Security aspects of Zero Knowledge identification schemes

Andreea Mihaela Panait

Department of Mathematics and Statistics,
McGill University, Montréal
Québec, Canada

February, 2008

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements of the degree of
Master of Science

Abstract

In this thesis we follow two directions: Zero Knowledge Protocols and the Discrete Logarithm Problem. In each direction we present the necessary background and we give a new approach for some parts of the existing protocols.

The new parts are dedicated to the soundness property of the Schnorr Identification Scheme and to the security of the $\Sigma+$ -Protocol. Since both directions are very well-known and studied in the field of cryptography, they are presented with many details so that the new results are easy to follow.

In writing this thesis we have tried to present the material in a specific order and in a manner easy to read even by beginners in cryptography.

Résumé

Dans cette thèse on suit deux directions : les protocoles Zero-Knowledge et le Problème du Logarithme Discret. Dans chaque cas on présente toutes les connaissances préliminaires nécessaires et on donne une nouvelle approche pour certaines parties des protocoles existants.

Les parties originales sont dédiées à la correction du Schèma d'Identification de Schnorr et à la sécurité du protocole $\Sigma+$. Comme les deux directions sont très connues et étudiées dans le domaine de la cryptographie, elles sont présentées avec beaucoup de détails pour mieux comprendre les nouveaux résultats.

Partout dans cette thèse on a essayé de présenter le matériel dans un ordre spécifique et dans une manière spéciale afin qu'elle soit facile à lire et à suivre par les débutants du domaine de la cryptographie.

Acknowledgments

I would like to express my gratitude to my supervisor Prof. Claude Crépeau for the patience that he has had with a new student in cryptography in accomplishing the desired goal. I would also like to thank my family and my unborn child for letting me finish this work.

Table of Contents

Introduction

We will begin speaking about cryptology by explaining the origin of the word itself. The term "**cryptology**"comes from ancient Greek and it has two components: "kryptos" which means hidden and "logos" which means word.

Besides the origin of the word it is interesting to mention where it first appeared. Many consider that the Egyptian hieroglyphs are the earliest form of cryptology. The Egyptian writing system is complex but relatively straightforward. The inventory of signs is divided in three major categories: (1) logograms, signs that write out morphemes, (2) phonograms, signs that represent one or more sounds and (3) determinatives, signs that denote neither morpheme nor sound but help with the meaning of a group of signs that precede them. It was not until the nineteenth century that Egyptian hieroglyphs were deciphered. Several people had been trying to crack the code when Jean-François Champollion discovered the secret to this ancient writing. He published in 1828 a paper " Précis " where he explains how to read hieroglyphs. His work was based on the discovery of the Rosetta Stone (a stone with writing on it in two languages, Egyptian and Greek using three scripts: hieroglyphic, demotic and Greek) found in 1799 by the French soldiers.

Used with the same purpose as today, that of protecting the military and diplomatic communications, one of the first cryptosystem used in history was due to the roman emperor Julius Caesar. He had the brilliant idea of protecting the messages sent to his military troops by shifting each letter three places to the right in the

alphabet.

In the field of cryptology we distinguish two major and well studied directions: cryptography and cryptanalysis. A fundamental goal of cryptography is to give to two people the possibility to transmit information over an insecure channel such that a possible adversary cannot understand the message. This idea can be organized formally by the notion of a cryptosystem. In a cryptosystem the sender's information is encrypted by using a predetermined key and the ciphertext, obtained after that, is sent over the channel to the receiver. The adversary seeing the ciphertext cannot determine the original message but the receiver, knowing the encryption key, can decrypt the ciphertext and reconstruct the initial message. People working in cryptography are interested in developing cryptosystems which are secure to use. Cryptanalysis is dedicated to developing new attacks to break the security of the existing cryptosystems.

Seeing all the various domains (where cryptography is used) such as e-mail, diplomatic and military communication and e-commerce we can understand the desire of cryptanalysts to break existing cryptosystems.

The major tools in " modern " cryptology are mathematical notions and results. They give us the possibility to encrypt and decrypt data.

In this thesis I will try to describe two existing cryptographic protocols and give a new extension for each one.

The first one is the Schnorr Identification Scheme which is one of the existing Zero Knowledge Protocols. We will give a detailed description of the scheme and we will demonstrate the security of the protocol.

The second one is the $\Sigma+$-Protocol which gives efficient proofs of knowledge for any exponentiation homomorphism in groups with hidden order. Studying the security of the protocol we raise a question never studied by the authors.

The thesis is structured as follows: in the first chapter we give a presentation of

the theoretical background necessary to follow the material. In the second chapter we describe the field of identification schemes, concentrating on the Zero Knowledge Protocols. Chapter 3 describes the Schnorr Identification Scheme together with the security of the scheme.

In Chapter 4 we develop our new result regarding the security of the scheme. Chapter 5 introduces us to the field of the Discrete Logarithm Problem with a focus on proofs of knowledge. In Chapter 6 we describe the Σ+-Protocol and in Chapter 7 we present the modification that can be easily done in the protocol and which counters some proposed attacks to the protocol as described in [16].

The thesis ends with conclusion and bibliography.

Chapter 1

Theoretical Background

Definition 1.1. A **proof** has the following two properties:

(1) Soundness

No false affirmation can be proven.

(2) Completeness

Every true affirmation can be proven.

Definition 1.2. An **interactive proof system** is a proof where the completeness is verified and there is a negligeable probability that the soundness property is not verified.

Definition 1.3. An **identification scheme** is a scheme which allows someone with a secret information to convince the other party of the knowledge of the secret information.

Definition 1.4. Impersonation is when an adversary is able to act as the sender and to answer all receiver's challenges.

Definition 1.5. An interactive proof system is called a **proof of knowledge** if the following affirmation is verified: if someone is able to impersonate the sender then this is equivalent with knowing the sender's private key.

Definition 1.6. A proof of knowledge is called **Zero Knowledge** if the following affirmation is verified: if the information exchanged between the two parties constitutes a proof of knowledge then a cheating verifier cannot deduce anything else except that fact.

Formally, Zero Knowledge is demonstrated via the notion of a simulator. For details see [15].

Definition 1.7. An **authentication scheme** is a five-tuple $(P, A, \boldsymbol{K}, S, V)$ where the following conditions are satisfied:

(1) P is a set of possible messages

(2) A is a set of possible signatures or authenticators

(3) $\boldsymbol{K}$, the keyspace, is a finite set of possible keys

(4) For each $K \in \boldsymbol{K}$, there is a signing algorithm $sig_K \in S$ and a corresponding verification algorithm $ver_K \in V$. Each $sig_K : P \to A$ and $ver_K : P \times A \to \{true, false\}$ are functions such that the following equations are satisfied:

for all $x, K, ver_K(x, sig_K(x)) = true$

for all $x, K, y \neq sig_K(x), ver_K(x, y) = false$

with the two algorithms sig_K and ver_K efficiently computable.

Definition 1.8. ElGamal Signature Scheme

Let p be a prime such that the Discrete Logarithm Problem in $\mathbf{Z}_p$ is intractable and let $\alpha \in \mathbf{Z}_p^*$ be a primitive element. Let $P = \mathbf{Z}_p^*$, $A = \mathbf{Z}_p^* \times \mathbf{Z}_{p-1}$ and define:

$$K = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

The values p, α, β are public and a is private.

For $K = (p, \alpha, a, \beta)$ we define:

$$sig_K(x) := (\gamma, \delta)$$

where $\gamma := \alpha^k \bmod p$ and

$$\delta := (x - a\gamma)k^{-1} \bmod p - 1$$

where k is a secret random number, $k \in \mathbf{Z}_{p-1}^*$

For $x, \gamma \in \mathbf{Z}_p^*$ and $\delta \in \mathbf{Z}_{p-1}$, we define:

$$ver_K(x, (\gamma, \delta)) := true \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p$$

In the following definition we use the same notations *sig* and *ver* for the keys and for the algorithms using these keys.

Definition 1.9. A **certificate** for someone in the network will consist of some identifying information for that person (e.g., their name, email address, etc), their public key(s) and the signature of the TA (the Trusted Authority) for that information.

Issuing a Certificate to Alice is a protocol with the following steps:

1. The TA establishes Alice's identity by means of conventional forms of identification such as a birth certificate, passport, etc. Then the TA forms a string denoted ID(Alice).

2. A private signing key for Alice, sig_{Alice} and a corresponding public verification key, ver_{Alice} are determined.

3. The TA generates its signature

$$s = sig_{TA}(ID(Alice) \parallel ver_{Alice})$$

on Alice's identity string and verification key.

The Certificate

$$\textbf{Cert(Alice)} = (ID(Alice) \parallel ver_{Alice} \parallel s)$$

is given to Alice, along with Alice's private key: sig_{Alice}

Definition 1.10. A language is a set of strings.

Definition 1.11. A language is assigned to the **NP** (nondeterministic polynomial time) class if it is solvable in polynomial time by a nondeterministic Turing machine. For details see [20].

Definition 1.12. A **one-way hash function** maps an arbitrary-length input message M to a fixed-length output hash H(M) such that the following properties hold:

One-way: Given a hash $H(M)$, it is difficult to find the message M.

Second preimage resistant: Given a message M_1, it is difficult to find another message M_2 such that $H(M_1) = H(M_2)$.

Collision resistant: It is difficult to find two messages M_1 and M_2 such that $H(M_1) = H(M_2)$.

Definition 1.13. Given a hash function $h : X \to Y$ and an element $y \in Y$ the **Preimage Problem** is to find an $x \in X$ such that $h(x) = y$.

Definition 1.14. The **Root Problem** for an arbitrary group H is to compute a $h \in H$ such that $h^e = u$ given an integer $e > 1$ and a group element $u \in H$.

Definition 1.15. In cryptography, **a commitment scheme** is a method that allows a user to commit to a value while keeping it hidden, and while preserving the user's ability to reveal the committed value later. A useful way to visualize a commitment scheme is to think of the Sender as putting the value in a locked box, and giving the box to the Receiver.

The value in the box is hidden from the Receiver, who cannot open the lock (without the help of the Sender), but since the Receiver has the box, the value inside cannot be changed. Commitment schemes are important to a variety of cryptographic protocols, especially Zero-Knowledge proofs and secure computation.

A commitment scheme take place in two phases: the **commit phase** during which a value is chosen and specified, and the **reveal phase** during which the value

is revealed and checked. In the simplest commitment schemes, the commit phase consists of a simple message from the Sender to the Receiver, while the reveal phase consists of a single message from the Sender to the Receiver, followed by a check performed by the Receiver.

Definition 1.16. A **graph** is an ordered pair $(V(G), E(G))$ consisting of a nonempty set $V(G)$ of vertices and a set $E(G)$, disjoint from $V(G)$, of edges which are unordered pairs of vertices of G.

Definition 1.17. Two graphs which contain the same number of vertices connected in the same way are said to be **isomorphic**.

Formally, two graphs **G** and **H** with vertices $V_n = \{1, 2, ..., n\}$ are said to be isomorphic if there is a permutation p of V_n such that $\{u, v\}$ is in the set of edges $E(\mathbf{G})$ if and only if $\{p(u), p(v)\}$ is in the set of edges $E(\mathbf{H})$.

Chapter 2

Identification Schemes. Zero Knowledge Protocols

An identification scheme gives the possibility to a person to confirm the identity of another person. The main property of an identification scheme is that if someone is watching the information flow he cannot easily impersonate the person who is proving its identity. The adversary is able to observe the information transmitted between the sender (say Alice) and the receiver (say Bob) and its first goal is to impersonate Alice.

An example of an identification scheme is a zero-knowledge protocol. Such a scheme allows Alice to prove her identity without revealing anything from her identifying information.

An intuitive good example is the following: suppose that Alice knows how to solve the Rubik's cube and she wants to convince Bob that she can, without revealing the solving procedure. Then they can act in the following way: Alice gives Bob the Rubik's cube, then Bob scrambles the cube and then he returns it to Alice. Alice finds a place where Bob cannot see her, she solves the cube and she gives it back to Bob.

On one hand, Bob is convinced that Alice knows how to solve the Rubik's cube because he saw that she was solving it and on the other hand, he clearly didn't see the solution.

For identification purpose we wish that each person has something unique. In reality this is true if we think of physical attributes but we need to adapt this to some other types of information.

In the field of identification schemes some useful and secure schemes have been discovered.

In design, a first component that these schemes should have, as explained below, is: "random challenges". A random challenge is a mathematical statement chosen at random for which the proof is known and by using it someone is asked to prove it.

If Alice transmits to Bob information to identify herself which never changes then the scheme can be easily proved as insecure.

There are two directions in building identification schemes known in the literature. One way is to use the simple cryptographic primitives like signature schemes or message authentication codes and the other way which we will present in this thesis is to build new identification schemes.

One of the basic examples of identification schemes can be found in Shafi Goldwasser et al in 1985 paper: "The knowledge complexity of interactive proof-systems" [15].

Let $\mathbf{Z}_n^*$ the set of integers from 1 to n relatively prime with n. We say that $a \in \mathbf{Z}_n^*$ is a *quadratic residue mod n* if $r^2 \equiv a \pmod{n}$ for some $r \in \mathbf{Z}_n^*$. r is called a *square root of a*. Otherwise, we say that $a \in \mathbf{Z}_n^*$ is a *quadratic nonresidue mod n*.

The quadratic residuosity problem is to decide, given n, if a is a *quadratic residue mod n*.

Consider now the following interactive protocol between a Prover that knows n, a and a square root r of a and a Verifier that knows n and a.

Prover Verifier

He picks at random $t \in \mathbf{Z}_n^*$, computes $y \equiv t^2 \pmod{n}$.

He sends it to the Verifier.

$$y \longrightarrow$$

He picks at random $b \in \{0, 1\}$.

He sends it to the Prover.

$$\longleftarrow b$$

If $b = 0$ then he sets $z = t$, otherwise he sets $z = rt$.

He sends z to the Verifier.

$$z \longrightarrow$$

If $b = 0$ then the Verifier accepts iff $z^2 \equiv y \pmod{n}$.

If $b = 1$ then the Verifier accepts iff $z^2 \equiv ya \pmod{n}$.

The protocol is complete because if all of the parties are honest and follow the protocol then the Verifier accepts with probability 1.

For the soundness we suppose that P' is a prover strategy that makes the Verifier accept with probability bigger than $\frac{1}{2}$. Then one of the possible first messages y sent by the Prover P' must be such that V accepts for both choices $b = 0$ and $b = 1$. Let z_0, z_1 be the third round messages sent by P' in such cases.

Then we have ${z_0}^2 \equiv y \pmod{n}$ and ${z_1}^2 \equiv ya \pmod{n}$, so that $a \equiv ({z_0}^{-1} z_1)^2 \pmod{n}$. Therefore a is a quadratic residue mod n.

In conclusion, if a is not a *quadratic residue mod n* then the Verifier accepts with

the probability $\leq \frac{1}{2}$.

We can construct a simulator for the zero knowledge property as follows:

Let V' be an arbitrary Verifier strategy. Given n, a the simulator algorithm for V' follows the steps:

1. Pick at random $b \in \{0, 1\}$ and $z \in \mathbf{Z}_n^*$; pick randomness R for V'.

2. If $b = 0$ set $y \equiv z^2 \pmod{n}$ else $y \equiv a^{-1}z^2 \pmod{n}$.

3. If V', using randomness R, given y as first message, outputs b, then halt and output transcript: " V' selects randomness R, P sends y at first round, V' sends b at second round, P sends z at third round, V' accepts."

4. Go to 1.

The most important information in the analysis of the simulator is that regardless of the choice of b, the simulator chooses y as a uniformly distributed quadratic residue in $\mathbf{Z}_n^*$. This means that y and b, as random variables, are statistically independent and the second message of V' given y is also statistically independent of b.

Therefore, the simulator has probability $\frac{1}{2}$ of outputting a simulation in each attempt.

With these three properties we have a zero knowledge interactive proof.

Efficiency determined in the last years the discovery of new identification schemes which can be implemented in practice like: **The Schnorr Identification Scheme**, **The Okamoto Identification Scheme** and **The Guillou-Quisquater Identification Scheme**.

In a zero knowledge scheme the Verifier can establish that the answer from the Prover is correct but nothing else except that. A typical round in a zero knowledge scheme is:

Prover		Verifier
"commitment" message	$\longrightarrow$	
	$\longleftarrow$	challenge
response to the challenge	$\longrightarrow$	

This can be repeated for several rounds but would be less efficient. Taking into consideration all the responses of the Prover in all the rounds, the Verifier then accepts or rejects the proof.

An intuitive idea of zero knowledge proofs was published by Jean-Jacques Quisquater et al in their paper: "How to explain Zero Knowledge Protocols to your children" [17].

The basic story takes place in the ancient city of Baghdad. In this town there was an old man named Ali Baba. Ali Baba was selling and buying different things almost every day. One day a thief took one of Ali Baba's purses and started to run. Immediately, Ali Baba ran after him until the thief went into a cave having two different passages: one to the right and one to the left.

Ali Baba didn't see where the thief was going so he picked by chance to go to the left. He searched all the way to the end of the left passage and he didn't discover the thief. He was then certain that the thief had gone to the right. After searching the right passage as well, he was confused by the disappearance of the thief, he gave up and went home.

The next day the situation was repeated and another thief stole one of Ali Baba's things; the second thief entered into the same cave as the first. Now knowing the first incident from the day before Ali Baba chose to go to the right side but he was surprised to not find the thief. He didn't check the left passage, assuming that it was a problem of bad luck in choosing the passage.

This was just the beginning since now, every day, a thief stole something from Ali Baba. As we can imagine each day the thief was not caught. After forty days

the only explanation was that each time the thief was lucky enough to escape. Being unsatisfied by this explanation, Ali Baba decided on the next day to hide into the cave and discover the mystery.

The next day he chose the right passage and he stayed there until a thief appeared. The thief approached the wall and he said the magic words: " Open Sesame" after which the wall started to open, letting the thief escape. The mystery was solved. He even heard the magic words and he tried himself to open the walls. He was amazed by the fact that the two passages communicate and give the possibility of escaping from the cave.

After discovering the words he decided to change them such that no one else could use the old ones. This gave the possibility of catching the thieves from now on.

He also decided to write his story but without revealing the new magic words. The manuscript can be seen in our days near Boston in the United States of America.

Using this story we can describe an example of a zero knowledge protocol between Alice(the Prover) and Bob(the Verifier):

The purpose of Alice is to prove to Bob that she knows the secret phrase which will open the gate R-S in the cave but of course without revealing the secret to Bob. Alice's commitment is to go to R or S. One round can be as follows: (see figure on the next page)

Bob goes to P and he waits until Alice goes to R or S. Then he goes to Q and he asks Alice to appear from the tunnel (from the right or left side). Supposing that Alice does not know the right words the probability of appearing from the good side is only 1/2. Bob will repeat this round until he is sure that Alice knows the secret phrase. We observe that no matter how many times the proof is repeated, Bob cannot discover the secret words.

Repeating the protocol, the probability for Alice to guess each time the correct side is $\leq \frac{1}{2}^{(\text{the number of repetitions})}$ and decreases significantly.

Another important fact is that by using randomization for the initial path that Alice takes and hiding it from Bob we can eliminate the possibility of Bob discovering the secret words (simply by following Alice).

Before we formalize the concept of zero knowledge protocol it is good to have many examples. I found in the literature another very well designed and significant example with two isomorphic graphs G_1, G_2. Let's take a look to it:

Prover Verifier

He knows $\phi(G_1) = G_2$.

He picks a random permutation π, $\pi(G_1) = H$.

$H \longrightarrow$

He picks $i \in \{1, 2\}$ at random.

He sends the index $i = 1$ or $i = 2$.

$\longleftarrow i$

If G_1 is received then take $\alpha := \pi$.

If G_2 is received then take $\alpha := \pi \circ \phi^{-1}$.

$$\alpha \qquad \longrightarrow$$

He checks if $\alpha(G_i) = H$.

The protocol is complete because the Verifier is convinced at the end that the graphs are isomorphic.

It is sound because:

-if G_1, G_2 are not isomorphic then the Prover is not able to construct H isomorphic to both G_1 and G_2;

-the Verifier has probability at least $\frac{1}{2}$ to pick the graph to which H is not isomorphic;

-after k rounds the probability of successful cheating is $\frac{1}{2^k}$.

With these two properties knowing that we have an interactive proof system then to prove that is zero knowledge we construct a simulator (see definition 1.6).

The simulator guesses G_1 or G_2. If it chooses, say G_1, then it chooses randomly a permutation π, calculates $\pi(G_1) = H$ and sends it to the Verifier.

We observe that the simulator knows the correct answer only if the Verifier chooses the same G_1 or G_2 as the simulator. If the other graph is chosen then we skip that round and start over with a new round.

Since the protocol has these three properties then we conclude that it is zero knowledge.

Now we can formalize the general concept of a Zero Knowledge protocol. The notion was first introduced by Shafi Goldwasser et al in 1985 in their paper: "The knowledge complexity of interactive proof-systems" [15].

The authors of the paper discuss interactive proof systems. They give a computational complexity measure of knowledge and measure the amount of additional

knowledge contained in proofs. Depending on the amount of additional knowledge that should be revealed to prove the identity they propose classification of different languages.

As they mention in the beginning of the paper their interest is " the case where the additional knowledge is essentially zero " and they show that it is possible to interactively prove that a number is a *quadratic residue mod m* while releasing zero additional knowledge. They state, "This is surprising as no efficient algorithm for deciding quadratic residuosity *mod m* is known when $m's$ factorization is not given. Moreover, all known NP proofs for this problem exhibit the prime factorization of m. This indicates that adding interaction to the proving process, may decrease the amount of knowledge that must be communicated in order to prove a theorem."

Suppose that we have a protocol between the Prover and the Verifier. We introduce the following notions:

(1) Soundness

No false affirmation can be proven.

(2) Completeness

Every true affirmation can be proven.

An **interactive proof system** is a proof where the completeness is verified and there is a negligeable probability that the soundness property is not verified.

An interactive proof system is called a **proof of knowledge** if the following affirmation is verified: if someone is able to impersonate the sender then this is equivalent with knowing the sender's private key.

In this context the notion of soundness can be adapted as: the Prover is able to convince the Verifier that he has the private key only if he actually has it.

The soundness property can be also used in the following form: a scheme is sound if there exists an efficient algorithm (a **knowledge extractor**) which uses the Prover as a blackbox to obtain the secret information.

A proof of knowledge is called **zero knowledge** if the following affirmation is verified: if the information exchanged between the two parties constitute a proof of knowledge then a cheating verifier cannot deduce anything else except that fact.

Formally zero knowledge is demonstrated via the notion of a simulator. For details see [15].

From mathematical point of view, interactive proofs are not very rigorous because we always have a small probability for the cheating Prover to convince the Verifier of his knowledge (see the soundness property). That's why some techniques have been developed to decrease this probability.

Different papers studying identification schemes are presently known. We mention [5] and [22].

In the field of identification schemes, A.Fiat and A. Shamir opened a very important path by showing how to use zero knowledge techniques to create new more efficient schemes. The most important ones are: **The Schnorr Identification Scheme** and **The Guillou-Quisquater Identification Scheme**.

When a new identification scheme is created the normal follow-up is to study its security under different types of attacks. For these two schemes passive and active attacks were studied. The adversary can impersonate the Verifier under both types of attack. The passive attack is weaker than the active one.

In the passive attack the adversary can obtain transcripts from the exchange of information between the two parties. In the active attack we can consider that the adversary will impersonate a cheating Verifier and is capable of opening new interactions with different presumed provers.

Until now the problem of proving security against impersonation under active attacks is still open. Mihir Bellare and Adriana Palacio studied the security of **The Schnorr Identification Scheme** and **The Guillou-Quisquater Identification Scheme** [4].

In the next chapter we will describe **The Schnorr Identification Scheme** and we will study in detail the soundness property of this scheme.

Chapter 3

The Schnorr Identification Scheme and its security

The scheme was first introduced by C.P. Schnorr in 1989 [19]. He presented at that time a new public-key signature scheme and a corresponding authentication scheme, both based on the discrete logarithms in a subgroup of units in $\mathbf{Z}_p$ with p a prime sufficiently large. The protocol wanted to improve the speed of the ElGamal signature scheme introduced in 1985.

The protocol combines ideas from the schemes of ElGamal and Fiat and Shamir schemes [12]. We now start to describe the protocol:

The scheme needs a trusted authority TA to choose common parameters (called domain parameters) with the properties:

p is a large prime, $p \simeq 10^{1024}$

q is a large prime divisor of $p-1$, $q \simeq 2^{160}$

α is an element in $\mathbf{Z}_p^*$ with order q

t is a security parameter such that $q > 2^t$

As an observation $t = 40$ will assure a safe level of security of the protocol.

The domain parameters introduced above are considered public information and

they do not change within the protocol.

The private key a of each participant in the protocol is generated inside the set $0 \leq a \leq q-1$. The user calculates $v \equiv \alpha^{-a} \pmod{p}$ or $v \equiv (\alpha^a)^{-1} \pmod{p}$ or $v \equiv \alpha^{q-a} \pmod{p}$ and makes v public information.

The parameters (public and private) for each participant to the protocol are certified by the TA. The Schnorr Identification Scheme can be described as follows:

Steps:

1. Alice chooses a random $k \in \{0, 1, ..., q-1\}$ and she computes
$\gamma \equiv \alpha^k \pmod{p}$

She sends the Certificate $Cert(Alice)$ and γ to Bob.

2. Bob verifies Alice's public key v on the Certificate $Cert(Alice)$. He chooses a random $r \in \{1, 2, ..., 2^t\}$ and he sends r to Alice.

3. Alice computes: $y \equiv k + ar \pmod{q}$ and she sends y to Bob.

4. Bob verifies that: $\gamma \equiv \alpha^y v^r \pmod{p}$ and if it is true then he accepts, otherwise Bob rejects.

To visualize the transmission of information we can represent everything as:

Alice (Prover) Bob(Verifier)

1. $k \in \{0, 1, ..., q-1\}$

$\gamma \equiv \alpha^k \pmod{p}$

$\gamma \longrightarrow$

2. $r \in \{1, 2, ..., 2^t\}$

$\longleftarrow r$

3. $y \equiv k + ar \pmod{q}$

$y \qquad \longrightarrow$

4. Verify that: $\gamma \equiv \alpha^y v^r \pmod{p}$

The step number 4 will allow Alice to convince Bob of her identity (assuming that both participants are honest). The following calculation can be done:

$\alpha^y v^r \equiv \alpha^{k+ar} v^r \pmod{p}$

$\equiv \alpha^{k+ar}(\alpha^{-a})^r \equiv \alpha^{k+ar-ar} \equiv \alpha^k \equiv \gamma \pmod{p}$

Therefore, Alice has a private key a which she uses along the protocol, she convinces Bob that she has it but she never reveals its value.

The initial purpose of this scheme was to improve the speed and the efficiency of the computations. In this direction we can track the amount of information exchanged during the protocol.

In the first step Alice sends γ (1024 bits of information) as well as her certificate. Then Bob sends r (40 bits of information). In the third step Alice gives y to Bob (160 bits of information). The last step consists of a verification with no transmission of information between the two parties. The conclusion is that the quantity of information is reasonable to have an efficient protocol.

The speed of the scheme depends on the computations done by each participant. For practical reasons it is desirable that Alice performs all the operations with a smart card of limited power and Bob will use a more powerful computer.

The amount of computation done by Alice is pretty small: she starts with an exponentiation (modulo p) in step 1 (that she can do offline even before the protocol starts) and then in step 3 she only does one multiplication and one addition (modulo q).

The scheme can be introduced in any finite group **G**. Any finite group with a multiplication rule which can be computed efficiently and with the property that the

discrete logarithm problem is hard to solve is a good candidate. In this general case the element $\alpha \in \mathbf{Z}_p^*$ should have order q with the property that q has a prime factor larger than 2^{140}. If the prime q is made public then we just follow the same idea of the initial protocol. If only an upper bound M of q is given, then the following modifications appear:

1. r is randomly chosen in the set $\{1, 2, 3, ..., M\}$

2. in step 3 the calculation modulo q is replaced by modulo M.

In the literature Girault proposed such an identification scheme in $\mathbf{Z}_n$ with n a composite modulus at Eurocrypt'91 [14]. Other examples of groups where the same theory can be developed are class groups and elliptic curves $E(K)$ over a finite field K.

Security of the Schnorr identification Scheme

We will take into consideration some aspects of the security of this protocol. An attack on the scheme will be directed into an adversary named Olga who tries to impersonate Alice.

The first possibility to break the protocol is for Olga to guess the correct value r chosen by Bob. Supposing that Olga has the correct value for r then she can pick any y and she calculates: $\gamma \equiv \alpha^y v^r \pmod{p}$. She sends γ to Bob who will send her the challenge r. Then she transmits y which is already chosen. When Bob verifies the specific result, it will be just the definition of γ. Therefore, he will automatically accept.

In this attack the probability of breaking the scheme is equal to the probability with which Olga can guess the value r. That means:

$P = \frac{\text{number of favorable cases}}{\text{total number of cases}}$

$P = \frac{1}{2^t} = 2^{-t}$

Using the fact that $r \in \{1, 2, ..., 2^t\}$ and t the security parameter is sufficiently

large ($t = 40$) then this attack is hard to implement in practice.

The most important thing about the challenge r is the fact that in each round Bob picks a statistically independent value for r. If this was not done then Olga could follow the previously mentioned attack.

Another possible method to break the protocol is for Olga to compute two values r_1, r_2 with the corresponding y_1, y_2 for the same value γ.

Then:

$\gamma \equiv \alpha^{y_1} v^{r_1} \pmod{p}$

$\gamma \equiv \alpha^{y_2} v^{r_2} \pmod{p}$

$\Leftrightarrow \alpha^{y_1} v^{r_1} \equiv \alpha^{y_2} v^{r_2} \pmod{p}$

$\Leftrightarrow \alpha^{y_1 - y_2} v^{r_1} \equiv v^{r_2} \pmod{p}$

$\Leftrightarrow \alpha^{y_1 - y_2} \equiv v^{r_2 - r_1} \pmod{p}$

Using the definition of $v \equiv \alpha^{-a} \pmod{p}$ we obtain:

$\alpha^{y_1 - y_2} \equiv (\alpha^{-a})^{r_2 - r_1} \pmod{p}$

$\Leftrightarrow \alpha^{y_1 - y_2} \equiv \alpha^{-a(r_2 - r_1)} \pmod{p}$

Since $ord(\alpha) = q$ in $\mathbf{Z}_p^*$ then we identify:

$y_1 - y_2 \equiv -a(r_2 - r_1) \pmod{q}$

Olga's goal is to find the value of the Alice's private key a.

Since $r_1, r_2 \in \{1, 2, ..., 2^t\}$ then $1 \leq r_1 \leq 2^t$, $1 \leq r_2 \leq 2^t$ and $0 < |r_2 - r_1| \leq 2^t$.

Since the prime $q > 2^t$ then the element $|r_2 - r_1|$ is invertible in $\mathbf{Z}_q$ and it exists $(|r_2 - r_1|)^{-1}$ in $\mathbf{Z}_q$.

This allows Olga to compute:

$y_1 - y_2 \equiv -a(r_2 - r_1) \pmod{q}$

$y_1 - y_2 \equiv a(r_1 - r_2) \pmod{q}$

which implies: $a = (y_1 - y_2)(r_1 - r_2)^{-1} \bmod q$

This method of finding the private key a demonstrates that if Olga can impersonate Alice with probability P then she can find the private key with probability P^2.

Now, of course, if Olga knows the private key a then she can impersonate Alice with probability $P = 1$. This implies that Olga finding the private key is at least as strong as Olga impersonating Alice. So there is a small probability for a cheating Prover to convince an honest Verifier that he has the secret information.

Stinson [21] argued that the above observation is sufficient to say that The Schnorr Identification Scheme has the **soundness property**. In the next chapter we will show a specific way of obtaining this property.

When the scheme was developed we assumed that from the public information $v \equiv \alpha^{-a} \pmod{p}$ someone cannot compute $a = -\log_\alpha v$ in $\mathbf{Z}_p^*$ efficiently. So we require that this computation is difficult for an adversary.

Another property of the Zero Knowledge protocol that is verified is **completeness**. We know that in step 4 of the scheme, Bob accepts Alice's proof of identity (if we assume that both the Prover and the Verifier are honest).

With the properties shown above we can say that The Schnorr Identification Scheme is a **proof of knowledge**.

Chapter 4

Our Result

In Stinson's book "Cryptography: Theory and Practice", Third Edition,[21], chapter 9, it is presented the security of the Schnorr Identification Scheme. Regarding the soundness property the author claims that if an adversary Olga knows a value γ for which she can compute the challenges r_1, r_2 with their corresponding y_1, y_2 then Olga can compute Alice's private key. No method is mentioned for the calculation of the values $\gamma, r_1, r_2, y_1, y_2$. In the present chapter we would like to present a reduction between a complete break of the soundness condition to an impersonation attack.

Consider a general impersonation attack:

We start with the public parameters produced by the TA: p, q, α, t and v the public key of each user.

$v \equiv \alpha^{-a} \pmod{p}$ where a is the private key

Now we suppose that we have two algorithms **A** and **B**. The goal of the algorithms **A** and **B** is to accomplish impersonation.

The two algorithms are described below:

Algorithm A

Input: p, q, α, t and v

Output: γ

Algorithm B

Input: α, γ, v, r

Output: y

In order for the algorithm **B** to produce a y satisfying $\gamma \equiv \alpha^y v^r \pmod{p}$, $\alpha^y \equiv \gamma v^{-r} \pmod{p}$ it must produce $y = \log_\alpha \gamma v^{-r} \bmod p-1$. Algorithm **B** outputs a guess for value y, the correct ones correspond only to the value $y = \log_\alpha \gamma v^{-r} \bmod p-1$.

Now we assume that for a certain set $R_\gamma \subset \{1, 2, 3, ..., 2^t - 1\}$ any $r \in R_\gamma$ is such that the algorithm **B** produces a correct y.

Two values r_1, r_2 and related y_1, y_2 correspond to the same value γ given by the algorithm **A**.

We now study the probability of cheating the protocol. To cheat the protocol we do the steps:

We use algorithm **A** to obtain the value γ. We continue with step 1 in the protocol so we transmit γ to Bob. Bob sends the value of r in step 2. Compute γv^{-r} and compare α^y with γv^{-r}.

If $\alpha^y \equiv \gamma v^{-r} \pmod{p}$ then we transmit y to Bob in step 3; otherwise we transmit any other value. Bob will accept or reject depending on the value y.

The probability of cheating the protocol using this method is:
$P = \frac{|R_\gamma|}{2^t}$

We now study the probability to obtain the discrete logarithm $\log_\alpha v$. First we run the algorithm **A** and get γ.

By choosing r_1, $r_2 < 2^t$ at random and with a bit of luck the algorithm **B** will produce correct solutions y_1, y_2. Now as shown in the previous chapter from:

$\gamma \equiv \alpha^{y_1} v^{r_1} \pmod{p}$

$\gamma \equiv \alpha^{y_2} v^{r_2} \pmod{p}$

we can calculate: $a = (y_1 - y_2)(r_1 - r_2)^{-1} \bmod q$

Therefore, the probability of breaking the discrete logarithm $\log_\alpha v$ is $\frac{(|R_\gamma|)(|R_\gamma|\text{-}1)}{2^{2t}}$,

which is much smaller than the probability of breaking the protocol.

We can conclude that we can break the Schnorr Identification Scheme with probability P^2, where P is the probability of finding (r_1, y_1) or (r_2, y_2).

More precisely, if algorithm **B** has a probability of success $P_1 = \frac{1}{\mathrm{p(t)}} \gg \frac{1}{2^t}$ where $p(t)$ is a polynomial in t then the probability of finding the correct values r_1, r_2, y_1, y_2 is $P_2 = \frac{1}{p^2(t)}$.

Now we can conclude:

$\frac{1}{\mathrm{p(t)}} \gg \frac{1}{2^t}$, because $p(t)$ is a polynomial and moreover

$\frac{1}{p^2(t)} \gg \frac{1}{2^t}$ which is equivalent to:

$\frac{1}{\mathrm{q(t)}} \gg \frac{1}{2^t}$, where $q(t)$ is a polynomial in t.

So we can write the following relations:

Pr[**A** outputs γ]·Pr[**B** outputs valid r|**A** outputs γ]=$\frac{1}{\mathrm{p(t)}}$

We know that there exist $p_{\mathbf{A}}(t)$, $p_{\mathbf{B}}(t)$ such that:

Pr[**A** outputs γ]$\geq\frac{1}{p_{\mathbf{A}}(t)}$

Pr[**B** outputs $r|\gamma$]$\geq\frac{1}{p_{\mathbf{B}}(t)}$

Therefore,

Pr[**A** outputs γ]·Pr[**B** outputs $r|\gamma$]·Pr[**B** outputs $r'|\gamma$]$\geq\frac{1}{\mathrm{p(t)}}\frac{1}{p_{\mathbf{B}}(t)}$

With all this observations we can formulate the following statement:

" If Olga can impersonate Alice successfully with probability $P = \frac{1}{\mathrm{p(t)}} > \frac{1}{2^t}$ then the probability of finding for a γ the correct values r_1, r_2, y_1, y_2 such that Bob will accept is: $P = \frac{1}{p^2(t)} > \frac{1}{2^t}$.

Therefore, if an adversary can impersonate Alice with a non negligeable probability P in polynomial time then the adversary can also compute Alice's private key with a non negligeable probability P' in polynomial time."

Chapter 5

The Discrete Logarithm Problem

If $\mathbf{G}$ is a cyclic group with the operation denoted by " $\times$" and a generator $g \in \mathbf{G}$ then we define:

$$g^a = \underbrace{g \times g \times g \times ... \times g}_{a \text{ terms}}$$

The operation defined above is called exponentiation. The inverse operation is defined as follows:

Given $y \in \mathbf{G}$, $\mathbf{G}$ cyclic with the generator g. Now, if $ord(g) = n$ then there is a unique exponent a $0 \leq a \leq n-1$ such that $y = g^a$. a is called the discrete logarithm of y and it is written as: $a = \log_g y$.

The exponentiation inside the group $\mathbf{G}$ can be calculated by using fast exponentiation in $O(|a|)$ group operations, where $|a|$ is the number of bits of the representation of a. It is believed that the discrete logarithm is essentially harder to compute than exponentiation in many groups.

The Discrete Logarithm Problem can be solved by computing g, g^2, g^3, ..., until we obtain: $y = g^a$. In this method we use: $g^i = g \times g^{i-1}$, so the necessary time is $\Omega(a)$.

Another solution would be to calculate all the pairs (i, g^i) and sort them with

respect to the second coordinate. Given $y \in \mathbf{G}$ then we make a binary search in the sorted list of (i, g^i) and we identify a such that $y = g^a$. This requires $\Omega(a)$ space.

In the literature there are many methods to find discrete logarithms. We can mention: **Shank's Algorithm**, **The Pollard Rho Discrete Logarithm Algorithm**, **The Pohlig-Hellman Algorithm**, **The Index Calculus Method** and **The Number Field Sieve**. A good description of all these methods can be found in [21]. But none of these methods provide a solution in polynomial time.

For the Discrete Logarithm Problem if $y \in \mathbf{G}$ is given, it is easy to prove that there exists a such that:

$g^a = y$ (always exists), but for the Prover to prove that he knows a is a harder question.

As we introduced in Chapter 2 a zero knowledge proof must satisfy the properties of soundness, completeness and zero-knowledge. In other words we have a zero knowledge proof when a Prover succeed in proving a statement to a Verifier without revealing any knowledge of his secret.

One interesting and new direction is to obtain efficient zero-knowledge proofs of knowledge for the discrete logarithm by considering the discrete logarithm as a preimage of exponentiation.

In cryptography there are efficient zero-knowledge proofs of knowledge of the preimage of many one-way homomorphisms.

Endre Bangerter, Jan Camenisch, Ueli Maurer introduced efficient zero-knowledge proofs of knowledge for exponentiation and multiexponentiation homomorphisms for the first time in groups with unknown order (for example an RSA group) [1].

To present the details of their work we need to first explain the necessary notions.

If $(G,+)$, $(H,\cdot)$ are groups then the map $\varphi : G \to H$ is called a homomorphism if:

$\varphi(g_1 + g_2) = \varphi(g_1) \cdot \varphi(g_2)$

for any $g_1, g_2 \in G$

A preimage of the element $h \in H$ under the homomorphism φ is an element $g \in G$ such that: $\varphi(g) = h$.

Supposing now that we have a protocol between a Prover and a Verifier with the given function φ and the element $h \in H$ to prove that the Prover has g with $\varphi(g) = h$. If the Prover can make the Verifier accept in the last step with a probability larger than a certain threshold probability (the knowledge error), then the Prover must know the preimage $g \in G$ of h such that: $\varphi(g) = h$.

This is a proof of knowledge of the preimage of the homomorphism φ because we can see it as the proof of the existence of a **knowledge extractor** which can compute a preimage g of h given access to the Prover as a rewinding oracle.

The most commonly used protocol for almost all the homomorphisms used in cryptography is the Σ-Protocol with binary challenges.

Let's now introduce first the Σ-Protocol together with its properties.

The original idea of the Σ-Protocol was first introduced by Cramer in his Ph.D thesis [8].

We need some basic definitions and notations. We denote by:

$R[\Phi(k)] = \{((\varphi, h), g) : \varphi \text{ homomorphism}, \varphi : G \to H, g \in G, \varphi(g) = h\}$ where k is an integer security parameter.

Definition

Let Φ be a collection of homomorphisms having a finite domain G and let's consider $((\varphi, h), g)$ an element in $R[\Phi(k)]$. If (P, V) is an interactive protocol with a Prover P and a Verifier V where the common information for P and V is the pair (φ, h) and the secret information of P is g, then we say that a Σ-Protocol with the

challenge set $C = \{0, 1, 2, ..., c(k)\}$ is the following 4-step protocol between the Prover and the Verifier:

Steps:

1. The Prover chooses a random value uniformly distributed $r \in_U G$; he then computes $t = \varphi(r)$ and he sends t to the Verifier.

2. The Verifier chooses a random $c \in_U C$ and he sends c to the Prover.

3. The Prover computes: $s = r + cg$ and he sends it to the Verifier.

4. The Verifier calculates $\varphi(s)$. If $\varphi(s) = \varphi(r)\varphi(g)^c$ then he outputs 1, otherwise he outputs 0.

Graphically the protocol can be represented as:

The Prover (P)		The Verifier (V)
1. $r \in_U G$ $t = \varphi(r)$		
	$t \longrightarrow$	
		2. $c \in_U C$
	$\longleftarrow c$	
3. $s = r + cg$		
	$s \longrightarrow$	
		4. If $\varphi(s) = th^c$ then he outputs 1, otherwise he outputs 0.

Now at step 4 the Verifier is convinced that the Prover has the secret information g because:

$\varphi(s) = \varphi(r + cg) = \varphi(r)\varphi(cg) =$

$= \varphi(r)\varphi(g)^c$

$= th^c$

The Σ-Protocol is proved to be honest-verifier zero-knowledge and zero-knowledge if the set C is polynomially bounded in k.

Because of the binary challenges, the Σ-Protocol has a knowledge error of $\frac{1}{2}$ and we need to repeat the protocol sufficiently many times to obtain a small enough overall knowledge error.

To obtain properties of the Σ-Protocol we need to introduce another helpful notion: the pseudo-preimage problem.

Given the homomorphism $\varphi : G \rightarrow H$ and $h \in H$ then we call a pseudo-preimage of h under φ a pair (c, g) with $h^c = \varphi(g)$ where c is a non-zero integer and $g \in G$.

The integer c is called the exponent of the pseudo-preimage (c, g).

In general to find a preimage g of h under φ given a pseudo-preimage (c, g) of h under φ where $h \in Image(\varphi)$ is called **The Pseudo-Preimage Problem**.

For the invertible homomorphisms the pseudo-preimage problem is trivial. Another easy case is for certain one-way homomorphisms like the ones underlying the Schnorr and the Guillou-Quisquater schemes. The fact that the pseudo-preimage problem is easy for these cases allows us to construct knowledge extractors for the Σ-Protocol.

The Pseudo-Preimage Problem is hard for homomorphisms in groups where the Root Problem is hard (ex. RSA groups [18]).

Pseudo-preimages have the property that if we are given two pseudo-preimages of the same element $h \in H$ then there is a method to calculate a preimage of h by using

Shamir's trick [11].

If (c_1, g_1), (c_2, g_2) are two pseudo-preimages of $h \in H$ under φ and if $\gcd(c_1, c_2) = 1$ then $g = ag_1 + bg_2$ is a preimage of h under φ where a, b are integers such that: $ac_1 + bc_2 = 1$.

The standard **knowledge extractor** of the Σ-Protocol can be constructed as follows:

We consider P' an arbitrary Prover who has the common input (φ, h). Suppose that he has arbitrary private information and his probability of success inside the protocol is: $A > \frac{1}{c(k)+1}$

Now, if we give access to rewinding the Prover P' then we can obtain the following vectors: $(t, c, s), (t', c', s')$ both verifying the condition in step 4 of the Σ-Protocol where $t = t'$, $c' > c$.

We denote: $\Delta s = s' - s$, $\Delta c = c' - c$. Then $\Delta c > 0$.

From step 4 we can obtain:

$\varphi(s) = th^c$

$\varphi(s') = t'h^{c'} = th^{c'}$

We divide these two equations as follows:

$\frac{\varphi(s')}{\varphi(s)} = \frac{th^{c'}}{th^c}$

$\varphi(s')(\varphi(s))^{-1} = h^{c'-c}$

$\varphi(s' - s) = h^{c'-c}$

$\varphi(\Delta s) = h^{\Delta c}$

If we choose $C = \{0, 1\}$ then $\Delta c = 1$, $h = \varphi(\Delta c)$ and we obtain a preimage of h under φ. In this way we construct a knowledge extractor for the challenge set with cardinality 2.

For the general case when the cardinality is greater than 2 we need to use the collision extractability property, the existence of pseudo-preimage finders for special homomorphisms and also Shamir's trick to find a preimage [8].

Chapter 6

The $\Sigma+$ -Protocol

The $\Sigma+$ -Protocol is a new protocol which provides efficient proofs of knowledge of x given h^x, for any exponentiation homomorphism $\varphi : \mathbf{Z} \rightarrow H$, $\varphi(x) = h^x$, where H is a group with hidden order. It was shown that the efficiency of the proofs of knowledge is related to the smallest order of the homomorphism's image.

Our intention is to present how this protocol will provide efficient proofs of knowledge for the discrete logarithm in RSA groups where the modulus n is a product of two safe primes [2].

Damgård and Fujisaki showed that the Σ -Protocol can be used to prove the knowledge of the discrete logarithm in groups with hidden order assuming that the Prover does not know the order of the group. They developed a new scheme which will be named the **DF Scheme** [10],[9]. Their work was based on [13].

The **DF Scheme** is *not* a computational proof of knowledge according to the definition presented in [3].

To obtain proofs of knowledge using the **DF Scheme** we need to make weaker assumptions so we can later obtain in a restricted case what we initially wanted. Ideas shared with the **DF Scheme** are used in the $\Sigma+$ -Protocol to obtain standard proofs of knowledge.

The **DF Scheme** requires some initial conditions such as that the order of the group H is not revealed and the generalized Root Problem is hard (in other words, for any $h \in_U H$ it is hard to find an integer $e \neq 1$ and $u \in H$ such that $u^e = h$). In comparison with the **DF Scheme**, the Σ+ -Protocol does not require any initial assumption.

Organized like the model of the **DF Scheme**, the Σ+ -Protocol has the Σ -Protocol executed several times. The computational cost of the Σ+ -Protocol is roughly three times bigger than the cost of the Σ -Protocol.

To describe in detail the protocol we will need some introductory notions and assumptions.

The Σ+ -Protocol - description

Let n be of the form $n = (2p+1)(2q+1)$ with $p, q, 2p+1, 2q+1$ prime numbers and let's consider k an integer security parameter. We assume that there exists a generator $G_S(k)$ which provides pairs (n, g), $g \in \mathbf{Z}_n^*$ with the property that it is hard to calculate a $u \in \mathbf{Z}_n^*$ and an integer $e > 1$ such that $u^e = g$.

If we denote by QR_n the subgroup of $\mathbf{Z}_n^*$ of the quadratic residues modulo n we assume in the following part that $g \in QR_n$.

As we mentioned at the beginning, the Σ+ -Protocol can be applied for simple or multiple exponentiation.

For simplicity we present the case of the simple exponentiation.

We start with:

$\varphi : \mathbf{Z} \to \mathbf{Z}_n^*$

$\varphi(x) = h^x$, for $h \in \mathbf{Z}_n^*$

By: $c^+ := c^+(k)$

$\Delta x := \Delta x(k)$

$l_z := l_z(k)$

we denote integer parameters.

Definition

We consider Φ a collection of simple-exponentiation homomorphisms and let

$\varphi : \mathbf{Z} \to \mathbf{Z}_n^*$, $\varphi(x) = h^x$, $h \in \mathbf{Z}_n^*$, $((\varphi, y), x) \in R[\Phi](:= R[\Phi(k)])$, $x \in [-\Delta x, \Delta x]$.

Let $\chi : \{0,1\}^{l_d} \to \{0,1\}^{l_c}$ be a crypto hash function (see definition 1.12). This means that finding collision is hard and given $\chi(\omega)$ is difficult to find parts of ω) with $l_d(k) := l_d$, $l_c(k) := l_c$ and $l_d > EncLen^+(\mathbf{Z}_n^*)$ (which is an upper bound on the length of the binary encoding of elements of $\mathbf{Z}_n^*$).

Now if we have a Prover P with the input $((\varphi, y), x)$ and a Verifier V with the input (φ, y) then a $\Sigma+$ -Protocol with the challenge set $C := \{0, 1, ..., c^+\}$ is described as follows:

Steps:

1. The Verifier chooses (n, g) generated by $G_S(k)$, $\rho \in_U [0, 2^k \lfloor n/4 \rfloor]$ and he calculates $g_1 := g^\rho \bmod n$

He sends g_1, g, n to the Prover.

For simplicity we denote:

$v(x_1, x_2) := g_1^{x_1} g^{x_2} \bmod n$

2. The Prover randomly chooses $r \in_U [-2^{l_z} c^+ \Delta x, 2^{l_z} c^+ \Delta x]$ then he calculates $t = \varphi(r)$; he chooses $X \in_U [0, 2^{l_z} n]$. He calculates $Y = v(x, X)$.

He chooses $R \in_U [-2^{2l_z} c^+ n, 2^{2l_z} c^+ n]$ and then calculates $T = v(r, R)$ and choose randomly $R_{\mathbf{Z}_n^*} \in_U \{0,1\}^{l_d}$.

Let $K := \chi(T \| Y \| R_{\mathbf{Z}_n^*})$ be a commitment to T and Y. The Prover sends K to the

Verifier.

3. The Verifier chooses randomly $c \in_U C = \{0, 1, ..., c^+\}$ and he sends it to the Prover.

4. The Prover calculates $s := r + cx$, $S := R + cX$ and he transmits (s, S) to the Verifier.

5. The Verifier sends ρ to the Prover.

6. The Prover calculates g^ρ. If $g^\rho \equiv g_1 \pmod{n}$ is not verified then he stops the protocol.

Otherwise, he continues and he unveils T and Y to the Verifier.

7. The Verifier checks the equalities:

$K = \chi(T \| Y \| R_{\mathbf{Z}_n^*})$

$\varphi(s) = ty^c$

$v(s, S) \equiv TY^c \pmod{n}$

If all of the equations are verified then he outputs 1 (he accepts); otherwise, he outputs 0 (he rejects).

We observe that $x \in [-\Delta x, \Delta x]$ is necessary for the Σ+ -Protocol to be statistical zero knowledge which means that someone needs to know how large x can be to blind x in the messages sent by the Prover.

The parameter l_z controls the tightness of the statistical zero knowledge property of the Σ+ -Protocol.

It should be mentioned that the common input for the Prover and the Verifier is (φ, y) and the secret information of the Prover is x.

Graphically, the protocol can be represented as:

Verifier(V) (φ, y) Prover (P) $((\varphi, y), x)$

Step 1.

$G_S(k) \to (n, g)$

$\rho \in_U [0, 2^k \lfloor n/4 \rfloor]$

$g_1 := g^\rho \bmod n$

$v(x_1, x_2) := g_1^{x_1} g^{x_2} \bmod n$

$(g_1, g, n) \quad \longrightarrow$

Step 2.

$r \in_U [-2^{l_z} c^+ \Delta x, 2^{l_z} c^+ \Delta x]$

$t = \varphi_M(r),\ X \in_U [0, 2^{l_z} n]$

$Y = v(x, X)$

$R \in_U [-2^{2l_z} c^+ n, 2^{2l_z} c^+ n]$

$T = v(r, R)$

$R_{\mathbf{Z}_n^*} \in_U \{0,1\}^{l_d}$

$K := \chi(T \| Y \| R_{\mathbf{Z}_n^*})$

$\longleftarrow \qquad (K, t)$

Step 3.

$c \in_U C = \{0, 1, ..., c^+\}$

$c \qquad \longrightarrow$

Step 4.

$s := r + cx$

$S := R + cX$

$\longleftarrow \qquad (s, S)$

Step 5.

$\rho \qquad \longrightarrow$

Step 6.

If $g^{\rho} \neq g_1 \pmod{n}$ he stops,

otherwise he continues.

$\longleftarrow \qquad (T, Y, R_{\mathbf{Z}_n^*})$

Step 7.

He verifies if the following equalities are true:

$K = \chi(T \| Y \| R_{\mathbf{Z}_n^*})$

$\varphi(s) = ty^c$

$v(s, S) \equiv TY^c \pmod{n}$

If all of the equations are verified then he outputs 1 (he accepts); otherwise, he outputs 0 (he rejects).

The key to the construction of the **knowledge extractor** for the The Σ+ -Protocol is the next theorem.

Assuming the above conditions are verified. We define a generator $G_V(l, k)$ that outputs multi-exponentiations $V : \mathbf{Z}^l \to QR_n$ as follows:

(1) Choose $G_S(k) \to (n, g)$

(2) For $i = 1, ..., (l-1)$ choose $\rho_i \in_U [0, 2^k \lfloor n/4 \rfloor]$

(3) We set $g_i := g^{\rho_i}$

(4) We define $V(x_1, ..., x_l) := {g_1}^{x_1} ... {g_{l-1}}^{x_{l-1}} g^{x_l}$

(5) Output (V, n)

Theorem

Under strong RSA assumption it is hard given $G_V(l,k) \rightarrow (V,n)$ to find a $y \in \mathbf{Z}_n^*$ and a pseudo-preimage $(v, (w_1, w_2, ..., w_l)$ of y under V such that $v \neq 1$ and v does not divide w_j for some $j \in \{1, ..., l\}$. The proof of the theorem can be found in [7].

We observe that the protocol beginning with step 2 is similar to two Σ -Protocols ran in parallel for the homomorphisms φ and v.

By using the fact that φ has the argument the same as the first argument of v we can construct a **knowledge extractor** for the $\Sigma+$ -Protocol.

In Chapter 5 we showed how to construct a knowledge extractor for the Σ -Protocol. Since the $\Sigma+$ -Protocol consists of repeating the Σ -Protocol then we conclude that we can create a pseudo-preimage $(\Delta c = c' - c, \Delta s = s' - s)$ of y under φ and a pseudo-preimage $(\Delta c, (\Delta s, \Delta S))$ of Y under v.

In other words:

$\varphi(\Delta s) = h^{\Delta s} = y^{\Delta c}$

$v(\Delta s, \Delta S) = g_1^{\Delta s} g^{\Delta S} = Y^{\Delta c}$

For the homomorphism v we can apply the theorem mentioned before for $l = 2$ and we obtain that $\Delta c | \Delta s$, $\Delta c | \Delta S$.

We suppose additionally that $\gcd(\Delta c, |Image(\varphi)|) = 1$ and we obtain a preimage of y under φ.

$y^{\Delta c} = h^{\Delta s}$

$y = h^{\frac{\Delta s}{\Delta c}}$ and $x := \frac{\Delta s}{\Delta c}$ is preimage of y under φ.

We can conclude that we use the application of v to construct the **knowledge extractor** for the homomorphism φ.

As in the case of the Σ -Protocol to obtain the **knowledge extractor** we need to give to the Verifier pairs (t, s), $(T, (s, S))$ such that the relations from step 7 are satisfied. Now using the fact that the Σ+ -Protocol is based on the Σ -Protocol in which we know that for given (φ, y), and (v, Y) we can create the vectors (t, c, s), $(T, c, (s, S))$ such that they verify the required equations.

In the Σ -Protocol the initial input (φ, y) and (v, Y) should be provided. The difference in the Σ+ -Protocol is that (v, Y) is chosen inside the protocol. In order to be zero-knowledge we have to simulate the choices of Y.

Now if $X \in_U [0, 2^{l_z} n]$ then by definition $Y = v(x, X)$ will be randomly chosen in the subgroup $< g >$ since the length of the interval is bigger than $ord(g)$.

In the protocol we want to prevent the situation where a dishonest Verifier can choose a v such that Y will reveal information about the secret x. Otherwise we will lose the zero-knowledge property of the Σ+ -Protocol. The change proposed in the protocol is to use a hash function χ as follows: in step 2 the Prover does not know if v is correctly chosen and he only sends the hash K of T and Y also combined with a random string R_H, instead of the values T and Y themselves. Then in steps 5 and 6 the Verifier convinces the Prover that $g_1 \in < g >$ and the function v is correctly formed. After the Verifier V succeed in convincing the Prover P about the correctness of v he then sends the discrete logarithm ρ of g_1 to the Prover P.

The Prover P, convinced by the Verifier, can now send the information (T, Y) to the Verifier V. The information (s, S) should be calculated by the Prover P without having the discrete logarithm ρ of g_1 (to apply the theorem mentioned in the construction of the **knowledge extractor**).

In the description of the Σ+ -Protocol there are some parameters n and g chosen by the Verifier V. The Prover P does not verify these parameters and he does not have the tools to check them. If we introduce these particular verifications inside the protocol then we will reduce its efficiency. This will be studied in the next chapter.

Chapter 7

Our Result

In the description of the Σ+ -Protocol there are some initial settings of some parameters. The Prover P accepts these conditions but he does not have the tools to verify them and in fact he does not check them during the algorithm. The efficiency of the protocol depends pretty much on the fact that this verification is not done by the prover P.

This observation will give the possibility of constructing different attacks to the security of the protocol. Sébastien Kunz-Jacques, Gwenaëlle Martinet, Guillaume Poupard and Jacques Stern tried in [16] to show that some attacks can be constructed to break the Σ+ -Protocol.

In the cited paper they studied the following cases when a dishonest Verifier can obtain the secret information of the Prover:(breaking the Zero Knowledge property)

(1) the parameter n can be chosen such that the discrete logarithms in $\mathbf{Z}_n^*$ can be efficiently computed. One way of doing that is to take n of the form $n = pq$, p, q primes and $p-1$, $q-1$ are smooth (which means that they are products of small primes). Camenisch and Michels have found some zero knowledge proofs to show that a modulus is a product of two safe primes [6];

In case 1 a cheating Verifier can choose the modulus n such that it is easy to

compute:

$x\rho + X \bmod ord(g) = \log_g y$

$r\rho + R \bmod ord(g) = \log_g T$

By executing the $\Sigma+$ -Protocol the Verifier obtains the values of:

$s := r + cx$ and

$S := R + cX$

In this way we have four equations with the given information ρ, $ord(g)$, s, S, c and four unknowns x, X, r, R. The equations are not independent therefore, the system cannot be solved to find the secret information x.

A possible idea to extend this case is to take $\rho = 1$. The attacker can obtain $x+X$ mod $ord(g)$ which can be seen as x mod $ord(g)$ combined with X.

Since $ord(g) \approx \frac{\mathrm{n}}{2}$ one bit of information is revealed if x is uniformly distributed mod $ord(g)$.

By repeating this method the secret information x can be deduced bit by bit.

(2) the parameter ρ can be chosen such that the multiplicative order of g_1 is small, n is such that p, q are primes, $p-1$, $q-1$ are smooth, $(p-1)/2$, $(q-1)/2$ are relatively prime and g is an element of $\mathbf{Z}_n^*$ of maximal order $\lambda(n) = (p-1)(q-1)/2$.

In this case the idea is similar to case 1 but the most significant bit of $\log_g y$ reveals the least significant bit of x, i.e. the value x mod 2. With a high probability this can be done from a single execution of the protocol. It is good to mention that a Prover who follows the protocol cannot detect this fact.

The attack can be extended and the next bit of x can be found. Each bit requires at least one execution of the protocol, therefore to recover a l-bit secret x the total number of protocol executions would be $l \times (1 + \frac{1}{\sqrt{n}})$.

All the attacks mentioned above are not possible if the Prover verifies the fact that n is an RSA modulus (which means that $n = pq$ with p, q, $2p + 1$, $2q + 1$ are primes) and g is a quadratic residue ($g \in QR_n$).

The authors of [16] assume that the Prover P checks the quadratic residuosity of g. By changing g_0 to $g = g_0^2 \bmod n$ then the Prover P makes sure that $g \in QR_n$.

The other parameter n which should have a specific form is n. Our attention is concentrated on the verification of the correctness of n. If we modify the $\Sigma+$ -Protocol such that we add two extra steps, then all these possible assumptions are no longer valid.

Verifier(V) (φ, y)		Prover (P) $((\varphi, y), x)$
Step 1.		
$G_S(k) \to (n, g)$		
$\rho \in_U [0, 2^k \lfloor n/4 \rfloor]$		
$g_1 := g^\rho \bmod n$		
$v(x_1, x_2) := g_1^{x_1} g^{x_2} \bmod n$		
(g_1, g, n)	$\longrightarrow$	
		Step 2.
		$r \in_U [-2^{l_z} c^+ \Delta x, 2^{l_z} c^+ \Delta x]$
		$t = \varphi(r),\ X \in_U [0, 2^{l_z} n]$
		$Y = v(x, X)$
		$R \in_U [-2^{2l_z} c^+ n, 2^{2l_z} c^+ n]$
		$T = v(r, R)$
		$R_H \in_U \{0,1\}^{l_d}$
		$K := \chi(T \| Y \| R_H)$
	$\longleftarrow$ (K, t)	

Step 3.

$c \in_U C = \{0, 1, ..., c^+\}$

$c \longrightarrow$

Step 4.

$s := r + cx$

$S := R + cX$

$\longleftarrow \quad (s, S)$

Step 4'.

The Verifier sends p, q primes
with $2p + 1, 2q + 1$ prime numbers
such that $n = (2p + 1)(2q + 1)$.

$p, q \longrightarrow$

Step 4".

The Prover checks if $p, q, 2p + 1, 2q + 1$ are prime numbers and $n = (2p + 1)(2q + 1)$. Then he continues the protocol, otherwise he stops.

Step 5.

$\rho \longrightarrow$

Step 6.

If $g^{\rho} \neq g_1 \pmod{n}$ he stops,

otherwise he continues.

$\longleftarrow$ (T, Y, R_H)

Step 7.

He verifies if the following equalities are true:

$K = \chi(T \| Y \| R_H)$

$\varphi(s) = ty^c$

$v(s, S) \equiv TY^c \pmod{n}$

If all of the equations are verified then he outputs 1 (he accepts); otherwise he outputs 0 (he rejects).

When we added the extra step 4' we added at the right place inside the protocol to keep it zero knowledge. If we add the step earlier in the protocol then we loose the soundness property. If we put it later in the protocol, after step 5 then we disclose some information and we loose the zero knowledge property.

Now we study how the verification of the correctness of n will affect the efficiency of the protocol. If we choose the Rabin Miller primality test then the cost will be $O(n^3)$.

If we run the test at most ${l_z}^3$ times, we do not change the over all time complexity of the algorithm while getting very small error probability. Notice that the most expensive operation of this protocol is in step 2 in the calculation of T.

Therefore in the case when the test is done more than ${l_z}^3$ we affect the efficiency of the protocol.

Another aspect is that if the step 4" fails then the information accumulated by the Verifier is not enough to recover the secret of the Prover. An important step in

this observation is the fact that we use the cryptographic hash function χ and the Verifier, by having $K = \chi(T\|Y\|R_H)$, he cannot find the values T and Y. Therefore, in this situation, the protocol is zero knowledge.

Conclusion

In the present thesis our purpose was to try to give more attention to some parts of two existing and well known protocols in cryptography.

The two results presented were developed by trying to understand the Schnorr Identification Scheme and the Σ+-Protocol. Considering in both cases the different approaches that exist in the literature we discovered that some parts were not very well studied and some cases were missing. Therefore, the general idea was to try to provide new arguments and new directions for each protocol.

This thesis should be interpreted as a simple attempt to give a slight improvement to the existing work done in each of the two protocols. Certainly this work can be continued and it will probably improve parts of the schemes presented.

The field of cryptography gives us an enormous number of protocols and schemes, each with different presentations in the literature. There is a great variety and by using mathematical tools we can try all the time to make the existing cryptographic protocols more and more complete and secure.

Bibliography

[1] E. Bangerter, J. Camenisch, and U. Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hiddden order. *PKC, Lecture Notes in Computer Science*, 3386(1):154–171, 2005.

[2] N. Baric and B. Pfitzmann. Collision-free acumulators and fail-stop signature schemes without trees. *Advances in Cryptology, Lecture Notes in Computer Science*, 1233(1):480–494, 1997.

[3] M. Bellare and O. Goldreich. On defining proofs of knowledge. *Advances in Cryptology, Lecture Notes in Computer Science*, 740(1):390–420, 1992.

[4] M. Bellare and A. Palacio. Identification schemes: Proofs of security against impersonation under active and concurrent attacks. *Advances in Cryptology, Lecture Notes in Computer Science*, 2442(1):162–177, 2002.

[5] M. Burmester, Y. Desmedt, and T. Beth. Efficient zero-knowledge identification schemes for smart cards. *The Computer Journal*, 35(1):21–29, 1992.

[6] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is a product of two safe primes. *Advances in Cryptology, Lecture Notes in Computer Science*, 1592(1):107–122, 1999.

[7] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. *Advances in Cryptology, Lecture Notes in Computer Science*, 2729(1):126–144, 2003.

[8] R. Cramer. Modular design of secure, yet practical cryptographic protocols. *PhD Thesis, University of Amsterdam*, 1(1), 1996.

[9] I. Damgard. On sigma-protocols. *Advances in Cryptology, Lecture Notes in Computer Science*, 1(1), 2002.

[10] I. Damgard and E. Fujisaki. An integer commitment scheme based on groups with hidden order. *Advances in Cryptology, Lecture Notes in Computer Science*, 2501(1), 2002.

[11] E.G.Strauss. Problems and solutions. *Americam Mathematical Monthly*, 71(1):806–808, 1964.

[12] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. *Advances in Cryptology, Lecture Notes in Computer Science*, 263(1):186–194, 1987.

[13] E. Fujisaki and T. Okamoto. Statistical zero-knowledge protocols to prove modular polynomials relations. *Advances in Cryptology, Lecture Notes in Computer Science*, 1294(1):16–30, 1997.

[14] M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. *Advances in Cryptology, Lecture Notes in Computer Science*, 1(1):481–486, 1991.

[15] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *Proceedings of 17th Symposium on the Theory Computation, Rhode Island*, 1(1), 1985.

[16] S. Kunz-Jacques, G. Martinet, G. Poupard, and J. Stern. Cryptanalysis of an efficient proof of knowledge of discrete logarithm. *Advances in Cryptology, Lecture Notes in Computer Science*, 3958(1):27–43, 2006.

[17] J.-J. Quisquater, L. C. Guillou, and T. A. Berson. How to explain zero-knowledge protocols to your children. *Advances in Cryptology, Lecture Notes in Computer Science*, 435(1):628–631, 1990.

[18] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[19] C. Schnorr. Efficient identification and signature schemes for smart cards. *Advances in Cryptology, Lecture Notes in Computer Science*, 435(1):235–251, 1990.

[20] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing, 1997.

[21] D. Stinson. *Cryptography: Theory and Practice*. Chapman/Hall and Taylor/Francis Group, 2006.

[22] D. D. Waleffe and J.-J. Quisquater. Better login protocols for computer networks. *Lecture Notes in Computer Science*, 741(1):50–70, 1993.