time t was credential transfer, then two nyms must already have been linked at time $t-1$.

Suppose at time t we just finished credential issue on one of these pseudonyms. But credential issue is a secure procedure as a result of which the organization does not receive any output. Therefore, if two nyms can be linked now, they could already be linked at time $t-1$.

Finally, can two pseudonyms be linked at the time when one of them is just generated? No, because nym generation is a secure procedure, and inferring information from its output would violate the decisional Diffie-Hellman assumption. Then the nyms can be linked before one of them is even generated, which is a contradiction.

Unforgeability of credentials: This follows from the new assumption and the zero-knowledge nature of protocol Γ. In particular, the process of establishing a nym with an organization, and then obtaining a credential from that organization, corresponds precisely to the behavior of the oracle from the new assumption.

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.

The modification of the model will have the issuing organization participate in the credential transfer procedure. Suppose the user has a credential C. He computes a function on it, $f(C)$, and hands $f(C)$ over to the receiving organization. He is able to prove that $f(C)$ corresponds to a nym that he owns, but he is unable to prove that $f(C)$ is a credential. However, if the receiving organization can ask the issuing organization's help with the verification that $f(C)$ corresponds to a credential, that would confirm that the user indeed has a credential.

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 cooperates 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). This is as secure as the scheme with one-time credentials.

Chapter 5

Conclusions and open questions

The present thesis poses a number of open problems which fall into three main categories. The first category of open problems is definitional: what are the desirable features of a pseudonym system not already captured by our model?

The second category is coming up with theoretical constructions for our model and its extensions that would not rely as heavily on zero-knowledge proofs.

Finally, the third category of open problems has to do with practical constructions for pseudonym systems in our model. Here, we have open problems of two flavors: a number-theoretic question of whether our assumption is equivalent to the known assumptions such as the decisional Diffie-Hellman assumption; and the problem of a practical construction of a pseudonym system scheme with multiple-use credentials.

5.1 Extensions to the model

We have considered some extensions to the model that would allow to incorporate such features as expiration dates, use l times credentials, distributed duties of each organization, credentials from a group of organizations, anonymity and credential revocation mechanisms, and credentials that cannot be used more than once.

However, the list of possibilities is by no means exhausted. Depending on the application of a pseudonym system, there may arise extensions of the model we have not considered.

In general, we believe that the model we have presented captures the essential ingredients of a pseudonym system: the notions of users and organizations are well defined, and from these we derive the unforgeability and unlinkability properties of a pseudonym system. As a result of the unforgeability property, the organizations rest assured that a user cannot forge a credential; as a result of the unlinkability property, a user can rest assured that whatever an organization can learn about him, it can only learn through him.

5.2 Theoretical constructions

The basic theoretical constructions we have given, although impractical due to the heavy use of zero-knowledge proofs, are optimal in the sense that the complexity of all operations depends on the security parameter only.

However, there are many ways to improve our theoretical constructions. The main open question here is the construction of a non-interactive theoretical pseudonym system, i.e. a pseudonym system where instead of having an interactive zero-knowledge proof of correctness for each credential, we can have non-interactive zero-knowledge proofs. It would be better still to be able to eliminate zero-knowledge proofs altogether and base the pseudonym system on general assumptions such as one-way functions and trapdoor permutations.

In addition, nice theoretical constructions are wanted for the extensions of the basic model. For example, our construction of credentials from a group of organization is impractical: the public key of the group is linear in the size of the group.

5.3 New practical constructions

We have presented a construction which is an improvement over the previous best practical construction due to Chen[13]: the notion of a user is well-defined, and the organizations in the system are protected against a cheating CA. We have also made it possible to eliminate the CA in some cases while Chen's scheme requires the presence of a CA.

On the other hand, although a significant improvement over the existing practical construction, our construction is still far from giving us the main mechanism we are looking for: a practical pseudonym system with multiple-use credentials.

In addition, our practical scheme is based on a new assumption. It is an open problem to come up with a practical scheme for a pseudonym system that would not be utilizing this assumption.

5.4 Conclusions

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.

We do not live in a world where everyone has a public key and a secret key they use to sign documents every day. However, it seems that we are moving towards it. As we are shifting into the computerized world, encryption and digital signatures are becoming increasingly important, and, hand in hand with them, the desire to protect one's keys.

Therefore, although not immediately practical, we believe that the model in which we are working will become so in its due time.

Bibliography

[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] M. Blaze, J. Feigenbaum, and J. Lacy. Managing trust in medical information systems. *AT&T Technical Report 96.14.1*, 1996.

[4] Dan Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, pages 48–63. Springer-Verlag, 1998.

[5] David Brin. *The Transparent Society: Will Technology Force Us to Choose between Privacy and Freedom?* Perseus Press, 1998.

[6] 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.

[7] Ran Canetti, Moses Charikar, Ravi Kumar, Sridhar Rajagopalan, Amit Sahai, and Andrew Tomkins. Non-transferable anonymous credentials. *Manuscript, 1998. Revision in submission*, 1999.

[8] 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.

[9] David Chaum. Security without identification: transaction systems to make Big Brother obsolete. *Communications of the ACM*, 28(10), 1985.

[10] David Chaum. Designated confirmer signatures. In *Advances in Cryptology—EUROCRYPT 94*, pages 86–91. Springer-Verlag, 1994.

[11] 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.

[12] David Chaum and Torben Pryds Pedersen. Wallet databases with observers (extended abstract). In *Advances in Cryptology—CRYPTO '92*, pages 89–105. Springer-Verlag, 1992.

[13] 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.

[14] 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.

[15] 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.

[16] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[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] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology—EUROCRYPT 96*, pages 143–154. Springer-Verlag, 1996.

[26] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology—CRYPTO '97*, pages 150–164. Springer-Verlag, 1997.

[27] Joe Kilian and Erez Petrank. Identity escrow. In *Advances in Cryptology—CRYPTO '98*, pages 169–185. Springer-Verlag, 1998.

[28] Anna Lysyanskaya, Ronald L. Rivest, and Amit Sahai. Pseudonym systems. *Manuscript, 1999. Revision in submission*, 1999.

[29] 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.

[30] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[31] Tatsuaki Okamoto. Designated confirmer signatures and public-key encryption are equivalent. In *Advances in Cryptology—CRYPTO '94*, pages 61–74. Springer-Verlag, 1994.

[32] 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.

[33] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[34] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[35] Paul F. Syverson, Stuart G. Stubblebine, and David M. Goldschlag. Unlinkable serial transactions. In *Financial Cryptography: First International Conference, FC '97*, pages 39–55. Springer-Verlag, 1997.