SHORT UNDENIABLE SIGNATURES: DESIGN, ANALYSIS, AND APPLICATIONS

THÈSE N^{O} 3691 (2006)

PRÉSENTÉE LE 15 DÉCEMBRE 2006

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Laboratoire de sécurité et cryptographie

SECTION DE SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDERALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jean MONNERAT

mathématicien diplômé EPF
de nationalité suisse et originaire de Vermes (JU)

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. S. Vaudenay, directeur de thèse
Prof. M. Franklin, rapporteur
Prof. A. Lenstra, rapporteur
Prof. J. Stern, rapporteur

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2006

to Susan

Abstract

Digital signatures are one of the main achievements of public-key cryptography and constitute a fundamental tool to ensure data authentication. Although their universal verifiability has the advantage to facilitate their verification by the recipient, this property may have undesirable consequences when dealing with sensitive and private information. Motivated by such considerations, undeniable signatures, whose verification requires the cooperation of the signer in an interactive way, were invented.

This thesis is mainly devoted to the design and analysis of short undeniable signatures. Exploiting their online property, we can achieve signatures with a fully scalable size depending on the security requirements. To this end, we develop a general framework based on the interpolation of group elements by a group homomorphism, leading to the design of a generic undeniable signature scheme. On the one hand, this paradigm allows to consider some previous undeniable signature schemes in a unified setting. On the other hand, by selecting group homomorphisms with a small group range, we obtain very short signatures.

After providing theoretical results related to the interpolation of group homomorphisms, we develop some interactive proofs in which the prover convinces a verifier of the interpolation (resp. non-interpolation) of some given points by a group homomorphism which he keeps secret. Based on these protocols, we devise our new undeniable signature scheme and prove its security in a formal way. We theoretically analyze the special class of group characters on $\mathbb{Z}_n^*$. After studying algorithmic aspects of the homomorphism evaluation, we compare the efficiency of different homomorphisms and show that the Legendre symbol leads to the fastest signature generation. We investigate potential applications based on the specific properties of our signature scheme. Finally, in a topic closely related to undeniable signatures, we revisit the designated confirmer signature of Chaum and formally prove the security of a generalized version.

Keywords: undeniable signatures, short signatures

Résumé

Les signatures numériques sont l'un des principaux accomplissements de la cryptographie à clef publique et constituent un outil indispensable pour assurer l'authenticité des données. Bien que leur vérifiabilité universelle facilite leur vérification par un destinataire, cette propriété peut avoir des conséquences indésirables dans un contexte lié à des informations sensibles ou privées. Motivées par ces considérations, les signatures incontestables, dont la vérification requiert la coopération du signataire de manière interactive, ont vu le jour.

Cette thèse se consacre principalement au développement et à l'analyse de signatures incontestables courtes. Grâce à leur propriété interactive, nous parvenons à développer des signatures dont la taille peut être ajustée librement par rapport à la sécurité requise. A cet effet, nous proposons un cadre général, basé sur l'interpolation d'éléments d'un groupe par un homomorphisme, nous permettant de concevoir un schéma de signature incontestable générique. D'une part, ce cadre théorique nous permet de considérer des précédents schémas de signature incontestable de manière unifiée. D'autre part, en choisissant un homomorphisme de groupe adéquat, des signatures courtes sont obtenues de manière naturelle.

Après la présentation de résultats théoriques liés à l'interpolation d'homomorphismes de groupe, nous développons des preuves interactives dans lesquelles le prouveur convainc un vérifieur de l'interpolation (ou, respectivement, de la non interpolation) de points donnés, par un homomorphisme secrètement connu de lui-même. A partir de ces protocoles, nous développons un schéma de signature incontestable et prouvons sa sécurité de manière formelle. Nous menons une analyse théorique concernant la classe particulière d'homomorphismes que sont les charactères de groupe sur $\mathbb{Z}_n^*$. Après une étude des aspects algorithmiques de l'évaluation d'un homomorphisme de groupe, nous comparons l'efficacité de différents homomorphismes et montrons que le symbole de Legendre conduit à la génération de signature la plus efficace. Nous étudions des applications potentielles utilisant les propriétés spécifiques de notre schéma de signature. Finalement, dans un domaine intimement lié aux signatures incontestables, nous décrivons une généralisation du schéma de signature à confirmeur désigné de Chaum et en prouvons la sécurité de manière rigoureuse.

Mots-clés : signatures incontestables, signatures courtes

Acknowledgments

First of all, I would like to express all my gratitude to Serge Vaudenay who gave me the opportunity to accomplish this thesis in an ideal environment, although my cryptographic knowledge was very limited when we first met. His excellent scientific advices and guidance throughout the last four years brought me so much!

It was a real pleasure to prepare this work in such a friendly atmosphere. This is mostly due to all my current and former colleagues of the LASEC: Gildas Avoine, Thomas Baignères, Claude Barral, Julien Brouchier, Brice Canvel, Martine Corval, Matthieu Finiasz, Yi Lu, Philippe Oechslin, Sylvain Pasini, and Martin Vuagnoux. In particular, I profoundly thank Gildas for the three years during which we shared the same office. His friendship and our very interesting discussions related to basically everything, and in particular, the French King “Pépin le Bref”, considerably contributes to make my day-to-day work quite enjoyable. Pascal, my first office mate, deserves special thanks for being the person who made me discover the exciting world of cryptography and for taking the time to proofread this thesis.

A part of my PhD student life allowed me to meet very interesting and friendly people of the cryptographic community at different conferences and workshops. There would be too many persons to thank and the risk to forget people is quite high, so I prefer to thank all of them here for very interesting scientific and non-scientific discussions. I warmly thank Emmanuel Bresson for having shared a very nice trip after the PKC workshop in 2004 and Damien Vergnaud and Fabien Laguillaumie who invited me in Caen (France) to present a part of this work at a seminar.

I thank the president of the jury Prof. Emre Telatar as well as Prof. Matthew Franklin, Prof. Arjen Lenstra, and Prof. Jacques Stern for having accepted to take a non-negligible part of their time for reviewing this work. This is an honour for me to have such brilliant researchers in my thesis committee.

I also express my thanks to the Swiss National Science Foundation which partially supported this thesis under the grants 200021-101453/1 and 200020-109133.

Finally, I would like to heartily thank my parents who always gave the best of themselves for my education as well as my brothers and my sister for their constant support and motivation. Last but not least, I am infinitely grateful to my wife Susan for her sincere love and her unconditional support during the whole time of this work. I dedicate this thesis to her.

Contents

Chapter 1

Introduction

Digital signatures undoubtedly constitute one of the most fundamental tools of public-key cryptography. They formally consist in binding some information with a player associated with a given public key. Provided that the signer's public key was given to the recipient through an authenticated channel, this one can securely verify signatures sent by the signer. In this way, transmitting a message with its respective signature over a communication channel guarantees the message's authenticity to the recipient. As explained above, classical digital signatures are verifiable by anybody who knows the signer's public key.

Although the universal verifiability of digital signatures is very convenient in most applications, it may lead to undesirable consequences in the case of private or commercially sensitive information. In particular, compromising authenticated documents can be publicly released in a very easy and efficient way. This motivates the introduction of *undeniable signatures* for which verification of a signature must be done in an interactive way with the signer. Besides privacy reasons, undeniable signatures can be used for additional purposes such as digital cash or the licensing of sensitive software.

Since the invention of undeniable signatures, several schemes with various properties and different underlying mathematical problems have been developed. Although a considerable amount of work has been dedicated to the design of undeniable signature schemes, results towards the design of very short (e.g., 40 bits) undeniable signatures have not been published. This thesis is mainly concerned with the design of a scheme offering very short signatures, or more precisely, signatures which are fully scalable with respect to a security level. To this end, we introduce a generic scheme based on group homomorphisms which are hard to evaluate if one does not possess the secret key. This generic setting allows us to make very short signatures by selecting the group homomorphism with a small range group, while the domain size can be adjusted so that an adversary with any given computational power (without the secret key) cannot evaluate the group homomorphism.

Most of the work presented in this thesis is dedicated to the development and the analysis of this scheme called MOVA. In addition to this, we devote a part to the so-called *designated confirmer signatures* which are closely related to undeniable signatures. Namely, designated confirmer signatures are like undeniable signatures except that the verification is shifted to a third party, called the confirmer.

Thesis Outline

Chapter 2 aims at providing a brief overview of the cryptographic background required to understand the sequel of this work. After recalling some definitions related to indistinguishability of probability ensembles, we discuss the computational model and summarize the methodology of reductionist proofs usually used in public-key cryptography to show the security of a cryptographic scheme. We present a commonly used idealization of a security model arisen by the introduction of idealized functions called random oracles. Then, we review the definition and security notions of several cryptographic primitives which will play an important role in this thesis such as commitment schemes. The last section of this chapter is dedicated to a survey of interactive proofs and zero-knowledge notion which are necessary in the verification protocols of an undeniable signature scheme.

Chapter 3 is devoted to general aspects of undeniable and designated confirmer signatures. First, we present the context and motivations of the introduction of these cryptographic primitives and discuss the main differences with classical digital signatures. The subsequent section provides a formal definition of these two kinds of signatures. Next to this, we develop the security properties related to them and the verification protocols. Certain security properties are considered with different requirements in terms of the ability of the adversary and goal this one needs to achieve. To conclude this chapter, we give an overview of the historical development of undeniable and designated confirmer signatures.

Then, **Chapter 4** is dedicated to the design and security analysis of our new undeniable signature scheme called MOVA. We first put forward the concept of interpolation of group homomorphisms, consider some related computational problems, and provide some technical results necessary for the rest of this chapter. Subsequently, we develop some interactive proofs which allow a prover to convince a verifier whether a given set of points interpolates in a group homomorphism, i.e., whether there exists a homomorphism whose graph contains the whole set of points. In addition, we propose an interactive proof which will be used by the signer to show that the public key is valid. Based on these results, we present the components of the MOVA scheme and prove security results according to the definitions given in the previous chapter. We also derive some security bounds which allow to quantify the security of the different properties with respect to the hardness of un-

derlying problems and the adversary's ability. At the end, we discuss the possibility of verifying several signatures at the same time and the possibility to convert some signatures into universally verifiable ones.
Most of the results obtained in this chapter have been published in [107] at the ASIACRYPT '04 conference and in [111] at the VIETCRYPT '06 conference. The MOVA scheme is actually a generalization on a previous scheme [109] based on group characters that we presented at the PKC '04 workshop. In addition, the MOVA scheme led to a patent application [112]. All these results have been achieved in a joint work with Serge Vaudenay.

Chapter 5 exposes an in-depth analysis of a special variant of MOVA for which homomorphisms are instantiated with group characters. We first present some general theoretical results related to group characters defined on $\mathbb{Z}_n^*$ by focusing to those of order 2, 3, and 4. The former case corresponds to Jacobi symbols while the other ones are naturally studied in the cubic and biquadratic (or quartic) residuosity theory. We show how characters of this kind can be selected for MOVA and discuss a variant with two levels of secret which only arises with cubic or quartic characters. To study the security of MOVA in the context of characters, we exhibit reductions of computational problems related to this MOVA variant with some more common problems such as the factorization and some square root problems. Finally, the feasibility of using in practice a MOVA instantiation without primes is treated as well as some specifications concerning the validation of the public key in the context of characters.
This work related to characters led to the first version of the MOVA scheme and to a joint publication [109] with Serge Vaudenay.

Chapter 6 deals with algorithmic aspects of different possible instantiations of MOVA. Besides those based on characters, we consider the RSA trapdoor permutation and one homomorphism which consists in sending elements in a hidden subgroup followed by a discrete logarithm computation. We particularly focus on algorithmic specifications related to the computation of the latter and the quartic residue symbol. We finish this chapter by exposing implementations results with practical parameters. A comparison of the different possible instantiations show that the Jacobi symbol leads to the fastest signature generation followed by the variant based on the discrete logarithm in a hidden subgroup.
This chapter has mainly the same contents as an article [106] published at the MYCRYPT '05 conference in collaboration with Yvonne Anne Oswald and Serge Vaudenay. The work dedicated to the quartic residue symbol and the implementations of the different instantiations were achieved in a student semester project of Yvonne Anne Oswald [122] under our supervision.

Chapter 7 investigates potential applications which may arise from the specific properties of the MOVA undeniable signature scheme. In particular, we focus on taking advantage of the short size of MOVA signatures. In the first part of this chap-

ter, we give a rather high-level description of some potential applications. Then, we dedicate the sequel of this chapter to the study of an SMS-based lottery application. In this protocol, the main role of the MOVA signature is to give a strong evidence that the player's bid was registered by the lottery.
The contents of this chapter arise from a joint work with Serge Vaudenay and results related to the lottery application were also done in collaboration with Florin Oswald during his master's thesis [121].

The only part devoted to designated confirmer signatures can be found in **Chapter 8**. We revisit the original scheme of Chaum with more general building blocks such as an existentially forgeable undeniable signature. For the first time, a formal security proof is provided. This shows that resistance against adaptive forgery attacks can be achieved, while the invisibility can be achieved against a non-adaptive adversary. After proposing a practical instantiation of this scheme, we discuss some feasibility results of such a primitive. To this end, we prove that the undeniable signature scheme considered as building block is equivalent to public-key encryption showing that our results are consistent with a previous article of Okamoto on the minimal assumptions for the existence of a designated confirmer signature.
These results were achieved in collaboration with Serge Vaudenay and led to a paper [110] presented at the ISC '05 conference.

Finally, we conclude this work in **Chapter 9** and suggest some future research directions related to our results, which are worth investigating from our point of view.

Additional Contributions

Besides the results presented in this work, we would like to mention some additional contributions achieved during these PhD studies on some topics which are not related to undeniable signatures. Originally, we wanted to explore some algebraic tools to seek applications in cryptography. In particular, our interests focused on the Hensel lifting and the cubic and quartic residuosity. The former was first used in cryptography by Nigel Smart [140] to show that a special class of curves called anomalous have an easy discrete logarithm problem. While Hensel lifting led us to study anomalous elliptic curves and algebraic extensions of block ciphers, we decided to concentrate on the latter topic after having developed the MOVA variant based on characters.

In symmetric-key cryptography, we constructed some weak algebraic extensions of the block ciphers BES [113] and AES [2, 45]. This was done in a joint work with Serge Vaudenay and has been presented in [108] at the ICICS '04 conference. In collaboration with Gildas Avoine to supervise the semester project of Thomas Peyrin, we have provided some new theoretical and algorithmic results related to non-adjacent form representation of integers. This paper [6] was published at the

INDOCRYPT '04 conference. In a joint work [97] with Franck Leprévost, Sébastien Varrette, and Serge Vaudenay, we have described a method to efficiently generate anomalous elliptic curves. In addition to scientific contributions, our teaching activities motivated us to write an exercise book [8] on cryptography for undergraduate students. This has been achieved with the following colleagues of the LASEC: Thomas Baignères, Pascal Junod, Yi Lu, and Serge Vaudenay.

Chapter 2

Preliminaries in Cryptography

2.1 Notation and Background

The sets denoting different kind of numbers are written in “blackboard” font such as the set of positive integers $\mathbb{N}$, the relative integers $\mathbb{Z}$, the real numbers $\mathbb{R}$, and the complex numbers $\mathbb{C}$. We also denote by $\mathbb{R}^+$ the set of non negative real numbers. The set $\{0,1\}^*$ stands for the set of the bitstrings of arbitrary length. For any bitstring $x \in \{0,1\}^*$, we use a “norm symbol” $|x|$ to denote its length, i.e., the number of bits it is composed of. We use $\mathrm{poly}(\cdot)$ to denote any given polynomial. Picking an element x uniformly at random in a given set S will be written $x \in_U S$ or $x \leftarrow_U S$.

Definition 2.1.1. *A function $f : \mathbb{N} \to \mathbb{R}^+$ is called negligible if for any polynomial $p : \mathbb{N} \to \mathbb{R}^+$ there exists an integer n_0 such that*

$$n \geq n_0 \Rightarrow f(n) < \frac{1}{p(n)}.$$

Notation. We will denote by $\mathrm{negl}(\cdot)$ any negligible function.

Indistinguishability. We recall the definition of the statistical distance between two random variables.

Definition 2.1.2. *The statistical distance Δ between two discrete random variables X_1 and X_2 with range $\mathcal{X}$ is*

$$\Delta(X_1, X_2) := \frac{1}{2} \sum_{x \in \mathcal{X}} \left| \Pr[X_1 = x] - \Pr[X_2 = x] \right|.$$

We now define the notion of statistical and computational indistinguishability between two sequences of random variables with the same range $\mathcal{X}$.

Definition 2.1.3. *A probability ensemble is a sequence of random variables* $(X_i)_{i\in I}$ *indexed by a set* I*.*

From now on, $I \subseteq \{0,1\}^*$ is a subset of strings of arbitrary length.

Definition 2.1.4 (Statistical Indistinguishability)**.** *Let* $(X_i)_{i\in I}$ *and* $(Y_i)_{i\in I}$ *be two probability ensembles of discrete random variables such that* X_i *and* Y_i *have the same range for any* $i \in I$*. The ensembles* $(X_i)_{i\in I}$ *and* $(Y_i)_{i\in I}$ *are statistically indistinguishable if*

$$\Delta(X_i, Y_i) = \mathrm{negl}(|i|).$$

Furthermore, if $\Delta(X_i, Y_i) = 0$ *for any* $i \in I$*, we say that these ensembles are perfectly indistinguishable.*

Definition 2.1.5 (Computational Indistinguishability)**.** *Let* $(X_i)_{i\in I}$ *and* $(Y_i)_{i\in I}$ *be two ensembles of discrete random variables such that* X_i *and* Y_i *have the same range for any* $i \in I$*. The ensembles* $(X_i)_{i\in I}$ *and* $(Y_i)_{i\in I}$ *are computationally indistinguishable if for any probabilistic polynomial-time (with respect to* $|i|$*) algorithm* $\mathcal{D}$

$$|\Pr[\mathcal{D}(i, X_i) = 1] - \Pr[\mathcal{D}(i, Y_i) = 1]| = \mathrm{negl}(|i|).$$

These definitions can be easily adapted to the case $I = \mathbb{N}$ and for which we normally require that the negligible function negl takes $n \in \mathbb{N}$ as input instead of its length. Note that this variant is trivially included in the above general definitions if we represent any integer $n \in \mathbb{N}$ with a unary representation, i.e., with the string $1^n := 111\ldots1$, where the symbol 1 appears n times.

For more information about issues related to indistinguishability, we refer to Section 3.2 of the textbook of Goldreich [69].

Computational Model

The algorithms encountered in this work are formally modeled by Turing machines which were already invented in 1936 by Turing [142] as a formal model of a computer, though computers did not exist at that time. A Turing machine is composed of a tape with an infinite number of cells each containing a symbol, a head which can write and read symbols on the tape and move left and right, an internal state (with two special states "initial" and "final"), and a transition function which tells the next manipulation the machine needs to execute depending on the symbol read by the head and the internal state. A Turing machine can have several work tapes (initially with blank symbols) in addition to its input tape. Probabilistic algorithms are modeled with probabilistic Turing machines which possess an additional tape called *random tape* containing uniformly random symbols (or bits).

In this thesis, we usually consider polynomial time algorithms, where the term "polynomial" refers to the size of the input. If the algorithm is probabilistic, the complexity of the machine should be meant as the expected complexity over the random tape distribution. We also note that the complexity is formally measured by the number of steps executed by the Turing machine until it reaches the final state.

In certain situations, it is required to consider Turing machines which have an additional access to so-called *oracles* which are able to perform a special functionality such as solving a computational (hard) problem. Such a machine is called an *oracle machine* and the access to the oracle is done via an additional special tape on which the machine writes the oracle input. When the machine is in a special state called "invocation", the oracle input is replaced by its output. For instance, a factorization oracle would return the prime factors of a given integer. It is important to note that an oracle call has a constant complexity whatever the oracle is able to achieve.

Once oracle machines are defined, one can now define the notion of problem reductions. One says that a problem $\mathcal{P}_1$ reduces to the problem $\mathcal{P}_2$ if there exists an oracle machine which solves the problem $\mathcal{P}_1$ using access to an oracle solving the problem $\mathcal{P}_2$. We can also say that $\mathcal{P}_2$ is at least as hard as $\mathcal{P}_1$ and we denote this fact by $\mathcal{P}_2 \geq \mathcal{P}_1$.

In public-key cryptography, we often use *reductionist security proofs* which consists in solving a supposedly hard problem from an adversary breaking a given cryptographic scheme. One then deduces that since the underlying problem is hard, the cryptographic scheme should be secure. Here, "breaking" is specified with respect to a security notion related to the scheme. Depending on the context, the adversary may also given an access to some oracles. The difficulty of a security proof consists in extracting information used for solving the underlying hard problem by simulating the adversary environment.

Random Oracles

To facilitate the security analysis of cryptographic schemes, idealized objects called *random oracles* [13,33,58] were introduced. Basically, a random oracle implements a uniformly distributed random function from $\{0,1\}^k$ to $\{0,1\}^\ell$ for some given positive integers k, ℓ, i.e., a function which is picked uniformly at random among all possible functions with a k-bit input and an ℓ-bit output. In the random oracle model, the participants have access to such idealized functions through so-called random oracles. Such an oracle can be implemented as follows. We maintain a list with some input-output pairs which is empty at the beginning. For any new query to the oracle, one checks whether it is already stored as input, if this is the case, one answers the corresponding output stored in the list. Otherwise, one picks a uniformly random output of ℓ bits and adds this new pair in the list.

When reductionist proofs are performed in the random oracle, the adversary is given access to random oracles. These ones can be simulated in a way which helps to achieve a reduction to a hard computational problem. This model makes security proof easier mainly because one can feed a challenge related to a hard problem through the simulation of random oracles.

Yet, it has been shown that a cryptographic scheme secure in the random oracle model may lead to insecure instantiations in the standard model, i.e., in the model without random oracles. As illustration, Bellare et al. [12] proposed a scheme which is secure in the random oracle model such that any instantiation would lead to an insecure scheme. We should emphasize that such schemes are some artificial examples and, as far as we know, no natural scheme (i.e., schemes which were not designed to explicitly show the insecurity of the random oracle model) provably secure in the random oracle has been broken. To summarize, though random oracle model is a strong idealization, it is commonly believed that a security proof in this model shows that the scheme was not badly designed. An interesting discussion about this issue can be found in Section 6 of Koblitz and Menezes [84].

2.2 Some Cryptographic Primitives

Throughout this section, $k \in \mathbb{N}$ denotes a security parameter.

2.2.1 Public-Key Cryptosystem

The realization of the first public-key cryptosystem RSA [131] is one of the main important achievements of modern cryptography. Namely, it allows to communicate in a confidential way with the help of an authenticated channel used to transmit the public-key. This ensures that a message encrypted with respect to this public-key will only be decrypted by the owner of the corresponding secret key. The concept of public-key cryptography was first[1] developed in the ground-breaking article of Diffie and Hellman [53] without proposing a concrete public-key encryption scheme.

Below, we denote the message space by $\mathcal{M}$ and the ciphertext (encrypted message) space by $\mathcal{C}$. A public-key encryption scheme is composed of three polynomial time algorithms.

Setup The key generator is a probabilistic algorithm which on input of the security parameter 1^k outputs a key pair. We have $(pk, sk) \leftarrow \mathsf{Setup}(1^k)$.

[1]Note that the role of Merkle who submitted already a paper [101] related to this topic in 1975 (see Chapter 9 in the textbook of Vaudenay [143]) should also be taken in consideration.

Enc The encryption algorithm is a probabilistic algorithm which takes a message $m \in \mathcal{M}$ and the public key pk as input and outputs a ciphertext c. We have $c \leftarrow \mathsf{Enc}(m, pk)$.

Dec The decryption algorithm is a deterministic algorithm which takes a ciphertext $c \in \mathcal{C}$ and the secret key sk to output a message $m \in \mathcal{M}$. We have $m \leftarrow \mathsf{Dec}(c, sk)$. In some cases, the decryption algorithm checks whether the ciphertext is of a given form and returns the symbol $\perp$ if c is not valid.

Correctness. *We also require that any encrypted message can be retrieved with the decryption algorithm. Namely, for any message $m \in \mathcal{M}$ and any key pair (pk, sk) generated by* Setup*, we always have*

$$c \leftarrow \mathsf{Enc}(m, pk);\ m \leftarrow \mathsf{Dec}(c, sk).$$

We now present a security notion called *indistinguishability under a chosen plaintext attack* (IND-CPA) which is based on the work of Goldwasser and Micali [72,73]. This notion formalizes the fact that two ciphertexts of two different messages should be indistinguishable for any adversary which does not know sk. Here, the adversary is only given pk which allows him to encrypt any message m. Therefore, the weakest adversary one can think of, is able to perform a chosen-plaintext attack.

IND-CPA Security Notion. *Let us first consider an adversary $\mathcal{A}$ modeled by a probabilistic polynomial algorithm and the two following games corresponding to $b \in \{0, 1\}$.*

$\mathbf{Game}^{\mathsf{ind\text{-}cpa\text{-}}b}$. *First, $\mathcal{A}$ is fed by a public key pk generated by $(pk, sk) \leftarrow \mathsf{Setup}(1^k)$. After a given time, the adversary $\mathcal{A}$ submits two messages m_0, m_1. The challenger answers $c \leftarrow \mathsf{Enc}(m_b, pk)$. Then, $\mathcal{A}$ outputs a bit b'.*

We define the advantage of the adversary as follows

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ind\text{-}cpa}} := \left|\Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\mathsf{ind\text{-}cpa\text{-}1}}\right] - \Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\mathsf{ind\text{-}cpa\text{-}0}}\right]\right|,$$

where the probabilities are over the random tapes of the involved algorithms. We say that a public-key encryption scheme satisfies indistinguishability under a chosen-plaintext attack *(*IND-CPA*) if there exists no probabilistic polynomial time adversary $\mathcal{A}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ind\text{-}cpa}}$ is non-negligible.*

2.2.2 Digital Signatures

The main goal of digital signatures is to reproduce the properties of a handwritten signature in the digital world, i.e., to bind some message or data with an entity. More generally, digital signatures serve to authenticate messages transmitted over a communication channel, thus ensuring to the recipient that the received information is genuine and was sent by the correct person. Since their invention, digital signatures become more and more crucial and are now used in countless cryptographic protocols. Historically, digital signatures were immediately developed with the invention of public-key cryptography. Namely, in the seminal paper of Rivest, Shamir, and Adleman [131], digital signatures are already proposed as an application of the RSA trapdoor one-way permutation.

The message space is denoted by $\mathcal{M}$ and the signature space by Σ. A digital signature scheme consists of the three following polynomial-time algorithms.

Setup This probabilistic algorithm generates a key pair which is associated with the signer. We have $(pk, sk) \leftarrow \mathsf{Setup}(1^k)$.

Sign This probabilistic algorithm generates a signature for a given message $m \in \mathcal{M}$ with respect to the above generated key pair. We have $\sigma \leftarrow \mathsf{Sign}(m, sk)$.

Verify The verification algorithm is usually deterministic and takes a message-signature pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ and the public key pk as input and outputs a bit $0, 1 \leftarrow \mathsf{Verify}(m, \sigma, pk)$ telling whether the pair (m, σ) is valid with respect to the key pair (pk, sk). The output bit 1 means that the signature is valid.

Correctness. *The signature scheme must furthermore have a verification algorithm consistent with the signing algorithm. Namely,* Verify *must return* 1 *to any signature generated by* Sign*. So, for any* $m \in \mathcal{M}$ *and any key pair* (pk, sk) *generated by* $\mathsf{Setup}(1^k)$*, we always have*

$$\sigma \leftarrow \mathsf{Sign}(m, sk);\ 1 \leftarrow \mathsf{Verify}(m, \sigma, pk).$$

In practice, the signer needs to send his public key pk to the verifier through an authenticated channel so that the verifier is ensured that pk really corresponds to the right signer. Once this step is performed, the signer can authenticate messages using digital signatures even through an insecure communication channel.

We present the most classical security notion related to digital signatures which is the *existential unforgeability under an adaptive chosen-message attack*. It was originally introduced in the famous article of Goldwasser, Micali, and Rivest [76].

Existential Unforgeability. *Let us first introduce $\mathcal{O}$ a signing oracle. More precisely, $\mathcal{O}$ is a kind of "magical" machine which on any message m sent to it answers a valid signature, i.e., it implements the algorithm* Sign. *We denote by* **L** *the list of all messages queried to $\mathcal{O}$. A signature scheme is secure against an existential forgery under an adaptive chosen-message attack, if for any probabilistic polynomial time forger (algorithm) $\mathcal{F}$, we have*

$$\Pr\left[1 \leftarrow \mathsf{Verify}(m,\sigma,pk) \wedge m \notin \mathbf{L} \;\middle|\; \begin{array}{l}(pk,sk) \leftarrow \mathsf{Setup}(1^k);\\ (m,\sigma) \leftarrow \mathcal{F}^{\mathcal{O}}(pk)\end{array}\right] = \mathrm{negl}(k),$$

where the probability is taken over the random tapes of the involved algorithms. Each invocation to the oracle $\mathcal{O}$ is counted in the complexity of $\mathcal{F}$ so that the number of queries made to $\mathcal{O}$ must also be polynomially bounded in k.

2.2.3 Hash Functions

One of the most useful tool in cryptography are hash functions. They are used in most of the protocols and digital signatures. A hash function consists in reducing messages of arbitrary length to some bitstrings of a given length, typically of 160 bits. One of their fundamental goal is to ensure the integrity of a message, i.e., to guarantee that a transmitted message over an insecure communication channel was not modified. To achieve this, it suffices to hash the message and compare hashed values of the original message and the received message. To ensure this kind of property, it should be very difficult to find a different message with the same hashed value. Below, we present in more details the desired properties a hash function should satisfy.

Let k be an integer and $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a function taking arbitrary long bitstrings as input. We say that H is a *cryptographic hash function* if the following properties hold.

Preimage Resistance. For a given bitstring $h \in \{0,1\}^k$, it is computationally infeasible to find a message $m \in \{0,1\}^*$ such that $H(m) = h$.

Second Preimage Resistance. For a given message $m \in \{0,1\}^*$, it is computationally infeasible to find a message $m' \in \{0,1\}^*$ such that $m \neq m'$ and $H(m) = H(m')$.

Collision Resistance. It is computationally infeasible to find two messages $m \neq m'$ such that $H(m) = H(m')$.

Hash functions are used in digital signature schemes in order to reduce the message to a given size so that subsequent algebraic transformations can be performed.

Namely, these algebraic computations are not defined on an arbitrary size. In addition to this, hash functions are also crucial to "break" the algebraic structure that may otherwise remain in digital signatures. For instance, any homomorphic trapdoor one-way permutation used to sign without a hash function would immediately succumb to a chosen-message attack. In such a case, using a hash function is fundamental in order to break the homomorphic property. To make some security analysis easier, hash functions will often be modeled by random oracles.

2.2.4 Pseudorandom Generators

Generating a long sequence of uniformly random bits in an efficient way is often not realizable if one does not have a dedicated device at disposal. This motivates the use of pseudorandom generators whose goal is to generate a sequence of bits from a short random bitstring often called the seed. More precisely, for two positive integers ℓ and k with ℓ typically much larger than k, we say that a function

$$\mathrm{Gen} : \{0,1\}^k \to \{0,1\}^\ell$$

is a *pseudorandom generator* if for a random variable X defined on $\{0,1\}^k$ with a uniform distribution the random variable $\mathrm{Gen}(X)$ is computationally indistinguishable from a uniformly distributed random variable on $\{0,1\}^\ell$.

It can be shown (see Chapter 3 of Goldreich [69]) that the sequence generated by a pseudorandom generator is *unpredictable* which roughly means that given a partial sequence there exists no efficient algorithm guessing the next sequence bit with a non-negligible advantage, i.e., with a success probability non-negligibly larger than 1/2. This shows that a pseudorandom generator necessarily breaks the structure of the input such as hash functions do.

As concrete scheme, we mention the famous Blum-Blum-Shub pseudorandom generator [17] whose security relies on the quadratic residuosity assumption on Blum prime integers. Yet, for efficiency reasons it is much more convenient to use the very recent scheme QUAD proposed by Berbain et al. [14] at the EUROCRYPT '06 conference.

In the rest of this work, we will make use of pseudorandom generators which will be modeled by random oracles in the security proofs.

2.2.5 Commitment Schemes

Commitment schemes play an important role in cryptography and their use is of particular importance within cryptographic protocols. These primitives can be seen as the "digital" analog of a safe or a sealed envelope. During the so-called commitment phase, a player (the sender) wants to commit on a value (or bitstring) to a receiver such that this one cannot deduce information about the committed value (hiding property). A second phase consists in opening the commitment by disclosing some extra information allowing the receiver to learn the value which was committed. It is also required that between the two phases, the sender is not able to change his mind so that it should be impossible for him to open the commitment on a different value from the committed one. This is called the binding property of the commitment scheme. To summarize, the hiding property prevents from a cheating receiver while the binding property prevents from a malicious sender.

In addition to classical commitment schemes satisfying the above properties, we will consider some variants in which a pair of keys is generated. In this settings, knowledge of the secret key gives additional ability such as opening a given commitment on any chosen value (*trapdoor commitment*) or retrieving from the commitment the initial value committed by the sender (*extractable commitment*).

Some parts of the material developed below are inspired from Section 3.4 of the Master's thesis of Pasini [124].

Commitment

We present here a formal definition of a commitment scheme. To begin with, we denote by $\mathcal{M}$ the space of the messages, $\mathcal{C}$ the space of all commitments, and $\mathcal{R}$ the space of the "decommitment" values which allow to open the commitments. A commitment scheme is composed of the three following polynomial time algorithms.

Setup This probabilistic algorithm takes k (in unary) as input and outputs a description of some parameters. We have $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, where param contains the necessary information for specifying the algorithms Commit and Open with respect to k.

Commit For any given $m \in \mathcal{M}$ as input, this probabilistic algorithm outputs a commitment and a decommitment value. We have $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(m)$.

Open This algorithm (typically deterministic) checks whether a commitment com corresponds to a message m and a decommitment value dec by outputting a bit. We have 0 or $1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec})$, where the output 1 means that the commitment was opened correctly.

For the sake of simplicity, we will also use the notation Commit to denote the whole commitment scheme when the context is clear. We require the following security properties.

Completeness. *For any $m \in \mathcal{M}$,*

$$\Pr\left[1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec}) \,\middle|\, \begin{array}{l}\mathsf{param} \leftarrow \mathsf{Setup}(1^k);\\ (\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(m)\end{array}\right] = 1 - \mathrm{negl}(k),$$

where the probability is taken over the random tapes of the involved algorithms.

Most of commitment schemes considered in practice have perfect completeness, i.e., the above probability is equal to 1.

Binding Property. *For any algorithm $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{l}1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec}) \wedge\\ 1 \leftarrow \mathsf{Open}(m', \mathsf{com}, \mathsf{dec}') \wedge\\ m \neq m'\end{array} \,\middle|\, \begin{array}{l}\mathsf{param} \leftarrow \mathsf{Setup}(1^k);\\ (m, m', \mathsf{com}, \mathsf{dec}, \mathsf{dec}') \leftarrow \mathcal{A}(\mathsf{param})\end{array}\right] = \mathrm{negl}(k),$$

where the probability is taken over all random tapes of the different algorithms.

If we restrict to probabilistic polynomial time algorithms $\mathcal{A}$, we say that the commitment scheme Commit is *computationally binding*. If the above property holds even against computationally unbounded adversary $\mathcal{A}$, we say that the commitment scheme Commit is *statistically binding*. As strongest binding notion, we say that Commit is *perfectly binding*, if the above probability is equal to zero for any (computationally unbounded) adversary $\mathcal{A}$. This means that any commitment $\mathsf{com} \in \mathcal{C}$ can be opened in at most one way, i.e., there exist at most one $m \in \mathcal{M}$ and one value $\mathsf{dec} \in \mathcal{R}$ such that $1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec})$.

We denote the above success probability of an adversary $\mathcal{A}$ breaking the binding property by $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{com\text{-}bnd}}$.

Hiding Property. *For any $m \in \mathcal{M}$, the distribution of the value* com *generated by* $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(m)$ *is indistinguishable from the uniform distribution over $\mathcal{C}$.*

Depending on the kind of indistinguishability (i.e., perfect, statistical, computational) of both distributions, the scheme Commit is said to be *perfectly*, *statistically*, or *computationally hiding*. More formally, we consider the probability ensemble $(\mathrm{COM}_k)_{k \in \mathbb{N}}$ indexed by the security parameter k where COM_k is the random variable corresponding to a commitment of the message m generated by the algorithm Commit with parameters $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$. Indistinguishability is then compared between this ensemble with the uniform one (only composed of uniformly random variables) for any message m. We recall that the different flavours of indistinguishability are defined in Definition 2.1.4 and Definition 2.1.5.

Trapdoor Commitment

This primitive is a commitment scheme associated with a pair of keys such that the knowledge of the secret key (trapdoor) allows to fully break the binding property by allowing to open any commitment $c \in \mathcal{C}$ on any chosen message $m \in \mathcal{M}$. Trapdoor commitments were first introduced by Brassard, Chaum, and Crépeau in [26] and they are sometimes called chameleon commitments in the literature [28, 85].

A trapdoor commitment scheme is composed of the four following polynomial time algorithms.

Setup This probabilistic algorithm takes k (in unary) as input and outputs a key pair $(pk, sk) \leftarrow \mathsf{Setup}(1^k)$.

Note that here the parameters are implicitly included in the key pair.

Commit For any given $m \in \mathcal{M}$ and any public key pk as input, this probabilistic algorithm outputs a commitment and a decommitment value. We have $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(m, pk)$.

Open This algorithm (typically deterministic) checks whether a commitment com corresponds to a message m and a decommitment value dec by outputting a bit. We have 0 or $1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec}, pk)$, where the output 1 means that the commitment was opened correctly.

Collide For any messages $m, m' \in \mathcal{M}$ with $m \neq m'$, values $\mathsf{dec} \in \mathcal{R}$ and $\mathsf{com} \in \mathcal{C}$ such that

$$1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec}, pk),$$

and the secret key as input, the algorithm Collide outputs

$$\mathsf{dec}' \leftarrow \mathsf{Collide}(m, m', \mathsf{com}, \mathsf{dec}, sk)$$

satisfying $1 \leftarrow \mathsf{Open}(m', \mathsf{com}, \mathsf{dec}', pk)$.

The hiding and binding property can be similarly defined as for classical commitment schemes. In the binding game, the adversary $\mathcal{A}$ is fed only with the public key and the indistinguishability for the hiding property must hold for any key pair.

Note that the existence of a trapdoor allowing to find collisions rules out the existence of trapdoor commitments which are statistically (and thus perfectly) binding.

As an example of a perfectly-hiding and computationally-binding trapdoor commitment scheme, we refer to Bresson et al. [28]. Its binding property relies on the difficulty of factoring an RSA modulus composed of safe primes. Additional concrete realizations of trapdoor commitments are given in [24, 26].

For additional information on trapdoor commitments and their applications, we refer to the PhD thesis of Fischlin [60].

Extractable Commitment

An extractable commitment scheme is a commitment scheme with associated pair of keys such that the secret key allows to retrieve the committed message.

All algorithms are like for trapdoor commitments except that we replace $\mathsf{Collide}$ by an extraction algorithm $\mathsf{Extract}$.

$\mathsf{Extract}$ For any message m and any $\mathsf{com} \in \mathcal{C}$ generated by $\mathsf{Commit}(m, pk)$, we have

$$m \leftarrow \mathsf{Extract}(\mathsf{com}, sk)$$

such that $1 \leftarrow \mathsf{Open}(m, \mathsf{com}, \mathsf{dec}, pk)$ for one $\mathsf{dec} \in \mathcal{R}$.

The extractability requirement implies that an extractable commitment scheme must be perfectly binding, since a commitment com uniquely defines a committed message m. Moreover, this property rules out the existence of statistically-hiding extractable commitments.

More information about extractable commitments and constructions can be found in the papers [50, 52].

2.3 Interactive Proofs

This section deals with the notion of interactive proofs which is fundamental for the design of cryptographic protocols. It is intended to recall some basic material and to restrict to the necessary background for the sequel of this work. Most of the subsequent results are taken from the book of Goldreich [69] and an article of Barak et al. [11].

The concept of interactive proofs was strongly motivated by the need of secure cryptographic protocols such as identification protocols. Interactive proofs arise when a player (called the *prover*) of the protocol needs to prove interactively the validity of a given statement to another player (called the *verifier*). As an example, we can think of a situation where a prover generates an RSA modulus n and wants to prove to the verifier that n is indeed an RSA modulus (usually without disclosing the prime factors of n). Interactive proofs are often the core component of an identification protocol, where the prover convinces the verifier, he possesses a secret with some specific properties (e.g., square root of an element modulo a composite number) which identifies him.

Formally, the prover and the verifier are modeled by some interactive Turing machines. Roughly, an interactive Turing machine consists in a classical Turing machine with additional tapes dedicated to the communication typically with another interactive machine. More precisely, an interactive Turing machine is a probabilistic

Turing machine with a read-only input tape, a read-only communication tape, a write-only communication tape, and a write-only output tape. When two interactive Turing machines are linked together in order to perform an interactive proof, the read-only communication tape of each machine coincide with the write-only communication tape of the other one. So, a message sent from a machine A to a machine B is modeled by a message written on the write-only communication tape of the machine A which is directly written on the read-only tape of the machine B. In addition to the communication tapes, two linked interactive Turing machines have the same read-only input tape, which corresponds to the common input of the interactive proof.

We denote the prover by $\mathbf{P}$ and the verifier by $\mathbf{V}$. The interaction between both $\mathbf{P}$ and $\mathbf{V}$ with a common input x, a private input w to $\mathbf{P}$, and private input y to $\mathbf{V}$ is written $\langle \mathbf{P}(w), \mathbf{V}(y) \rangle (x)$. As the name suggests, a private input to a given player is only known by himself and is often called the *auxiliary* input. Usually, the prover takes a secret as auxiliary input which allows him to prove the validity of the statement related to the interactive proof. In some cases, the verifier is also given an auxiliary input which may consist of a private key.

As said earlier, in an interactive proof, the prover aims at proving the validity of a given statement and normally uses some secret information to do so. In order to formalize the statement notion, we consider a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ with the corresponding formal language

$$L_R := \{x \mid \exists w \text{ such that } (x,w) \in R\}.$$

In this settings, an interactive proof consists for a prover in showing to a verifier that a common input x is in the language L_R by using the corresponding witness w as private input. At the end of the protocol, if the prover was able to answer correctly to the challenges sent by the verifier, the latter accepts the proof. Otherwise, if the prover fails to correctly prove the given statement, the verifier rejects the proof. This acceptance or rejection of the verifier is expressed by outputting a bit at the end of the interaction. This bit represents the output of the interaction as well. We write this as

$$\langle \mathbf{P}(w), \mathbf{V}(y) \rangle (x) \to 0 \text{ or } 1,$$

where 1 stands for “`accept`” and 0 for “`reject`”.

We are now in a position to define an interactive proof system of membership for the language L_R. From now on, we simply write L instead of L_R if a reference to R is not required.

Definition 2.3.1. *A pair of interactive machines $\langle \mathbf{P}, \mathbf{V} \rangle$ is called an* interactive proof system *for a language L if there are two functions $c, s : \mathbb{N} \to [0,1]$ such that $1 - c(n) > s(n) + 1/\mathrm{poly}(n)$ and the following properties are satisfied.*

Efficiency. *The interactive machine* $\mathbf{V}$ *is polynomial-time with respect to* $|x|$.

Completeness. *For any* $x \in L$ *and any* w *such that* $(x, w) \in R$, *and any auxiliary input* y, *we have*

$$\Pr[\langle \mathbf{P}(w), \mathbf{V}(y) \rangle(x) \to 1] \geq 1 - c(|x|),$$

where the probability is over the random tapes of $\mathbf{P}$ *and* $\mathbf{V}$.

Soundness. *For any* $x \notin L$, *any auxiliary input* y, *and any interactive machine* $\mathbf{P}^*$ *(deviating from the protocol specifications and possibly computationally unbounded)*

$$\Pr[\langle \mathbf{P}^*, \mathbf{V}(y) \rangle(x) \to 1] \leq s(|x|),$$

where the probability is over the random tapes of $\mathbf{P}^*$ *and* $\mathbf{V}$.

The functions $c(\cdot)$ and $s(\cdot)$ are respectively called the *completeness error* and *soundness error.*

If the soundness property only holds against a polynomial-time (in $|x|$) malicious prover, we say that $\langle \mathbf{P}(\cdot), \mathbf{V}(\cdot) \rangle(\cdot)$ is a *computationally sound* interactive proof or an interactive *argument* [26].

Zero-Knowledge

As shown earlier, soundness property in an interactive proof prevents a malicious prover to pass the protocol with an invalid statement, i.e., $x \notin L$. On the other hand, Definition 2.3.1 does not guarantee anything about a malicious behaviour of the verifier. In some practical applications such as identification protocols one usually does not want that the prover discloses more information than the validity of the statement. Namely, we clearly want to avoid that some malicious verifier retrieves private information of the prover. In an identification protocol, the major risk comes from a malicious verifier who attempts to retrieve the secret allowing the prover to identify himself.

This motivated to introduce the notion of zero-knowledge which captures the fact that no information should leak to the verifier except the validity of the statement. Informally, the paradigm behind this concept is based on the simulation of the interactive proof by the verifier himself showing that computation made by the prover does not provide additional information to the verifier. The simulation should work provided that the prover is honest and the statement is valid. This is in fact not a restriction since we usually do not care about the security consequences when both the prover and the verifier are malicious. Before introducing a formal definition, we would like to mention that the concept of zero-knowledge was put forward and formalized by Goldwasser, Micali, and Rackoff [75].

All the messages exchanged between a prover $\mathbf{P}$ and a verifier $\mathbf{V}$ in an interactive proof form the *transcript* of the protocol. We denote by $\mathtt{View}(\langle\mathbf{P},\mathbf{V}\rangle)$ the transcript (random variable) of the interactive proof $\langle\mathbf{P},\mathbf{V}\rangle$.

Definition 2.3.2 (Zero-Knowledge). *We say that an interactive proof system $\langle\mathbf{P},\mathbf{V}\rangle$ for the language L_R (with corresponding relation R) is (perfect, statistical, or computational)* zero-knowledge *if for any probabilistic polynomial-time interactive machine $\mathbf{V}^*$ and any polynomial p, there exists a probabilistic polynomial-time algorithm $\mathcal{B}$ called simulator, such that the ensembles*

$$\{\mathtt{View}(\langle\mathbf{P}(w),\mathbf{V}^*(y)\rangle(x))\}_{(x,w)\in R,\ y\in\{0,1\}^{p(|x|)}} \quad \textit{and} \quad \{\mathcal{B}(x,y)\}_{(x,w)\in R,\ y\in\{0,1\}^{p(|x|)}}$$

are (perfectly, statistically, or computationally) indistinguishable, where $\mathcal{B}(x,y)$ denotes the random variable of $\mathcal{B}$'s output with the inputs x,y.

This definition of zero-knowledge with an auxiliary input to the verifier is an extension of the classical definition in which the verifier is only given the common input x. The auxiliary input can be some information known by the verifier before the beginning of the interaction such as a private key.

We present now a stronger notion of zero-knowledge which is called black-box zero-knowledge which is satisfied if there exists a universal simulator for any verifier having an oracle access to the verifier as a black-box, i.e., the simulator can interact with the verifier $\mathbf{V}$ by feeding him with a random tape and interacting by exchanging messages without getting access to the verifier's internal code. More precisely, the simulator has access to a "next-message function" which on input takes a random tape and the first messages satisfying the specifications of the protocol (partial transcript) and outputs the next message $\mathbf{V}$ would send if he was interacting with the honest prover $\mathbf{P}$. Note that this allows to reset the verifier to a previous state (with the same random tape).

Definition 2.3.3 (Black-Box Zero-Knowledge). *We say that an interactive proof system $\langle\mathbf{P},\mathbf{V}\rangle$ for the language L_R (with corresponding relation R) is (perfect, statistical, or computational)* black-box zero-knowledge *if there exists a probabilistic polynomial-time oracle machine $\mathcal{B}$ such that for every probabilistic polynomial-time interactive machine $\mathbf{V}^*$ and any polynomial p, the ensembles*

$$\{\mathtt{View}(\langle\mathbf{P}(w),\mathbf{V}^*(y)\rangle(x))\}_{(x,w)\in R,\ y\in\{0,1\}^{p(|x|)}} \quad \textit{and} \quad \{\mathcal{B}^{\mathbf{V}^*}(x,y)\}_{(x,w)\in R,\ y\in\{0,1\}^{p(|x|)}}$$

are (perfectly, statistically, or computationally) indistinguishable.

Remark 2.3.4. Our definition of black-box zero-knowledge is less "strict" than definition given in [69] where the simulator $\mathcal{B}$ is not directly given the auxiliary input. Note however that our definition remains stronger than classical zero-knowledge as defined in Definition 2.3.2.

As pointed out in [11], all known zero-knowledge interactive proofs prior to an article of Barak [9] in 2001 are black-box zero-knowledge. In this way, all zero-knowledge proofs in this work will be black-box as well. To output an indistinguishable transcript, the simulator $\mathcal{B}$ usually needs to rewind the verifier. When the simulator does not need to do so, we call this notion (black-box) *straight-line zero-knowledge*. More information about this special kind of zero-knowledge can be found in an article of Dwork and Sahai [55].

Important questions about zero-knowledge proofs concerns the minimal number of moves which are required to achieve different flavours of zero-knowledge. Besides the theoretical interest raised by this question, number of moves are important in environments with some constrained communication. Considerable results about this issue have been developed from the introduction of zero-knowledge. In the standard model, Barak et al. [10] proved that zero-knowledge proofs of an NP-complete language (possibly non-black-box) requires at least 3 moves and Goldreich and Krawczyk [71] showed that 4 moves are necessary in the black-box case for non-trivial languages.

To overcome this limitation, the notion of zero-knowledge was extended in the random oracle model (for more details, see Bellare and Rogaway [13]) in which the queries to the random oracles are controlled by the simulator, i.e., it can simulate the output of the oracles provided that the output distribution is correct. Recently, Pass [125] proposed the notion of *deniable zero-knowledge* in the random oracle. The difference with the classical zero-knowledge in the random oracle is that the simulator is no longer allowed to simulate the output of the random oracles, but is only able to observe the queries made by the verifier to the random oracles as well as the corresponding answers. This actually means that the simulator's transcript really corresponds to the view of the verifier. In this model, Pass [125] showed that 2 moves are necessary to achieve zero-knowledge for NP and proposed a general 2-move protocol for NP which is not very convenient for practical purposes. In our results, proofs of zero-knowledge in the random oracle will be deniable in general.

Non-interactive zero-knowledge achieves the minimal number of moves since this term is used to designate zero-knowledge interactive proof with only one move, i.e., the prover releases one message to the verifier. This can be achieved in the random oracle model and due to results of Pass [125] cannot be deniable. Usually, non-interactive zero-knowledge is achieved in a different (less idealized) model called *common reference string model* where the players have access to a random string (often a public key). In this model, the simulator is allowed to simulate the common reference string with respect to its predefined distribution. Non-interactive zero-knowledge was originally defined in a less idealized model called the *common random string model*, where the string is uniformly distributed. More information about the original definition of non-interactive zero-knowledge is given in the article of Blum, Feldman, and Micali [18].

Non-Transferability

We introduce the concept of non-transferability which will be particularly useful for undeniable signatures and designated confirmer signatures. We define this notion in the context of interactive proofs where the verifier is associated with a pair of keys $(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}})$ depending on a security parameter $k \in \mathbb{N}$ and generated by a probabilistic polynomial time algorithm $\mathsf{Setup}^{\mathbf{V}}$, i.e.,

$$(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k).$$

We assume furthermore that the verifier's public key was transmitted to the prover in an authenticated way.

Non-transferability aims at preventing from a malicious verifier $\mathbf{V}^*$ (associated with a known public key) to convince another party of the validity of the statement proven by the prover. Note that zero-knowledge does not automatically prevent from this, since $\mathbf{V}^*$ learns anyway that the statement is valid, i.e., that a given string x lies in a formal language. A generic way to transfer a proof may be achieved if $\mathbf{V}^*$ forwards messages generated by a hidden verifier $\widetilde{\mathbf{V}}$ to the prover $\mathbf{P}$. In this way, $\mathbf{V}^*$ only stays in the middle between $\mathbf{P}$ and $\widetilde{\mathbf{V}}$ and the prover does not notice that he is actually convincing $\widetilde{\mathbf{V}}$ of the validity of the proven statement. In the context of language proof of membership, we define this notion as follows.

Definition 2.3.5. *Let consider an interactive proof system $\langle \mathbf{P}, \mathbf{V} \rangle$ for the language L_R, where a random generated pair of keys*

$$(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k)$$

is associated with $\mathbf{V}$ such that $\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}$ is in the common input and $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ is the verifier's auxiliary input. We say that $\langle \mathbf{P}, \mathbf{V} \rangle$ is (perfect, statistical, or computational) non-transferable *if there exists a probabilistic polynomial-time interactive machine $\mathcal{B}$ with input $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ such that for any probabilistic polynomial-time interactive machine $\widetilde{\mathbf{V}}$ (with or without $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$), both transcript ensembles*

$$\{\mathtt{View}(\langle \mathbf{P}(w), \widetilde{\mathbf{V}} \rangle (x, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}}))\}_{(x,w),k} \quad \text{and} \quad \{\mathtt{View}(\langle \mathcal{B}(\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}), \widetilde{\mathbf{V}} \rangle (x, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}}))\}_{(x,w),k}$$

are (perfectly, statistically, or computationally) indistinguishable, and where $\mathcal{B}$ is given a bit telling whether $x \in L_R$.

Note that contrary to zero-knowledge, $\mathcal{B}$ does not model the view of the verifier. So, in the random oracle model, $\mathcal{B}$ does not see any queries made by $\widetilde{\mathbf{V}}$. The role of $\mathcal{B}$ is actually to show that the verifier $\mathbf{V}^*$ with $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ can actually replace the prover. This implies that $\widetilde{\mathbf{V}}$ cannot determine whether $\mathbf{V}^*$ is interacting with the prover or not.

Chapter 3

Overview on Undeniable and Designated Confirmer Signatures

3.1 Signatures with Online Verification

The main goal of digital signatures is to bind some data with an entity, i.e., formally with a given public key. What makes public-key cryptography techniques much more attractive over the conventional one, is mainly due to the easier way of managing keys through a broadcast of the public key. In the context of signatures, this paradigm makes signatures easily verifiable by basically anybody. Though this property called *universal verifiability* is usually very convenient, it might lead to undesirable consequences, in particular when sensitive or private information is signed. For instance, a compromising certified (with a signature) document signed by a personality might be easily spread in newspapers or Internet. This motivates to cope with privacy problems for certain applications of digital signatures.

To keep advantage of the key management offered by public-key techniques but in order to avoid universal verifiability, Chaum and van Antwerpen [40] came up with a new kind of signatures called *undeniable signatures* for which the recipient needs to participate in an interactive protocol with the signer in order to verify the validity of a signature. In this way, the signer has a control on his signatures spread by verifying signatures only to concerned or authorized person. To prevent from a malicious signer who would claim that he did not sign a contract, a mechanism allows the signer to prove interactively that a signature is invalid. In case of dispute, legal means can be used to compel a non cooperating signer to prove the invalidity of a signature. This ensures that a signer cannot repudiate his signatures, since this one has the possibility to prove invalidity of an invalid signature.

In addition to a setup algorithm for generating keys and a signing algorithm, an undeniable signature scheme is composed of two verification protocols called respectively the *confirmation* and *denial* protocols. The former is used to prove that a valid signature is valid, while the later is used to prove that an invalid signature is indeed invalid. Note that a failure of the signer in the confirmation protocol does not imply that the signature is invalid, but may simply come from a lack of cooperation of the signer.

Undeniable signatures clearly offer a better privacy for the signer, but the recipient of the signature looses in terms of guarantee from the signer. For instance, if the signer becomes unavailable, the recipient cannot make use of the signature. In order to give a greater guarantee to the recipient that the he will be able to verify a signature at any time, Chaum [38] proposed in 1994 the notion of *designated confirmer signatures*. Such a signature works like an undeniable signature with the main difference that the verification is shifted to a third party called the *confirmer*. This one is designated at the time the signature is generated and is assumed to cooperate when signature verifications are requested. However, the recipient does not trust the confirmer completely in the sense that this one really has to prove the validity or invalidity of a signature, i.e., the recipient will not accept a simple acknowledgment of the validity or invalidity of a requested signature.

Online Versus Offline Security

As explained above, the fundamental difference between classical digital signatures and undeniable signatures (or also designated confirmer signatures) lies in the verification which is done offline in the former case and online in the later case. We briefly explain why these two different kinds of verification imply different levels of security. We consider generic attacks against a classical digital signature scheme and an undeniable one. A generic attack to forge a signature of a given message would simply consist in checking all possible signatures with the verification algorithm until this one outputs 1. It is important to note that the adversary can perform much more verifications for the classical signature than for the undeniable signature. Namely, in the latter the adversary needs to have an online access to the signer in order to perform the verification. This requires much more time and the signer can easily limit the number of verifications a given verifier is allowed to perform. This discussion shows that a classical signature cannot be smaller than a certain threshold, since an adversary cannot be limited in the context of classical signatures. Usually, we require an offline security of about 2^{80}, which means that a classical signature cannot be smaller than 80 bits. In contrast, an undeniable signature may potentially have a size of 20 or 30 bits if the number of verifications is strongly limited.

3.2 Definitions

3.2.1 Undeniable Signatures

We consider two players who are the signer ($\mathbf{S}$) and the verifier ($\mathbf{V}$). Let $k \in \mathbb{N}$ be a security parameter, $\mathcal{M}$ be the message space and Σ be the signature space. An undeniable signature scheme is composed of the four following algorithms.

Setup The setup is composed of probabilistic polynomial time algorithms $\mathsf{Setup}^{\mathbf{S}}$ and $\mathsf{Setup}^{\mathbf{V}}$ producing the signer's key pair $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$ and the verifier's key pair $(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k)$. We set $\mathcal{K}^{\mathbf{S}} := (\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$ and $\mathcal{K}^{\mathbf{V}} := (\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}})$.
Furthermore, we assume the existence of a public-key infrastructure allowing to authenticate the public keys of the different players.

Sign Let $m \in \mathcal{M}$ be a message to sign. On the input of the signer's secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$, the (probabilistic) polynomial time algorithm Sign generates a signature $\sigma \leftarrow \mathsf{Sign}(m, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$ of m (which lies in Σ).

Confirm Let $(m, \sigma) \in \mathcal{M} \times \Sigma$ be a supposedly valid message-signature pair. $\mathsf{Confirm}$ is an interactive protocol between $\mathbf{S}$ and $\mathbf{V}$ i.e., a pair of interactive probabilistic polynomial time algorithms $\mathsf{Confirm}_{\mathbf{S}}$ and $\mathsf{Confirm}_{\mathbf{V}}$ such that m, σ, $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$, $\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}$ is input of both, $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$ is the auxiliary input of $\mathsf{Confirm}_{\mathbf{S}}$, $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ is the auxiliary input of $\mathsf{Confirm}_{\mathbf{V}}$. At the end of the protocol, $\mathsf{Confirm}_{\mathbf{V}}$ outputs a boolean value which tells whether σ is accepted as a valid signature of m.

Deny Let $(m, \sigma') \in \mathcal{M} \times \Sigma$ be an alleged invalid message-signature pair. Deny is an interactive protocol between $\mathbf{S}$ and $\mathbf{V}$, i.e., a pair of interactive probabilistic polynomial time algorithms $\mathsf{Deny}_{\mathbf{S}}$ and $\mathsf{Deny}_{\mathbf{V}}$ such that m, σ', $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$, $\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}$, is input of both, $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$ is the auxiliary input of $\mathsf{Deny}_{\mathbf{S}}$, $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ is the auxiliary input of $\mathsf{Deny}_{\mathbf{V}}$. At the end of the protocol, $\mathsf{Deny}_{\mathbf{V}}$ outputs a boolean value which tells whether σ' is accepted as an invalid signature.

In addition to this, we provide a definition related to the *validity* of a given message-signature pair with respect to a key pair $\mathcal{K}^{\mathbf{S}} = (\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$.

Definition 3.2.1. *Let $(m, \sigma) \in \mathcal{M} \times \Sigma$ be a message-signature pair and $\mathcal{K}^{\mathbf{S}}$ a given key pair which was generated by* $\mathsf{Setup}^{\mathbf{S}}$. *We say that the pair (m, σ) is* valid *with respect to $\mathcal{K}^{\mathbf{S}}$ if there exists a random tape such that* $\mathsf{Sign}(m, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$ *outputs σ. Otherwise, we say that (m, σ) is* invalid.

We point out that Setup algorithm is not often defined in the literature to produce a pair of keys for the verifier. This will however be crucial when considering non-transferability of the confirmation and denial protocols. In most of the articles related to this topic, the authors consider that zero-knowledge verification protocols are sufficient for achieving non-transferability or only mention standard techniques allowing to make them non-transferable.

We also note that the definition of an undeniable signature scheme may optionally contain some convertibility algorithms and related verification algorithm. An algorithm of perfect convertibility consists in releasing some extra information by the signer which makes all signatures universally verifiable, i.e., all signatures related to the corresponding public key (signer) become ordinary digital signatures. Also considered, is the concept of selective convertibility which consists in transforming a given signature in an ordinary one. In this variant, the signer has a complete control on the signature choice he aims to convert. Although we will mention this notion in this thesis, we will not thoroughly deal with selective convertibility.

3.2.2 Designated Confirmer Signatures

A designated confirmer signature requires the introduction of an additional player called the confirmer whose role is to interact with a verifier in the verification phase of the signatures. Thus, we consider three entities which are the signer ($\mathbf{S}$), the confirmer ($\mathbf{C}$) and the verifier ($\mathbf{V}$). They all possess a pair of keys $\mathcal{K}^{\mathbf{U}} := (\mathcal{K}_{\mathrm{p}}^{\mathbf{U}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{U}})$ for $\mathbf{U} \in \{\mathbf{S}, \mathbf{C}, \mathbf{V}\}$ which are generated by some setup algorithms. Again, we let $k \in \mathbb{N}$ be a security parameter and denote the message space by $\mathcal{M}$ and the signature space by Σ. In order to keep a notation as simple as possible, the different algorithms of a designated confirmer signature are named as for the definition of an undeniable signature. A designated confirmer signature is composed of the four following algorithms.

Setup The setup is composed of three probabilistic polynomial time algorithms $\mathsf{Setup}^{\mathbf{U}}$ for $\mathbf{U} \in \{\mathbf{S}, \mathbf{C}, \mathbf{V}\}$ producing keys $\mathcal{K}^{\mathbf{U}} \leftarrow \mathsf{Setup}^{\mathbf{U}}(1^k)$.
Furthermore, we assume the existence of a public-key infrastructure allowing to authenticate the public keys of the different players.

Sign Let $m \in \mathcal{M}$ be a message. On the input of the signer's secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$ and confirmer's public key $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}$, the (probabilistic) polynomial time algorithm Sign generates a signature $\sigma \leftarrow \mathsf{Sign}(m, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{C}})$ of m (which lies in Σ).

Confirm Let $(m, \sigma) \in \mathcal{M} \times \Sigma$ be a supposedly valid message-signature pair. $\mathsf{Confirm}$ is an interactive protocol between $\mathbf{C}$ and $\mathbf{V}$ i.e., a pair of interactive probabilistic polynomial time algorithms $\mathsf{Confirm}_{\mathbf{C}}$ and $\mathsf{Confirm}_{\mathbf{V}}$ such that m, σ, $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}$, $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$, $\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}$ are input of both, $\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$ is the auxiliary input of $\mathsf{Confirm}_{\mathbf{C}}$ and $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ is the auxiliary input of $\mathsf{Confirm}_{\mathbf{V}}$. At the end of the protocol, $\mathsf{Confirm}_{\mathbf{V}}$ outputs a boolean value which tells whether σ is accepted as a valid signature of m.

Deny Let $(m, \sigma') \in \mathcal{M} \times \Sigma$ be an alleged invalid message-signature pair. Deny is an interactive protocol between $\mathbf{C}$ and $\mathbf{V}$ i.e., a pair of interactive probabilistic polynomial time algorithms $\mathsf{Deny}_{\mathbf{C}}$ and $\mathsf{Deny}_{\mathbf{V}}$ such that m, σ', $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}$, $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$, $\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}$ are input of both, $\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$ is the auxiliary input of $\mathsf{Deny}_{\mathbf{C}}$ and $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ is the auxiliary input of $\mathsf{Deny}_{\mathbf{V}}$. At the end of the protocol, $\mathsf{Deny}_{\mathbf{V}}$ outputs a boolean value which tells whether σ' is accepted as an invalid signature.

The notion of *validity* or *invalidity* of a given pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ can be defined in a very similar way as for an undeniable signature. The only difference is that this notion is determined with respect to both key pairs $\mathcal{K}^{\mathbf{S}}$ and $\mathcal{K}^{\mathbf{C}}$. So, we say that a pair (m, σ) is valid with respect to $\mathcal{K}^{\mathbf{S}}$ and $\mathcal{K}^{\mathbf{C}}$ if there exists a random tape such that $\mathsf{Sign}(m, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{C}})$ outputs σ. Otherwise, we say that this pair is invalid.

Note also that convertibility notions of signatures can be considered in an identical manner as for the undeniable signatures. Besides, this property is commonly encountered in some papers dedicated to designated confirmer signatures. As this will not play a role at all in this work, we omit again details concerning convertibility issues.

3.3 Security Model

This section is devoted to the different security notions which are required for an undeniable and a designated confirmer signature to be secure. Due to the strong similarity of both primitives, the security notions are almost identical except we have to deal with an additional player (the confirmer) for designated confirmer signatures. Following a natural order, we first develop the security model of undeniable signatures. Unavoidably, our definitions are not universal since several different security models, though closely related, can be found in the literature. In general, some notions are considered with different flavours in terms of the adversary ability and goals to achieve in order to break the security property. In this thesis, we will sometimes make use of different flavours for the same security property as well.

Two fundamental security notions about signature schemes with online verifications are the *unforgeability* and *invisibility* of the signatures. The former ensures that no other party than the signer (possessing $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$) could have generated a valid message-signature pair with respect to $\mathcal{K}^{\mathbf{S}}$ (and optionally $\mathcal{K}^{\mathbf{C}}$ for designated confirmer signatures). The aim of the latter is to guarantee the privacy of the signer by making impossible to other players to determine whether a given message-signature is valid or not. This opposes to the universal verifiability of ordinary digital signatures. Since initial motivations of undeniable signatures were to get rid of this property, this naturally gives rise to the invisibility property.

We consider four basic security notions related to the confirmation and denial protocols which are the *completeness*, the *soundness*, *zero-knowledge*, and the *non-*

transferability. The last one ensures that a malicious verifier is not able to convince any third party of the validity of the statement (e.g., a given message signature is valid) proven in the protocol. The non-transferability notion may be important in some applications where the validity of the proof itself is valuable (like for licensing software).

Throughout this section, k will denote a security parameter related to an undeniable signature or designated confirmer signature. Most of the security definitions will contain some expressions related to this parameter. In particular, terms such as "negligible", "non-negligible" or "polynomial" will refer to k.

3.3.1 Undeniable Signatures

Existential Unforgeability

We consider the standard security notion of existential forgery under an adaptive chosen-message attack as defined in the seminal paper of Goldwasser, Micali, and Rivest [76] for classical digital signatures. In this setting, the adversary has to produce a valid message-signature pair corresponding to a challenged public key using the help of signing oracles. In the context of undeniable signatures, we additionally give to the adversary an access to a verification oracle playing the role of the prover in the confirmation and denial protocols.

We use the following definition which is similar to the one proposed in an article of Kurosawa and Heng [86] considering an active adversary.

Definition 3.3.1. *An undeniable signature scheme is* secure against an existential forgery under an adaptive chosen-message attack *if there exists no probabilistic polynomial time algorithm $\mathcal{F}$ which wins the following game with a non-negligible probability.*

Game$^{\textsf{ef-cma}}$. *$\mathcal{F}$ receives a signer's public key $\mathcal{K}_\text{p}^\mathbf{S}$ from $(\mathcal{K}_\text{p}^\mathbf{S}, \mathcal{K}_\text{s}^\mathbf{S}) \leftarrow \textsf{Setup}^\mathbf{S}(1^k)$ and a verifier's key pair $(\mathcal{K}_\text{p}^\mathbf{V}, \mathcal{K}_\text{s}^\mathbf{V}) \leftarrow \textsf{Setup}^\mathbf{V}(1^k)$. Then, $\mathcal{F}$ can make some queries of its choice to the following oracles:*

- *a signing oracle which answers to any queried message $m \in \mathcal{M}$ a signature $\sigma \leftarrow \textsf{Sign}(m, \mathcal{K}_\text{s}^\mathbf{S})$*
- *a confirmation protocol oracle which to any queried pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ interacts with $\mathcal{F}$ playing the role of the prover in a confirmation protocol, i.e., the oracle implements the algorithm* $\textsf{Confirm}_\mathbf{S}$
- *a denial protocol oracle which to any queried pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ interacts with $\mathcal{F}$ playing the role of the prover in a denial protocol, i.e., the oracle implements the algorithm* $\textsf{Deny}_\mathbf{S}$

All queries can be sent in any order and adaptively, but must be polynomially bounded in k. Then, $\mathcal{F}$ wins the game if it outputs a valid pair $(m^*, \sigma^*) \in \mathcal{M} \times \Sigma$ such that m^* was not queried to the signing oracle.
The success probability of $\mathcal{F}$ in this game is denoted by $\mathsf{Succ}_{\mathcal{F}}^{\textsf{ef-cma}}$.

Invisibility

We consider an active adversary who has access to some oracles and who will have to distinguish a valid message-signature pair from a randomly picked one. We use a similar definition as Kurosawa-Heng [86].

Definition 3.3.2. *Consider first a probabilistic polynomial time algorithm $\mathcal{D}$ called* invisibility distinguisher *and the two following games with respect to a bit b.*

$\mathbf{Game}^{\textsf{inv-cma-}b}$. *$\mathcal{D}$ receives $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$ from $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$ and a verifier's key pair $(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k)$, it can query some chosen messages to a signing oracle and some chosen message-signature pairs $(m, \sigma) \in \mathcal{M} \times \Sigma$ to some oracles running the confirmation and denial protocols. After a given time, $\mathcal{D}$ chooses one message $m^* \in \mathcal{M}$ which was not queried to the signing oracle and submits it to the challenger. If $b = 0$, he sets $\sigma^* = \mathsf{Sign}(m^*, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$. Otherwise, σ^* is picked uniformly at random in the signature space Σ. $\mathcal{D}$ receives σ^*. After that, the distinguisher can query the signing, confirmation, and denial oracles again provided that m^* is not a query of the signing oracle and (m^*, σ^*) is not a query of the confirmation or denial protocols. Finally, $\mathcal{D}$ outputs a guess bit b'.*

We define the advantage of the distinguisher as follows

$$\mathsf{Adv}_{\mathcal{D}}^{\textsf{inv-cma}} := \left|\Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\textsf{inv-cma-1}}\right] - \Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\textsf{inv-cma-0}}\right]\right|,$$

where probabilities are over the random tapes of the involved algorithms. An undeniable signature scheme is said to be invisible under a chosen-message attack *if there exists no probabilistic polynomial time algorithm $\mathcal{D}$ such that the advantage $\mathsf{Adv}_{\mathcal{D}}^{\textsf{inv-cma}}$ is non-negligible.*

Note that this definition is similar to that of Galbraith and Mao [62] except that the distinguisher is not allowed to query m^* to the signing oracle. The invisibility notion of Galbraith-Mao cannot be satisfied when the signature is deterministic. Another way to define invisibility consists in considering two messages m_0, m_1, a signature $\sigma = \mathsf{Sign}(m_b, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$ and to determine the bit b, i.e., which message was actually signed. A definition following this manner is given in Camenisch and Michels [30]. Galbraith and Mao showed that both definitions are equivalent provided that the signatures are uniformly distributed in the signature space for a uniformly distributed message.

We introduce now a weaker invisibility notion following the spirit of Camenisch-Michels definition.

Definition 3.3.3. *Consider a probabilistic polynomial time algorithm $\mathcal{D}$ called* invisibility distinguisher *and the two following games with respect to a bit b.*

$\mathbf{Game}^{\textsf{inv-lkma-}b}$. *At the beginning, the game works as in $\mathbf{Game}^{\textsf{inv-cma-}b}$ until the challenger submits a message. After a given time (lunchtime), $\mathcal{D}$ does not have access to the oracles anymore. He receives two messages $m_0, m_1 \in_U \mathcal{M}$ uniformly distributed and a signature $\sigma = \textsf{Sign}(m_b, \mathcal{K}_{\rm s}^{\mathbf{S}})$. Finally, $\mathcal{D}$ outputs a bit b'.*

We define the advantage of the distinguisher as follows

$$\textsf{Adv}_{\mathcal{D}}^{\textsf{inv-lkma}} := \left|\Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\textsf{inv-lkma-1}}\right] - \Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\textsf{inv-lkma-0}}\right]\right|,$$

where probabilities are over the random tapes of the involved algorithms. An undeniable signature scheme is said to be invisible under a lunchtime known-message attack *if there exists no probabilistic polynomial time algorithm $\mathcal{D}$ such that the advantage $\textsf{Adv}_{\mathcal{D}}^{\textsf{inv-lkma}}$ is non-negligible.*

Notions Related to the Confirmation and Denial Protocols

We mainly adapt security properties of interactive proofs presented in Section 2.3 for the confirmation and denial protocols. The main difference here, is that the terms "indistinguishability", "non-negligible" refer to the security parameter k and that some key pairs for both the prover and verifier are generated. The security properties of the confirmation (resp. denial) protocol are given in the following definition.

An execution of the confirmation (resp. denial) protocol will be denoted by $\textsf{Confirm}_{\mathbf{S},\mathbf{V}}(\star)$ (resp. $\textsf{Deny}_{\mathbf{S},\mathbf{V}}(\star)$), where $\star$ is the common input of the players.

Definition 3.3.4. *Let consider an undeniable signature scheme defined according to Subsection 3.2.1. The different security notions such as* completeness, soundness, zero-knowledge, *and* non-transferability *of the confirmation (resp. denial) protocol hold if the following respective definitions are satisfied.*

Completeness. *Given random key pairs generated by*

$$(\mathcal{K}_{\rm p}^{\mathbf{S}}, \mathcal{K}_{\rm s}^{\mathbf{S}}) \leftarrow \textsf{Setup}^{\mathbf{S}}(1^k), \quad (\mathcal{K}_{\rm p}^{\mathbf{V}}, \mathcal{K}_{\rm s}^{\mathbf{V}}) \leftarrow \textsf{Setup}^{\mathbf{V}}(1^k),$$

for any valid (resp. invalid) message-signature pair $(m, \sigma) \in \mathcal{M} \times \Sigma$, the confirmation (resp. denial) protocol $\textsf{Confirm}_{\mathbf{S},\mathbf{V}}(m, \sigma, \mathcal{K}_{\rm p}^{\mathbf{S}}, \mathcal{K}_{\rm p}^{\mathbf{V}})$ (resp. $\textsf{Deny}_{\mathbf{S},\mathbf{V}}(m, \sigma, \mathcal{K}_{\rm p}^{\mathbf{S}}, \mathcal{K}_{\rm p}^{\mathbf{V}})$) outputs 1 with probability 1 when $\mathbf{S}$ and $\mathbf{V}$ correctly follow all steps of the protocol. The probability is taken over the random tapes of the different involved algorithms.

Soundness. *Given random key pairs generated by*

$$(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k), \quad (\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k),$$

for any invalid (resp. valid) message-signature pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ *and any cheating signer* $\mathbf{S}^*$ *(modeled as a probabilistic polynomial time interactive machine with access to* $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$*), the probability that the confirmation protocol* $\mathsf{Confirm}_{\mathbf{S}^*,\mathbf{V}}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$ *(resp. denial protocol* $\mathsf{Deny}_{\mathbf{S}^*,\mathbf{V}}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$*) outputs* 1 *is negligible with respect to the size of* k.
The success probability of $\mathbf{S}^*$ *is denoted by* $\mathsf{Succ}_{\mathbf{S}^*}^{\mathsf{sd\text{-}con}}$ *(resp.* $\mathsf{Succ}_{\mathbf{S}^*}^{\mathsf{sd\text{-}den}}$*).*

Black-Box Zero-Knowledge. *Let us consider some random key pairs generated as follows*

$$(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k), \quad (\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k).$$

The confirmation (resp. denial) protocol is black-box zero-knowledge if there exists a probabilistic polynomial time oracle machine $\mathcal{B}$ *called simulator such that for any probabilistic polynomial cheating verifier* $\mathbf{V}^*$ *(with or without* $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$*) and any valid (resp. invalid) pair* $(m, \sigma) \in \mathcal{M} \times \Sigma$*,* $\mathcal{B}^{\mathbf{V}^*}$ *outputs a transcript (random variable indexed by k) which is indistinguishable from the transcript of the protocol* $\mathsf{Confirm}_{\mathbf{S},\mathbf{V}^*}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$ *(resp.* $\mathsf{Deny}_{\mathbf{S},\mathbf{V}^*}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$*), where* $\mathbf{S}$ *is the honest signer. We assume that* $\mathcal{B}$ *and* $\mathbf{V}^*$ *share the same information (e.g.,* $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ *if any). Namely, when* $\mathbf{V}^*$ *has access to some random oracles,* $\mathcal{B}$ *can see the queries (and answers) as well. Moreover, we say that the protocol is straight-line zero-knowledge if* $\mathcal{B}$ *does not need to rewind* $\mathbf{V}^*$.

Non-Transferability. *Let us consider some random key pairs generated as follows*

$$(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k), \quad (\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k).$$

The confirmation (resp. denial) protocol is non-transferable if there exists a probabilistic polynomial time interactive machine $\mathcal{B}$ *with input* $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ *such that for any computationally unbounded cheating verifier* $\tilde{\mathbf{V}}$ *and any pair* $(m, \sigma) \in \mathcal{M} \times \Sigma$*, the transcript of* $\mathsf{Confirm}_{\mathcal{B},\tilde{\mathbf{V}}}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$ *(resp.* $\mathsf{Deny}_{\mathcal{B},\tilde{\mathbf{V}}}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$*) is indistinguishable from that of* $\mathsf{Confirm}_{\mathbf{S},\tilde{\mathbf{V}}}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$ *(resp.* $\mathsf{Deny}_{\mathbf{S},\tilde{\mathbf{V}}}(m, \sigma, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$*). When* $\tilde{\mathbf{V}}$ *has access to some random oracles,* $\mathcal{B}$ *does not see any queries (nor answers) made to them. However,* $\mathcal{B}$ *is assumed to be given a bit telling whether* (m, σ) *is valid or not.*

Remark 3.3.5. Note that the soundness is also crucial for the *non-repudiation* of the signatures, i.e., the signer cannot claim that a signature generated by himself is not valid. Namely, the main role of the denial protocol is to give the possibility for the signer to deny an invalid signature. As a consequence, the signer cannot repudiate valid signatures if the denial protocol is sound.

We note that definition of non-transferability allows to avoid some attacks in which the verifier $\mathbf{V}^*$ identified with $\mathcal{K}_\mathrm{p}^\mathbf{V}$ forwards messages to the honest signer which were generated by an hidden verifier $\tilde{\mathbf{V}}$. Namely, our definition ensures that $\mathbf{V}^*$ with knowledge of $\mathcal{K}_\mathrm{s}^\mathbf{V}$ could simulate the answer of $\mathbf{S}$ (without any help from the honest signer $\mathbf{S}$) so that $\tilde{\mathbf{V}}$ does not have evidence of the proof validity.

Our definition of non-transferability is similar to that proposed by Camenisch and Michels [30] with the main difference that our version assumes that $\tilde{\mathbf{V}}$ is computationally unbounded. We can thus assume that $\tilde{\mathbf{V}}$ makes no queries to the signing and confirmation/denial oracles. Therefore, the non-transferability of the protocols presented below will also hold with respect to the Camenisch-Michels definition.

3.3.2 Designated Confirmer Signatures

Existential Forgery

We define this security notion in a similar way as for undeniable signatures following the same spirit as the existential forgery under adaptive chosen-message attack defined by Goldwasser et al. [76] for classical digital signatures. The main specificity in this context is that the adversary may be the confirmer, so that we give the key pair $\mathcal{K}^\mathbf{C}$ to the adversary. A similar definition can be found in Camenisch and Michels [30].

Definition 3.3.6. *A designated confirmer signature is secure against an existential forgery under an adaptive chosen-message attack if there exists no probabilistic polynomial time algorithm $\mathcal{F}$ which wins the following game with a non-negligible probability.*

Game$^{\textsf{ef-cma}}$. *$\mathcal{F}$ receives a signer's public key $\mathcal{K}_\mathrm{p}^\mathbf{S}$ from $(\mathcal{K}_\mathrm{p}^\mathbf{C}, \mathcal{K}_\mathrm{s}^\mathbf{C}) \leftarrow \mathsf{Setup}^\mathbf{C}(1^k)$, and two key pairs $(\mathcal{K}_\mathrm{p}^\mathbf{C}, \mathcal{K}_\mathrm{s}^\mathbf{C}) \leftarrow \mathsf{Setup}^\mathbf{C}(1^k)$ and $(\mathcal{K}_\mathrm{p}^\mathbf{V}, \mathcal{K}_\mathrm{s}^\mathbf{V}) \leftarrow \mathsf{Setup}^\mathbf{V}(1^k)$. Then, $\mathcal{F}$ can query some chosen messages to a signing oracle. All queries can be sent adaptively and the number of queries must be polynomially bounded in k. $\mathcal{F}$ wins the game if it outputs a valid pair $(m^*, \sigma^*) \in \mathcal{M} \times \Sigma$ such that m^* was not queried to the signing oracle.*
We denote this probability of success by $\mathsf{Succ}_\mathcal{F}^{\mathsf{ef-cma}}$.

Remark 3.3.7. Note that the access to the confirmation (resp. denial) protocol oracle is omitted here, since the adversary can easily simulate these protocols by itself using $\mathcal{K}_\mathrm{s}^\mathbf{C}$.

Lunchtime Invisibility

Here, we present a definition which is adapted from Camenisch and Michels [30] and which is a chosen-message version of Definition 3.3.3.

Definition 3.3.8. *Consider a probabilistic polynomial time algorithm $\mathcal{D}$ called* invisibility distinguisher *and the two following games with respect to a bit b.*

Game$^{\textsf{inv-lcma-}b}$. *$\mathcal{D}$ receives $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$ (possibly $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$) from $(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}}) \leftarrow \mathsf{Setup}^{\mathbf{C}}(1^k)$, $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$, and a verifier's key pair $(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}}) \leftarrow \mathsf{Setup}^{\mathbf{V}}(1^k)$. He can query some chosen messages to a signing oracle and some message-signature pairs $(m, \sigma) \in \mathcal{M} \times \Sigma$ to some oracles running the confirmation and denial protocol. After a given time (lunchtime), $\mathcal{D}$ does not have access to the oracles anymore. Then, he chooses two messages $m_0, m_1 \in \mathcal{M}$ and submits them to a challenger. He receives $\sigma = \mathsf{Sign}(m_b, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{C}})$. Finally, $\mathcal{D}$ outputs a guess bit b'.*

We define the advantage of the distinguisher as follows

$$\mathsf{Adv}_{\mathcal{D}}^{\textsf{inv-lcma}} := \left|\Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\textsf{inv-lcma-1}}\right] - \Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\textsf{inv-lcma-0}}\right]\right|,$$

where probabilities are over the random tapes of the involved algorithms. An undeniable signature scheme is said to be invisible under a lunchtime chosen-message attack *if there exists no probabilistic polynomial time algorithm $\mathcal{D}$ such that the advantage $\mathsf{Adv}_{\mathcal{D}}^{\textsf{inv-lcma}}$ is non-negligible.*

Remark 3.3.9. When the signer's secret key is given to the distinguisher, the access to the signing oracle can be omitted.

Note that this definition is a little weaker than the definition of [30] in which $\mathcal{D}$ can continue to send queries to the oracles after the selection of m_0, m_1.

Non-Coercibility

This notion prevents that the signer **S** is coerced by anybody who would like to get a proof that a given signature was really generated by **S** after the signature is released. As far as the signer erases his intermediate computations, this notion can be regarded as an extension of the invisibility property in which the adversary is given $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$. Indeed a signer who would keep in memory the random values needed to generate a signature could be coerced to prove later how this one was generated. Note also that we should distinguish the non-coercibility from the receipt-freeness where the signer would be unable to keep a proof that he really generated a given signature even if he meant to. This extends the non-coercibility to the non-corruptibility.

Notions Related to the Confirmation and Denial Protocols

Security properties related to the confirmation and denial protocols are like for an undeniable signature, i.e., the *completeness*, the *soundness*, *zero-knowledge* and the *non-transferability.* The definitions given in Subsection 3.3.1 directly apply except that we replace the signer by the confirmer and that an additional key pair needs to be considered.

3.4 Related Work

This part mainly offers a short overview of the different contributions dedicated to undeniable signatures and designated confirmer signatures. Of course, it is not intended to be exhaustive but we aim at recalling the most important achievements made in these fields. Although a few results are related to both topics, we prefer to provide separate treatments in order to make a clear distinction between the contributions of both respective subjects.

Undeniable Signatures

Introduction of undeniable signatures dates back to 1989 with the article of Chaum and van Antwerpen [40] presented at the CRYPTO conference. Their original motivation was to protect the signer's privacy and argued that this property may be particularly important in applications where commercially or personally sensitive data are signed. Namely, using classical digital signatures in such a context may lead to the dissemination of sensitive information verifiable by anybody due to the universal verifiability of ordinary signatures.

One year later, Chaum [36] proposed a new version of this scheme with modified confirmation and denial protocols satisfying zero-knowledge property. Contrary to the previous protocols, one is then ensured that no information (except validity or invalidity of the signature) leaks to a possible malicious verifier. In particular, a verifier cannot convince another party that a given signature is valid by attaching the transcript of the confirmation protocol.

The same year, Boyar et al. [23] introduced the concept of *convertible* undeniable signature for which the signer can turn all the previous signature into universally verifiable ones by releasing some additional information. They also introduce the concept of *selective convertibility* and proposed a generic construction of a convertible undeniable signature with selective convertibility. In addition, this construction shows that convertible undeniable signatures exist if and only if one-way functions exist. They furthermore developed a practical scheme based on the ElGamal signature scheme [56]. As it sometimes turns out in cryptography, this scheme was later shown insecure in 1996 by Michels et al. [103]. More precisely, the signature scheme becomes forgeable once the information to convert all signatures is released by the signer. A way to repair this scheme was also developed in [103], but the authors did not give a formal proof. Also based on ElGamal signatures, Damgård and Pedersen [49] presented at Eurocrypt '96 two provably secure schemes with convertibility property. Finally, following the tradition of designing discrete logarithm based scheme, Michels and Stadler [104] proposed a convertible undeniable signature scheme based on the Schnorr signature scheme [137] which is also suitable for a threshold variant, i.e., where several signers share the signing ability.

First undeniable signature schemes which are not based on discrete logarithms were developed by Gennaro et al. [65, 66] in 1997. The signature generation works like for an RSA ordinary signature where the modulus is composed of safe primes. A variant of this RSA based signature with a general modulus were done by Galbraith et al. [63] and a detailed analysis of invisibility properties of RSA based signatures were studied by Galbraith and Mao in [62].

A few years ago, topic of undeniable signatures has become quite active and several new schemes were published. In 2004, Biehl et al. [15] used quadratic orders to design a new scheme and Libert and Quisquater [98] introduced an identity-based undeniable signature scheme using bilinear pairings. These ones allowed Laguillaumie and Vergnaud [89] to design in 2005 a scheme offering the possibility of converting all signatures pertaining at a given period of time. Using again some pairings, the same authors have been able to get rid of the use of random oracles in [88].

Original scheme of Chaum [36] was further studied last few years. In 2001, Okamoto and Pointcheval [119] introduced some "gap-problems" and showed that the security of the Chaum's scheme can be based on the Gap Diffie-Hellman problem. Later, Ogata et al. [116] showed that one can actually prove the security of this scheme using the Computational Diffie-Hellman problem. Finally, Kurosawa and Heng [86] proposed some 3-move verification protocols which are not zero-knowledge. They showed that unforgeability and invisibility still holds. However, their Chaum's scheme variant does not achieve non-transferability.

Besides the development of the design of new schemes, additional work about different issues related to undeniable signatures has been achieved. In 1991, Desmedt and Yung [51] (see also some Chaum's criticisms in [37]) presented some weaknesses of undeniable signatures. In particular, they showed that several verifiers can be convinced during a verification protocol, while the signer believes that he is interacting only to one legitimate verifier. This may cause some problems in applications where the verification is valuable, for instance, if this is used to check the authenticity of a software to customers who paid for a license. In 1994, Jakobsson [80] showed that similar attacks can be performed to blackmail the signer. In order to prevent these kinds of attacks, Jakobsson et al. [81] introduced non-transferable protocols which ensures that a verifier cannot transfer the validity/invalidity proof of a given signature to another party during a confirmation or denial protocol. To achieve this, they developed so-called *designated verifier proofs* in which the prover designate the verifier he is going to convince on a given statement. Moreover, a generic construction based on trapdoor commitments [26] is also given in this article.

Before to conclude this overview, we would like to mention a few additional results. In 1991, Pedersen [126] developed a threshold variant of undeniable signature. The signer's secret key is shared among several provers and the recipient of a message can verify a signature by interacting with a subset of enough many (with respect to the threshold) provers. Another useful contribution is due to Fujioka et al. [61] who introduced the bi-proof concept which allows to prove in one sole protocol whether the signature is valid or invalid. So, one bi-proof protocol can replace both the confirmation and denial protocols at the same time. Finally, Chaum et al. [41] have investigated undeniable signatures which are unconditionally secure for the signer, i.e., even against a computationally unbounded forger, the signer will be able to deny forged signatures.

For another overview on the development of undeniable signatures, we refer to the PhD thesis of Laguillaumie [87].

Designated Confirmer Signatures

Designated confirmer signatures were introduced in 1994 in an article of Chaum [38]. The main motivation was that undeniable signatures do not offer an ideal solution when the signer may become unavailable. In his original article, Chaum proposed a scheme with verification protocols similar to those of his undeniable signature scheme [36]. Yet, he did not provide any formal proof of his scheme nor a security model for a designated confirmer signature. The same year, Okamoto [118] presented a formal security model for this cryptographic primitive and proposed a generic way of constructing a designated confirmer signature. This allowed him to prove that a primitive equivalent to public-key encryption is required to achieve a secure designated confirmer signature. More practical results are also given in his article, where he developed practical constructions based on 3-move identification protocols without any formal security proof.

In 1998, Michels and Stalder [105] pointed out that practical constructions of Okamoto succumb under a forgery attack if the adversary is given the confirmer's secret key. A countermeasure is proposed without a security proof. They introduced the notion of confirmer commitments allowing to construct designated confirmer signatures. However, two years later Camenisch and Michels [30] showed that adaptive attacks can break the invisibility of this scheme as well as that of Chaum [38]. They also proposed a construction based on verifiable encryption but with an inefficient denial protocol. The efficiency has been strongly improved and made practical by Camenisch and Shoup in [31]. Recently, Goldwasser and Waisbard [77] developed a general construction without random oracles using general (inefficient) zero-knowledge proofs. Finally, Gentry et al. [67] used commitments and techniques of Camenisch and Shoup to design a scheme with some practical confirmation and denial protocols without making use of random oracles.

Chapter 4

MOVA Undeniable Signature

Up until now, undeniable signature schemes did not fully exploit online security properties towards the design of schemes offering very short signatures. One of the main contributions of this thesis is to remedy to this situation. To this goal, we develop a very general framework based on the sole notion of the interpolation of group homomorphisms. Based on this, we define a decisional and computational problem, which generalize several fundamental problems related to public-key cryptography. Among them, we find the decision and computational Diffie-Hellman problems as well as the quadratic residuosity problem.

The interest of this new perspective to undeniable signatures is twofold. First, group homomorphisms allow to express the well-known Chaum's undeniable signature [36] and the RSA undeniable signature of Gennaro et al. [66] in a unified formalism. Secondly, our technique allows to develop very short signatures in a quite natural way, namely by instantiating our scheme with group homomorphisms with a range group of small size.

In what follows, we introduce the concept of interpolation of group homomorphisms and related problems. We then dedicate a part of this chapter to deal with interactive proof protocols related to the interpolation of group homomorphisms. In particular, we consider two protocols in which the prover proves to a verifier that a given set of points interpolates (resp. does not interpolate) in a group homomorphism. From this, we develop the MOVA undeniable signature scheme whose confirmation and denial protocols are directly based on the previously defined protocols. Finally, we develop the security properties of MOVA by providing security reductions to some supposedly hard problems. These results allow to quantify different parameters of the scheme given the security level of the different security properties.

4.1 Interpolation of Group Homomorphisms

4.1.1 Problem Definitions

In what follows, we will mainly deal with finite Abelian groups written in an additive way. In this case, the following notation for a scalar multiplication will apply.

Notation. For a non negative integer k and an element g of an Abelian group, kg will denote the element

$$\underbrace{g+g+\cdots+g}_{k \text{ times}}.$$

If k is negative, kg denotes the element $-(-k)g$.

The concept of group homomorphism interpolation is defined below.

Definition 4.1.1. *Let G, H be two Abelian groups and S a subset of $G \times H$ written in the form $S := \{(x_1, y_1), \ldots, (x_s, y_s)\}$.*

1. *We say that the set of points S* interpolates in a group homomorphism *if there exists a group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for $i = 1, \ldots, s$.*
2. *We say that a set of points $B \subseteq G \times H$* interpolates in a group homomorphism with another set of points $A \subseteq G \times H$ *if $A \cup B$ interpolates in a group homomorphism.*

Group Homomorphism Interpolation Problem

We state here the Group Homomorphism Interpolation problem (GHI problem) and its corresponding decisional problem (GHID problem).

n-S-**GHI Problem** (n-S-Group Homomorphism Interpolation Problem)

Parameters: Two Abelian groups G and H, a set $S \subseteq G \times H$ of s points, and a positive integer n.

Instance Generation: n elements $x_1, \ldots, x_n$ picked uniformly at random in G.

Problem: Find $y_1, \ldots, y_n \in H$ such that $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ interpolates with S in a group homomorphism.

n-S-**GHID Problem** (n-S-GHI Decisional Problem)

Parameters: Two Abelian groups G and H, a set $S \subseteq G \times H$ of s points, and a positive integer n.

Instance Generation: The instance T is generated according to one of the two following ways and is denoted T_0 or T_1 respectively. T_0 is a set of n points $\{(x_1, y_1), \dots, (x_n, y_n)\} \in (G \times H)^n$ picked uniformly at random such that it interpolates with S in a group homomorphism. T_1 is picked uniformly at random in $(G \times H)^n$.

Problem: Decide whether the instance T is of type T_0 or T_1.

Remark 4.1.2. We point out, the distribution T_0 does not seem easy to produce in general. However, when the set of points uniquely determines a homomorphism f, one can generate T_0 by picking the x_i's uniformly at random and setting $y_i := f(x_i)$ for $i = 1, \dots, s$.

In practice, we consider the n-S-GHI and n-S-GHID problems for sets S which interpolate in a unique group homomorphism. Hence, S defines a homomorphism. The n-S-GHI problem consists in computing it on n elements. The n-S-GHID problem consists in deciding whether all points of T lie in its graph. From now on, we always assume that this homomorphism is unique.

Additionally, we define the special S-GHID problem which is defined as for n-S-GHID with no input and the problem consists in deciding whether S interpolates in a group homomorphism.

Note also that GHI and GHID are some generic problems and that parameters G, H, S can be of full diversity since no restriction about their form is made a priori. Later, when more practical aspects will be required, some precise specifications of these parameters will be presented leading to different possible instantiations.

Related Computational Problems

We also consider the following (non-probabilistic) problems.

d-**MGGD Problem** (Modular Group Generation Decisional Problem)

Parameters: An Abelian group G, a positive integer d.

Instance: A set of values $S_1 = \{x_1, \dots, x_s\} \subseteq G$.

Problem: Does S_1 modulo dG span G/dG?

(d, S_1)-**MSR Problem** (Modular System Representation Problem)

Parameters: An Abelian group G, a set $S_1 = \{x_1, \dots, x_s\} \subseteq G$, and a positive integer d.

Instance: An element $x \in G$.

Problem: Find $a_1, \dots, a_s \in \mathbb{Z}$ such that $x \in a_1x_1 + \cdots + a_sx_s + dG$. If no solution exists, output $\perp$.

d**-Root Problem** (dth Root Problem)

Parameters: An Abelian group G and a positive integer d.

Instance: An element $x \in dG$.

Problem: Find $r \in G$ such that $x = dr$.

Problem Solvability

The hardness of the different above problems against a given adversary is measured by the success probability of this one. As usual, the adversary is formalized by a probabilistic polynomial time algorithm (Turing machine). The success probability of such an adversary $\mathcal{A}$ of solving a problem $\mathcal{P}$ (with some specified parameters and possibly an instance) is denoted

$$\mathsf{Succ}_{\mathcal{A}}^{\mathcal{P}}.$$

For instance, if we take a 1-S-GHI problem such that S defines a unique group homomorphism f, we have

$$\mathsf{Succ}_{\mathcal{A}}^{1\text{-}S\text{-GHI}} = \Pr_{x,\varpi}[y = f(x) \mid x \leftarrow_U G;\ y \leftarrow \mathcal{A}(x;\varpi)],$$

where ϖ denotes the random tape of $\mathcal{A}$.

When the underlying problem $\mathcal{P}$ is decisional, we usually call the adversary "distinguisher" and denotes him by $\mathcal{D}$. We denote the input of the problem as x and the goal of $\mathcal{D}$ is to decide whether x was drawn according to the distribution D_0 or D_1. In the decisional setting, we prefer to consider the following measure of success

$$\mathsf{Adv}_{\mathcal{D}}^{\mathcal{P}} := \left|\Pr_{x,\varpi}[0 \leftarrow \mathcal{D}(x;\varpi) \mid x \leftarrow D_0] - \Pr_{x,\varpi}[0 \leftarrow \mathcal{D}(x;\varpi) \mid x \leftarrow D_1]\right|$$

called the advantage of the distinguisher $\mathcal{D}$ and where ϖ denotes the random tape of $\mathcal{D}$. The motivation to consider the advantage instead of the probability of success is based on the fact that a trivial distinguisher can always solve a decisional problem with probability $1/2$. The advantage of such algorithms is 0 which is a far more appropriate measure. If the input x is picked according to D_0 and D_1 with a probability $1/2$, the advantage is equal to

$$\mathsf{Adv}_{\mathcal{D}}^{\mathcal{P}} = \left|2 \cdot \Pr_{x,\varpi,b}[b \leftarrow \mathcal{D}(x;\varpi) \mid b \leftarrow_U \{0,1\}; x \leftarrow D_b] - 1\right|.$$

Throughout this thesis, we will consider some instances of the n-S-GHI and n-S-GHID problems which are assumed to fulfil the following assumption.

Assumption. For any probabilistic algorithms $\mathcal{A}$ and $\mathcal{D}$ running in polynomial time with respect to the instance size, the functions

$$\mathsf{Succ}_{\mathcal{A}}^{n\text{-}S\text{-GHI}} \quad \text{and} \quad \mathsf{Adv}_{\mathcal{D}}^{n\text{-}S\text{-GHID}},$$

are negligible with respect to the instance size.

4.1.2 Preliminaries

In this subsection, we give some technical results related to the interpolation of group homomorphisms. The importance of the presented material will become clear in the subsequent sections. We first present some properties a set of points $S = \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq G \times H$ should satisfy in order to interpolate in a unique group homomorphism. We begin by proposing a criterion on the x_i's such that we are ensured that at most one group homomorphism exists. Then, we present an additional property involving the elements y_i's which guarantees the existence of such a homomorphism.

Uniqueness of the Interpolation

Lemma 4.1.3. *Let G, H be two finite Abelian groups. We denote d and λ the order and the exponent of H respectively. Let $x_1, \ldots, x_s \in G$ which span a subgroup denoted by G'. The following properties are equivalent. In this case, we say that $x_1, \ldots, x_s$ H-generate G.*

1. *For any elements $y_1, \ldots, y_s \in H$, there exists at most one group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for all $i = 1, \ldots, s$.*
2. *There exists a unique group homomorphism $\varphi : G \longrightarrow H$ such that $\varphi(x_i) = 0$ for $i = 1, \ldots, s$, namely $\varphi = 0$.*
3. *The set $\mathrm{Hom}(G/G', H)$ of all group homomorphisms from G/G' to H is restricted to $\{0\}$.*
4. $\gcd(\#(G/G'), d) = 1$.
5. *$G' + dG = G$.*
6. *$G' + \lambda G = G$.*
7. *The cosets $x_1 + dG, \ldots, x_s + dG$ span G/dG.*
8. *The cosets $x_1 + \lambda G, \ldots, x_s + \lambda G$ span $G/\lambda G$.*

Proof. **1 ⇒ 2.** This directly follows by choosing $y_i = 0$ for all $i = 1, \ldots, s$.

2 ⇒ 1. Assume that there exist two group homomorphisms f_1, f_2 from G to H such that $f_1(x_i) = f_2(x_i) = y_i$ for all $i = 1, \ldots, s$. Then, by assertion 2, we deduce that the group homomorphism $f_1 - f_2$ must be equal to the homomorphism 0.

2 ⇒ 3. Suppose that there exists a homomorphism $\bar{\varphi} : G/G' \to H$ which is not equal to 0. Let $\pi_{G'} : G \to G/G'$ denote the canonical projection. We define the homomorphism $\varphi := \bar{\varphi} \circ \pi_{G'}$ from G to H. By definition, $\pi_{G'}(x_i) = 0$ and therefore $\varphi(x_i) = 0$ for any $i = 1, \ldots, s$. Moreover, since $\pi_{G'}$ is onto and $\bar{\varphi}$ is not trivial, φ must be different from 0, which contradicts the assertion 2.

3 ⇒ 4. Suppose the existence of a common prime factor p of $\#(G/G')$ and d. Then, from the structure of Abelian groups (see Appendix A.1), G/G' and H must both possess one cyclic subgroup U and V respectively of order p. Let λ' denote the exponent of the group G/G'. By the structure of Abelian groups, we can choose U of the form $\lambda'/p \cdot (G/G')$. Hence, we have a group homomorphism

$$\begin{array}{rccc} \varphi : & G/G' & \longrightarrow & U \\ & x & \longmapsto & \frac{\lambda'}{p} x \end{array}$$

which is onto. So, we can define a non trivial homomorphism which is the composition of φ and the isomorphism between U and V. This contradicts 3.

4 ⇒ 5. Let $x \in G$ and $k := \mathrm{ord}(x \bmod G')$ be the order of $x \bmod G'$ in the quotient group G/G'. By the assertion 4, d must be invertible modulo k. Let $m \in \mathbb{Z}$ such that $m \cdot d \equiv 1 \pmod{k}$. We have $m \cdot d \cdot x \equiv x \pmod{G'}$. Hence, $x - d(m \cdot x) \in G'$ and therefore $x \in G' + dG$.

5 ⇒ 2. Let $\varphi \in \mathrm{Hom}(G, H)$ such that $\varphi|_{G'} = 0$ and $x \in G$. By assertion 5, we can write $x = a_1 x_1 + \cdots + a_s x_s + dr$ for some integers $a_1, \ldots, a_s$ and an element $r \in G$. Thus, $\varphi(x) = d\varphi(r) = 0$. This holds for any $x \in G$, i.e., $\varphi = 0$.

5 ⇒ 6. This directly follows from $dG \subseteq \lambda G$, since $\lambda | d$.

6 ⇒ 2. Let $\varphi \in \mathrm{Hom}(G, H)$ such that $\varphi|_{G'} = 0$ and $x \in G$. By assertion 6 we can write $x = a_1 x_1 + \cdots + a_s x_s + \lambda r$ for some integers $a_1, \ldots, a_s$ and an element $r \in G$. Thus, $\varphi(x) = \lambda\varphi(r) = 0$. This holds for any $x \in G$, i.e., $\varphi = 0$.

5 ⇔ 7. This follows from $G' + dG = G \Leftrightarrow \{x' + dG \mid x' \in G'\} = G/dG$.

6 ⇔ 8. This follows from $G' + \lambda G = G \Leftrightarrow \{x' + \lambda G \mid x' \in G'\} = G/\lambda G$. □

Remark 4.1.4. Note that the criteria 4-8 suggest that H is only involved by the prime factors of its order. Later, the smallest prime factor p will play an important role. Note also that if $G = H$, these criteria mean that $x_1, \ldots, x_s$ generate G. Furthermore, from the assertion 7, we see that a positive answer to the d-MGGD problem for $S_1 = \{x_1, \ldots, x_s\}$ is equivalent to say that $x_1, \ldots, x_s$ H-generate G.

Expert Group Knowledge. The assertion 5 states that any $x \in G$ can be written in the form $x = dr + a_1x_1 + \cdots + a_sx_s$ for some coefficients $a_1, \ldots, a_s \in \mathbb{Z}_d$ and $r \in G$. Such a representation of x can be always found when one is able to solve both (d, S_1)-MSR and d-Root problems perfectly. We say that one has an *expert group knowledge* of G with the set $S_1 = \{x_1, \ldots, x_s\}$, if one is able to find such a representation for any $x \in G$.

Existence of the Interpolation

Here is a condition allowing to determine whether a set of points interpolates in a group homomorphism. The following result assumed that the G-coordinates of this set of points H-generate G so that the group homomorphism is unique when it exists.

Lemma 4.1.5. *Let G, H be two finite Abelian groups. We denote d the order of H. Let $x_1, \ldots, x_s \in G$ which H-generate G. The set $S = \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq G \times H$ interpolates in a group homomorphism if and only if for any $a_1, \ldots, a_s \in \mathbb{Z}$ such that*

$$a_1x_1 + \cdots + a_sx_s \in dG,$$

we have

$$a_1y_1 + \cdots + a_sy_s = 0.$$

Proof. **"⇒":** By assumption, there exists a homomorphism $f : G \to H$ such that $f(x_i) = y_i$ for $i = 1, \ldots, s$. Since dG lies in the kernel of f, we have

$$f(a_1x_1 + \cdots + a_sx_s) = a_1y_1 + \cdots + a_sy_s = 0,$$

whenever $a_1x_1 + \cdots + a_sx_s \in dG$.
"⇐". By the assertion 5 of Lemma 4.1.3, we know that any element $x \in G$ can be written in the form $x = dr + a_1x_1 + \cdots + a_sx_s$ for some integers $a_1, \ldots, a_s$ and an element $r \in G$. We now define a function $f : G \to H$ defined by

$$f(dr + a_1x_1 + \cdots + a_sx_s) := a_1y_1 + \cdots + a_sy_s \qquad (4.1)$$

for any $a_1, \ldots, a_s \in \mathbb{Z}$ and $r \in G$. It remains to prove that f is well-defined on G and that it is homomorphic. Assume that an element $x \in G$ admits two different representations, i.e.,

$$x = dr + a_1x_1 + \cdots + a_sx_s = dr' + a'_1x_1 + \cdots + a'_sx_s.$$

By assumption, we must have

$$(a_1 - a'_1)y_1 + \cdots + (a_s - a'_s)y'_s = 0,$$

and therefore

$$f(dr + a_1x_1 + \cdots + a_sx_s) = f(dr' + a'_1x_1 + \cdots + a'_sx_s).$$

The homomorphic property directly follows from the linearity of the right hand side of (4.1). □

Remark 4.1.6. Note that we can replace d by the exponent λ of the group H in Lemma 4.1.5.

Remark 4.1.7. Lemma 4.1.5 does not hold anymore if we relax the assumption stating that $x_1, \ldots, x_s$ H-generate G. The following example illustrates this statement. Let $G = \mathbb{Z}_{27}$, $H = \mathbb{Z}_9 \oplus \mathbb{Z}_3$, $x_1 = 3$, and $y_1 = (0, 1)$. We have $d = 27$ and $\lambda = 9$. For any group homomorphism $f : \mathbb{Z}_{27} \to \mathbb{Z}_9 \oplus \mathbb{Z}_3$, we have $f(x_1) = 3 \cdot f(1)$ showing that $f(x_1)$ cannot be equal to y_1 since the second component should vanish. However, $a_1x_1 \in 27G$ implies that a_1 is a multiple of 9, thus $a_1y_1 = 0$. This shows that Lemma 4.1.5 does not apply in this case. Note also that replacing d by λ in this example would lead to the same conclusion.

By the above remark, relaxing the assumption of the H-generation of G in Lemma 4.1.5 can be achieved only if we add some assumptions on the groups G and H. In this thesis, we usually do not care about situations where more than one group homomorphism can exist. We will always assume that the $x_1, \ldots, x_s$ H-generate G. So, we prefer to consider the above Lemma 4.1.5 with no restriction on the groups G and H.

Examples of GHI and GHID Problems

We can often meet the GHI and GHID problems in cryptography as the following examples suggest. Here, we exclusively consider 1-GHI and 1-GHID variants.

Example 1. We take a cyclic group G of order q, $H = \mathbb{Z}_q$, and a generator g of G. The set $S = \{(g, 1)\}$ interpolates in a unique group homomorphism, and the GHI problem is exactly the discrete logarithm problem.

Example 2. We take a cyclic group $G = H$ of order q, and a generator g of G. For any $a \in \mathbb{Z}_q$, $S = \{(g, ag)\}$ interpolates in a unique group homomorphism: the exponentiation to the power a. The GHI and GHID problems correspond to the Diffie-Hellman problem [53] and the decisional Diffie-Hellman problem when S is refreshed for each instance with an element $a \in_U \mathbb{Z}_q$ picked uniformly at random.

Example 3. Let $n = pq$ such that p, q are different odd primes and $H = \{-1, +1\}$. We let $x_1, x_2 \in \mathbb{Z}_n^*$ be such that x_1 is a quadratic residue modulo p and not modulo q, and that x_2 is a quadratic residue modulo q, and not modulo p. We notice that

$S = \{(x_1, 1), (x_2, -1)\}$ interpolates in a unique group homomorphism which is the Legendre symbol $(\cdot/p)$. Since it is easy to compute $(\cdot/n)$, the quadratic residuosity problem [73] with the information x_1 and x_2 is equivalent to the GHI and GHID problems restricted to the inputs $x \in \mathbb{Z}_n^*$ such that $(x/n) = 1$.

In Chapter 5, we will deal with characters on $\mathbb{Z}_n^*$, which are a natural generalization of the Legendre symbol.

Example 4. Here, we consider the well known RSA cryptosystem [131]. Let $n = pq$ be an RSA modulus and $G = H = \mathbb{Z}_n^*$. Let $f : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ be defined by $f(x) = x^e \bmod n$ for an exponent e such that $\gcd(e, \varphi(n)) = 1$. Given enough many pairs $(x_i^e \bmod n, x_i) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$, $i = 1, \ldots, s$, such that the first coordinates generate $\mathbb{Z}_n^*$, the RSA decryption problem corresponds to the GHI problem with S composed of the above pairs.

Example 5. We show here how we can apply the GHI problem to the Bilinear Diffie-Hellman Problem (BDHP). This problem was used in the seminal paper of Boneh and Franklin [20, 21] to propose an identity-based encryption scheme based on it. Let $\hat{e} : G_1 \times G_1 \to G_2$ be a bilinear, non-degenerate and computable mapping, where G_1 and G_2 are cyclic groups of a large prime order p. Let P be a generator of G_1, we can state the BDHP as follows: given three random elements aP, bP and $cP \in G_1$, compute $\hat{e}(P, P)^{abc}$. (G_1 resp. G_2 is written additively resp. multiplicatively.) BDHP is equivalent to GHI problem with the set $S = \{(P, \hat{e}(aP, bP))\}$ and $x_1 = cP$ when S is refreshed for each instance with some a and b picked uniformly at random in $\mathbb{Z}_p$.

Example 6. Let consider the Paillier's trapdoor function [123] that maps an element $(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$ to the element $g^x \cdot y^n \bmod n^2$ of $\mathbb{Z}_{n^2}^*$, with g an element of $\mathbb{Z}_{n^2}^*$ of order n. For such a g, the Paillier trapdoor function is an isomorphism. Thus, assuming we have s pairs of plaintext/ciphertext that generate $\mathbb{Z}_n \times \mathbb{Z}_n^*$ resp. $\mathbb{Z}_{n^2}^*$, the decryption problem of a challenged ciphertext corresponds to the GHI problem with $G = \mathbb{Z}_{n^2}^*$ and $H = \mathbb{Z}_n \times \mathbb{Z}_n^*$. This application of GHI problem to the decryption problem can be adapted to every homomorphic trapdoor function.

Note that Examples 2,3,4,5,6 include trapdoors in order to interpolate the group homomorphism. Furthermore, Example 3 includes a trapdoor in order to solve the MSR problem. Also note that the order d of H is publicly known in Examples 1,2,3,5. It is further quite small in Example 3. In what follows we focus on publicly known d and on trapdoor homomorphisms. We will also consider the following example inspired by [3].

Example 7. Let $n = pq$ such that $p = rd + 1$ and q are prime, $\gcd(r, d) = 1$, $\gcd(q - 1, d) = 1$, with d small prime. We take $G = \mathbb{Z}_n^*$ and $H = \mathbb{Z}_d$. We can easily compute a group homomorphism by first raising to the power $r(q - 1)$ then computing a discrete logarithm in a small cyclic subgroup of order d.

Sampling G uniformly

We first state a basic but useful lemma.

Lemma 4.1.8. *Let $f : G \to H$ be a surjective group homomorphism from the group G to the group H. Then, f is balanced, i.e.,*

$$\#f^{-1}(y) = \#\mathrm{Ker}(f) \text{ for any } y \in H.$$

Proof. Let $x, x' \in G$ and $y := f(x)$. The lemma follows by noticing that

$$f(x') = y \text{ if and only if } x' \in x + \mathrm{Ker}(f).$$

□

We finally provide a useful lemma to sample group elements.

Lemma 4.1.9. *Let G, H, d be defined as in Lemma 4.1.5. Let $x_1, \dots, x_s \in G$ which H-generate G. The following mapping from $G \times \mathbb{Z}_d^s$ to G is balanced*

$$g : \quad (r, a_1, \dots, a_s) \quad \longmapsto \quad dr + a_1x_1 + \dots + a_sx_s. \tag{4.2}$$

Proof. Let n be the order of G. Let $h : G \times \mathbb{Z}_{nd}^s \to G$ be a function defined by $h(r, a_1, \dots, a_s) := dr + a_1x_1 + \dots + a_sx_s$. Obviously, h is a homomorphism. It is onto due to the assertion 5 of Lemma 4.1.3. Hence, it is balanced by Lemma 4.1.8. Let $\varphi : G \times \mathbb{Z}_{nd}^s \to G \times \mathbb{Z}_d^s$ be a function defined by

$$\varphi(r, a_1, \dots, a_s) := (r + q_1x_1 + \dots + q_sx_s, a_1 \bmod d, \dots, a_s \bmod d),$$

where $a_i - (a_i \bmod d) = dq_i$, for $i = 1, \dots, s$. We have $g \circ \varphi = h$. We note that φ is balanced onto $G \times \mathbb{Z}_d^s$ since

$$\varphi^{-1}(r, a_1, \dots, a_s) = \{(r - q_1x_1 - \dots - q_sx_s, a_1 + dq_1, \dots, a_s + dq_s) \mid (q_1, \dots, q_s) \in \mathbb{Z}_n^s\}.$$

If $\#g^{-1}(x) = m$, we have $mn^s = \#\varphi^{-1}\left(g^{-1}(x)\right) = \#h^{-1}(x) = (dn)^s$. Hence, $m = d^s$ does not depend on x, so g is balanced. □

Remark 4.1.10. We can prove a variant of Lemma 4.1.9, where d is replaced by the exponent λ of the group H.

Sampling Representations with Expert Group Knowledge

Note that the expert group knowledge corresponds to the ability of finding an element of the set $g^{-1}(x)$ for any $x \in G$. We show here that this ability even allows to pick an element in $g^{-1}(x)$ uniformly at random for any $x \in G$. At first, we remark that two representations $(r, a_1, \dots, a_s)$, $(r', a'_1, \dots, a'_s)$ represent the same element x if and only if we have

$$d(r - r') + (a_1 - a'_1)x_1 + \cdots + (a_s - a'_s)x_s = 0.$$

In other words, two representations lie in the same set $g^{-1}(x)$ for a given x if and only if they difference lie in $g^{-1}(0)$. This shows that sampling elements of $g^{-1}(x)$ uniformly at random can be achieved by finding any representation of x (expert group knowledge) and adding this one with a representation of the neutral element 0 picked uniformly at random.

We now explain how to pick some representation $(r, a_1, \dots, a_s)$ uniformly at random in $g^{-1}(0)$, i.e., such that $dr + a_1x_1 + \cdots + a_sx_s = 0$. We pick a tuple $(r', a'_1, \dots, a'_s) \in_U G \times \mathbb{Z}_d^s$ uniformly at random and compute

$$x' := dr' + a'_1x_1 + \cdots + a'_sx_s. \tag{4.3}$$

We apply our expert group knowledge and retrieve a tuple $(r'', a''_1, \dots, a''_s) \in g^{-1}(x')$. We set $r := r' - r''$ and $a_i := a'_i - a''_i$ for $i = 1, \dots, s$ and conclude by noting that the obtained representation is uniform in $g^{-1}(0)$.

From now on, we will use this property directly if it is required and assuming one has an expert group knowledge of G.

4.1.3 Problem Approximations

In this subsection we present some results related to the approximation of a group homomorphism which interpolates in a set of points S. They are inspired from the theory of checkable proofs [5,7] and linear cryptanalysis [82]. We first develop some results about the existence of an interpolating group homomorphism in S which provides an answer to the special S-GHID problem. Next, we give another result focusing on the computability of this one.

Homomorphism Existence

Lemma 4.1.11. *Given two finite Abelian groups G and H, and a set of s points $S = \{(x_i, y_i) \mid i = 1, \dots, s\} \subseteq G \times H$, we assume that $x_1, \dots, x_s$ H-generate G. We let d be the order of H and p be its smallest prime factor. We assume that there exists a function $f : G \longrightarrow H$ such that*

$$\rho := \Pr_{(r, a_1, \dots, a_s) \in_U G \times \mathbb{Z}_d^s} [f(dr + a_1x_1 + \cdots + a_sx_s) = a_1y_1 + \cdots + a_sy_s] > \frac{1}{p}.$$

The set of points S interpolates in a group homomorphism. Furthermore, given a random $x \in_U G$, the value $y = f(x)$ matches the unique interpolation with probability ρ.

Proof. Let K be the subgroup of $\mathbb{Z}_d^s$ defined by

$$K := \{(a_1, \dots, a_s) \in \mathbb{Z}_d^s \mid a_1x_1 + \cdots + a_sx_s \in dG\}.$$

We notice that the representation (4.3) of any element $x \in G$ as a combination of $x_1, \dots, x_s$ is uniquely defined modulo K. By Lemma 4.1.5, the first assertion is proved if one shows that $a_1y_1 + \cdots + a_sy_s = 0$ for any $(a_1, \dots, a_s) \in K$.

Let us consider a random tuple $(r, a_1, \dots, a_s) \in_U G \times \mathbb{Z}_d^s$. ρ is the probability over this random tuple that $f(dr + a_1x_1 + \cdots + a_sx_s)$ equals $a_1y_1 + \cdots + a_sy_s$. An alternative way to pick the a_i's uniformly at random consists in first picking a coset of $\mathbb{Z}_d^s/K$ uniformly at random and then picking a tuple of K uniformly at random. Since all cosets have the same probability to be picked, we deduce the existence of a coset $(a_1, \dots, a_s) + K$ such that

$$\Pr_{(r,b_1,\dots,b_s)\in_U G\times K}[f(dr+(a_1+b_1)x_1+\cdots+(a_s+b_s)x_s) = (a_1+b_1)y_1+\cdots+(a_s+b_s)y_s] \geq \rho.$$

Note that $a_1x_1 + \cdots + a_sx_s$ is now a constant x and that $dr + b_1x_1 + \cdots + b_sx_s$ can be written dr' where r' is uniformly sampled in G and independent from $b_1, \dots, b_s$. Hence, there exists an element $r' \in G$ such that

$$\Pr_{(b_1,\dots,b_s)\in_U K}[f(dr' + x) = (a_1 + b_1)y_1 + \cdots + (a_s + b_s)y_s] \geq \rho.$$

So we have

$$\Pr_{(b_1,\dots,b_s)\in_U K}[b_1y_1 + \cdots + b_sy_s = \text{constant}] > \frac{1}{p}.$$

Since $(b_1, \dots, b_s) \mapsto b_1y_1+\cdots+b_sy_s$ is a group homomorphism from K to a subgroup of H it must be a balanced function. Its image is either a subgroup of size at least p or the trivial subgroup $\{0\}$. Hence, the probability must actually be 1 and we have $b_1y_1 + \cdots + b_sy_s = 0$ for all $(b_1, \dots, b_s) \in K$.

To prove the second assertion, we first note that the unique homomorphism g interpolating in S is such that $g(x) := a_1y_1 + \cdots + a_sy_s$ and is uniquely defined by $x = dr + a_1x_1 + \cdots + a_sx_s$. Moreover, since the mapping defined in (4.2) is balanced by Lemma 4.1.9, we deduce that

$$\rho = \Pr_{x\in_U G}[f(x) = g(x)],$$

which concludes the proof. □

In the following lemma, we show that a similar result holds, even if we restrict on the tuples $(r, a_1, \dots a_s)$ whose combinations generate an arbitrary fixed value in G.

Lemma 4.1.12. *Let G, H, S, d, and p as in Lemma 4.1.11. We assume that S does not interpolate in any group homomorphism and define for any $x \in G$ the set*

$$U_x := \{(r, a_1, \dots, a_s) \in G \times \mathbb{Z}_d^s \mid dr + a_1x_1 + \cdots + a_sx_s = x\}.$$

Then, for any $x \in G$ and any $y \in H$, we have

$$\Pr_{(r,a_1,\dots,a_s)\in_U U_x}[a_1y_1 + \cdots + a_sy_s = y] = 0 \text{ or } \delta,$$

for a constant $\delta \leq 1/p$. Therefore, for any $x \in G$ and any function $f : G \to H$, we have

$$\Pr_{(r,a_1,\dots,a_s)\in_U U_x}[f(x) = a_1y_1 + \cdots + a_sy_s] \leq \frac{1}{p}.$$

Proof. Let K be defined as in the proof of Lemma 4.1.11. By Lemma 4.1.5, the image of $g : (b_1, \dots, b_s) \mapsto b_1y_1 + \cdots + b_sy_s$ defined on K is a subgroup of order greater or equal to p. Moreover, by Lemma 4.1.8, g is balanced on its image, which shows that

$$\Pr_{(b_1,\dots,b_s)\in_U K}[b_1y_1 + \cdots + b_sy_s = y] = 0 \text{ or } \delta$$

for any $y \in H$, where $\delta = 1/|\mathrm{Im}(g)|$. Let x be an arbitrary element of G. We can deduce that for any fixed tuple $(r', b_1', \dots, b_s') \in U_x$, we also have

$$\Pr_{(b_1,\dots,b_s)\in_U K}[(b_1 + b_1')y_1 + \cdots + (b_s + b_s')y_s = y] = 0 \text{ or } \delta$$

for any $y \in H$. This is equivalent to

$$\Pr_{(a_1,\dots,a_s)\in_U V_x}[a_1y_1 + \cdots + a_sy_s = y] = 0 \text{ or } \delta,$$

for any $y \in H$, where $V_x := \{(a_1, \dots, a_s) \mid \exists r \in G \text{ s.t. } (r, a_1, \dots, a_s) \in U_x\}$. Here, we remark that for any tuple $(a_1, \dots, a_s) \in V_x$, there exists the same number of elements $r \in G$ such that $(r, a_1, \dots, a_s) \in U_x$. Namely, this number is equal to the cardinality of the kernel of the homomorphism $r \mapsto dr$ defined on G, which is equal to $\#G/\#dG$. From this, we finally deduce that

$$\Pr_{(r,a_1,\dots,a_s)\in_U U_x}[a_1y_1 + \cdots + a_sy_s = y] = \Pr_{(a_1,\dots,a_s)\in_U V_x}[a_1y_1 + \cdots + a_sy_s = y] = 0 \text{ or } \delta,$$

for any $y \in H$. □

Homomorphism Computability

The next result says that the function f can be used to compute the unique homomorphism g, i.e., to solve the 1-S-GHI problem.

Lemma 4.1.13. *Given two finite Abelian groups G and H, and a set of s points $S = \{(x_i, y_i) \mid i = 1, \ldots, s\}$, we assume that $x_1, \ldots, x_s$ H-generate G. We assume that we are given the order d of H whose smallest prime factor is p and that we can sample elements in G with respect to a uniform distribution. We assume that we have an oracle function $f : G \longrightarrow H$ such that*

$$\Pr_{(r,a_1,\ldots,a_s)\in_U G\times\mathbb{Z}_d^s}[f(dr + a_1x_1 + \cdots + a_sx_s) = a_1y_1 + \cdots + a_sy_s] = \frac{1}{p} + \theta$$

with $\theta > 0$. Let $\varepsilon > 0$ be arbitrarily small. There exists a group homomorphism g which interpolates S and which is computable within

$$\frac{16}{\theta^2}\log\left(\frac{p}{\varepsilon}\right)(1 + o(1))$$

oracle calls to f with an error probability less or equal to $\varepsilon(1 + o(1))$ when $\theta \to 0$.

Proof. Due to Lemma 4.1.11, the homomorphism g exists and we have

$$\Pr_{x\in_U G}[f(x) = g(x)] = \frac{1}{p} + \theta. \tag{4.4}$$

We use the same techniques which are used in linear cryptanalysis and consider Algorithm 4.1. We choose the parameters related to this algorithm as follows

$$n = \frac{8}{\theta^2}\left(\frac{1}{p} + \theta\right)\log\left(\frac{p}{\varepsilon}\right) \qquad \text{and} \qquad \tau = \frac{1}{p} + \frac{\theta}{2}$$

and we first estimate the error probability of the acceptance test given in this algorithm. We consider the two following types of error

$$\varepsilon_1 = \Pr_{x\in_U G}[c \le \tau n \mid y = g(x)] \quad \text{and} \quad \varepsilon_2 = \Pr_{x\in_U G}[c > \tau n \mid y \ne g(x)].$$

We will now estimate these two values and show that they are negligible. If $y \ne g(x)$, then the test $(\mathbf{T})$ works with probability $t_2 \le 1/p$ due to Lemma 4.1.11. We also notice that if $y = g(x)$, the probability that the test works is $\rho := \frac{1}{p} + \theta$ by (4.4), since $dr + a_1x_1 + \cdots + a_sx_s + ax$ is uniformly distributed in G by Lemma 4.1.9. Therefore, we have

$$\varepsilon_1 = \sum_{i=0}^{\lfloor\tau n\rfloor}\binom{n}{i}\rho^i(1-\rho)^{n-i} \quad \text{and} \quad \varepsilon_2 = \sum_{i=\lfloor\tau n\rfloor+1}^{n}\binom{n}{i}t_2^i(1-t_2)^{n-i}. \tag{4.5}$$

Algorithm 4.1. Algorithm computing the group homomorphism g

```
Parameters: $n \in \mathbb{N}$ and $\tau \in [0,1]$
Input: $x \in G$
 1: repeat
 2:    pick $r \in_U G, a_1, \dots, a_s \in_U \mathbb{Z}_d$ uniformly at random
 3:    $y = f(x + dr + a_1x_1 + \cdots + a_sx_s) - a_1y_1 - \cdots - a_sy_s$
 4:    $c = 0$
 5:    for $i = 1$ to $n$ do
 6:       pick $r \in_U G, a_1, \dots, a_s, a \in_U \mathbb{Z}_d$ uniformly at random
 7:       if $f(dr + a_1x_1 + \cdots + a_sx_s + ax) = a_1y_1 + \cdots + a_sy_s + ay$ (T) then
 8:          $c = c + 1$
 9:       end if
10:    end for
11: until $c > \tau n$
Output: $y$
```

Let
$$\varphi(t) := \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}$$
be the probability density function of the normal distribution and
$$\Phi(x) := \int_{-\infty}^{x} \varphi(t)dt$$
the corresponding cumulative distribution function. Applying the Central Limit Theorem on both equations given in (4.5) we obtain[1]
$$\varepsilon_1 = \Phi\left(\frac{\lfloor \tau n \rfloor - \rho n}{\sqrt{n\rho(1-\rho)}}\right) + o(1) \tag{4.6}$$
and
$$\varepsilon_2 = 1 - \Phi\left(\frac{\lfloor \tau n \rfloor + 1 - nt_2}{\sqrt{nt_2(1-t_2)}}\right) + o(1), \tag{4.7}$$
when $\theta \to 0$ (i.e., $n \to \infty$). Since Φ is a monotone function and $\Phi(-x) = 1 - \Phi(x)$ for any $x \in \mathbb{R}$, we get the inequalities
$$\varepsilon_1 \leq \Phi\left(\sqrt{n}\frac{\tau - \rho}{\sqrt{\rho(1-\rho)}}\right) + o(1) \tag{4.8}$$

[1]Using the inequality of Slud [139] we can show that ε_1 is a lower bound of the term $\Phi(\ )$ in (4.6) while ε_2 is an upper bound of the term $1 - \Phi(\ )$ in (4.7).

and

$$\varepsilon_2 \leq \Phi\left(-\sqrt{n}\frac{\tau - t_2}{\sqrt{t_2(1-t_2)}}\right) + o(1).$$

Let h be a real function defined by $h(t) = (\tau - t)/(\sqrt{t(1-t)})$. By looking at the logarithmic derivative of the function h

$$\frac{d}{dt}\log(h(t)) = \frac{-1}{\tau - t} + \frac{t - \frac{1}{2}}{t(1-t)}$$

on the interval $[0, p^{-1}]$, we notice that this one is negative. Hence, we deduce that

$$\varepsilon_2 \leq \Phi\left(-\sqrt{n}\frac{\tau - p^{-1}}{\sqrt{p^{-1}(1-p^{-1})}}\right) + o(1) = \Phi\left(-\sqrt{n}\frac{\theta}{2\sqrt{p^{-1}(1-p^{-1})}}\right) + o(1).$$

For $x > 0$, it can be shown (see Lemma 2 in Chapter VII, Section 1 of Feller [57]) that

$$\Phi(-x) < \frac{\varphi(x)}{x}.$$

Hence, when x is large enough ($x > 1$), we even have

$$\Phi(-x) < \varphi(x). \tag{4.9}$$

From this, it follows that

$$\varepsilon_2 \leq \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{n\theta^2}{8(p^{-1}(1-p^{-1}))}} + o(1), \tag{4.10}$$

when ε is small enough, i.e., when $\sqrt{n} \cdot \theta$ is large enough. Now, we substitute the expression of n in (4.10) and we obtain

$$\varepsilon_2 \leq \frac{1}{\sqrt{2\pi}}\left(\frac{\varepsilon}{p}\right)^{\frac{p+p^2\theta}{p-1}} + o(1).$$

Since

$$\frac{p + p^2\theta}{p-1} \geq 1 \quad \text{and} \quad \frac{\varepsilon}{p} < 1$$

when ε is small, we finally get

$$\varepsilon_2 \leq \frac{\varepsilon}{p\sqrt{2\pi}} + o(1) \leq \rho\frac{\varepsilon}{2} + o(1). \tag{4.11}$$

In a similar way, we show that $\varepsilon_1 \leq \varepsilon/2 + o(1)$. From (4.8) and by applying the inequality (4.9) when ε is small enough, we have

$$\varepsilon_1 \leq \varphi\left(-\frac{\sqrt{n}\cdot\theta}{2\sqrt{\rho(1-\rho)}}\right) + o(1) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{n\theta^2}{8\rho(1-\rho)}} + o(1),$$

which leads to

$$\varepsilon_1 \leq \frac{1}{\sqrt{2\pi}}\left(\frac{\varepsilon}{p}\right)^{\frac{1}{1-\rho}} + o(1) \leq \frac{\varepsilon}{\sqrt{2\pi}\cdot p} + o(1) \leq \frac{\varepsilon}{2} + o(1).$$

It remains to compute the complexity and the error probability of the algorithm. At first, we observe that the probability for the "**until**" condition to be not satisfied is given by

$$\alpha := \Pr_{x \in_U G}[c \leq \tau n] = \rho\varepsilon_1 + (1-\rho)(1-\varepsilon_2).$$

If ε is small enough ($\varepsilon \leq 1/2$), we have

$$\alpha = 1 - \rho + \rho\varepsilon_1 - \varepsilon_2(1-\rho) \leq 1 - \rho + \rho\frac{\varepsilon}{2} + o(1) \leq 1 - \frac{3\rho}{4} + o(1).$$

Moreover, the expected number of iterations is equal to

$$\sum_{i=1}^{\infty} i\alpha^{i-1}(1-\alpha) = \frac{1}{1-\alpha} \leq \frac{4}{3\rho} + o(1).$$

Hence, the expected complexity in terms of oracle calls to f admits the following upper bound

$$\frac{4(n+1)}{3\rho}(1+o(1)) \leq \frac{2n}{\rho}(1+o(1)) = \frac{16}{\theta^2}\log\left(\frac{p}{\varepsilon}\right)(1+o(1)).$$

Using (4.11), the probability of error is given by

$$\sum_{i=1}^{\infty} \alpha^{i-1}(1-\rho)\varepsilon_2 \leq (1-\rho)\varepsilon_2\left(\frac{2}{\rho} + o(1)\right) \leq \varepsilon\left(1 + o(1)\right),$$

which concludes the proof. □

4.1.4 Problem Amplifications and Reductions

In this subsection, we consider GHID and GHI problems with a set of points $S = \{(g_1, h_1), \ldots, (g_s, h_s)\}$ such that $g_1, \ldots, g_s$ H-generate G and S interpolates in a group homomorphism f. Note that this homomorphism is unique by Lemma 4.1.3.

Amplification of the GHI and GHID Problems

We show here how GHI and GHID problem solvers can be amplified so that both problems can be perfectly solved under certain conditions.

GHID Amplification. Here, we need to assume that H is cyclic and has a prime order. For any integer n and any set $\{(x_1, y_1), \ldots (x_n, y_n)\} \in (G \times H)^n$, we explain how one can correctly decide whether $f(x_i) = y_i$ for all $i = 1, \ldots, n$ or not with an overwhelming probability, using an n-S-GHID distinguisher $\mathcal{D}$ with an advantage $\varepsilon > 0$. For this purpose, the main task consists in generating an n-S-GHID instance from $\{(x_1, y_1), \ldots (x_n, y_n)\}$ such that this one is of type T_0 if $f(x_i) = y_i$ for all $i = 1, \ldots, n$ and of type T_1 otherwise. We generate such an instance by picking $r_i \in_U G$, $a_{i,j} \in_U \mathbb{Z}_d$ uniformly at random for $i = 1, \ldots, n$, $j = 1, \ldots, s + n$ and by setting

$$x'_i := dr_i + a_{i,1}g_1 + \cdots + a_{i,s}g_s + a_{i,s+1}x_1 + \cdots + a_{i,s+n}x_n$$

and

$$y'_i := a_{i,1}h_1 + \cdots + a_{i,s}h_s + a_{i,s+1}y_1 + \cdots + a_{i,s+n}y_n.$$

Note that if $f(x_i) = y_i$ for $i = 1, \ldots, n$, then the above instance is of type T_0, since x'_i is uniformly distributed for $i = 1, \ldots, n$ by Lemma 4.1.9 and $f(x'_i) = y'_i$ for $i = 1, \ldots, n$ due to the homomorphic property of f. On the other hand, if $f(x_j) \neq y_j$ for at least one j, the above instance is of type T_1 provided that H is a cyclic group of a prime order. Namely, we are ensured that $a_{i,s+j}(f(x_j) - y_j)$ can take any value of H in this case. Therefore, the value y'_i is independent from x'_i for any i, which shows that the instance defined by x'_i and y'_i for $i = 1, \ldots, n$ is of type T_1. As in the article of Boneh [19] about the decision Diffie-Hellman problem, we can correctly determine whether $f(x_i) = y_i$ for $i = 1, \ldots, n$ or not with an overwhelming probability in polynomial time. We summarize this technique below. We need to consider two experiments. The first one consists in generating k instances as above and feeding them to $\mathcal{D}$. Let w_1 be the random variable counting the number of times $\mathcal{D}$ decides that the instance was of type T_0. In a second experiment, we generate k random instances of type T_1 and feed them to $\mathcal{D}$. Let w_2 be the number of "type T_0" answers by $\mathcal{D}$. One can decide whether $f(x_i) = y_i$ for $i = 1, \ldots, n$, by testing whether $|w_1 - w_2|$ is greater than a given threshold. An adequate choice of the threshold value allows to succeed with an overwhelming success probability.

GHIP Amplification. We can amplify the success probability of a 1-S-GHIP solver $\mathcal{A}$ to an overwhelming success probability by directly applying results of Lemma 4.1.13. This amplification technique works in polynomial time provided that $\mathcal{A}$ has a success probability greater than $1/p + \theta$, where θ is non-negligible. Since an n-S-GHIP instance consists in n independent 1-S-GHIP instances, one can assume that an n-S-GHIP solver solves every component with the same probability. Hence, amplification techniques work for n-S-GHIP provided that the solver succeeds with a probability greater than $1/p^n + \theta$.

These amplification results show that GHID and GHI problems can be related to their non-probabilistic variants if certain conditions are fullfiled. When H is cyclic of prime order, GHID distinguishers can be used to determine whether any set $\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (G \times H)^n$ interpolates in f. When GHIP solvers are good enough, one can evaluate f on any given points $(x_1, \ldots, y_n) \in G^n$.

We are now in position to show that these non-probabilistic versions can be interpreted in terms of languages so that they can be analyzed using tools of the theory of complexity.

Non-Probabilistic GHI and GHID Problems are in $\mathcal{NP} \cap$ co-$\mathcal{NP}$

We associate a language L to the 1-S-GHID problem which is composed of any instance $(x, y) \in G \times H$ such that $f(x) = y$. We show that L is in $\mathcal{NP}$. In order to decide whether a given pair (x, y) lies in L in polynomial time, it suffices to provide a witness containing the coefficients $r \in G$, $a_1, \ldots, a_s \in \mathbb{Z}_d$ such that

$$x = dr + a_1 g_1 + \cdots + a_s g_s. \tag{4.12}$$

Namely, by the assertion 5 of Lemma 4.1.3, such a representation always exists and checking the validity of the following equation

$$y = a_1 h_1 + \cdots + a_s h_s$$

directly provides the answer. Note that this is the case also for instances $(x, y) \notin L$. As a consequence of this, we deduce that the non-probabilistic variant of the 1-S-GHID problem lies in $\mathcal{NP} \cap$ co-$\mathcal{NP}$. This result generalizes in a straightforward way to the n-S-GHID problems with any $n \in \mathbb{N}$.

To proceed in a similar way for the 1-S-GHI problem, we first define some closely related decisional problems. Assume that the group H is equipped with a totally ordered relation denoted $\preccurlyeq$ and consider the language

$$L_h := \{x \in G \mid f(x) \preccurlyeq h\}$$

for any $h \in H$. We note that if one is able to decide whether $x \in L_h$ for any $x \in G$ and $h \in H$, then one can easily compute $f(x)$ by dichotomy. As above, we can show that coefficients $r \in G$, $a_1, \ldots, a_s \in \mathbb{Z}_d$ satisfying (4.12) provides a witness for x with respect to L_h for any $h \in H$. This shows that non-probabilistic GHI problems can be easily represented with languages lying in $\mathcal{NP} \cap$ co-$\mathcal{NP}$.

Remark 4.1.14. In the above discussion, we did not specify to which parameter, the term "polynomial" refers to. To formally specify this, we can consider a family of problems indexed by a parameter k and whose respective instances are of size k. So, we should consider such a family of n-S_k-GHID (resp. GHI) problems with some groups G_k and H_k satisfying the above required properties.

Reductions to MSR Problem

Note that finding the coefficients $a_1, \ldots, a_s$ in (4.12) suffices to solve the 1-S-GHI and 1-S-GHID problems with the input x. Namely, these coefficients allow to compute $a_1 h_1 + \cdots + a_s h_s$ which readily gives the answer of the 1-S-GHI and 1-S-GHID problems. Since the coefficients $a_1, \ldots, a_s$ can be found by solving a (d, S)-MSR problem on x, the n-S-GHI and n-S-GHID problems reduce to (d, S)-MSR problem.

4.2 Interactive Proof Protocols

In this section, we develop some interactive proof protocols related to the interpolation of group homomorphisms. They serve as a preparation towards the presentation of our undeniable signature scheme MOVA. Namely, the confirmation and the denial protocols of MOVA are directly based on them.

Throughout this section, we use a commitment scheme denoted Commit with corresponding opening algorithm Open. For the notation related to the commitment scheme and further technical details, we refer to Subsection 2.2.5.

4.2.1 Interactive Proof for the GHID Problem

Let G, H, and $S = \{(g_1, e_1), \ldots, (g_s, e_s)\}$ be parameters of a GHID problem. Let d be the order of H and $\ell \in \mathbb{N}$ be a security parameter. We present here an interactive proof in which a prover wants to convince a verifier that S interpolates in a group homomorphism $f : G \longrightarrow H$ known by himself. This provides a positive answer to the special S-GHID problem. This protocol is denoted $\text{GHIproof}_\ell(S)$ and is depicted in Figure 4.1.

We remark that this protocol makes use of a commitment scheme. This one is crucial to achieve zero-knowledge, since it allows the prover to disclose the answers v_i's after having checked that the verifier generated the challenges correctly. Otherwise, a malicious verifier could use the prover as an oracle to evaluate the function f on any element of G. Note that the parameter ℓ corresponds to the number of challenges sent by the verifier. So, ℓ plays a direct role in the security of GHIproof. This protocol can actually be seen as the parallelization of ℓ simpler protocols with one challenge u. Such a version with one challenge is depicted in Figure 4.2.

```
**GHIproof**$_\ell(S)$
**Parameters:** $G, H, d$
**Input:** $\ell$, $S = \{(g_1, e_1), \dots, (g_s, e_s)\} \subseteq G \times H$
1: The verifier picks $r_i \in_U G$ and $a_{i,j} \in_U \mathbb{Z}_d$ uniformly at random for
   $i = 1, \dots, \ell$ and $j = 1, \dots, s$. He computes $u_i = dr_i + a_{i,1}g_1 + \dots + a_{i,s}g_s$
   and $w_i = a_{i,1}e_1 + \dots + a_{i,s}e_s$ for $i = 1, \dots, \ell$. He sends $u_1, \dots, u_\ell$ to the
   prover.
2: The prover checks whether $f(g_i) = e_i$ for $i = 1, \dots, \ell$. If it is not
   the case, he aborts the protocol. Then, he computes $v_i = f(u_i)$ for
   $i = 1, \dots, \ell$ and the commitment $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(v_1, \dots, v_\ell)$. He
   sends the committed value com to the verifier.
3: The verifier sends all $r_i$'s and $a_{i,j}$'s to the prover.
4: The prover checks that the $u_i$'s were computed correctly by verifying
   that $u_i = dr_i + a_{i,1}g_1 + \dots + a_{i,s}g_s$ holds for $i = 1, \dots, \ell$. If not, he aborts
   the protocol. He then opens his commitment by sending the values $v_i$'s
   and dec.
5: The verifier checks that $v_i = w_i$ holds for $i = 1, \dots, \ell$ and that the
   commitment is opened correctly, i.e., $1 \leftarrow \mathsf{Open}(v_1, \dots, v_\ell, \mathsf{com}, \mathsf{dec})$. If
   this is the case, the verifier accepts the proof. Otherwise, he rejects it.
```

Figure 4.1: Interactive proof for the GHID problem

Security results related to GHIproof are presented here.

Theorem 4.2.1. *Let G, H be some Abelian groups and $\ell \in \mathbb{N}$. We denote by d the order of H and p the smallest prime factor of d. Assume that we are given a set of points $S = \{(g_1, e_1), \dots, (g_s, e_s)\} \subseteq G \times H$ such that the elements $g_1, \dots, g_s$ H-generate the group G. We consider the* GHIproof$_\ell(S)$ *protocol.*

1. *The* GHIproof *protocol is complete.*

2. *Assuming that* Commit *is perfectly hiding, the* GHIproof *protocol is perfect black-box zero-knowledge against any verifier. Moreover, if* Commit *is a perfectly hiding trapdoor commitment scheme, the* GHIproof *protocol is perfect black-box straight-line zero-knowledge against any verifier who has the secret key $\mathcal{K}_\mathrm{s}^\mathbf{V}$ associated to* Commit.

3. *Assuming that* Commit *is a perfectly hiding trapdoor commitment scheme with associated secret key $\mathcal{K}_\mathrm{s}^\mathbf{V}$ of the (honest) verifier, the* GHIproof *protocol is perfect non-transferable.*

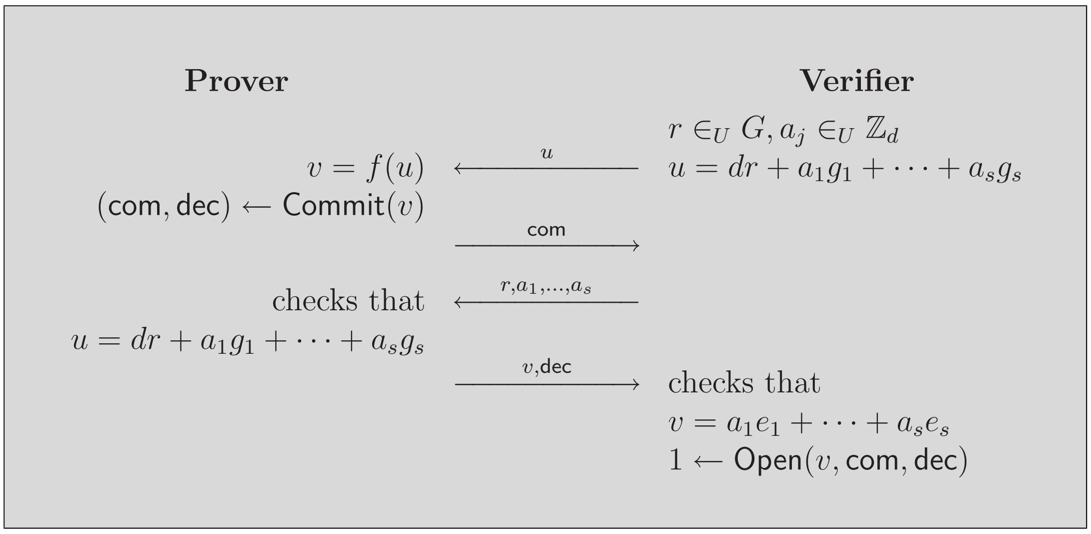
Prover
Verifier
$r \in_U G, a_j \in_U \mathbb{Z}_d$
$u = dr + a_1 g_1 + \cdots + a_s g_s$
u
$v = f(u)$
$(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(v)$
com
$r, a_1, \ldots, a_s$
checks that
$u = dr + a_1 g_1 + \cdots + a_s g_s$
v, dec
checks that
$v = a_1 e_1 + \cdots + a_s e_s$
$1 \leftarrow \mathsf{Open}(v, \mathsf{com}, \mathsf{dec})$

Figure 4.2: Simplified interactive proof for the GHID problem

4. *The* GHIproof *protocol is sound: from any cheating prover* $\mathbf{P}^*$ *who passes the protocol on an invalid set* S *(i.e., with no interpolating homomorphism) with a probability* $\mathsf{Succ}^{\mathsf{sd\text{-}GHI}}_{\mathbf{P}^*} = \varepsilon$ *and an expert group knowledge of* G*, we can construct an algorithm* $\mathcal{B}$ *which finds a collision on the commitment scheme* Commit *with a probability* $\mathsf{Succ}^{\mathsf{com\text{-}bnd}}_{\mathcal{B}} \geq \varepsilon(\varepsilon - p^{-\ell})$ *by rewinding* $\mathbf{P}^*$ *once.*

5. *("Proof of knowledge") For any* $\theta > 0$*, assuming that the protocol succeeds with probability greater than* $(p^{-1}+\theta)^{\ell}$ *with an honest verifier and that* Commit *is extractable, for any* $\varepsilon > 0$ *there exists an extractor with a time complexity factor* $\mathcal{O}(\log(1/\varepsilon))$ *which can compute an interpolating group homomorphism from the (possibly cheating) prover using the secret key of* Commit *with probability at least* $1 - \varepsilon$*.*

Proof.

1. The completeness follows by the homomorphic property of f. Namely,

$$f(dr_i + a_{i,1}g_1 + \cdots + a_{i,s}g_s) = a_{i,1}e_1 + \cdots + a_{i,s}e_s \quad \text{for } i = 1, \ldots \ell,$$

holds when $f(g_j) = e_j$ for $j = 1, \ldots, s$.

2. First, we handle the case of the trapdoor commitment with an associated pair of key $(\mathcal{K}^{\mathbf{V}}_{\mathrm{p}}, \mathcal{K}^{\mathbf{V}}_{\mathrm{s}})$ owned by the cheating verifier $\mathbf{V}^*$. We construct a simulator $\mathcal{B}$ with input $\mathcal{K}^{\mathbf{V}}_{\mathrm{s}}$ which interacts with $\mathbf{V}^*$. The simulator $\mathcal{B}$ first receives the challenge

$u := (u_1, \dots, u_\ell)$ from $\mathbf{V}^*$. Otherwise, the simulator answers the flag abort. Then, $\mathcal{B}$ picks a tuple $v' := (v'_1, \dots, v'_\ell) \in_U H^\ell$ uniformly at random, computes

$$(\mathsf{com}', \mathsf{dec}') \leftarrow \mathsf{Commit}(v', \mathcal{K}_\mathrm{p}^\mathbf{V})$$

and sends com' to $\mathbf{V}^*$. Then, the verifier sends the values r_i's and $a_{i,j}$'s to $\mathcal{B}$. The simulator checks whether the received values satisfy $u_i = dr_i + \sum_{j=1}^s a_{i,j}g_j$ for $i = 1, \dots, \ell$. If it is not the case the simulator answers abort. If it is the case, the simulator deduces the right tuple v and computes

$$\mathsf{dec} \leftarrow \mathsf{Collide}(v', v, \mathsf{com}', \mathsf{dec}', \mathcal{K}_\mathrm{s}^\mathbf{V})$$

when $v \neq v'$. The simulator then sends v, dec (or dec' if $v = v'$) to open the commitment com'. Note that in this case, the transcript of this interaction is

$$(u, \mathsf{com}', r_i\text{'s}, a_{i,j}\text{'s}, v, \mathsf{dec}).$$

Since the commitment is perfectly hiding, the transcript corresponding to the above interaction between the simulator and $\mathbf{V}^*$ has exactly the same distribution as a real transcript produced between an honest prover and the verifier $\mathbf{V}^*$.

We now consider Commit without trapdoor. The main difficulty in this situation consists in solving the challenges while we cannot open the commitment on any value of our choice. This will be solved by rewinding $\mathbf{V}^*$ with the same random tape. At the beginning, $\mathcal{B}$ receives the challenge $u := (u_1, \dots, u_\ell)$ from $\mathbf{V}^*$ and aborts if it is not the case. He picks a committed value com' uniformly at random and sends it to $\mathbf{V}^*$. The verifier answers the coefficients r_i's and $a_{i,j}$'s. If $u_i = dr_i + \sum_{j=1}^s a_{i,j}g_j$ for $i = 1, \dots, \ell$, the simulator stops and rewinds $\mathbf{V}^*$ with the same random tape. Otherwise, $\mathcal{B}$ outputs the transcript $(u, \mathsf{com}', r_i\text{'s}, a_{i,j}\text{'s}, \mathsf{abort})$. After the rewinding, $\mathbf{V}^*$ sends again the same challenge u. The simulator now computes the answers $v_i := \sum_{j=1}^s a_{i,j}e_j$ and computes

$$(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(v).$$

$\mathcal{B}$ sends com to the verifier. This one answers some values r_i's and $a_{i,j}$'s which generate u. Otherwise, the simulator aborts and outputs $(u, \mathsf{com}, r_i\text{'s}, a_{i,j}\text{'s}, \mathsf{abort})$ as transcript. Finally, the simulator would send dec and v to the verifier and outputs the transcript

$$(u, \mathsf{com}, r_i\text{'s}, a_{i,j}\text{'s}, v, \mathsf{dec}).$$

As above, we see that except when the simulator stops the simulation and rewinds $\mathbf{V}^*$, the interaction is identical as with an honest prover. Since Commit is perfectly hiding, the first value com' has the correct distribution. Thus, the simulation works and the transcript generated by the simulator has the same distribution as the one produced by an honest signer and the verifier $\mathbf{V}^*$.

3. The simulator $\mathcal{B}$ has to simulate an honest prover interacting with a verifier $\tilde{\mathbf{V}}$. We recall that the simulator is given the secret key of the honest verifier (and of Commit). If the set S does not interpolate in a group homomorphism ($\mathcal{B}$ knows this fact thanks an information bit by definition of the non-transferability), the simulator simply answers abort in the second step of the protocol. Otherwise, the simulator follows exactly the same simulation as for proving straight-line zero-knowledge. So, the simulator has exactly the same behaviour as an honest prover. Hence, the non-transferability is perfect.

4. By Lemma 4.1.11, if the set S does not admit any interpolating homomorphism, it is impossible to find any procedure to deduce $v_i = a_{i,1}e_1 + \cdots + a_{i,s}e_s$ from the challenge $u_i = dr + a_{i,1}g_1 + \cdots + a_{i,s}g_s$ with probability greater than $1/p$, for any $i = 1, \ldots, \ell$. Since the challenges are generated independently, no prover is able to find the correct $v = (v_1, \ldots, v_\ell)$ from the challenge $u = (u_1, \ldots, u_\ell)$ with probability greater than $p^{-\ell}$. Below, we show that a (cheating) prover $\mathbf{P}^*$ must break the binding property of Commit with non-zero probability in order to succeed in the protocol with a probability $\varepsilon > p^{-\ell}$.

We construct below a simulator $\mathcal{B}$ which interacts with $\mathbf{P}^*$ and plays the role of an honest verifier. The simulator will launch 2 protocol runs sequentially with $\mathbf{P}^*$ in such a way that a collision on Commit can be found with a certain probability. For this, he rewinds the prover and follows a behaviour which depends on the first run. We emphasize here that this is allowed as long as the prover receives messages which are correctly distributed in both runs of the protocol. So, if we look at both runs separately, the prover cannot see any difference from an interaction with an honest verifier.

The simulator $\mathcal{B}$ picks some coefficients $a_{i,j}$'s and r_i's uniformly at random and computes the challenges $u_i := dr_i + \sum_{j=1}^{s} a_{i,j}g_j$ for $i = 1, \ldots, \ell$. He then sends u to $\mathbf{P}^*$. This one answers a committed value com. The simulator releases the coefficients $a_{i,j}$'s and r_i's to $\mathbf{P}^*$. At the end, $\mathbf{P}^*$ succeeds if he opens the commitment correctly on the values $v_i := \sum_{j=1}^{s} a_{i,j}e_j$ for $i = 1, \ldots, \ell$. If he does not succeed, the simulator aborts. Otherwise, using his expert group knowledge of G, the simulator finds some coefficients $a^*_{i,j}$'s and r^*_i's satisfying

$$u_i = dr^*_i + \sum_{j=1}^{s} a^*_{i,j}g_j \text{ for } i = 1, \ldots, \ell,$$

and which are picked uniformly at random among all possible representations of the u_i's. Now, $\mathcal{B}$ rewinds the prover with the same random tape. The simulator sends the same challenges u_i's as before. Therefore, the prover answers the same commitment com. At this time, $\mathcal{B}$ releases the coefficients $a^*_{i,j}$'s and r^*_i's to $\mathbf{P}^*$. This one succeeds if he is able to open com on the values $v^*_i = \sum_{j=1}^{s} a^*_{i,j} e_j$ for $i = 1, \dots, \ell$. In case of success, the simulator directly finds a collision with respect to Commit if $v_i \neq v^*_i$ holds for at least one i.

It remains to compute the probability that this event occurs. At first, we note that the simulations are perfect in both runs of the protocol. Namely, if we look at both protocol runs independently, we remark that all coefficients $r_i, a_{i,j}$ are chosen uniformly at random. We now have to take into account that rewinding the prover with the same random tape and the same challenges is a restriction in the space of all possible protocol runs. So, we decompose the probability of success according to the different random tapes ϖ and challenges u. Note that once the random tape and the challenge are fixed, both protocol runs are independent.

Let A be the probability event that $\mathbf{P}^*$ succeeds in the first protocol run and $A_{\varpi,u}$ be the same event conditioned by the random tape ϖ and challenge u. Similarly, we define the same events A^* and $A^*_{\varpi,u}$ for the second protocol run. We set $\Pr[A_{\varpi,u}] = \Pr[A^*_{\varpi,u}] := \varepsilon_{\varpi,u}$. We also denote by B the event that $v \neq v^*$, where $v^* := (v^*_1, \dots, v^*_\ell)$ and $B_{\varpi,u}$ the same event conditioned by the random tape ϖ and challenge u. Note that $\Pr[\neg B_{\varpi,u}] \leq p^{-\ell}$ for any random tape ϖ and any challenge u by Lemma 4.1.12. Since $A_{\varpi,u}$ and $A^*_{\varpi,u}$ are independent, we have

$$\Pr[A_{\varpi,u} \wedge A^*_{\varpi,u}] = \varepsilon^2_{\varpi,u}.$$

From this, we deduce that the following

$$\Pr[A_{\varpi,u} \wedge A^*_{\varpi,u} \wedge B_{\varpi,u}] \geq \varepsilon^2_{\varpi,u} - \Pr[A_{\varpi,u} \wedge A^*_{\varpi,u} | \neg B_{\varpi,u}] \cdot p^{-\ell} \geq \varepsilon^2_{\varpi,u} - \Pr[A_{\varpi,u} | \neg B_{\varpi,u}] \cdot p^{-\ell}$$

holds. Here, we show that $A_{\varpi,u}$ and $\neg B_{\varpi,u}$ are independent. Let U and V be the random variables corresponding to the values u and v respectively. By Lemma 4.1.12, for any u and v, $\Pr[V = v | U = u] = 0$ or δ, for a constant $\delta \leq p^{-\ell}$. Let $\mathcal{V}$ be the support of V. By decomposing the events $A_{\varpi,u}$ and $\neg B_{\varpi,u}$ with respect to the values taken by V, we obtain

$$\Pr[A_{\varpi,u}] := \sum_{v \in \mathcal{V}} \Pr[A_{\varpi,u} | V = v] \cdot \delta$$

and

$$\Pr[\neg B_{\varpi,u}] := \sum_{v \in \mathcal{V}} \Pr[V = v | U = u]^2 = \delta.$$

Then,

$$\Pr[A_{\varpi,u} \wedge \neg B_{\varpi,u}] = \sum_{v \in \mathcal{V}} \Pr[A_{\varpi,u} | V = v] \cdot \Pr[V = v | U = u]^2 = \Pr[A_{\varpi,u}] \cdot \Pr[\neg B_{\varpi,u}]$$

shows the independence of $A_{\varpi,u}$ and $\neg B_{\varpi,u}$, so that $\Pr[A_{\varpi,u}|\neg B_{\varpi,u}] = \varepsilon_{\varpi,u}$. Therefore, we obtain

$$\mathsf{Succ}_{\mathcal{B}}^{\mathsf{com\text{-}bnd}} \geq \sum_{\varpi,u} q_{\varpi,u} \cdot \varepsilon^2_{\varpi,u} - \sum_{\varpi,u} q_{\varpi,u} \cdot \varepsilon_{\varpi,u} \cdot p^{-\ell},$$

where $q_{\varpi,u}$ is the probability that the protocol runs with the random tape ϖ and challenge u. Since the function $x \mapsto x^2$ defined on $\mathbb{R}$ is convex, we can apply Jensen's inequality on the first term. Thus, we finally get

$$\mathsf{Succ}_{\mathcal{B}}^{\mathsf{com\text{-}bnd}} \geq \varepsilon^2 - \varepsilon \cdot p^{-\ell},$$

which concludes the proof.
5. We describe an extractor $\mathcal{E}$ which knows the secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{com}}$ associated with Commit and plays with the (possibly cheating) prover $\mathbf{P}^*$. Since all challenges u_i's are independent and have the same distribution, without loss of generality, we can assume that $\mathbf{P}^*$ solves any challenge u_i with the same probability. By assumption, this probability is greater than $p^{-1} + \theta$. By Lemma 4.1.11, there exists a group homomorphism g which interpolates the set of points S. The extractor now uses $\mathbf{P}^*$ as an oracle function f which satisfies the condition of Lemma 4.1.13. Namely, $\mathcal{E}$ launches $\mathbf{P}^*$ and sends as challenge u_1 a value chosen with respect to Algorithm 4.1. The prover sends com and $\mathcal{E}$ computes $(v_1, \ldots, v_s) \leftarrow \mathsf{Extract}(\mathsf{com}, \mathcal{K}_{\mathrm{s}}^{\mathbf{com}})$ and retrieves v_1 the corresponding answer of f. Then, $\mathcal{E}$ simply follows Algorithm 4.1 and proceeds identically for subsequent values to evaluate by f. Note that $\mathcal{E}$ may need to rewind the prover for each value to evaluate by f. The complexity of this extractor follows by applying Lemma 4.1.13. $\square$

Remark 4.2.2. Note that the soundness relies on a computational assumption when the commitment is computationally binding. In this case, such an interactive proof is called a "computationally sound interactive proof" or an "interactive argument".

Remark 4.2.3. The assertion 5 shows that GHIproof possesses a kind of extraction of the computability of the function f. Since f is not characterized by a single parameter here, we cannot extract f with respect to the formal definition of a proof knowledge.

Remark 4.2.4. We note that the security properties of GHIproof strongly depends on Commit. In practice, it is impossible to have a commitment satisfying both the extractability and a perfectly hiding property. According to the desired security goals, we can adjust Commit as long as it is possible. For our thesis purposes, we will usually consider Commit as a perfectly hiding and computationally binding trapdoor commitment. If the non-transferability is not required, we can content ourselves with a perfectly hiding and computationally binding commitment.

4.2.2 Interactive Proof for the co-GHID Problem

Let G, H, and $S = \{(g_1, e_1), \dots, (g_s, e_s)\} \subseteq G \times H$ be parameters of a GHID problem, and let d be the order of H with smallest prime factor p. Let $T = \{(x_1, z_1), \dots, (x_t, z_t)\} \subseteq G \times H$ be a set of t points. We assume that we have a prover who wants to convince a verifier that for at least one k the answer to the 1-S-GHID problem with input (x_k, z_k) is negative. For this, the prover makes use of the knowledge of a group homomorphism f uniquely interpolating S. Let $\ell \in \mathbb{N}$ be a security parameter. He performs the interaction depicted in Figure 4.3 with a verifier.

```
coGHIproof_ℓ(S, T)
Parameters: $G, H, d, p$
Input: $\ell$, $S = \{(g_1, e_1), \dots, (g_s, e_s)\}$, $T = \{(x_1, z_1), \dots, (x_t, z_t)\}$
1: The verifier picks $r_{i,k} \in_U G$, $a_{i,j,k} \in_U \mathbb{Z}_d$, and $\lambda_i \in_U \mathbb{Z}_p$ uniformly at random for $i = 1, \dots, \ell$, $j = 1, \dots, s$, $k = 1, \dots, t$. He computes $u_{i,k} := dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$ and $w_{i,k} := \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all $i$ and $k$. Set $u := (u_{1,1}, \dots, u_{\ell,t})$ and $w := (w_{1,1}, \dots, w_{\ell,t})$. He sends $u$ and $w$ to the prover.
2: The prover computes $y_k = f(x_k)$ for $k = 1, \dots, t$ and verifies that $y_k \neq z_k$ for at least one $k$. Otherwise, he aborts the protocol. Then, he computes $v_{i,k} := f(u_{i,k})$ for $i = 1, \dots, \ell$, $k = 1, \dots, t$. From the equation $w_{i,k} - v_{i,k} = \lambda_i(z_k - y_k)$, he should be able to find each $\lambda_i$ if the verifier is honest, since $y_k \neq z_k$ for at least one $k$. Otherwise, he picks $\lambda_i \in_U \mathbb{Z}_p$ uniformly at random for $i = 1, \dots, \ell$. He computes $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(\lambda)$, where $\lambda := (\lambda_1, \dots, \lambda_\ell)$. The prover sends the committed value $\mathsf{com}$ to the verifier.
3: The verifier sends all $r_{i,k}$'s and $a_{i,j,k}$'s to the prover.
4: The prover checks that $u$ and $w$ were correctly computed by verifying that $u_{i,k} := dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$ and $w_{i,k} := \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all $i$ and $k$. If not, he aborts the protocol. He then opens the commitment by sending $\lambda$ and $\mathsf{dec}$.
5: The verifier checks that the prover has found the right $\lambda$ and that the commitment is correctly opened by checking $1 \leftarrow \mathsf{Open}(\lambda, \mathsf{com}, \mathsf{dec})$. If this is the case, the verifier accepts the proof. Otherwise, he rejects it.
```

Figure 4.3: Interactive proof for the co-GHID problem

Remark 4.2.5. Note that retrieving the values λ_i's in coGHIproof requires to extract discrete logarithms in H which are restricted in the set $\mathbb{Z}_p$. So, the group H

should be selected to satisfy this property. Another way to get rid of this problem is to restrict the challenge λ in a smaller interval of $\mathbb{Z}_p$.

This protocol was inspired from the denial protocol of Gennaro et al. [66]. This one can actually be seen as a special case of ours with the RSA encryption function as homomorphism.

We also notice that λ was chosen such that it can uniquely be retrieved from every nonzero values of H that can be taken by the elements $z_k - y_k$'s. Namely, this is shown by the following result.

Lemma 4.2.6. *Let H be an Abelian group of order d, and $a, b \in H$ such that $b \neq 0$. Let λ be in $\mathbb{Z}_p$ where p is the smallest prime dividing d. Then, if the equation $a = \lambda b$ has a solution in λ, then this one is unique.*

Proof. Let us first consider the subgroup $\langle b \rangle$ generated by b. If there exists a solution to the above equation, we must have $a \in \langle b \rangle$. Moreover, the coefficient λ is uniquely defined modulo $\mathrm{ord}(b)$. By definition of p, we have $\mathrm{ord}(b) \geq p$. Therefore, λ is uniquely defined in $\mathbb{Z}_p$. $\square$

As for the GHIproof protocol, we can see that this interactive proof is the parallelization of ℓ times a simpler protocol with one challenge. We depict this simplified version in Figure 4.4.

Security results related to coGHIproof are given in the following theorem.

Theorem 4.2.7. *Let G, H be some Abelian groups and $\ell \in \mathbb{N}$. We denote by d the order of H and p the smallest prime factor of d. Assume that we are given a set of points $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$ such that the elements $g_1, \ldots, g_s$ H-generate the group G and such that S interpolates in exactly one homomorphism f known by the prover. We consider the* coGHIproof$_\ell(S, T)$ *protocol, where $T = \{(x_1, z_1), \ldots, (x_t, z_t)\} \subseteq G \times H$.*

1. *The* coGHIproof *protocol is complete.*
2. *Assuming that* Commit *is perfectly hiding, the* coGHIproof *protocol is perfect black-box zero-knowledge against any verifier. Moreover, if* Commit *is a perfectly hiding trapdoor commitment scheme, the* coGHIproof *protocol is perfect black-box straight-line zero-knowledge against any verifier who has the secret key $\mathcal{K}_s^{\mathbf{V}}$ associated to* Commit.
3. *Assuming that* Commit *is a perfectly hiding trapdoor commitment scheme with associated secret key $\mathcal{K}_s^{\mathbf{V}}$ of the (honest) verifier, the* coGHIproof *protocol is perfect non-transferable.*

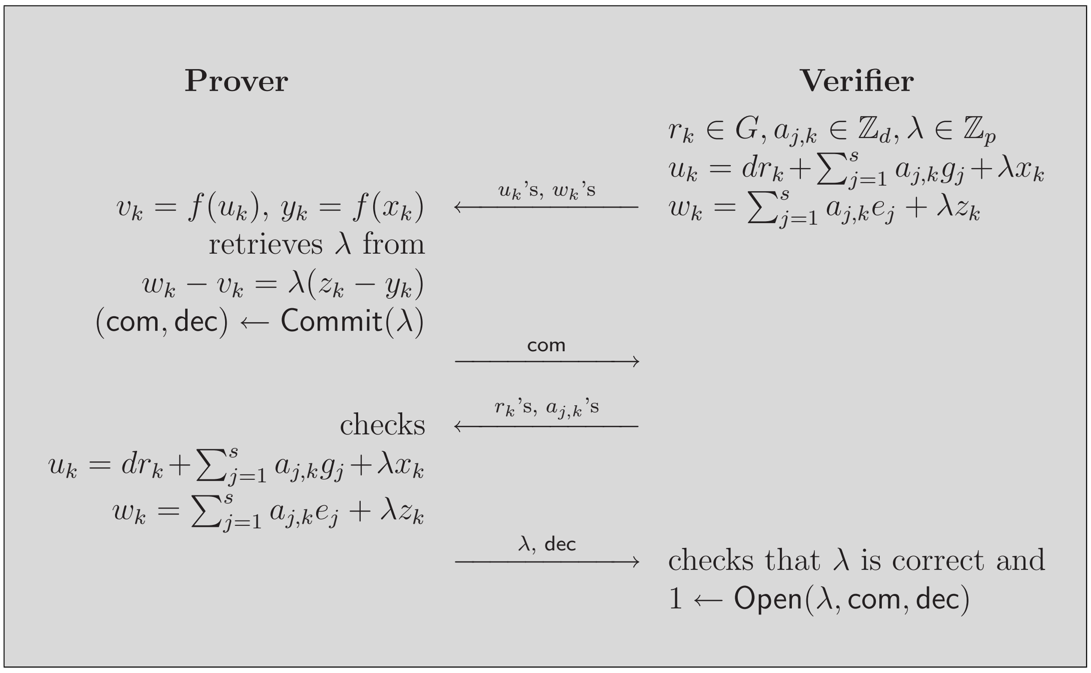
Prover
Verifier
$r_k \in G, a_{j,k} \in \mathbb{Z}_d, \lambda \in \mathbb{Z}_p$
$u_k = dr_k + \sum_{j=1}^{s} a_{j,k} g_j + \lambda x_k$
$w_k = \sum_{j=1}^{s} a_{j,k} e_j + \lambda z_k$
u_k's, w_k's
$v_k = f(u_k)$, $y_k = f(x_k)$
retrieves λ from
$w_k - v_k = \lambda(z_k - y_k)$
$(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(\lambda)$
com
r_k's, $a_{j,k}$'s
checks
$u_k = dr_k + \sum_{j=1}^{s} a_{j,k} g_j + \lambda x_k$
$w_k = \sum_{j=1}^{s} a_{j,k} e_j + \lambda z_k$
λ, dec
checks that λ is correct and
$1 \leftarrow \mathsf{Open}(\lambda, \mathsf{com}, \mathsf{dec})$

Figure 4.4: Simplified interactive proof for the co-GHID problem

4. *The* coGHIproof *protocol is sound: from any cheating prover* $\mathbf{P}^*$ *who passes the protocol on a set* T *interpolating with* S *in a group homomorphism (i.e.,* f*) with a probability* $\mathsf{Succ}_{\mathbf{P}^*}^{\mathsf{sd}\text{-coGHI}} = \varepsilon$ *and an expert group knowledge of* G*, we can construct an algorithm* $\mathcal{B}$ *which finds a collision on the commitment scheme* Commit *with a probability* $\mathsf{Succ}_{\mathcal{B}}^{\mathsf{com}\text{-}\mathsf{bnd}} \geq \varepsilon(\varepsilon - p^{-\ell})$ *by rewinding* $\mathbf{P}^*$ *once.*

Proof.

1. The completeness follows directly from the uniqueness of the values λ_i's as shown in Lemma 4.2.6, when for at least one k we have $y_k \neq z_k$. This is fullfiled by assumption.

2. The proof of zero-knowledge works in a very similar way as that of GHIproof. Namely, in the case of the trapdoor commitment scheme, the simulator $\mathcal{B}$ first answers a committed value com' picked uniformly at random. After having received the values $r_{i,k}$'s and $a_{i,j,k}$'s, $\mathcal{B}$ deduces λ_i from the equation

$$u_{i,k} - dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j = \lambda_i x_k$$

for $i = 1, \ldots, \ell$ and $k = 1, \ldots, t$ and checks that each λ_i is identical for all k. He can then checks that the $w_{i,k}$'s were correctly generated. Then, the simulator opens com'

on the right λ_i's using the secret key of Commit. As for GHIproof, the simulator outputs a transcript with identical distribution as the one produced between the same verifier and an honest prover, when Commit is perfectly hiding.

When Commit is not a trapdoor commitment, the simulator $\mathcal{B}$ needs to stop after the verifier discloses $r_{i,k}$'s and $a_{i,j,k}$'s. After rewinding the verifier, $\mathcal{B}$ is now able to commit to the right answer λ_i's. Following the same methodology as for GHIproof, the simulator finally outputs a perfectly simulated transcript.

3. The simulator behaves exactly as for the straight-line zero-knowledge, except when the set T interpolates with S in f. In this case, the simulator aborts the protocol. So, the simulator behaves identically as an honest prover. Thus, this protocol is perfect non-transferable.

4. We first remark that $u_{i,k}$ is uniformly distributed for any fixed λ_i for $i = 1, \dots, \ell$ and $k = 1, \dots, t$ by Lemma 4.1.9. Moreover, if the set of points T interpolates in f, we have $f(x_k) = z_k$ for all $k = 1, \dots, t$. By the homomorphic property of f, we have $f(u_{i,k}) = w_{i,k}$ for any i and k. Putting all together implies that the distribution of the challenge $(u_{i,k}, w_{i,k})$ is completely independent of the value λ_i for any i and k. Thus, a prover cannot deduce the right λ_i's with a probability greater than $p^{-\ell}$ from the challenges. Below, we show how we can break the binding property of Commit using a prover succeeding with a probability $\varepsilon > p^{-\ell}$. To this, we follow the same methodology as to prove assertion 4 of Theorem 4.2.1 (soundness of GHIproof) which consists in running the protocol once with the prover, rewinding this one, and running the protocol with carefully chosen coefficients.

The simulator $\mathcal{B}$ first picks the values $a_{i,j,k}$'s, $r_{i,k}$'s, and λ_i's uniformly at random and computes the tuples $u = (u_{1,1}, \dots, u_{\ell,t})$ and $w = (w_{1,1}, \dots, w_{\ell,t})$ according to the co-GHIproof protocol. Then, $\mathcal{B}$ sends u, w to the prover $\mathbf{P}^*$. This one answers com. The simulator releases the coefficients $a_{i,j,k}$'s, $r_{i,k}$'s, λ_i's, and the prover succeeds if he opens com on $\lambda = (\lambda_1, \dots, \lambda_\ell)$. In this case, $\mathcal{B}$ picks $\lambda^* = (\lambda_1^*, \dots, \lambda_\ell^*)$ uniformly at random. By using his expert group knowledge, he is able to find some uniformly random coefficients $a^*_{i,j,k}$'s, $r^*_{i,k}$'s satisfying

$$u_{i,k} - \lambda_i^* x_k = dr^*_{i,k} + \sum_{j=1}^{s} a^*_{i,j,k} g_j \text{ for } i = 1, \dots, \ell \text{ and } k = 1, \dots, t.$$

The simulator rewinds $\mathbf{P}^*$ with the same random tape and the same challenges. He answers the same commitment com. This time, $\mathcal{B}$ sends $a^*_{i,j,k}$'s, $r^*_{i,k}$'s. The prover wins if he is able to open com on the value λ^*. If $\lambda^* \neq \lambda$, the simulator breaks the computationally binding property of Commit.

Note that $\mathcal{B}$ simulates an honest verifier perfectly in both protocol runs. We can compute the success probability that $\mathcal{B}$ finds a collision for Commit in a very similar way as for GHIproof. Namely, we decompose the probability of success for the different random tapes ϖ and challenges u. Let $\varepsilon_{\varpi,u}$ be the probability that the prover wins in one protocol run with the random tape ϖ and the challenge u (and thus $w = f(u)$). Since the probability that $\lambda = \lambda^*$ for any random tape ϖ and challenge u is equal to $p^{-\ell}$, we can show as for GHIproof that the success probability of $\mathcal{B}$ satisfies

$$\mathsf{Succ}_{\mathcal{B}}^{\text{com-bnd}} \geq \sum_{\varpi,u} q_{\varpi,u} \cdot (\varepsilon_{\varpi,u}^2 - \varepsilon_{\varpi,u} \cdot p^{-\ell}),$$

where $q_{\varpi,u}$ denotes the probability that the protocol runs with the random tape ϖ and the challenge u. Again, applying Jensen's inequality leads to the desired bound $\varepsilon(\varepsilon - p^{-\ell})$. □

Remark 4.2.8. The security properties of Theorem 4.2.7 can easily be proved when the λ_i's are picked in a set $\{0, 1, \ldots, q\}$, for an integer $q < p - 1$. The soundness probability becomes $q^{-\ell}$ in this setting.

Remark 4.2.9. As for GHIproof, we will usually consider coGHIproof with Commit as a perfectly hiding and computationally binding trapdoor commitment scheme. However, in a context where non-transferability is not required coGHIproof can be instantiated with a classical perfectly hiding and computationally binding commitment.

4.2.3 Interactive Proof for the MGGD Problem

Let G, d be some parameters and S_1 some input of a d-MGGD problem. We propose here an interactive proof in which a prover proves that $S_1 = \{g_1, \ldots, g_s\}$ H-generate G, for any group H of order d. In other words, by assertion 5 of Lemma 4.1.3 this corresponds to show that $\langle S_1 \rangle + dG = G$ or by assertion 7 of the same lemma that the answer to the d-MGGD problem is positive. For this, the prover needs an expert group knowledge of G with S_1. Let $\ell \in \mathbb{N}$ be a security parameter. This protocol called MGGDproof is depicted in Figure 4.5.

A simplified version of this protocol with only one challenge is depicted in Figure 4.6.

Before stating security properties of this protocol, we introduce the following technical result. This one will play a crucial role for the soundness of MGGDproof.

Lemma 4.2.10. *Given a finite Abelian group G, a subset $S_1 = \{g_1, \ldots, g_s\} \subseteq G$, and an integer d with smallest prime factor p. We assume that there exists a function $f : G \to G \times \mathbb{Z}_d^s$ satisfying*

$$\Pr_x[x = dr + a_1 g_s + \cdots + a_s g_s \mid x \leftarrow_U G; f(x) = (r, a_1, \ldots a_s)] > \frac{1}{p}.$$

```
**MGGDproof**$_\ell(S_1)$
**Parameters:** $G, d$
**Input:** $\ell$, $S_1 = \{g_1, \ldots, g_s\} \subseteq G$
1: The prover picks $x_1, \ldots, x_\ell \in_U G$ uniformly at random and computes
   $(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(x_1, \ldots, x_\ell)$. He sends com to the verifier.
2: The verifier picks $y_1, \ldots, y_\ell \in_U G$ uniformly at random and sends
   $y_1, \ldots, y_\ell$ to the prover.
3: Using expert group knowledge, the prover finds $r_i \in G$, $a_{i,1}, \ldots, a_{i,s} \in \mathbb{Z}_d$
   such that $x_i + y_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \ldots, \ell$. He sends the values
   $r_i$'s, $a_{i,j}$'s to the verifier and opens the commitment by sending the $x_i$'s
   and dec.
4: The verifier checks that $x_i + y_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ holds for $i =
   1, \ldots, \ell$ and that the commitment is correctly opened by checking
   $1 \leftarrow \mathsf{Open}(x_1, \ldots, x_\ell, \mathsf{com}, \mathsf{dec})$. If this is the case, the verifier accepts
   the proof. Otherwise, he rejects it.
```

Figure 4.5: Interactive proof for the MGGD problem

Then, we have $G = \langle S_1 \rangle + dG$. In other words, S_1 H-generate G for any Abelian group H of order d.

Proof. First, we notice that if $\gcd(\#G, d) = 1$, we have $dG = G$ which trivially leads to $\langle S_1 \rangle + dG = G$. Now, assuming that $\gcd(\#G, d) \neq 1$, there exists a smallest prime p' such that $p' | \#G$ and $p' | d$. Consider now the unique prime factor decomposition

$$\#G = \prod_{i=1}^{k} q_i^{a_i},$$

where $q_1 < q_2 < \cdots < q_k$. Note that $p' = q_\ell$ for an integer $\ell \leq k$. By the structure of the Abelian groups (see Appendix A.1), we can deduce that $dG(q_i) = G(q_i)$ for any integer $i < \ell$, where $A(q)$ denotes the q-subgroup of an Abelian group A. This shows that the structure of dG is of the form

$$dG \simeq G(q_1) \oplus \cdots \oplus G(q_{\ell-1}) \oplus dG(q_\ell) \oplus \cdots \oplus dG(q_k)$$

and that

$$\#dG = \prod_{i=1}^{\ell-1} q_i^{a_i} \cdot \prod_{i=\ell}^{k} q_i^{b_i},$$

for some integers b_i's satisfying $b_i \leq a_i$ for $i = \ell, \ldots, k$. Since dG is a subgroup of $K := \langle S_1 \rangle + dG$, we are ensured that $\#G/\#K$ is whether 1 or greater or equal

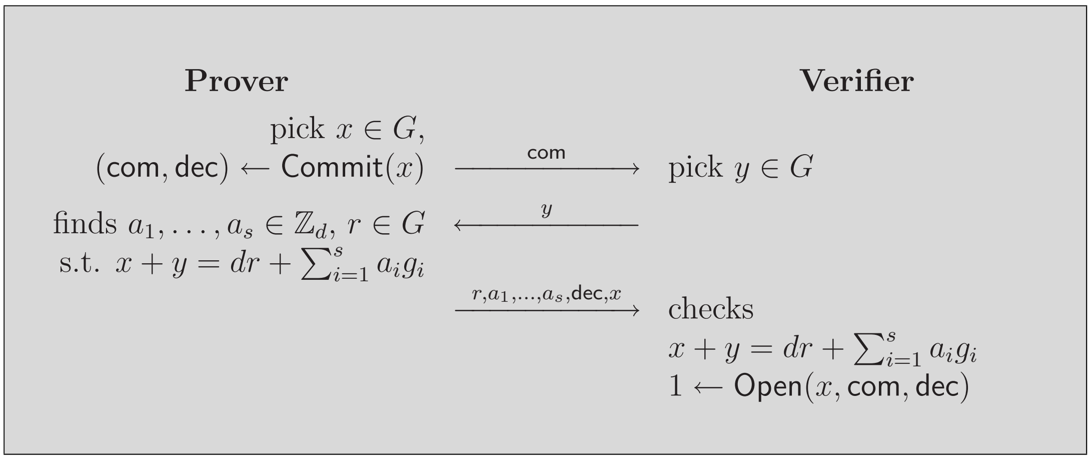
Prover
Verifier
pick $x \in G$,
$(\mathsf{com}, \mathsf{dec}) \leftarrow \mathsf{Commit}(x)$
$\xrightarrow{\mathsf{com}}$
pick $y \in G$
$\xleftarrow{y}$
finds $a_1, \dots, a_s \in \mathbb{Z}_d$, $r \in G$
s.t. $x + y = dr + \sum_{i=1}^{s} a_i g_i$
$\xrightarrow{r, a_1, \dots, a_s, \mathsf{dec}, x}$
checks
$x + y = dr + \sum_{i=1}^{s} a_i g_i$
$1 \leftarrow \mathsf{Open}(x, \mathsf{com}, \mathsf{dec})$

Figure 4.6: Simplified interactive proof for the MGGD problem

to p'. By the existence of f, we know that $\#G/\#K < p \leq p'$ which implies that $G = K$. □

We can now state the following security results.

Theorem 4.2.11. *Let G be an Abelian group and d be an integer with smallest prime factor p. We consider the parameters $S_1 = \{g_1, \dots, g_s\} \subseteq G$ and an integer ℓ. Assuming that the prover has an expert group knowledge of G with S_1, the* MGGDproof$_\ell(S_1)$ *protocol satisfies the following security properties.*

1. *The* MGGDproof *protocol is complete.*

2. *Assuming that* Commit *is a perfectly hiding trapdoor commitment scheme, the* MGGDproof *protocol is perfect black-box straight-line zero-knowledge against any verifier who has the secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ associated to* Commit.

3. *The* MGGDproof *protocol is sound: from any cheating prover $\mathbf{P}^*$ who passes the protocol with a set S_1 satisfying $\langle S_1 \rangle + dG \subsetneq G$ with a success probability $\mathsf{Succ}_{\mathbf{P}^*}^{\mathsf{sd}\text{-MGGD}} = \varepsilon$, there exists an algorithm $\mathcal{B}$ which finds a collision on* Commit *with a probability $\mathsf{Succ}_{\mathcal{B}}^{\mathsf{com}\text{-}\mathsf{bnd}} \geq \varepsilon(\varepsilon - p^{-\ell})$ by rewinding $\mathbf{P}^*$ once.*

Proof.

1. The completeness follows from the expert group knowledge of an honest prover.

2. We consider a simulator $\mathcal{B}$ who uses black-box access to the cheating verifier $\mathbf{V}^*$. They are both assumed to know the secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ of Commit. The simulator first picks a tuple $x' := (x'_1, \dots, x'_\ell) \in_U G^\ell$ uniformly at random and computes

$(\mathsf{com}', \mathsf{dec}') \longleftarrow \mathsf{Commit}(x', \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$. He then sends com' to $\mathbf{V}^*$. The simulator then receives a tuple $y := (y_1, \ldots, y_\ell) \in G^\ell$. If it is not the case, the simulator stops the simulation and outputs the transcript $(\mathsf{com}', \mathsf{abort})$. Otherwise, he picks $r_i \in_U G$, $a_{i,j} \in_U \mathbb{Z}_d$ uniformly at random for $i = 1, \ldots, \ell$, $j = 1, \ldots, s$ and computes

$$z_i := dr_i + \sum_{j=1}^{s} a_{i,j} g_j \text{ for } i = 1, \ldots, \ell.$$

By Lemma 4.1.9, z_i is uniformly distributed in G for $i = 1, \ldots, \ell$. He computes $x_i := z_i - y_i$ for $i = 1, \ldots, \ell$ and sets $x := (x_1, \ldots, x_\ell)$. He computes

$$\mathsf{dec} \longleftarrow \mathsf{Collide}(x', x, \mathsf{com}', \mathsf{dec}', \mathcal{K}_{\mathrm{s}}^{\mathbf{V}})$$

when $x \neq x'$. Finally, $\mathcal{B}$ answers the coefficients r_i's, $a_{i,j}$'s and opens com' on the tuple x by releasing x and dec (dec' when $x = x'$). He outputs the transcript $(\mathsf{com}', y, r_i\text{'s}, a_{i,j}\text{'s}, x, \mathsf{dec})$. We note that the simulator perfectly simulates the behaviour of an honest prover. This shows that the transcript distribution produced by $\mathcal{B}$ is identical as the one produced between the honest prover and $\mathbf{V}^*$.

3. Assume that S_1 is such that $\langle S_1 \rangle + dG \subsetneq G$. We describe a simulator $\mathcal{B}$ playing with the prover $\mathbf{P}^*$ in order to find a collision on Commit. Again, we proceed similarly as to prove the soundness of GHIproof. $\mathcal{B}$ first runs a MGGDproof protocol with $\mathbf{P}^*$ playing the role of an honest verifier. The transcript of this interaction is $(\mathsf{com}, y, r_i\text{'s}, a_{i,j}\text{'s}, x, \mathsf{dec})$, where $y = (y_1, \ldots, y_\ell) \in G^\ell$ and $x = (x_1, \ldots, x_\ell) \in G^\ell$. Then, the simulator rewinds $\mathbf{P}^*$ with the same random tape. This one sends again com. The simulator picks a tuple $y^* \in_U G^\ell$ uniformly at random and sends it to $\mathbf{P}^*$. Finally, the prover sends $r_i^*, a_{i,j}^*, x^*, \mathsf{dec}^*$ to $\mathcal{B}$. Note that the simulator perfectly simulates an honest verifier in both protocol runs.

We now consider a random tape ϖ and estimate the success probability for $\mathcal{B}$ conditioned by this event. Note that under this restriction both protocol runs are independent. Let A_ϖ be the probability event that $\mathbf{P}^*$ succeeds in the first run with the random tape ϖ and A_ϖ^* be the same event in the second run. We denote by B_ϖ the event $x \neq x^*$ with ϖ. Note that the simulator has found a collision on Commit, when the event $A_\varpi \wedge A_\varpi^* \wedge B_\varpi$ occurs.

At this time, we only consider random tapes ϖ satisfying $\varepsilon_\varpi := \Pr[A_\varpi] > p^{-\ell}$. Note that a protocol run is completely characterized by the random tape ϖ and the challenge y. Since this one is picked uniformly at random, our probability analysis can be performed by estimating the number of "good" protocol executions. Hence, we have

$$\Pr[A_\varpi \wedge A_\varpi^* \wedge B_\varpi] = \frac{\#\{(y, y^*) \in G^\ell \times G^\ell \mid \mathbf{P}^* \text{ wins in both runs and } x \neq x^*\}}{\#G^{2\ell}},$$

where x (resp. x^*) denotes the released value by the prover in a MGGDproof run with the random tape ϖ and challenge y (resp. y^*). We set $k := \varepsilon_\varpi \cdot \#G^\ell$. Let us denote $y^1, \dots, y^k \in G^\ell$, the k different challenges sent by $\mathcal{B}$ which produce the event A_ϖ. Let x^j be the corresponding x element for the challenge y^j for $j = 1, \dots, k$. We define $K := \langle S_1 \rangle + dG$ and $m := \#K^\ell$. For any $j = 1, \dots, k$, the ℓ components of the element $x^j + y^j$ must lie in the subgroup $K := \langle S_1 \rangle + dG$. By Lemma 4.2.10, $\#K/\#G \leq 1/p$, so that $k > m$. Using the fact that the elements $x^j + y^j$'s cannot take more than m different values, we deduce that

$$\#\{j \mid x^j = g\} \leq m \quad \text{for any } g \in G^\ell.$$

Namely, the existence of an element g contradicting the above assertion would lead to more than m values $g + y^j$ which should lie in K^ℓ. We denote by $c_1, \dots, c_t$, the t different values taken by the elements x^j's and set $n_j := \#\{i \mid x^i = c_j\}$ for $j = 1, \dots, t$. We have

$$\#\{(i,j) \mid x^i \neq x^j\} = \sum_{j=1}^{t} n_j(k - n_j)$$

and since $n_j \leq m$ for $j = 1, \dots, t$,

$$\sum_{j=1}^{t} n_j(k - n_j) \geq (k - m) \sum_{j=1}^{t} n_j = k(k - m).$$

From this, we deduce that

$$\Pr[A_\varpi \wedge A^*_\varpi \wedge B_\varpi] \geq \frac{k(k-m)}{\#G^{2\ell}} \geq \varepsilon_\varpi \cdot (\varepsilon_\varpi - p^{-\ell}).$$

This upper bound obviously holds for any random tape ϖ, since $\varepsilon_\varpi \leq p^{-\ell}$ leads to a non positive value for $\varepsilon_\varpi \cdot (\varepsilon_\varpi - p^{-\ell})$. From this, we get

$$\mathsf{Succ}^{\mathsf{com\text{-}bnd}}_{\mathcal{B}} \geq \sum_{\varpi} q_\varpi \cdot \varepsilon^2_\varpi - p^{-\ell} \cdot \sum_{\varpi} q_\varpi \cdot \varepsilon_\varpi,$$

where q_ϖ denotes the probability that the protocol runs with the random tape ϖ. Applying Jensen's inequality on the first term leads to the desired result. □

Remark 4.2.12. Using the techniques of Damgård [46], we can prove that the MGGDproof protocol is zero-knowledge in the auxiliary string model. The auxiliary string is the public key of Commit. The simulator first generates a key pair for Commit and feeds the verifier with the public key. Then, the simulator uses the knowledge of the secret key in order to simulate the transcript of the protocol as above. The main difference in this model is that the public key might be implemented by a trusted third party so that the verifier does not have to possess a secret key. However, we note that to ensure non-transferability the verifier needs to possess his own pair of key related to Commit. While it might be interesting here to make use of a trusted third party, this is not desirable to apply this for GHIproof and coGHIproof. Namely, non-transferability of both protocols will be crucial when applied to the MOVA undeniable signature.

Non-Interactive Variant of MGGDproof

Applying the Fiat-Shamir paradigm [58], the MGGDproof protocol can be turned non-interactive. Namely, the prover can simulate the challenges by himself from a seed using a pseudorandom generator. In fact, apart from finding representation of some elements of G, the only thing the prover needs to do is to convince the verifier that the challenges are drawn uniformly at random. An alternative to the Fiat-Shamir paradigm would consist in using a trusted third party (trusted by **P** and **V**) to generate challenges uniformly at random. We present in Figure 4.7 a non-interactive MGGDproof variant without trusted party in the random oracle model. For this, we need to consider a pseudorandom generator GenM (modeled by a random oracle) defined on the set $\{0,1\}^{k_m}$, for some integer k_m.

NIMGGDproof$_\ell(S_1)$
Parameters: G, d
Input: ℓ, $S_1 = \{g_1, \dots, g_s\} \subseteq G$

```
1: The prover picks seedM $\in_U \{0,1\}^{k_m}$ uniformly at random and using the pseudorandom generator GenM produces some challenges $x_1, \dots, x_\ell$. Then, using his expert group knowledge, he finds $r_i \in G$ and $a_{i,1}, \dots, a_{i,s} \in \mathbb{Z}_d$ such that $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \dots, \ell$. He sends seedM and the coefficients $r_i$'s and $a_{i,j}$'s to the verifier.
2: Using GenM, the verifier generates $x_1, \dots, x_\ell$ from seedM. He checks that $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ holds for $i = 1, \dots, \ell$. If this is the case, the verifier accepts the proof. Otherwise, he rejects it.
```

Figure 4.7: Non-interactive proof for the MGGD problem

Security results related to NIMGGDproof are given in the following theorem.

Theorem 4.2.13. *Let G be an Abelian group and d be an integer with smallest prime factor p. We consider $S_1 = \{g_1, \dots, g_s\} \subseteq G$ and an integer ℓ. Assuming that* GenM *is a random oracle and the prover has an expert group knowledge of G with S_1, the* NIMGGDproof$_\ell(S_1)$ *protocol satisfies the following security properties.*

1. *The* NIMGGDproof *protocol is complete.*
2. *The* NIMGGDproof *protocol is perfect black-box zero-knowledge*[2]*.*
3. *The* NIMGGDproof *protocol is sound: for any set S_1 such that $\langle S_1 \rangle + dG \subsetneq G$, any cheating prover $\mathbf{P}^*$ limited to q_{GenM} queries to* GenM*, has a success probability $\mathsf{Succ}_{\mathbf{P}^*}^{\mathsf{sd}\text{-NIMGGD}} \leq q_{\mathrm{GenM}} \cdot p^{-\ell} + (\#G)^{-\ell}$.*

Proof.

1. The completeness follows from the expert group knowledge of the prover.

2. We describe a simulator $\mathcal{B}$ who simulates the message sent by an honest prover. Although the verifier $\mathbf{V}^*$ does not send any message here, the simulator needs to simulate the random oracle GenM which can be queried by $\mathbf{V}^*$. $\mathcal{B}$ picks $r_i \in_U G$, $a_{i,j} \in_U \mathbb{Z}_d$ uniformly at random and computes

$$x_i := dr_i + \sum_{j=1}^{s} a_{i,j} g_j,$$

for $i = 1, \ldots, \ell$, $j = 1, \ldots, s$. The simulator picks seedM $\in_U \{0,1\}^{k_m}$ uniformly at random and adds the pair (seedM, x), where $x = (x_1, \ldots, x_\ell)$ in a list maintained to simulate GenM. Then, the simulator can run $\mathbf{V}^*$ and sends him seedM and the coefficients r_i's, $a_{i,j}$'s. $\mathcal{B}$ simulates GenM as usual by maintaining a list of the previous queries and corresponding answers. For any new query, the simulator simply picks the answer uniformly at random. We note that the simulation is done perfectly since seedM was added in the list before the first query made by $\mathbf{V}^*$.

3. We describe here a simulator $\mathcal{B}$ who uses $\mathbf{P}^*$ in order to win the following game.
Game: A challenger picks $x_i \in_U G$ uniformly at random for $i = 1, \ldots, \ell$ and sends $x_1, \ldots, x_\ell$ to $\mathcal{B}$. The simulator wins if he is able to find coefficients r_i's and $a_{i,j}$'s such that $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \ldots, \ell$.

The simulator first receives $x = (x_1, \ldots, x_\ell)$ according to the above game and runs $\mathbf{P}^*$. $\mathcal{B}$ picks an integer $n \in_U \{1, \ldots, q_{\mathrm{GenM}}\}$ uniformly at random. The GenM queries made by $\mathbf{P}^*$ are simulated by maintaining a list of the queries and corresponding answers. When the query is not new, $\mathcal{B}$ returns the answer stored in the list. Otherwise, the simulator outputs a uniformly random answer and adds the new pair in the list. However, we handle the nth query in a special way. Namely, we answer x to this query (Without loss of generality, we assume that $\mathbf{P}^*$ does not submit the same query more than once.). Since x was picked uniformly at random, $\mathcal{B}$ simulates the oracle GenM perfectly. At the end, $\mathbf{P}^*$ outputs a seed seedM

[2] Note that non-interactive zero-knowledge is anyway straight-line.

and coefficients r_i's and $a_{i,j}$'s. The simulator forwards the same coefficients to his challenger.

Let A be the event "$\mathcal{B}$ wins the game". By Lemma 4.2.10, $\Pr[A] \leq p^{-\ell}$. We also note that event A occurs if $\mathbf{P}^*$ sends seedM as nth query made to GenM. Let B be the event "$\mathbf{P}^*$ queried seedM to GenM" and C be the event "$\mathbf{P}^*$ succeeds". We have

$$\Pr[A] = \frac{1}{q_{\text{GenM}}} \cdot \Pr[B \wedge C].$$

Since GenM is a random oracle,

$$\Pr[\neg B \wedge C] \leq (\#G)^{-\ell},$$

because the prover needs to guess x which corresponds to some seedM. Putting all together leads to

$$p^{-\ell} \geq \Pr[A] \geq \frac{1}{q_{\text{GenM}}} \left(\mathsf{Succ}_{\mathbf{P}^*}^{\mathsf{sd}\text{-NIMGGD}} - (\#G)^{-\ell}\right),$$

which concludes the proof. □

Remark 4.2.14. Note that zero-knowledge here is not deniable, since we need to simulate answers of the random oracle GenM.

4.2.4 2-Move Interactive Proofs

This part focuses on some 2-move variants of the GHIproof and coGHIproof protocols. In order to achieve the different security properties such as *zero-knowledge* and *soundness* the introduction of some random oracles is required.

The variants are achieved by removing the two messages sent in the middle of the protocols which allow to achieve zero-knowledge through the commitment scheme. In order to maintain zero-knowledge, the verifier sends a kind of commitment on a seed which generates the coefficients producing the challenges sent to the prover. This commitment can be only opened by the prover after this one solved these challenges. We notably add a trapdoor one-way permutation with associated secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$ to the verifier. Namely, we consider the permutation $\mathsf{TPOW}(\cdot, \mathcal{K}_{\mathrm{p}}^{\mathbf{V}})$ and its inverse $\mathsf{TPOW}^{-1}(\cdot, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}})$. We denote $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{inv\text{-}tp}}$ the probability that an adversary $\mathcal{A}$ can compute $\mathsf{TPOW}^{-1}(y, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}})$ given a random y, without knowing $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$. For the sake of simplicity, we use the same notation TPOW for both protocols. We also need to introduce a pseudorandom generator GenC (resp. GenD) as well as a cryptographic hash function denoted H_c (resp. H_d) in this 2-move variant of GHIproof (resp. coGHIproof). Later, to analyze the security of the protocols these primitives will be idealized by random oracles.

The 2-move variants notably have a very similar complexity as the 4-move ones. In particular, the prover needs to perform the same number of homomorphism evaluations. Actually, in one hand we increase the complexity of one additional evaluation of each primitive GenC (resp. GenD), H_c (resp. H_d), TPOW for both the prover and the verifier. On the other hand, we have a gain due to all computations related to Commit which were required in the 4-move variants.

2-Move Variant of GHIproof

The 2-move variant of GHIproof called 2-GHIproof is given in Figure 4.8.

```
**2-GHIproof**$_\ell(S)$
**Parameters:** $G, H, d$
**Input:** $\ell$, $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$
1: The verifier picks seedC $\in_U \{0,1\}^{k_c}$ uniformly at random, and by applying a pseudorandom generator GenC on this seed, generates values $r_i \in G$ and $a_{i,j} \in \mathbb{Z}_d$ for $i = 1, \ldots, \ell$ and $j = 1, \ldots, s$. He computes $u_i = dr_i + a_{i,1}g_1 + \cdots + a_{i,s}g_s$, $w_i = a_{i,1}e_1 + \cdots + a_{i,s}e_s$ for $i = 1, \ldots, \ell$, and $\vartheta_c = \mathsf{TPOW}(\text{seedC}, \mathcal{K}_\text{p}^\mathbf{V})$. Using a cryptographic hash function $H_c : \{0,1\}^* \to \{0,1\}^{k_c}$, he computes $h_c := H_c(w_1, \ldots, w_\ell) \oplus \text{seedC}$. The verifier sends $u_1, \ldots, u_\ell$, $h_c$ and $\vartheta_c$ to the prover.
2: The prover checks whether $f(g_i) = e_i$ for $i = 1, \ldots, \ell$. If it is not the case, he aborts the protocol. Then, he computes $v_i = f(u_i)$ for $i = 1, \ldots, \ell$, $\text{seedC}' = H_c(v_1, \ldots, v_\ell) \oplus h_c$. He checks that $\vartheta_c = \mathsf{TPOW}(\text{seedC}', \mathcal{K}_\text{p}^\mathbf{V})$ and that GenC(seedC$'$) generates values $a_{i,j}$'s and $r_i$'s such that $u_i := dr_i + a_{i,1}g_1 + \cdots + a_{i,s}g_s$ for $i = 1, \ldots, \ell$. If not, the prover aborts the protocol. He then sends seedC$'$ to the verifier.
3: The verifier accepts the proof if seedC$'$ = seedC holds. Otherwise, he rejects it.
```

Figure 4.8: 2-move interactive proof for the GHID problem

The 2-GHIproof protocol possesses similar security properties as its 4-move variant, except that assertion 5 of Theorem 4.2.1 cannot be adapted here. The main reason is due to the fact that the verifier cannot fool the prover by sending challenges in G of his choice, since the prover does not commit the answers before having checked that the challenges were correctly generated. Hence, the verifier cannot use the prover as an oracle of the group homomorphism so that we are unable to apply the techniques used for GHIproof. The security results are given here.

Theorem 4.2.15. *Let G, H be some Abelian groups and $\ell \in \mathbb{N}$. We denote by d the order of H and p the smallest prime factor of d. Assume that we are given a set of points $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$ such that the elements $g_1, \ldots, g_s$ H-generate the group G. Assuming that* GenC, *H_c are random oracles and* TPOW *is a trapdoor one-way permutation, the* 2-GHIproof$_\ell(S)$ *protocol satisfies the following security properties.*

1. *The* 2-GHIproof *protocol is complete.*
2. *The* 2-GHIproof *protocol is statistical black-box straight-line zero-knowledge against any verifier. If the verifier possesses the secret key of* TPOW, *the* 2-GHIproof *protocol is perfect black-box straight-line zero-knowledge.*
3. *The* 2-GHIproof *protocol is perfect non-transferable.*
4. *The* 2-GHIproof *protocol is sound: for any invalid set S (i.e., with no interpolating homomorphism), any cheating prover $\mathbf{P}^*$ limited to q_{H_c} queries to H_c, has a success probability $\mathsf{Succ}_{\mathbf{P}^*}^{\mathsf{sd}\text{-}2\mathrm{GHI}} < \mathsf{Succ}^{\mathsf{inv}\text{-}\mathsf{tp}} + q_{H_c} p^{-\ell}$, where $\mathsf{Succ}^{\mathsf{inv}\text{-}\mathsf{tp}}$ is the maximum of $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{inv}\text{-}\mathsf{tp}}$ among all adversaries $\mathcal{A}$ which have similar complexity as $\mathbf{P}^*$.*

Proof.

1. The completeness follows by noticing that the prover retrieves the right seedC when $f(g_i) = e_i$ for $i = 1, \ldots, \ell$. Namely, we have $v_i = w_i$ for all $i = 1, \ldots, \ell$ in this case.

2. First, we consider that $\mathbf{V}^*$ (and the simulator $\mathcal{B}$) is not given $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$. $\mathcal{B}$ runs the verifier $\mathbf{V}^*$ and looks at the queries made by $\mathbf{V}^*$ to the oracle GenC. $\mathcal{B}$ puts these q_{GenC} queries seedC$_k$ for $1 \leq k \leq q_{\mathrm{GenC}}$ as well as the corresponding answers of GenC in memory. The simulator then receives the first message M of $\mathbf{V}^*$. If this one has not a correct format, the simulator outputs the transcript (M, abort). Otherwise, the simulator checks whether one answer among those queries seedC$_k$'s made to GenC correctly generates the challenges u_i's and the image of this query by TPOW is equal to ϑ_c. If it is not the case, $\mathcal{B}$ outputs the transcript $(u_1, \ldots, u_\ell, h_c, \vartheta_c, \mathsf{abort})$. Otherwise, the simulator is able to compute the right w_i's from this answer (the right r_i's and $a_{i,j}$'s) namely $w_i = \sum_{j=1}^{s} a_{i,j} e_j$ for $i = 1, \ldots, \ell$. From the w_i's, $\mathcal{B}$ computes seedC$^* := h_c \oplus H_c(w_1, \ldots, w_\ell)$ and checks whether seedC* generates the right r_i's and $a_{i,j}$'s. In the positive case, $\mathcal{B}$ outputs the transcript $(u_1, \ldots, u_\ell, h_c, \vartheta_c, \mathrm{seedC}^*)$. In the negative case, it outputs the following transcript $(u_1, \ldots, u_\ell, h_c, \vartheta_c, \mathsf{abort})$.

It remains to show that the two transcript distributions are statistically indistinguishable. When the first message has not a correct format, the two transcripts are clearly identical. Let consider the case where the verifier did not query any seedC$_k$ which produces the challenges u_i's and whose image by TPOW leads to ϑ_c. In this case, the honest prover does not abort the protocol only if he retrieves a seedC $= H(w_1, \ldots, w_\ell) \oplus h_c$ which generates the challenges u_i's and ϑ_c. This occurs only if the verifier $\mathbf{V}^*$ was able to guess that the output values of the query seedC to the oracle GenC generate the right r_i's and a_{ij}'s. Since GenC is a random oracle, no verifier $\mathbf{V}^*$ (even computationally unbounded) can succeed to do that with a non-negligible probability. We still have to consider the case where the verifier queried a seedC$_k$ which produces the challenges u_i's and ϑ_c. We see that the two transcripts are always identical, since the simulator clearly knows the answer of the honest prover by learning the right w_i's. Therefore, we can conclude that the two transcript distributions are statistically indistinguishable.

If $\mathbf{V}^*$ is given $\mathcal{K}_\mathrm{s}^\mathbf{V}$, the simulation can be done perfectly as follows. First, $\mathcal{B}$ launches $\mathbf{V}^*$ and receives the first message (which should be $u = (u_1, \ldots, u_\ell)$, h_c, and ϑ_c). The simulator computes seedC$' = \mathsf{TPOW}^{-1}(\vartheta_c, \mathcal{K}_\mathrm{s}^\mathbf{V})$ and using GenC generates coefficients a'_{ij} and r'_i and corresponding u'_i and w'_i for $i = 1, \ldots, \ell$ and $j = 1, \ldots, s$. Then, $\mathcal{B}$ checks whether $u'_i = u_i$ for $i = 1, \ldots, \ell$, seedC$' = H_c(w'_1, \ldots, w'_\ell) \oplus h_c$. If it is the case, $\mathcal{B}$ outputs the transcript $(h_c, w, \vartheta_c, \mathrm{seedC}')$. Otherwise, it outputs $(h_c, w, \vartheta_c, \mathsf{abort})$. Note that an honest prover would check exactly the same equalities (in a different way) and would answer in the same way. Hence, zero-knowledge is perfect.

3. The simulator first has a look at the bit telling whether S interpolates in f or not. If S does not interpolate, the simulator aborts in the second step of the protocol. Otherwise, he behaves as the simulator in the proof of straight-line zero-knowledge with a verifier knowing the secret key of TPOW. Namely, this simulation did not use the random oracle queries of the verifier so that the simulation applies here. Thus, this works perfectly for the non-transferability as well.

4. Let $\mathbf{P}^*$ be a cheating prover who wants to show that a non-interpolating set S interpolates in a homomorphism. Without loss of generality, we can assume that $\mathbf{P}^*$ always responds correctly to the verifier whenever he queries seedC to GenC. Indeed, he can check that seedC is the preimage of ϑ_c by TPOW and answer seedC to the challenge if correct. (With an honest verifier, there is no need to check whether the challenge is valid.) Hence, the verifier always accepts when the prover queries seedC to GenC. Similarly, we can assume that $\mathbf{P}^*$ always responds correctly to the verifier whenever he queries the right $w = (w_1, \ldots, w_\ell)$ to H_c because he can deduce seedC from h_c afterwards. Note that when $\mathbf{P}^*$ interacts with an honest verifier, the verifier only accepts if $\mathbf{P}^*$ outputs seedC.

We transform $\mathbf{P}^*$ into an algorithm to invert the trapdoor permutation as follows.

1. We receive a random challenge ϑ_c, whose preimage by TPOW is denoted seedC.

2. We generate some random values r_i's and $a_{i,j}$'s. We deduce some u_i's and w_i's and pick a random h_c. Then (u, h_c, ϑ_c) is a challenge for the prover. We simulate GenC as follows: for any query except seedC (we can check whether a value is seedC by checking that its image by TPOW is ϑ_c) we simulate a random oracle as usual i.e., we maintain a list of elements queried to GenC with corresponding answers and simulate according to this list. If the query is new, we simply pick the answer uniformly at random and add the pair in the list. For the query seedC, we stop the overall simulation and yield seedC: the inversion of ϑ_c succeeded. We simulate H_c as follows: for any query except w we simulate a random oracle (like for GenC). For the query w we stop: the inversion of ϑ_c failed.

3. We run $\mathbf{P}^*$ according to our simulation rules. If $\mathbf{P}^*$ outputs some value, we check whether it is seedC. If it is, we output it, otherwise we fail.

The algorithm succeeds to invert the trapdoor permutation at the condition that either (event A) $\mathbf{P}^*$ succeeds without even querying seedC to GenC nor w to H_c, or (event B) that $\mathbf{P}^*$ queries seedC to GenC without querying w to H_c beforehand. Let C be the event that $\mathbf{P}^*$ queries w to H_c before querying seedC to GenC. Since the simulation is perfect, $\Pr[A \cup B] + \Pr[C]$ is the probability that $\mathbf{P}^*$ passes the protocol with an honest verifier. We have $\Pr[A \cup B] \leq \mathsf{Succ}^{\text{inv-tp}}$. Below we exhibit an upper bound for $\Pr[C]$. To this, we consider a simulator $\mathcal{B}$ which plays with $\mathbf{P}^*$ to win the following game.
Game: A challenger picks elements r_i's and $a_{i,j}$'s uniformly at random and compute $u_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ and $w_i = \sum_{j=1}^{s} a_{i,j} e_j$ for $i = 1, \ldots, \ell$. The simulator $\mathcal{B}$ receives the u_i's and wins the game if he finds all the values w_i's.

$\mathcal{B}$ simply forwards the received challenges u_i's and picks h_c and ϑ_c uniformly at random in $\{0,1\}^{k_c}$. $\mathcal{B}$ simulates the oracle H_c as above, except that he guesses when the w_i's are queried. For this, he just picks an integer $\ell \in \{1, \ldots, q_{H_c}\}$ uniformly at random and stops the simulation at the ℓth query made to H_c. The simulator then answers the values w_i's. Note that $\mathbf{P}^*$ cannot query seedC to GenC when event C occurs. The simulation is perfect in the C case provided that ℓ is correctly guessed. Thus, we have $\Pr[D] \geq 1/q_{H_c} \cdot \Pr[C]$, where D denotes the event of winning the above game. By Lemma 4.1.11, $\Pr[D] \leq p^{-\ell}$, which implies

$$\Pr[C] \leq q_{H_c} p^{-\ell}.$$

So, the prover cannot succeed with probability larger than $\mathsf{Succ}^{\text{inv-tp}} + q_{H_c} p^{-\ell}$. $\square$

Remark 4.2.16. When the non-transferability is not necessary, the 2-GHIproof protocol can be simplified by removing operations related to TPOW and ϑ_c. Such simplified version can be shown to satisfy the same security properties except the non-transferability using very similar proofs as for Theorem 4.2.15.

2-Move Variant of coGHIproof

The interactive coGHIproof protocol can be transformed in a 2-move protocol in a similar way. This variant called 2-coGHIproof is presented in Figure 4.9

```
2-coGHIproof$_\ell(S)$
Parameters: $G, H, d, p$
Input: $\ell$, $S = \{(g_1, e_1), \dots, (g_s, e_s)\} \subseteq G \times H$, $T = \{(x_1, z_1), \dots, (x_t, z_t)\}$
1: The verifier picks seedD $\in_U \{0,1\}^{k_d}$ uniformly at random, and by applying a pseudorandom generator GenD on this seed, generates values $r_{i,k} \in G$, $a_{i,j,k} \in \mathbb{Z}_d$, $\lambda_i \in \mathbb{Z}_p$ for $i = 1, \dots, \ell$, $j = 1, \dots, s$, $k = 1, \dots, t$. He computes $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$, $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for $i = 1, \dots, \ell$, and $\vartheta_d = \mathsf{TPOW}(\text{seedD}, \mathcal{K}_{\text{p}}^{\mathbf{V}})$. Using a cryptographic hash function $H_d : \{0,1\}^* \to \{0,1\}^{k_d}$, the verifier computes $h_d := H_d(\lambda_1, \dots, \lambda_\ell) \oplus \text{seedD}$. Set $u := (u_{1,1}, \dots, u_{\ell,t})$ and $w := (w_{1,1}, \dots, w_{\ell,t})$. He sends $u$, $w$, $h_d$ and $\vartheta_d$ to the prover.
2: The prover computes $y_k = f(x_k)$ for $k = 1, \dots, t$ and verifies that $y_k \neq z_k$ for at least one $k$. Otherwise, he aborts the protocol. Then, he computes $v_{i,k} := f(u_{i,k})$ for $i = 1, \dots, \ell$, $k = 1, \dots, t$. From the equations $w_{i,k} - v_{i,k} = \lambda_i(z_k - y_k)$, he should be able to find every $\lambda_i$ if the verifier is honest since $y_k \neq z_k$ for at least one $k$. The prover computes seedD$' = H_d(\lambda_1, \dots, \lambda_\ell) \oplus h_d$. He checks that $\vartheta_d = \mathsf{TPOW}(\text{seedD}', \mathcal{K}_{\text{p}}^{\mathbf{V}})$ and that seedD$'$ generates coefficients $r_{i,k}$'s, $a_{i,j,k}$'s, $\lambda_i$'s such that $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$, $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all $i$ and $k$. If not, he aborts the protocol. He then sends seedD$'$ to the verifier.
3: The verifier accepts the proof if seedD$'$ = seedD holds. Otherwise, he rejects it.
```

Figure 4.9: 2-move interactive proof for the co-GHID problem

We obtain similar security results as Theorem 4.2.7 as shown here.

Theorem 4.2.17. *Let G, H be some Abelian groups and $\ell \in \mathbb{N}$. We denote by d the order of H and p the smallest prime factor of d. Assume that we are given a set of points $S = \{(g_1, e_1), \dots, (g_s, e_s)\} \subseteq G \times H$ such that the elements $g_1, \dots, g_s$ H-generate the group G and a set $T = \{(x_1, z_1), \dots, (x_t, z_t)\} \subseteq G \times H$. Assuming that* GenD, *$H_d$ are random oracles and* TPOW *is a trapdoor one-way permutation, the* 2-coGHIproof$_\ell(S, T)$ *protocol satisfies the following security properties.*

1. *The* 2-coGHIproof *protocol is complete.*

2. *The* 2-coGHIproof *protocol is statistical black-box straight-line zero-knowledge against any verifier. If the verifier possesses the secret key of* TPOW*, the* 2-coGHIproof *protocol is perfect black-box straight-line zero-knowledge.*

3. *The* 2-coGHIproof *protocol is perfect non-transferable.*

4. *The* 2-coGHIproof *protocol is sound: for any set T interpolating with S in a group homomorphism (i.e., with f), any cheating prover $\mathbf{P}^*$ limited to q_{H_d} queries to H_d, has a success probability $\mathsf{Succ}_{\mathbf{P}^*}^{\text{sd-2coGHI}} < \mathsf{Succ}^{\text{inv-tp}} + q_{H_d}p^{-\ell}$, where $\mathsf{Succ}^{\text{inv-tp}}$ is the maximum of $\mathsf{Succ}_{\mathcal{A}}^{\text{inv-tp}}$ among all adversaries $\mathcal{A}$ which have similar complexity as $\mathbf{P}^*$.*

Proof.

1. When T does not interpolate with S, the prover retrieves the correct answer $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ and so retrieves the correct seedD by computing $h_d \oplus H_d(\lambda)$.

2. This can be shown exactly as for 2-GHIproof. We follow the proof of assertion 2 of Theorem 4.2.15 except that we replace H_c by H_d, GenC by GenD, seedC by seedD, h_c by h_d, ϑ_c by ϑ_d. When the verifier is not given the secret key of TPOW, the simulator can find seedD by looking at the queries made to GenD by the verifier except with a negligible probability. If the verifier is given the secret key of TPOW, the simulator can retrieve seedD$' = \mathsf{TPOW}^{-1}(\vartheta_d, \mathcal{K}_{\text{s}}^{\mathbf{V}})$ and proceed to the same verifications as an honest prover. Hence, the simulation is perfect in this case.

3. This proof is very similar to the proof of the non-transferability of 2-GHIproof. The simulator first checks whether the set T interpolates with S and in the positive case aborts in the second step of the protocol. Otherwise, the simulator can follow the simulation done for proving straight-line zero-knowledge of 2-coGHIproof with a verifier knowing the secret key of TPOW.

4. The soundness can be proved in a similar way as for the 2-GHIproof protocol. We follow the proof of the assertion 4 of Theorem 4.2.15 except that we replace H_c by H_d, GenC by GenD, seedC by seedD, h_c by h_d, ϑ_c by ϑ_d. We use the prover to invert TPOW as for 2-GHIproof with the main difference that we generate here the challenges u and w using some random values $r_{i,k}$'s, $a_{i,j,k}$'s, λ_i's. The simulation of the oracles are also done in the same way and we can detect when seedD is queried to GenD. We stop the simulation when λ is queried to H_d. By considering similar events A, B and C, we can show that $\Pr[A \cup B] + \Pr[C]$ is the probability that $\mathbf{P}^*$ passes the protocol with an honest verifier. We also have $\Pr[A \cup B] \leq \mathsf{Succ}^{\text{inv-tp}}$. Now, the probability that $\mathbf{P}^*$ queries λ to H_d before sending seedD to GenD (event C) can be upper bounded as follows. We consider a simulator $\mathcal{B}$ playing with the

prover to win the following game.
Game: A challenger picks some elements $a_{i,j,k}$'s, $r_{i,k}$'s, λ_i's uniformly at random and compute $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k}g_j + \lambda_i x_k$, $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k}e_j + \lambda_i z_k$ for $i = 1, \ldots, \ell$ and $k = 1, \ldots, t$. The simulator receives the $u_{i,k}$'s and $w_{i,k}$'s and wins the game if he is able to retrieve $\lambda = (\lambda_1, \ldots, \lambda_\ell)$.

Following again the proof of soundness for 2-GHIproof, $\mathcal{B}$ can similarly solve the above game by picking an integer in $\{1, \ldots, q_{H_d}\}$ uniformly at random to guess when λ is queried to H_d by $\mathbf{P}^*$. Hence, we have $\Pr[D] \geq 1/q_{H_d} \cdot \Pr[C]$, where D is the event that $\mathcal{B}$ wins the above game. It remains to show that no algorithm wins the above game with a probability greater than $p^{-\ell}$. By Lemma 4.1.11, the challenges $u_{i,k}$ are all uniformly distributed and independent of the value λ. Moreover, since T interpolates in f, we have $w_{i,k} = f(u_{i,k})$ for any i and k so that the $w_{i,k}$'s are uniquely determined by the $u_{i,k}$'s. This shows that the challenges in the above game do not leak any information about λ, which implies that no algorithm can win with a larger probability than $p^{-\ell}$. □

Remark 4.2.18. Again, if one is willing to relax the non-transferability of the protocol 2-coGHIproof, we can slightly modify it by removing computations related to TPOW.

4.3 The MOVA Scheme

In this section, we develop one of the core results of our thesis. Namely, we present our undeniable scheme called MOVA which is based on a secret group homomorphism. This scheme was first proposed at ASIACRYPT '04 [107] and was inspired by a preliminary version restricted to group characters (with a less efficient denial protocol) presented at PKC '04 [109].

We now describe the MOVA signature scheme. We consider several setup variants in order to generate the related signer's key pair. Depending on the situation one will be preferred on the others.

Domain Parameters. We let integers Lkey, Lsig, Icon, Iden be security parameters as well as "group types" for Xgroup and Ygroup. The group types should define what groups and which sizes to use in order to achieve a certain level of security. An optional parameter Ival $\in \mathbb{N}$ is used in Setup Variants 3 and 4 below.

Primitives. We use two deterministic pseudorandom generators GenK and GenS (modeled by some random oracles) which produce elements of Xgroup. Depending on whether we consider some 4-move verification protocols or 2-move protocols, we consider a trapdoor commitment scheme Commit or a trapdoor one-way permutation TPOW. In both cases, the associated pair of key $(\mathcal{K}_{\mathrm{p}}^{\mathbf{V}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{V}})$ is that of the verifier.

Setup Variant 1. (signer without expert group knowledge)

The signer selects Abelian groups Xgroup and Ygroup of given types together with a secret group homomorphism Hom : Xgroup $\longrightarrow$ Ygroup. He computes the order d of Ygroup. He then picks a random string seedK and computes the Lkey first values $(\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}})$ from GenK(seedK) and $\mathrm{Ykey}_j := \mathrm{Hom}(\mathrm{Xkey}_j)$, for $j = 1, \ldots, \mathrm{Lkey}$.

Setup Variant 2. (signer with a Registration Authority and without expert group knowledge)

We use here a Registration Authority (RA) whose role consists of making sure that seedK was randomly selected.

1. The signer selects Abelian groups Xgroup and Ygroup of given type together with a group homomorphism Hom : Xgroup $\longrightarrow$ Ygroup. He computes the order d of Ygroup. He submits his identity Id together with Xgroup, Ygroup and d to RA.
2. RA first checks the identity of the signer and that he did not submit too many registration attempts. He then picks a random string seedK that is sent to the signer together with a signature C of the data

$$(\mathrm{Id}, \mathrm{Xgroup}, \mathrm{Ygroup}, d, \mathrm{seedK}).$$

3. The signer computes the Lkey first values $(\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}})$ from GenK(seedK) and $\mathrm{Ykey}_j := \mathrm{Hom}(\mathrm{Xkey}_j)$ for $j = 1, \ldots, \mathrm{Lkey}$.

Setup Variant 3. (signer with an expert group knowledge with validity proof of public key)

In this variant we assume that the signer has an expert group knowledge of Xgroup. It works exactly like in the Setup Variant 1, but the signer can further run a $\mathrm{MGGDproof}_{\mathrm{Ival}}$ in order to validate the public key so that Lkey can be further reduced to the smallest possible one.

Setup Variant 4. (signer with an expert group knowledge with non-interactive validity proof of public key)

This variant is the same as Setup Variant 3 except that the signer produces the non-interactive proof $\mathrm{NIMGGDproof}_{\mathrm{Ival}}$.

Public Key. $\mathcal{K}_\mathrm{p}^\mathbf{S} = (\mathrm{Xgroup}, \mathrm{Ygroup}, d, \mathrm{seedK}, (\mathrm{Ykey}_1, \ldots, \mathrm{Ykey}_\mathrm{Lkey}))$ with an optional (Id, C) for the variant 2, an optional Ival for the variants 3,4, and an optional non-interactive proof for the variant 4.

We say that the public key $\mathcal{K}_\mathrm{p}^\mathbf{S}$ is *valid* if $\{\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_\mathrm{Lkey}\}$ Ygroup-generate Xgroup.

Secret Key. $\mathcal{K}_\mathrm{s}^\mathbf{S} = \mathrm{Hom}$.

Signature Generation. Let m be a message to sign. The signer generates

$$\mathrm{GenS}(m) \to \mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_\mathrm{Lsig}.$$

He then computes $\mathrm{Ysig}_k = \mathrm{Hom}(\mathrm{Xsig}_k)$ for $k = 1, \ldots, \mathrm{Lsig}$. The signature is

$$\sigma = (\mathrm{Ysig}_1, \ldots, \mathrm{Ysig}_\mathrm{Lsig}).$$

It is $\mathrm{Lsig} \cdot \log_2 d$ bits long.

Confirmation Protocol. Let (m, σ) be a supposedly valid message-signature pair. Both the signer and the verifier (signature's recipient) compute the elements $\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_\mathrm{Lkey}$ from the signer's public key. They also generate

$$\mathrm{GenS}(m) \to \mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_\mathrm{Lsig}.$$

The signer playing the role of the prover runs $\mathrm{GHIproof}_\mathrm{Icon}$ with the verifier on the set

$$S = \{(\mathrm{Xkey}_j, \mathrm{Ykey}_j) \mid j = 1, \ldots, \mathrm{Lkey}\} \cup \{(\mathrm{Xsig}_k, \mathrm{Ysig}_k) \mid k = 1, \ldots, \mathrm{Lsig}\}.$$

Alternatively, they can run 2-$\mathrm{GHIproof}_\mathrm{Icon}$ on the same set S.

Denial Protocol. Let (m, σ') be an alleged invalid message-signature pair. We denote

$$\sigma' = (\mathrm{Zsig}_1, \ldots, \mathrm{Zsig}_\mathrm{Lsig}).$$

The signer and the verifier compute $\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_\mathrm{Lkey}$ from the public key as well as $\mathrm{GenS}(m) \to \mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_\mathrm{Lsig}$. The signer playing the role of the prover run $\mathrm{coGHIproof}_\mathrm{Iden}$ with the verifier on the sets

$$S = \{(\mathrm{Xkey}_j, \mathrm{Ykey}_j) \mid j = 1, \ldots, \mathrm{Lkey}\}$$

and

$$T = \{(\mathrm{Xsig}_k, \mathrm{Zsig}_k) \mid k = 1, \ldots, \mathrm{Lsig}\}.$$

Alternatively, they can run 2-$\mathrm{coGHIproof}_\mathrm{Icon}$ on the same sets S and T.

Remark 4.3.1. The main goal of the setup variants is to ensure that the elements $(\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}})$ must Ygroup-generate Xgroup in order to ensure non-repudiation of signatures. Otherwise, a malicious signer might use a different homomorphism interpolating also in the set $\{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ to deny a signature. The above property actually guarantees the unicity of the secret key. In Setup Variant 1, Lkey must be large enough so that it is impossible to maliciously select a key which does not fullfil this condition. In the variant 2, Lkey can be reduced since we assume the RA does not attempt to generate an invalid key. In the variant 3 and 4, Lkey can be maximally reduced since the signer selects seedK such that a minimal number of elements $\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}$ produces a valid key. With appropriate instantiations, the two last variants often lead to $\mathrm{Lkey} = 1$ or 2.

Apart from the different setup variants, we can consider two main variants for the confirmation and denial protocols. The one with 4-move confirmation and denial protocols is called 4-move MOVA, while the other one with 2-move confirmation and denial protocols is called 2-move MOVA.

4.4 Security Properties

We devote this part to present the security results related to the MOVA scheme. We first assume that the key generated in the different setup variants is valid and prove the security of 2-move (resp. 4-move) MOVA under this assumption. We postpone the security analysis of the setup variants in Subsection 4.4.3.

4.4.1 2-Move MOVA Scheme

Here, we prove that the 2-move version of the MOVA scheme satisfies the security properties mentioned in Section 3.3. The proofs of resistance against forgery attacks and invisibility were inspired from Kurosawa and Heng [86].

Theorem 4.4.1. *Let $S = \{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ and e denote the natural logarithm base. Assuming that* GenC*,* GenS*,* GenD*, H_d, and H_c are random oracles, that signer's public key is valid, and that* TPOW *is a trapdoor one-way permutation, the* MOVA *scheme with 2-move confirmation and denial protocols satisfies the following security properties.*

1. *The confirmation (resp. denial) protocol is complete.*
2. *Let p be the smallest prime factor of d. The confirmation (resp. denial) protocol is sound: for any invalid (valid) message-signature pair, any cheating*

signer $\mathbf{S}^*$ limited to q_{H_c} (resp. q_{H_d}) queries to H_c (resp. H_d), is such that the probability

$$\mathsf{Succ}_{\mathbf{S}^*}^{\mathsf{sd\text{-}con}} < \mathsf{Succ}^{\mathsf{inv\text{-}tp}} + q_{H_c} p^{-\mathrm{Icon}}$$

(resp. $\mathsf{Succ}_{\mathbf{S}^*}^{\mathsf{sd\text{-}den}} < \mathsf{Succ}^{\mathsf{inv\text{-}tp}} + q_{H_d} p^{-\mathrm{Iden}}$), where $\mathsf{Succ}^{\mathsf{inv\text{-}tp}}$ is the maximum of $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{inv\text{-}tp}}$ among all adversaries $\mathcal{A}$ which have similar complexity as $\mathbf{S}^*$.

3. *The confirmation (resp. denial) protocol is statistical black-box straight-line zero-knowledge in the random oracle model.*

4. *The confirmation (resp. denial) protocol is perfect non-transferable.*

5. *Consider the* Lsig-S-GHI *problem with the same parameters as for the* MOVA *scheme, i.e.,* $G = \mathrm{Xgroup}$, $H = \mathrm{Ygroup}$. *Assume that for any solver $\mathcal{B}$ with a given complexity, we have*

$$\mathsf{Succ}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-}\mathrm{GHI}} \leq \varepsilon.$$

Then, any forger $\mathcal{F}$ with similar complexity using q_S signing queries and q_V queries to the confirmation/denial oracle wins the existential forgery game under an adaptive chosen-message attack with a probability

$$\mathsf{Succ}_{\mathcal{F}}^{\mathsf{ef\text{-}cma}} \leq e(1+q_S)(1+q_V)\varepsilon.$$

6. *Consider the* Lsig-S-GHID *problem with the same parameters as for the* MOVA *scheme, i.e.,* $G = \mathrm{Xgroup}$, $H = \mathrm{Ygroup}$. *Assume that for any algorithm $\mathcal{B}$ with a given complexity, we have*

$$\mathsf{Adv}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-}\mathrm{GHID}} \leq \varepsilon \quad \textit{and} \quad \mathsf{Succ}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-}\mathrm{GHI}} \leq \varepsilon'.$$

Then, any distinguisher $\mathcal{D}$ with similar complexity using q_S signing queries and q_V queries to the confirmation/denial oracle wins the invisibility game under a chosen-message attack with advantage

$$\mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv\text{-}cma}} \leq e(1+q_S)(\varepsilon + 2e(1+q_V)\varepsilon').$$

Proof. First, we note that assertions 1-4 are direct consequences of Theorem 4.2.15 and Theorem 4.2.17. It suffices only to notice that a MOVA message-signature pair (m, σ) is valid iff the corresponding set $\{(\mathrm{Xsig}_1, \mathrm{Ysig}_1), \ldots, (\mathrm{Xsig}_{\mathrm{Lsig}}, \mathrm{Ysig}_{\mathrm{Lsig}})\}$ interpolates with S in the unique homomorphism Hom.

5. Let $\mathcal{F}$ be a forger who succeeds to existentially forge a signature under an adaptive chosen-message attack with a non-negligible probability ε. We will construct an algorithm $\mathcal{B}$ which solves the Lsig-S-GHI problem with

$$S := \{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$$

using the forger $\mathcal{F}$ and the verifier's secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$. At the beginning, $\mathcal{B}$ receives the challenges $x_1, \ldots, x_{\mathrm{Lsig}} \in$ Xgroup of the Lsig-S-GHI problem. Then, $\mathcal{B}$ runs the forger and simulates the queries to the random oracle GenS, q_S queries to the signing oracle Sign and q_V queries to the denial/confirmation oracle Ver. We can assume that all messages sent to Sign resp. Ver were previously queried to GenS (since the oracle Sign resp. Ver has to make such queries anyway). $\mathcal{B}$ simulates the oracles GenS and Sign as follows.

GenS. For each message m queried to GenS, $\mathcal{B}$ maintains a list of each message and corresponding answer $(m, \mathrm{Xsig}_1, \ldots, \mathrm{Xsig}_{\mathrm{Lsig}})$. If the message was already queried, $\mathcal{B}$ outputs the corresponding answer from the list. Otherwise, he picks $a_{i,j} \in_U \mathbb{Z}_d$ and $r_i \in_U$ Xgroup uniformly at random for $i = 1, \ldots, \mathrm{Lsig}$, $j = 1, \ldots, \mathrm{Lkey}$. With probability q, he answers

$$\mathrm{Xsig}_i := dr_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j} \mathrm{Xkey}_j \quad \text{for } i = 1, \ldots, \mathrm{Lsig}.$$

We call it type-1 answer. With probability $1 - q$, the answer is

$$\mathrm{Xsig}_i := dr_i + x_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j} \mathrm{Xkey}_j \quad \text{for } i = 1, \ldots, \mathrm{Lsig}.$$

We call it type-2 answer. For each message, $\mathcal{B}$ keeps the coefficients $a_{i,j}$'s and r_i's and answer type in memory. Note that the simulation is perfect by Lemma 4.1.9, since the public key is valid.

Sign. For a message m, if the answer to the GenS query of m was of type-1, then $\mathcal{B}$ answers $\mathrm{Ysig}_i := \sum_{j=1}^{\mathrm{Lkey}} a_{i,j} \mathrm{Ykey}_j$ for $i = 1, \ldots, \mathrm{Lsig}$. Otherwise, it aborts the simulation.

Let (m_i, σ_i) denote the ith query to Ver for $1 \le i \le q_V$ and $(m_{q_V+1}, \sigma_{q_V+1})$ denote the $\mathcal{F}$ output. In order to simulate the answers of the queries made to Ver, $\mathcal{B}$ guesses the smallest i such that (m_i, σ_i) is a valid forged pair (i.e., m_i was not queried to Sign). To this, $\mathcal{B}$ simply picks ℓ uniformly at random in $\{1, \ldots, q_V + 1\}$. $\mathcal{B}$ deals with the ith query as follows.

$i < \ell$**.** To any query (m_i, σ_i), $\mathcal{B}$ checks whether m_i was submitted to Sign. If it is the case, $\mathcal{B}$ is able to decide whether (m_i, σ_i) is valid and simulates the appropriate protocol. Otherwise, $\mathcal{B}$ guesses that (m_i, σ_i) is invalid and simulate the appropriate protocol. The simulation is done perfectly as in the proof of non-transferability of the confirmation (resp. denial) protocol. For this, see the proof of the assertion 3 of Theorem 4.2.15 and Theorem 4.2.17.

$i = \ell$. Let $(m_\ell, \sigma_\ell) := (m_\ell, \mathrm{Ysig}_1, \dots, \mathrm{Ysig}_{\mathrm{Lsig}})$. If the corresponding Xsig_i's were of type-1, $\mathcal{B}$ aborts. Otherwise, when ℓ was correctly guessed

$$\mathrm{Ysig}_i = y_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Ykey}_j \quad \text{for } i = 1, \dots, \mathrm{Lsig}$$

and $\mathcal{B}$ is able to deduce the y_i's of the Lsig-S-GHI problem.

It remains to compute the probability that $\mathcal{B}$ retrieves the y_i's and did not abort. For this, we can assume that m_{q_V+1} was queried to GenS, since the probability of guessing the output of GenS is negligible. This event occurs if $\mathcal{B}$ is able to simulate all Sign queries, guess the right ℓ and use the message m_ℓ to deduce the y_i's. Therefore,

$$\Pr[\mathcal{B} \text{ succeeds} | \mathcal{F} \text{ succeeds}] = q^{q_S}(1-q)/(q_V+1).$$

As for the full-domain hash technique [43] and as in [86], the optimal value for q is $q_{\mathrm{opt}} = q_S/(q_S+1)$. Thus, the success probability is

$$\mathsf{Succ}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-GHI}} = \left(1 - \frac{1}{1+q_S}\right)^{q_S} \frac{\varepsilon}{(1+q_S)(1+q_V)}$$

which admits the lower bound $(1/e(1+q_S)(1+q_V))\varepsilon$.

6. Let $\mathcal{D}$ be a distinguisher which breaks the invisibility of the MOVA scheme with an advantage ε. We construct an algorithm $\mathcal{B}$ which solves the Lsig-S-GHID problem by using $\mathcal{D}$ and $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$. At the beginning, $\mathcal{B}$ is challenged with a tuple $\{(x_1, y_1), \dots, (x_{\mathrm{Lsig}}, y_{\mathrm{Lsig}})\} \in (\mathrm{Xgroup} \times \mathrm{Ygroup})^{\mathrm{Lsig}}$ for which it has to determine whether $\mathrm{Hom}(x_i) = y_i$ for all $1 \le i \le \mathrm{Lsig}$ or if this tuple was picked uniformly at random. Like for the proof of the existential forgery, the simulator $\mathcal{B}$ runs $\mathcal{D}$ and simulates the queries made to the random oracle GenS, q_S queries to the signing oracle Sign and the queries to the denial/confirmation oracle Ver. We can assume that each message queried to Sign or Ver was previously queried to the random oracle GenS. We assume that no query m to Ver was submitted to Sign beforehand. (Otherwise, we can just simulate them with $\mathcal{K}_{\mathrm{s}}^{\mathbf{V}}$.) Let Forge be the event in which $\mathcal{D}$ sends a valid message-signature pair to Ver. We first remove all instances for which the event Forge occurs. So, we can now assume that $\mathcal{D}$ never submits any valid pair (m, σ) to Ver. $\mathcal{B}$ simulates the oracles just like in the proof of unforgeability with $\ell = q_V + 1$ (we excluded valid forged pairs).

After a given time, the distinguisher $\mathcal{D}$ sends a message m^* to the challenger of the invisibility game which is simulated by $\mathcal{B}$. If the answer of m^* to GenS was of type-1, $\mathcal{B}$ aborts the simulation. Otherwise, it sends the challenge signature $(\mathrm{Ysig}_1^*, \dots, \mathrm{Ysig}_{\mathrm{Lsig}}^*)$ where $\mathrm{Ysig}_i^* := y_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Ykey}_j$ for $1 \le i \le \mathrm{Lsig}$. Then, $\mathcal{D}$ continues to query the oracles which are simulated by $\mathcal{B}$ as above.

Finally, $\mathcal{D}$ outputs a guess bit b'. The simulator $\mathcal{B}$ outputs the same bit b' as guess bit to the Lsig-S-GHID challenger or a random bit when $\mathcal{B}$ aborted.

Using the homomorphic property of Hom, we deduce that the set $\{(x_i, y_i)\}_{i=1}^{\text{Lsig}}$ interpolates in a group homomorphism with the set of points S if and only if $(m^*, \text{Ysig}_1^*, \ldots, \text{Ysig}_{\text{Lsig}}^*)$ is a valid message-signature pair. Hence, when the simulator does not abort and the event Forge does not occur, $\mathcal{B}$ perfectly simulates the invisibility games. It remains to compute the advantage of $\mathcal{B}$.

For a bit b, we denote A_b the probability event that $\mathcal{B}$ does not abort when the challenge to $\mathcal{B}$ was of the form T_b (thus, $\mathcal{B}$ simulates the game $\mathbf{Game}^{\textsf{inv-cma-}b}$ to $\mathcal{D}$). Note that the probability $\Pr[A_1] = \Pr[A_0]$ can be bounded in an optimal way as in the proof of existential forgery attacks, namely, by choosing q adequately we get $\Pr[A_1] \geq (1/e(1+q_S))$. We now define the events B_b and D_b which occur when $\mathcal{B}$ and $\mathcal{D}$ respectively outputs the bit 0 when the challenge was of the form T_b. Note that if A_b happens, both events B_b and D_b occurs simultaneously. Let us denote ε_0 resp. ε_1, the probability for $\mathcal{D}$ to output 0 in the game $\mathbf{Game}^{\textsf{inv-cma-0}}$ resp. $\mathbf{Game}^{\textsf{inv-cma-1}}$. We now estimate $\Pr[B_0|A_0]$ and $\Pr[B_1|A_1]$ with respect to ε_0 and ε_1. To this end, we notice that the event $B_0|A_0$ resp. $B_1|A_1$ occurs simultaneously with the event where $\mathcal{D}$ outputs 0 in the game $\mathbf{Game}^{\textsf{inv-cma-0}}$ resp. $\mathbf{Game}^{\textsf{inv-cma-1}}$, provided that the event Forge does not occur. Hence, applying the difference lemma of Shoup [138] leads to

$$|\Pr[B_b|A_b] - \varepsilon_b| \leq \Pr[\mathsf{Forge}]$$

for $b = 0, 1$. From this, we can deduce that $\Pr[B_0|A_0] \geq \varepsilon_0 - \Pr[\mathsf{Forge}]$ and $\Pr[B_1|A_1] \leq \varepsilon_1 + \Pr[\mathsf{Forge}]$. Here, we can assume that $\Pr[B_0] \geq \Pr[B_1]$. The advantage of $\mathcal{B}$ is then equal to

$$\begin{aligned}\Pr[B_0] - \Pr[B_1] = {} & \Pr[\neg A_0] \cdot (\Pr[B_0|\neg A_0] - \Pr[B_1|\neg A_1]) \\ & + \Pr[A_0] \cdot (\Pr[B_0|A_0] - \Pr[B_1|A_1]).\end{aligned}$$

Since $\Pr[B_0|\neg A_0] = \Pr[B_1|\neg A_1] = 1/2$ and $\varepsilon_0 - \varepsilon_1 = \mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv-cma}}$, we finally have

$$\mathsf{Adv}_{\mathcal{B}}^{\text{Lsig-}S\text{-GHID}} \geq \frac{1}{(1+q_S)e}\left(\mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv-cma}} - 2\Pr[\mathsf{Forge}]\right).$$

We can conclude by noting that Forge occurs with a probability bounded from above by $e(1+q_S)(1+q_V)\varepsilon'$ by assertion 5. □

Remark 4.4.2. Similarly to Laguillaumie and Vergnaud [88], the efficiency of the security reduction for the existential forgery can be improved (factor $(1+q_V)^{-1}$ is removed) by replacing the GHI problem by its *gap* variant [119]. This problem consists in solving the GHI problem using an access to an oracle which solves the corresponding GHID problem. This one helps to simulate the confirmation and denial oracles. So, we do not need to guess $\ell \in \{1, \ldots, q_V + 1\}$ to simulate these oracles correctly.

Remark 4.4.3. MOVA scheme can be made probabilistic so that the invisibility notion defined in Galbraith and Mao [62] is satisfied. To this, it suffices to append some randomness r to the message to sign and to add r in the signature. The drawback is that the signature enlarges.

4.4.2 Security of the 4-Move MOVA Scheme

We present security properties of the 4-move MOVA scheme in the security model of Section 3.3.

Theorem 4.4.4. *Let* $S = \{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ *and* e *denote the natural logarithm base. Assuming that the signer's public key is valid and* GenS *is a random oracle, the* MOVA *signature scheme with 4-move confirmation and denial protocols satisfies the following security properties.*

1. *The confirmation (resp. denial) protocol is complete.*
2. *Let* p *be the smallest prime factor of* d*. The confirmation (resp. denial) protocol is sound. From any cheating signer* $\mathbf{S}^*$ *who passes the confirmation (resp. denial) protocol on an invalid (resp. valid) signature with a probability* $\mathsf{Succ}^{\mathsf{sd\text{-}con}}_{\mathbf{S}^*} = \varepsilon$ *and an expert group knowledge of* Xgroup*, we can construct an algorithm* $\mathcal{B}$ *which finds a collision on the commitment scheme with a probability*
$$\mathsf{Succ}^{\mathsf{com\text{-}bnd}}_{\mathcal{B}} = \varepsilon(\varepsilon - p^{-\mathrm{Icon}})$$
(resp. $\varepsilon(\varepsilon - p^{-\mathrm{Iden}})$ *) by rewinding* $\mathbf{S}^*$ *once.*
3. *With a perfectly hiding trapdoor commitment, the confirmation (resp. denial) protocol is perfect black-box straight-line zero-knowledge.*
4. *With a perfectly hiding trapdoor commitment, the confirmation (resp. denial) protocol is perfect non-transferable.*
5. *Consider the* Lsig-S-GHI *problem with the same parameters as for the* MOVA *scheme, i.e.,* $G = \mathrm{Xgroup}$, $H = \mathrm{Ygroup}$*. Assume that for any solver* $\mathcal{B}$ *with a given complexity, we have*
$$\mathsf{Succ}^{\text{Lsig-}S\text{-GHI}}_{\mathcal{B}} \leq \varepsilon.$$
Then, any forger $\mathcal{F}$ *with similar complexity using* q_S *signing queries and* q_V *queries to the confirmation/denial oracle wins the existential forgery game under an adaptive chosen-message attack with a probability*
$$\mathsf{Succ}^{\mathsf{ef\text{-}cma}}_{\mathcal{F}} \leq e(1+q_S)(1+q_V)\varepsilon.$$

6. *Consider the* Lsig-S-GHID *problem with the same parameters as for the* MOVA *scheme, i.e.,* $G =$ Xgroup, $H =$ Ygroup. *Assume that for any algorithm* $\mathcal{B}$ *with a given complexity, we have*

$$\mathsf{Adv}_{\mathcal{B}}^{\text{Lsig-}S\text{-GHID}} \leq \varepsilon \quad \textit{and} \quad \mathsf{Succ}_{\mathcal{B}}^{\text{Lsig-}S\text{-GHI}} \leq \varepsilon'.$$

Then, any distinguisher $\mathcal{D}$ *with similar complexity using* q_S *signing and* q_V *queries to the confirmation/denial oracle queries wins the invisibility game under a chosen-message attack with advantage*

$$\mathsf{Adv}_{\mathcal{D}}^{\text{inv-cma}} \leq e(1+q_S)(\varepsilon + 2e(1+q_V)\varepsilon').$$

Proof. The assertions 1-4 directly follow from Theorem 4.2.1 and Theorem 4.2.7. The unforgeability and invisibility proofs work exactly as for the 2-move version of MOVA, except that we simulate the confirmation and denial oracles according to the simulator used to show non-transferability of the 4-move confirmation and denial protocols. Again, the simulation of these oracles is done perfectly. For more details, see the proof of the assertion 3 of Theorem 4.2.1 and Theorem 4.2.7. □

4.4.3 Security of the Setup Variants

Before stating our security results on the setup variants 1 and 2, we need to introduce the following technical lemma.

Lemma 4.4.5. *Let* A *be a finite Abelian* p*-group such that*

$$A \simeq \mathbb{Z}_{p^{e_1}} \oplus \mathbb{Z}_{p^{e_2}} \oplus \cdots \oplus \mathbb{Z}_{p^{e_k}}.$$

for some integers $0 < e_1 \leq e_2 \leq \cdots \leq e_k$. *Set* $e := \sum_{i=1}^{k} e_i$. *The number of maximal subgroups of* A*, i.e., of order* p^{e-1} *is equal to*

$$\frac{p^k - 1}{p-1}.$$

Proof. According to some results in combinatorial theory (see Butler [29]), the number of subgroups of A of order p^ℓ is equal to those of order $p^{e-\ell}$, for any integer $\ell < e$. Hence, our problem can be solved by enumerating all subgroups of A of order p. For this, we consider all elements of A of order p. Since any group $\mathbb{Z}_{p^{e_i}}$ contains exactly p elements of order p or which are 0 for $i = 1, \ldots, k$, we have $p^k - 1$ elements in A of order p (we just need to remove the neutral element). To conclude, it suffices to remark that any subgroup generated by an element of order p is in fact generated by $p - 1$ such elements. □

We now state our security result related to Setup Variants 1 and 2.

Theorem 4.4.6 (Setup Variants 1,2)**.** *Let* Xgroup*,* Ygroup *be some Abelian groups, and d the order of* Ygroup*. Given a prime q, we let A_q be the subgroup of* Xgroup *of any element whose order is a power of q. Given q there exists a unique sequence of integers $a_{q,1} \leq \cdots \leq a_{q,k_q}$ such that A_q is isomorphic to $\mathbb{Z}_{q^{a_{q,1}}} \oplus \cdots \oplus \mathbb{Z}_{q^{a_{q,k_q}}}$. The probability P_{gen} that some elements* $\text{Xkey}_1, \ldots, \text{Xkey}_{\text{Lkey}} \in_U$ Xgroup *picked uniformly at random* Ygroup*-generate* Xgroup *satisfies*

$$P_{\text{gen}} \geq \prod_{q \in \mathbf{P}_d} \left(1 - \frac{q^{k_q} - 1}{(q-1) \cdot q^{\text{Lkey}}}\right),$$

where $\mathbf{P}_d$ is the set of all prime factors of $\gcd(\#\text{Xgroup}, d)$*.*

Proof. By Lemma 4.1.3, we need to study the probability to generate the quotient group Xgroup$/(d \cdot$ Xgroup$)$ with some elements picked uniformly at random. Classical results on the structure of Abelian groups (see Appendix A.1) states the decomposition

$$\text{Xgroup} \simeq A_{p_1} \oplus \cdots \oplus A_{p_n}.$$

Note that

$$\text{Xgroup}/(d \cdot \text{Xgroup}) \simeq A_{p_1}/dA_{p_1} \oplus \cdots \oplus A_{p_n}/dA_{p_n}.$$

We consider $B_q := A_q/dA_q$ and study the probability that elements generate this group. If $\gcd(d, q) = 1$, then $dA_q = A_q$ and B_q is trivial. So, we only focus on the q's that divide d and denote e_q the largest integer such that $q^{e_q} | d$. We deduce that the structure of B_q satisfies

$$B_q \simeq \mathbb{Z}_{q^{a_{q,1}}} \oplus \cdots \oplus \mathbb{Z}_{q^{a_{q,r}}} \oplus \mathbb{Z}_{q^{e_q}} \oplus \cdots \oplus \mathbb{Z}_{q^{e_q}},$$

where r is the largest integer such that $a_{q,r} < e_q$. The probability P_q that Lkey elements does not generate B_q is equal to the probability that these elements stay in one of the maximal subgroups of B_q. By Lemma 4.4.5, the number of such subgroups is equal to $(q^{k_q} - 1)/(q-1)$. Therefore,

$$P_q \leq \frac{q^{k_q} - 1}{(q-1) \cdot q^{\text{Lkey}}}.$$

Since these events are independent for the different B_q's, the final probability is obtained by multiplying the terms $1 - P_q$. □

Remark 4.4.7. As an application, if d is prime and if the d-subgroup of Xgroup is a product of k cyclic groups, we have $P_{\text{gen}} \geq 1 - (d^k - 1)/(d-1) \cdot d^{-\text{Lkey}}$. In practice, we will rarely have k greater than 2 so that we approximately have a probability of $1 - d^{-\text{Lkey}+1}$.

4.4.4 Security Parameters

Signature Parameters.

Security results on the unforgeability and the invisibility of MOVA given in Theorem 4.4.1 (and also Theorem 4.4.4) allows to directly derive some bounds on the signature size provided certain assumptions on the GHI and GHID problems. We emphasize here, that the hardness of solving some GHI and GHID problems can often be scaled by adjusting the size of Xgroup independently of Ygroup.

For instance, consider Xgroup $= \mathbb{Z}_n^*$ with $n = pq$ for two large primes p, q and the Legendre symbol $(\cdot/p)$. Breaking the corresponding GHI problem corresponds here to break the quadratic residuosity assumption. Moreover, as far as we know there does not exist any algorithm which solves this problem more efficiently than factoring the modulus n, which only depends on Xgroup.

As illustrated with the above example, the advantage of the MOVA scheme is the scalable signature size depending on the required security. This one precisely depends on the adversary computational resources (as well as the number of allowed queries to the oracles), the hardness of the GHI and GHID problems and the security we want to achieve. We assume here that Xgroup is adjusted such that

$$\mathsf{Succ}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-}\mathrm{GHI}} \approx d^{-\mathrm{Lsig}} \quad \text{and} \quad \mathsf{Adv}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-}\mathrm{GHID}} \approx 0,$$

for all algorithms $\mathcal{B}$ with similar complexity as the adversary. This leads to

$$\mathsf{Succ}_{\mathcal{F}}^{\mathsf{ef\text{-}cma}} \approx e q_S q_V d^{-\mathrm{Lsig}} \quad \text{and} \quad \mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv\text{-}cma}} \approx 2e^2 q_S q_V d^{-\mathrm{Lsig}}.$$

Since the verification of an undeniable signature must be done online, we can consider some security probabilities of about 2^{-20} instead of the classical offline security of 2^{-80}. In Table 4.1, we give the required MOVA signature size in order to achieve $\mathsf{Succ}_{\mathcal{F}}^{\mathsf{ef\text{-}cma}} \approx 2^{-20}$ depending on q_S and q_V. Similar results in order to achieve $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv\text{-}cma}} \approx 2^{-20}$ are presented in Table 4.2.

q_S	q_V	$\mathrm{Lsig} \cdot \log_2(d)$ bits
2^{10}	2^{10}	42
2^{10}	2^{20}	52
2^{20}	2^{10}	52
2^{20}	2^{20}	62

Table 4.1: Signature size with unforgeability of 2^{-20}

q_S	q_V	$\mathrm{Lsig} \cdot \log_2(d)$ bits
2^{10}	2^{10}	44
2^{10}	2^{20}	54
2^{20}	2^{10}	54
2^{20}	2^{20}	64

Table 4.2: Signature size with invisibility of 2^{-20}

Protocol Parameters

Results for the soundness of the 2-move confirmation and denial protocols can be obtained using the assertion 2 of Theorem 4.4.1. Under the assumption $\mathsf{Succ}^{\text{inv-tp}} \approx 0$, we obtain

$$\mathsf{Succ}^{\text{sd-con}}_{\mathbf{S}^*} \approx q_{H_c} p^{-\mathrm{Icon}} \quad \text{and} \quad \mathsf{Succ}^{\text{sd-den}}_{\mathbf{S}^*} \approx q_{H_d} p^{-\mathrm{Iden}}.$$

For instance, we can achieve a soundness probability of 2^{-20} with the parameters $\mathrm{Icon} = \mathrm{Iden} = 60/\log_2(p)$, $q_{H_c} = q_{H_d} = 2^{40}$. Similarly, for the 4-move protocols, we assume that Commit satisfies

$$\mathsf{Succ}^{\text{com-bnd}}_{\mathcal{B}} \approx 0$$

for any algorithm $\mathcal{B}$ with similar complexity as the adversary. This shows that

$$\mathsf{Succ}^{\text{sd-con}}_{\mathbf{S}^*} \approx p^{-\mathrm{Icon}} \quad \text{and} \quad \mathsf{Succ}^{\text{sd-den}}_{\mathbf{S}^*} \approx p^{-\mathrm{Iden}},$$

which leads to some smaller parameters Icon and Iden. Namely, for a soundness probability of 2^{-20}, we get $\mathrm{Icon} = \mathrm{Iden} = 20/\log_2(p)$.

Setup Parameters

We examine here the size of parameters of the different setup variants according to their specificity. First, we note that security of Setup Variant 1 and 2 is directly deduced from the results of Theorem 4.4.6. The main difference is simply due to the number of attempts the adversary can perform until he gets some "bad" elements $\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}$. In the first variant, the signer can try as many attempts as he can so that we require an "offline" probability $P_{\mathrm{gen}} \geq 1 - 2^{-80}$, while in the second one he is very limited so that we require an "online" probability $P_{\mathrm{gen}} \geq 1 - 2^{-20}$. Assuming similar assumptions as in Remark 4.4.7, we get $\mathrm{Lkey} = 81/\log_2(d)$ for Setup variant 1 and $\mathrm{Lkey} = 21/\log_2(d)$ for Setup Variant 2. This means that the tuple $(\mathrm{Ykey}_1, \ldots, \mathrm{Ykey}_{\mathrm{Lkey}})$ which is contained in the public key would be 81 and 21 bits long respectively. For Setup Variant 3, we derive security bounds from the results obtained for MGGDproof given in Theorem 4.2.11. As for the soundness of the confirmation and denial protocols, we get a probability that the signer passes the protocol with an invalid public key of 2^{-20} with $\mathrm{Ival} = 20/\log_2(p)$ assuming that no efficient adversary breaks the computationally binding property of Commit. Finally, Theorem 4.2.13 gives results for Setup Variant 4. Since Xgroup is usually greater than p, we have

$$\mathsf{Succ}_{\mathbf{P}^*}^{\mathsf{sd}\text{-NIMGGD}} \approx q_{\mathrm{GenM}} \cdot p^{-\mathrm{Ival}},$$

which leads to a probability of 2^{-20} with $q_{\mathrm{GenM}} = 2^{60}$ and $\mathrm{Ival} = 80/\log_2(p)$.

4.5 Additional Properties

We briefly discuss two additional properties of the MOVA scheme, one may desire under certain circumstances.

4.5.1 Batch Verification

We point out that the MOVA scheme allows a batch verification of signatures. Indeed, the confirmation protocol can be easily adapted in order to confirm several signatures at the same time by putting all pairs $(\mathrm{Xsig}_k, \mathrm{Ysig}_k)$ in a single set S. We propose a batch confirmation protocol in which the prover performs a batch verification of the valid message-signature pairs among those sent by the verifier. This protocol is depicted in Figure 4.10.

```
BatchConf_ℓ(SIG)
Parameter: $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$, $\ell$
Input: A set of message-signature pairs $\mathrm{SIG} = \{(m_1,\sigma_1),\ldots,(m_s,\sigma_s)\}$
    supposedly valid with respect to the MOVA scheme's public key $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$.
 1: The prover and the verifier generate $\mathrm{GenS}(m_i) \to \mathrm{Xsig}_1^i,\ldots,\mathrm{Xsig}_{\mathrm{Lsig}}^i$
    and directly retrieve $\mathrm{Ysig}_1^i,\ldots,\mathrm{Ysig}_{\mathrm{Lsig}}^i$ from the signa-
    ture $\sigma_i$ for $i = 1,\ldots,s$. They also both retrieve the set
    $\mathrm{SK} := \{(\mathrm{Xkey}_1,\mathrm{Ykey}_1),\ldots,(\mathrm{Xkey}_{\mathrm{Lkey}},\mathrm{Ykey}_{\mathrm{Lkey}})\}$ from $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$.
 2: For any $i = 1,\ldots,s$, the prover checks whether $\mathrm{Hom}(\mathrm{Xsig}_j^i) = \mathrm{Ysig}_j^i$
    for all $j = 1,\ldots,\mathrm{Lsig}$. He puts in memory all indices $(i_1,\ldots,i_k) =: I$
    which do not satisfy this property and sends them to the verifier.
 3: The prover and the verifier perform $\mathrm{GHIproof}_\ell(\mathrm{SIG}' \cup \mathrm{SK})$, where the
    set $\mathrm{SIG}'$ corresponds to SIG without the message-signature pairs whose
    index is in $I$.
```

Figure 4.10: Batch confirmation protocol

Security results related to GHIproof given in Theorem 4.2.1 directly applies to this batch variant. In particular, the prover cannot pass the protocol with a probability greater than $p^{-\ell}$ if SIG' contains an invalid signature and assuming that the commitment is binding. However, note that the prover may send some indices corresponding to valid signatures. To avoid this, the verifier can subsequently require to perform denial protocols to the signatures corresponding to these indices. In addition to the clear gain in terms of rounds, this technique allows to decrease the computational complexity mainly for the prover. Namely, when all signatures are valid, the prover has $\ell(s-1)$ less homomorphism evaluations to perform compared to a sequential verification of all signatures. This is particularly important to focus on the prover's complexity, since one prover is likely to share several verifiers in most practical applications.

The denial protocol can be modified similarly to show that at least one signature is invalid among a bunch of signatures using one protocol run. However, a batch version of the denial protocol is not likely to be used for concrete applications, since we usually want to precisely know which signatures are invalid.

4.5.2 Selective Convertibility

We show here how a signer with an expert group knowledge of Xgroup with the set $S_1 = \{\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}\}$ can convert a given MOVA signature in a universally verifiable signature. From a valid message-signature (m, σ) with $\sigma = (\mathrm{Ysig}_1, \ldots, \mathrm{Ysig}_{\mathrm{Lsig}})$ the signer finds some coefficients $r_i \in \mathrm{Xgroup}$ and $a_{i,j} \in \mathbb{Z}_d$ for $i = 1, \ldots, \mathrm{Lsig}$, $j = 1, \ldots, \mathrm{Lkey}$ such that

$$\mathrm{Xsig}_i = dr_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Xkey}_j \quad \text{for } i = 1, \ldots, \mathrm{Lsig}. \tag{4.13}$$

The converted signature is $\sigma_{\mathrm{conv}} := (\sigma, r_i\text{'s}, a_{i,j}\text{'s})$. The verification of converted pair $(m, \sigma_{\mathrm{conv}})$ consists in retrieving the Xsig_i's from m and verifying Equation (4.13) and the following equation

$$\mathrm{Ysig}_i = \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Ykey}_j \quad \text{for } i = 1, \ldots, \mathrm{Lsig}.$$

Note that this holds if and only if σ is a valid MOVA signature. Although the conversion may be useful in some applications, the major drawback is that the converted signature is usually quite large. Moreover, the advantage of this method compared to signing the message m with another universally verifiable signature is not clear at all. In other words, the link between the converted signature and the regular MOVA signature does not seem to be crucial. In general, converting a signature is performed to allow the recipient to verify the authenticity of the message without the help of the signer.

Chapter 5

Characters on $\mathbb{Z}_n^*$ and Applications to MOVA

Group characters on $\mathbb{Z}_n^*$ played a central role in the development of MOVA scheme since the first variant of the scheme [109] was based on this sole concept. They are typical examples of homomorphisms with a very small range allowing to basically produce MOVA signatures of any size depending on a security parameter. Thereby, they perfectly illustrate the full scalability of MOVA signatures. In this chapter, we focus on characters of order $2, 3, 4$ which arise naturally from the theory of quadratic, cubic, and biquadratic residuosity, respectively. So, characters of small order defined on $\mathbb{Z}_n^*$ can be seen as a natural generalization of the Legendre symbol and related problems generalize the quadratic residuosity problem.

We also want to mention that characters were already used in public-key cryptography for the design of algorithms as well as a useful tool in security proofs. Namely, the former is illustrated by the design of public-key cryptosystems due to Scheidler and Williams [135, 136]. The latter is a recent result due to Okamoto and Stern [120] who used the theory of characters to complete the security proof of the signature scheme ESIGN [117].

This chapter mainly deals with the theoretical aspects of characters and their implications to the MOVA scheme. After an introduction of the theoretical background, we show how to select characters which are particularly relevant for MOVA instantiations. Related computational problems are presented in the subsequent section as well as a treatment of the reductions (we could exhibit) between one another. Then, as a rather purely theoretical interest, we focus on a special variant which does not make use of prime numbers. Finally, we examine the concept of expert group knowledge in this setting clarifying how to use Setup Variants 3 and 4 with these MOVA instantiations.

5.1 Characters on $\mathbb{Z}_n^*$

In this section, we introduce the notion of multiplicative characters. Although characters can be defined on any group, we concentrate here on $\mathbb{Z}_n^*$. Namely, characters based on another kind of group will not be considered for MOVA instantiations. In particular, our treatment will focus on the characters of order 2, 3, and 4. Most of the material developed in this section is taken from the book of Ireland and Rosen [79] and that of Nathanson [114].

Definition 5.1.1. *Let G be a finite Abelian group. A character χ on G is a group homomorphism from G to $(\mathbb{C}\backslash\{0\},\cdot)$, i.e., a map $\chi : G \rightarrow \mathbb{C}\backslash\{0\}$ satisfying*

$$\chi(a+b) = \chi(a)\chi(b)$$

for any $a, b \in G$.

From the definition of a character, we can quickly deduce that $\chi(1) = 1$ and that the value $\chi(a)$ is always a $\lambda(G)$th root of the unity for all $a \in G$, where $\lambda(G)$ denotes the exponent of the group G, i.e., the maximal order an element of G can have. A group structure can be defined on the set of all characters on G. In this group, the product (group operation) $\chi_1\chi_2$ of two characters χ_1 and χ_2 is defined by

$$\chi_1\chi_2(a) = \chi_1(a)\chi_2(a)$$

for all $a \in G$. Similarly, the inverse χ^{-1} of a character χ is defined by

$$\chi^{-1}(a) = \chi(a)^{-1}$$

for any $a \in G$. The group of the characters defined on G is called the *dual* group of G and is denoted $\widehat{G}$. The structure of this group is given by the following theorem.

Theorem 5.1.2 ([114], pp. 129)**.** *A finite Abelian group G is isomorphic to its dual, i.e.,*

$$G \simeq \widehat{G}.$$

In the rest of this section, we restrict to the multiplicative group $G = \mathbb{Z}_n^*$. Applying the above results to $\mathbb{Z}_n^*$ shows that $\chi(a)$ is a $\lambda(n)$th root of the unity for any $a \in \mathbb{Z}_n^*$ and any character χ defined on $\mathbb{Z}_n^*$. We also note that a character can be naturally extended on the whole ring $\mathbb{Z}_n$ by setting $\chi(a) = 0$ for any non invertible element a. From Theorem 5.1.2, we readily get a characterization of the characters on $\mathbb{Z}_n^*$. This result for a prime n is given here.

Corollary 5.1.3. *Let p be a prime and d be an integer such that $d|p-1$.*

1. *The group of characters defined on $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$.*

2. *The characters on $\mathbb{Z}_p^*$ of order dividing d form a cyclic subgroup of order d.*

A proof of this corollary which does not make use of Theorem 5.1.2 can be found in Chapter 8 of [79].

The second part of this corollary is especially interesting for us, because we will consider characters of small order (e.g., 2, 3, 4). In addition, we deduce that a character on $\mathbb{Z}_p^*$ of order d maps the elements of $\mathbb{Z}_p^*$ to the set

$$\{\zeta_d^j \mid j = 0, \ldots, d-1\},$$

where ζ_d denotes the unit $e^{2\pi i/d}$ and $i := \sqrt{-1}$.

In the following subsections we present additional material which is specific to the cases $d = 2, 3, 4$.

5.1.1 Characters of Order 2

Let p be an odd prime. By Corollary 5.1.3, we know that there are only two characters on $\mathbb{Z}_p^*$ of order dividing 2, namely the trivial character ϵ that maps every elements to 1 and the Legendre symbol. We recall that the Legendre symbol (a/p) for an integer a with $\gcd(a, p) = 1$, is 1 if a is congruent to a square modulo p (quadratic residue) and -1 if it is not the case (quadratic non-residue). It turns out that there are $\frac{p-1}{2}$ quadratic residues resp. non quadratic residues in $\mathbb{Z}_p^*$.

For an odd integer n, the Jacobi symbol (a/n) for an $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$ is defined as

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{i_1} \cdot \left(\frac{a}{p_2}\right)^{i_2} \cdots \left(\frac{a}{p_k}\right)^{i_k},$$

where the factorization into primes of n is $p_1^{i_1} \cdots p_k^{i_k}$. The main properties of the Jacobi symbol are given below.

Proposition 5.1.4. *Let p be an odd prime, $a, b \in \mathbb{Z}$ and an odd $n \in \mathbb{Z}$. The following properties hold.*

1. $a^{(p-1)/2} \equiv (a/p) \pmod p$.
2. Multiplicativity: $(ab/n) = (a/n)(b/n)$.
3. Modularity: *If* $a \equiv b \pmod n$, *then* $(a/n) = (b/n)$.
4. Law of Quadratic Reciprocity: *If a and b are odd and positive, we have*
$$\left(\frac{a}{b}\right)\left(\frac{b}{a}\right) = (-1)^{\frac{a-1}{2} \cdot \frac{b-1}{2}}.$$
5. Complementary Reciprocity Law: $(2/n) = (-1)^{\frac{n^2-1}{8}}$.

We emphasize that the above result shows that the Legendre symbol is a well defined homomorphism on $\mathbb{Z}_p^*$.

5.1.2 Characters of Order 3

From now on, we assume some basic knowledge about integral domains, Euclidean domains, and related notions such as irreducible and prime elements. A short reminder about these notions is developed in Appendix A.2.

Eisenstein Integers

Here, we introduce the ring of Eisenstein integers. Indeed, this ring is the natural structure to study the characters of order 3 or the cubic residuosity. Most of the results below are taken from Chapter 9 of Ireland and Rosen [79].

In what follows, ω will denote the complex number

$$\omega = \frac{-1+\sqrt{-3}}{2}.$$

Note that ω is a non trivial cube root of 1 and satisfies $\omega^2+\omega+1=0$. We define the ring of the *Eisenstein integers* as the set

$$\mathbb{Z}[\omega] := \{a+b\omega \mid a,b \in \mathbb{Z}\}$$

with the classical operations (addition, multiplication) inherited from $\mathbb{C}$.

For an element $\alpha \in \mathbb{Z}[\omega]$, we define the norm $N(\alpha)=\alpha\bar{\alpha}$, where $\bar{\alpha}$ denotes the complex conjugate of α. This is the classical (squared) norm induced by the complex plane. From this definition, we have

$$N(a+b\omega) = a^2 - ab + b^2.$$

It can be shown that $\mathbb{Z}[\omega]$ is an Euclidean domain with respect to this norm. Hence, it is a unique factorization domain (see Appendix A.2), i.e., every element can be decomposed in a product of prime elements uniquely up to a unit. Note that the units (invertible elements) of $\mathbb{Z}[\omega]$ are ± 1, $\pm\omega$, $\pm\omega^2$. This leads that any non zero element in $\mathbb{Z}[\omega]$ has 6 associates. We write $\alpha \sim \beta$ if two elements α and β are associates. The greatest common divisor is defined up to an associate. So, we will always use the notation $\gcd(\alpha,\beta) \sim \gamma$ to say that γ is a greatest (with respect to $|\cdot|$) common divisor of α and β. To avoid some confusion, a prime of $\mathbb{Z}$ will be called a *rational prime* when the context is not clear. The classification of all prime numbers of $\mathbb{Z}[\omega]$ is given below.

Proposition 5.1.5. *The following statements describe all primes of $\mathbb{Z}[\omega]$ (up to a unit).*

1. *Let p be a rational prime such that $p \equiv 1 \pmod 3$. There exists a prime π satisfying*
$$N(\pi) = \pi\bar{\pi} = p.$$

2. *If q is a rational prime such that $q \equiv 2 \pmod 3$, then q is also a prime in $\mathbb{Z}[\omega]$.*

3. *$1 - \omega$ is prime and $N(1-\omega) = 3$.*

Note that the above list contains implicitly all the related associate elements as well. The ideal generated by a $\sigma \in \mathbb{Z}[\omega]$ is denoted by (σ) and is equal to $\sigma \cdot \mathbb{Z}[\omega]$. The following proposition provides a characterization of the ring of residue classes modulo a prime.

Proposition 5.1.6. *Let π be a prime in $\mathbb{Z}[\omega]$. Then $\mathbb{Z}[\omega]/(\pi)$ is a finite field with $N(\pi)$ elements.*

In addition, when $N(\pi) = p$ is a rational prime with $p \equiv 1 \pmod 3$, we can prove that the set $\{0, 1, 2 \ldots, p-1\}$ forms all representatives of the residue class field $\mathbb{Z}[\omega]/(\pi)$.

Cubic Residue Character

We are now in position to define a generalization of the Legendre symbol of order 3.

Proposition 5.1.7. *Let π be a prime such that $N(\pi) \neq 3$ and $\alpha \in \mathbb{Z}[\omega]$ such that $\pi \nmid \alpha$. There exists a unique integer $i \in \{0, 1, 2\}$ such that*

$$\alpha^{\frac{N(\pi)-1}{3}} \equiv \omega^i \pmod \pi. \tag{5.1}$$

Definition 5.1.8. *Let α and π be as in Proposition 5.1.7. The* cubic residue character *of α modulo π is the element ω^i given by* (5.1)*. It is denoted $\chi_\pi(\alpha)$ or $(\alpha/\pi)_3$.*

Remark that $\chi_\pi(\alpha) = 1$ if and only if α is a cubic residue modulo π, i.e., there exists $x \in \mathbb{Z}[\omega]$ such that $x^3 \equiv \alpha \pmod \pi$. We can extend the cubic residue character by setting $\chi_\pi(\alpha) = 0$, for any $\alpha \equiv 0 \pmod \pi$.

Let α and β be in $\mathbb{Z}[\omega]$ with $3 \nmid N(\beta)$. Assume the prime factorization of β is $u \prod_{i=1}^k \pi_i^{e_i}$ where $N(\pi_i) \neq 3$ for all $1 \leq i \leq k$ and u is a unit. Then the Jacobi-like symbol $\chi_\beta(\alpha)$ (also denoted $(\alpha/\beta)_3$) is defined as

$$\chi_\beta(\alpha) = \prod_{i=1}^k \chi_{\pi_i}(\alpha)^{e_i}.$$

In order to formulate the law of cubic reciprocity, we have to introduce the concept of "primary".

Definition 5.1.9. *We say that an element α of $\mathbb{Z}[\omega]$ is primary if $\alpha \equiv -1 \pmod 3$.*

Note that the term "primary" does not only apply to prime numbers. Every element α such that $1-\omega \nmid \alpha$ possesses exactly one associate that is primary. This notion allows to choose exactly one associate among the possible 6. This is similar as choosing the sign of an element in $\mathbb{Z}$. The analog notion of "primary" in $\mathbb{Z}$ is the notion of negative integer.

Theorem 5.1.10. *Let α, α', β be some Eisenstein integers such that* $\gcd(\alpha,\beta) \sim \gcd(\alpha',\beta) \sim 1$ *and* $1-\omega \nmid \beta$. *The following properties hold.*

1. Modularity: *If $\alpha \equiv \alpha' \pmod{\beta}$, then $\chi_\beta(\alpha) = \chi_\beta(\alpha')$.*
2. Multiplicativity: $\chi_\beta(\alpha\alpha') = \chi_\beta(\alpha)\chi_\beta(\alpha')$.
3. Law of Cubic Reciprocity: *If α and β are primary, then*
$$\chi_\beta(\alpha) = \chi_\alpha(\beta).$$
4. Complementary Reciprocity Laws: *For a primary $\sigma = 3(a+b\omega)-1$ with $a, b \in \mathbb{Z}$, we have*
$$\chi_\sigma(\omega) = \omega^{a+b} \quad \textit{and} \quad \chi_\sigma(1-\omega) = \omega^{2a}.$$

The above treatment on the cubic residuosity allows us to define the characters of order 3 on $\mathbb{Z}_p^*$ for a rational prime p. We consider only the case where $p \equiv 1 \pmod{3}$, since the characters are trivial otherwise. Set $p = \pi\bar{\pi}$. Recall first that the field $\mathbb{Z}[\omega]/(\pi)$ can be represented by $\mathbb{Z}_p^*$ since the set $\{0, 1, \ldots, p-1\}$ contains all representatives and the multiplications are equivalent in both cases. Thus, the cubic residue character χ_π is naturally defined on $\mathbb{Z}_p^*$. We directly deduce that χ_π^2 is another non trivial character of order 3 and is equal to $\chi_{\bar{\pi}}$ on the rational integers.

5.1.3 Characters of Order 4

Gaussian Integers

Studying the characters of order 4 consists mainly in the theory of biquadratic residuosity. This one is quite similar to that of cubic residuosity and arises in the ring of *Gaussian integers*
$$\mathbb{Z}[i] := \{a + bi \mid a, b \in \mathbb{Z}\}.$$

The norm of an element $\alpha = a + bi$ is defined by
$$N(\alpha) = \alpha \cdot \bar{\alpha} = a^2 + b^2,$$

where $\bar{\alpha}$ denotes the complex conjugate of α. $\mathbb{Z}[i]$ is well known to be Euclidean which implies that the factorization is unique up to a unit (see Appendix A.2). The units (invertible elements) of $\mathbb{Z}[i]$ are ± 1, $\pm i$, which implies that any non zero element has 4 associates. Again, we use the symbol $\sim$ to say that two Gaussian integers are associate and the greatest common divisor of two Gaussian integers is uniquely defined up to an associate. The classification of all primes of $\mathbb{Z}[i]$ is provided here.

Proposition 5.1.11. *The following statements describe all primes of $\mathbb{Z}[i]$ (up to a unit).*

1. *Let p be a rational prime such that $p \equiv 1 \pmod 4$. There exists a prime π satisfying*
$$N(\pi) = \pi\bar{\pi} = p.$$
2. *If q is a rational prime such that $q \equiv 3 \pmod 4$, then q is also a prime in $\mathbb{Z}[i]$.*
3. *$1+i$ is prime and $N(1+i) = 2$.*

Proposition 5.1.12. *Let $\pi \in \mathbb{Z}[i]$ be a prime. Then, $\mathbb{Z}[i]/(\pi)$ is a finite field with $N(\pi)$ elements.*

As for $\mathbb{Z}[\omega]$, if $N(\pi) = p$ is a rational prime with $p \equiv 1 \pmod 4$, the set $\{0, \ldots, p-1\}$ form all representatives of $\mathbb{Z}[i]/(\pi)$.

Quartic Residue Character

To prepare the definition of the quartic residue character, we first consider the following result.

Proposition 5.1.13. *Let π be a prime such that $N(\pi) \neq 2$ and $\alpha \in \mathbb{Z}[i]$ such that $\pi \nmid \alpha$. There exists a unique integer $0 \leq j \leq 3$ satisfying*
$$\alpha^{\frac{N(\pi)-1}{4}} \equiv i^j \pmod \pi. \tag{5.2}$$

Definition 5.1.14. *Let α and π be as in Proposition 5.1.13. The* quartic residue character *of α modulo π is the element i^j given by* (5.2)*. It is denoted $\chi_\pi(\alpha)$ or $(\alpha/\pi)_4$.*

Note that $\chi_\pi(\alpha) = 1$ is equivalent to show that α is a quartic residue modulo π. The quartic residue character can also be trivially extended on any $\alpha \equiv 0 \pmod \pi$ by setting $\chi_\pi(\alpha) = 0$. We define the corresponding Jacobi-like character by setting
$$\chi_\beta(\alpha) = \prod_{j=1}^{k} \chi_{\pi_j}(\alpha)^{e_j}$$

for any $\alpha, \beta \in \mathbb{Z}[i]$ such that $1+i \nmid \beta$ and where $u \prod_{j=1}^{k} \pi_j^{e_j}$ is the prime decomposition of β.

Definition 5.1.15. *An element $\alpha \in \mathbb{Z}[i]$ is primary if $\alpha \equiv 1 \pmod{(1+i)^3}$. If we set $\alpha = a + bi$, this is equivalent to*

$$a \equiv 1,\ b \equiv 0 \pmod 4 \quad \textit{or} \quad a \equiv 3,\ b \equiv 2 \pmod 4.$$

One can prove that any element α with $1+i \nmid \alpha$ has a unique primary associate.

Theorem 5.1.16. *Let $\beta = a + bi$, α and α' be some Gaussian integers such that $\gcd(\beta, \alpha) \sim \gcd(\beta, \alpha') \sim 1$ and $(1+i) \nmid \beta$. The following properties hold.*

1. Modularity: *If $\alpha \equiv \alpha' \pmod \beta$, then $\chi_\beta(\alpha) = \chi_\beta(\alpha')$.*
2. Multiplicativity: *$\chi_\beta(\alpha\alpha') = \chi_\beta(\alpha)\chi_\beta(\alpha')$.*
3. Quartic Reciprocity Law: *If α and β are primary,*
$$\chi_\alpha(\beta) = \chi_\beta(\alpha) \cdot (-1)^{\frac{N(\alpha)-1}{4} \cdot \frac{N(\beta)-1}{4}}.$$
4. Complementary Reciprocity Laws: *If β is primary,*
$$\chi_\beta(i) = i^{\frac{N(\beta)-1}{4}} \quad \textit{and} \quad \chi_\beta(1+i) = i^{\frac{a-b-b^2-1}{4}}.$$

From the above theory, we can find the characters of order dividing 4 on $\mathbb{Z}_p^*$. Note that p leads to 4 characters only if $p \equiv 1 \pmod 4$ by Corollary 5.1.3. Otherwise, the group of characters is trivial or is composed of the characters of order 2 if $p \equiv 3 \pmod 4$. Such a p is the norm of a prime element π. Since the set $\{0, 1, \ldots, p-1\}$ is a complete set of representatives of $\mathbb{Z}[i]/(\pi)$, the character χ_π is also well defined on $\mathbb{Z}_p^*$. Moreover, χ_π generates the group of all characters on $\mathbb{Z}_p^*$ of order dividing 4. We note that χ_π^2 is the Legendre symbol $(\cdot/p)_2$ and that $\chi_\pi^3 = \chi_{\bar{\pi}}$.

5.1.4 Characters of Higher Order

It is possible to extend our character constructions to some orders greater than 4. This is done by introducing a power residue symbol defined on the integers of a cyclotomic field. A general treatment of these cases would be beyond the scope of this work. Moreover, the computation seems to be more difficult to deal with and the ring of these integers becomes a non unique factorization domain when the order is large. Since such a ring is not a principal ideal domain, we should work with ideals that are generated by more than one element. However, we do not loose the existence of the reciprocity laws, namely there exists a so called Kummer's reciprocity law (see the textbook of Lemmermeyer [94]).

5.2 Instantiations for the MOVA Scheme

We provide a way to define certain multiplicative characters on $\mathbb{Z}_n^*$ for an $n = pq$ being the product of two primes. So, we will consider MOVA instantiations with Xgroup $= \mathbb{Z}_n^*$. The way of defining all characters on $\mathbb{Z}_n^*$ is mainly based on the following result.

Lemma 5.2.1 ([114], pp. 128)**.** *Let G be a finite Abelian group and let $G_1, \ldots, G_k$ be subgroups of G such that $G = G_1 \oplus \cdots \oplus G_k$. For every character $\chi \in \widehat{G}$, there exist unique characters $\chi_i \in \widehat{G_i}$ such that if $g \in G$ and $g = g_1 + \cdots + g_k$ with $g_i \in G_i$ for $i = 1, \ldots, k$, then*

$$\chi(g) = \chi_1(g_1) \cdots \chi_k(g_k).$$

Moreover,

$$\widehat{G} \simeq \widehat{G_1} \oplus \cdots \oplus \widehat{G_k}.$$

From this result and Chinese Remainder Theorem on $\mathbb{Z}_n^*$, we deduce that any character on $\mathbb{Z}_n^*$ can be defined from any characters on $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$. The corresponding character on $\mathbb{Z}_n^*$ can be obtained in the same way as the Jacobi symbol is defined from the Legendre symbol. To illustrate this technique, assume we have an integer d such that $d|p-1$ and $d|q-1$. From two characters χ_1 and χ_2 of order dividing d defined on $\mathbb{Z}_p^*$ respectively $\mathbb{Z}_q^*$, the character χ of order dividing d is obtained from

$$\chi(a) := \chi_1(a \bmod p) \cdot \chi_2(a \bmod q).$$

for any $a \in \mathbb{Z}_n^*$. Lemma 5.2.1 ensures that all characters χ on $\mathbb{Z}_n^*$ of order dividing d can be defined in this way.

For any character χ of order d we will associate a logarithm function denoted as $\log_\chi$. We set $\zeta_d := e^{2\pi i/d}$. For an element $a \in \mathbb{Z}_n^*$, we know that $\chi(a)$ is of the form ζ_d^j for a $j \in \{0, 1, \ldots d-1\}$. We define $\log_\chi(a) := \log_{\zeta_d}(\chi(a))$ which is equal to this j. Hence, the homomorphism $\log_\chi$ has the associated range group Ygroup $= \mathbb{Z}_d$.

5.2.1 Characters of Order 2

Here, p and q stands for any different odd primes. From the above discussion, we deduce that the complete list of characters of order 2 on $\mathbb{Z}_n^*$ is $(\cdot/p)$, $(\cdot/q)$, $(\cdot/n)$ and the trivial character. Note that the properties given in Proposition 5.1.4 are used in order to compute the Jacobi symbol $(\cdot/n)$ in a time complexity of $O(\log(n)^2)$. Furthermore, note that the ability to compute $(\cdot/p)$ (without knowing p) is equivalent to the ability to compute $(\cdot/q)$ and implies to solving the quadratic residuosity problem.[1]

[1]We point out that security of the first semantically secure public-key cryptosystem proposed by Goldwasser and Micali [73] is based on this problem.

5.2.2 Characters of Order 3

Let p, q be two different rational odd primes such that $p \equiv q \equiv 1 \pmod 3$ and π, $\sigma \in \mathbb{Z}[\omega]$ such that $N(\pi) = p$ and $N(\sigma) = q$. Set $n = pq$. The character on $\mathbb{Z}_n^*$ produced by χ_π and χ_σ is denoted by $\chi_{\pi\sigma}$ and is defined by

$$\chi_{\pi\sigma}(a) = \chi_\pi(a) \cdot \chi_\sigma(a).$$

The other characters are defined exactly in the same multiplicative way. There are 8 non trivial characters of order 3 defined on $\mathbb{Z}_n^*$, namely χ_π, $\chi_{\bar\pi}$, χ_σ, $\chi_{\bar\sigma}$, $\chi_{\pi\sigma}$, $\chi_{\bar\pi\sigma}$, $\chi_{\pi\bar\sigma}$ and $\chi_{\bar\pi\bar\sigma}$. Without loss of generality, it suffices to consider only χ_π and $\chi_{\pi\sigma}$.

Here, we explain how these characters can be found[2]. The main problem consists in finding a prime $\pi \in \mathbb{Z}[\omega]$ such that $N(\pi) = p \equiv 1 \bmod 3$ for a given rational odd prime p. We first show that -3 is a quadratic residue modulo p. Namely, from the law of quadratic reciprocity we have

$$\left(\frac{-3}{p}\right) = (-1)^{\frac{p-1}{2}} \left(\frac{p}{3}\right) (-1)^{\frac{p-1}{2}\cdot\frac{3-1}{2}} = \left(\frac{1}{3}\right) = 1.$$

Therefore, we can find a square root u of -3 modulo p and we obtain the following equation on $\mathbb{Z}[\omega]$

$$u^2 + 3 = (u + 1 + 2\omega)(u - 1 - 2\omega) = kp$$

for an integer k. Since $p > 2$, we deduce that $p \nmid u + 1 + 2\omega$ and $p \nmid u - 1 - 2\omega$, which shows that $\gcd(u + 1 + 2\omega, p)$ is not trivial. Thus, we choose

$$\pi = \gcd(u + 1 + 2\omega, p).$$

We mention that u can be computed by the algorithm of Tonelli and Shanks which computes the square root of any quadratic residue modulo a prime. Details about this algorithm can be found in the book of Cohen [42]. Note also that computing the gcd can be done with the classical Euclid algorithm, since Euclidean division exists in $\mathbb{Z}[\omega]$.

We would now like to summarize another method for finding π. It is based on solving the equation $a^2 - ab + b^2 = p$ with respect to the integer variables a and b. By making the change of variables $s = 2a - b$ and $t = b$, a and b can be found by solving the equation

$$s^2 + 3t^2 = 4p.$$

Applying the modified Cornacchia algorithm allows to solve this equation and to find a and b. The details about this algorithm are given in the book of Cohen [42].

Once cubic residue characters are found, they can be evaluated using properties of Theorem 5.1.10. Namely, based on these properties algorithms with quadratic complexity were developed. As an example, we refer to Damgård and Frandsen [48].

[2]This method is very similar as the proof of Proposition 9.1.4 of Ireland and Rosen [79]

5.2.3 Characters of Order 4

We consider two rational primes p and q such that $p \equiv q \equiv 1 \pmod 4$ and $\pi, \sigma \in \mathbb{Z}[i]$ such that $N(\pi) = p$ and $N(\sigma) = q$. We set $n = pq$. All the characters of order dividing 4 defined on $\mathbb{Z}_n^*$ are generated by χ_π and χ_σ. If we exclude the trivial character and those of order 2 (Legendre and Jacobi symbols), it remains 12 characters of order 4. In this thesis, we will only work with χ_π and $\chi_{\pi\sigma}$.

We show below how these characters can concretely be found. To this end, we first find an integer u which is a square root of -1 modulo p. Such a u always exists since $(-1/p) = (-1)^{(p-1)/2} = 1$ whenever $p \equiv 1 \pmod 4$. From this, we get the equality (on $\mathbb{Z}[i]$)

$$u^2 + 1 = (u+i)(u-i) = kp$$

for an integer k. We note that $p > 1$ implies that $p \nmid u + i$ and $p \nmid u - i$. Hence, $\gcd(u+i, p)$ cannot be trivial and we can choose

$$\pi = \gcd(u+i, p).$$

As for the characters of order 3, u can be computed by the algorithm of Tonelli and Shanks and the gcd can be found using the classical Euclid algorithm, since Euclidean division exists in $\mathbb{Z}[i]$. It is also worth to mention that $\pi = a + bi$ can be found by solving the equation $a^2 + b^2 = p$ in a and b using the algorithm of Cornacchia (see also Cohen [42]). Moreover, efficient algorithms using properties of Theorem 5.1.16 evaluate quartic residue characters in $O(\log(n)^2)$. These algorithmic aspects will be treated more precisely in Chapter 6.

5.2.4 A Variant with Two Levels of Secret

We discuss in this part special characters with two levels of secret. We consider the characters of the form $\chi_{\pi\sigma}$ as defined in the previous subsections. Such characters can be of order 3 or 4 depending on whether π and σ are picked in $\mathbb{Z}[\omega]$ or in $\mathbb{Z}[i]$. We point out that the knowledge of the value $\alpha := \pi\sigma$ does not easily lead to the factorization of $n = N(\alpha)$. This shows that a signer may have the ability to generate MOVA signatures without factorization knowledge of n. The latter provides expert group knowledge of $\mathbb{Z}_n^*$ with some appropriate set. Details about this precise point will be given in Section 5.5. At this time, it is only important to remark that we have two levels of secrecy, a partial one allowing to generate, confirm and deny MOVA signatures while the second allows also to convert signatures.

In addition to the academic and conceptual interest of such a property, a possible useful application is illustrated with the following scenario. We consider a mobile delegate of a company who signs some pre-agreement on some contracts or transactions using $\chi_{\pi\sigma}$ which can be confirmed by a server of the company. Later, the delegate sends a report to his company, which then can issue an ordinary signature for a final agreement by converting the signature of the delegate. In such a scenario, even if the delegate loses his key or it is stolen, he can contact the company before a confirmation of the signature is performed. In any case, the company never converts a signature before being convinced that the delegate key was not lost or stolen. This is ensured by waiting on the report sent by the delegate.

As for selective convertibility, we need a concrete situation where it is important that the MOVA signature is converted. Otherwise, the company can simply generate an ordinary signature with respect to another algorithm such as DSA or ECDSA.

5.3 On the Hardness of Related Problems

Here we present some different computational problems related to character instantiations in MOVA. Again, we focus this treatment to the case of characters of order $d \in \{2, 3, 4\}$ defined on $\mathbb{Z}_n^*$. However, the results presented below do not only restrict to an RSA modulus $n = pq$ but usually hold for a general integer n.

Here, for two problems $\mathcal{P}$ and $\mathcal{P}'$, we use some Karp reductions, i.e., we say that $\mathcal{P}$ is at most as hard as $\mathcal{P}'$ or equivalently $\mathcal{P}$ reduces to $\mathcal{P}'$ if the problem $\mathcal{P}$ can be solved in polynomial time by using only one access to an oracle $\mathcal{O}_{\mathcal{P}'}$ which solves $\mathcal{P}'$. We denote this fact by $\mathcal{P} \leq_K \mathcal{P}'$. This is also equivalent to say that $\mathcal{P}'$ is at least as hard as $\mathcal{P}$. We say also that two problems $\mathcal{P}$ and $\mathcal{P}'$ are equivalent if $\mathcal{P} \leq_K \mathcal{P}'$ and $\mathcal{P}' \leq_K \mathcal{P}$ are satisfied. We denote this property by $\mathcal{P} \equiv_K \mathcal{P}'$.

Let $\zeta_d = e^{2\pi i/d}$ with d equal to $2, 3, 4$ and denote H_d the set $\{\zeta_d^j \mid j = 0, \ldots, d-1\}$. Below we develop different computational problems.

FACT (Factorization in $\mathbb{Z}$)

Input: An integer $n \in \mathbb{Z}$.

Problem: Find the factorization of n in $\mathbb{Z}$.

CYCLOFACT^d (Factorization in $\mathbb{Z}[\zeta_d]$)

Input: An element $\sigma \in \mathbb{Z}[\zeta_d]$.

Problem: Find the factorization of σ.

ROOT(-3) (Square Root of -3 modulo n)

Input: $n \in \mathbb{Z}$ such that -3 is a square modulo n.

Problem: Find a $u \in \mathbb{Z}$ such that $u^2 \equiv -3 \pmod{n}$.

ROOT($-$1) (Square Root of -1 modulo n)

Input: $n \in \mathbb{Z}$ such that -1 is a square modulo n.

Problem: Find a $u \in \mathbb{Z}$ such that $u^2 \equiv -1 \pmod{n}$.

FERMAT^d (Finding an Element of a Given Norm)

Input: $n \in \mathbb{Z}$ such that $n = \alpha\bar{\alpha}$ for an $\alpha \in \mathbb{Z}[\zeta_d]$.

Problem: Find α.

MOVA^d (MOVA Problem)

Parameters: Let $n \in \mathbb{Z}$ and a set of points $\{(\alpha_i, h_i) \mid i = 1, \ldots, s\} \subseteq \mathbb{Z}_n^* \times H_d$ be such that there exists a unique character χ of order d with $\chi(\alpha_i) = h_i$ for $i = 1, \ldots, s$.

Input: $x \in \mathbb{Z}_n^*$.

Problem: Compute $\chi(x)$.

Remark 5.3.1. The MOVA problem can be seen as a non-probabilistic variant of the 1-S-GHI problem with a set of points S which interpolates uniquely in a character of order d. We will usually consider an RSA modulus $n = pq$, α_i's which H-generate $\mathbb{Z}_n^*$ for any group H of order d, and take a "hard character" χ. By "hard character" we mean a nontrivial character and for $d = 2$ we also exclude the Jacobi symbol $(\cdot/n)$. Note that if n is an RSA modulus, the MOVA problem corresponds to the quadratic residuosity problem for $d = 2$ if we restrict the input to the elements with a Jacobi symbol equal to 1. See also Landrock [90] for another cryptographic application of Fermat numbers (i.e., FERMAT^4 and ROOT(-1)).

Remark 5.3.2. Note that -3 is not necessarily invertible modulo n in the problem ROOT(-3). So, it is not ensured that -3 formally fullfils the definition of a quadratic residue modulo n. It is trivially the case under the assumption $\gcd(n, 3) = 1$. On the other hand, -1 is obviously a quadratic residue modulo n in ROOT(-1).

The following technical lemma provides assumptions on n such that the existence of a solution of a ROOT problem implies existence of the corresponding FERMAT problem and vice versa.

Lemma 5.3.3. *Let $n \in \mathbb{Z}$, the following assertions hold.*

1. *If $n \not\equiv 2 \pmod 4$ and -3 is a square modulo n, there exists an $\alpha \in \mathbb{Z}[\omega]$ such that $N(\alpha) = n$.*
2. *If n is square free and is of the form $n = N(\alpha)$ for an $\alpha \in \mathbb{Z}[\omega]$, then -3 is a square modulo n.*
3. *If -1 is a quadratic residue modulo n, there exists an element $\alpha \in \mathbb{Z}[i]$ such that $N(\alpha) = n$.*
4. *If n is square free and is of the form $n = N(\alpha)$ for an $\alpha \in \mathbb{Z}[i]$, then -1 is a quadratic residue modulo n.*

Proof.

1. By assumption, there exists an integer u such that $u^2 \equiv -3 \pmod n$. We can rewrite this equality in $\mathbb{Z}[\omega]$ as

$$kn = (u + \sqrt{-3})(u - \sqrt{-3}) = (u + 1 + 2\omega)(u - 1 - 2\omega),$$

for an integer k. If $\pi \in \mathbb{Z}[\omega]$, is a common prime of n and $u + 1 + 2\omega$, its conjugate $\bar{\pi}$, must also divide n and $u - 1 - 2\omega$. This shows that we have

$$\overline{\gcd(n, u + 1 + 2\omega)} = \gcd(n, u - 1 - 2\omega),$$

when the gcd are carefully chosen among the possible associates. We now show that $\alpha = \gcd(n, u + 1 + 2\omega)$ has the desired property. To this end, we remark that the only integer greater to 1 which may divide $u+1+2\omega$ is 2. Since $n \not\equiv 2 \pmod 4$, we have n odd or $4|n$. Since 2 is a prime in $\mathbb{Z}[\omega]$, $2|n$ implies that $2|\alpha$ and $2|\bar{\alpha}$ so that $4|n$. Note also that no greater power of 2 can divide n. Hence, except 2, all primes dividing n are splitted in two parts so that one lies in α while the other one lies in $\bar{\alpha}$. If the factor 4 occurs in n, we have a factor 2 in α and $\bar{\alpha}$ appearing exactly once in each term. Putting all together shows that $n = \alpha\bar{\alpha}$.

2. If we write $\alpha = a + b\omega$, we have

$$n = a^2 - ab + b^2. \tag{5.3}$$

Since n is square free, it must hold that $\gcd(a, n) = \gcd(b, n) = 1$. To show this, suppose the existence of a rational prime p such that $p|n$ and $p|a$. It follows that $p|b^2$ and since p is prime we also have $p^2|b^2$ and $p^2|a^2$, which leads to the contradiction $p^2|n$. From Equation (5.3), we get

$$4n = (2a - b)^2 + 3b^2.$$

and since b is invertible modulo n, we finally obtain

$$\left((2a-b)b^{-1}\right)^2 \equiv -3 \pmod{n}.$$

3. By assumption, we have an integer u such that $u^2 \equiv -1 \pmod{n}$. This can be rewritten in $\mathbb{Z}[i]$ as follows

$$kn = (u+i)(u-1),$$

for an integer k. We note that no integer greater than 1 divides $u+i$ and $u-i$. So, as for the proof of the assertion 1, we have any prime factors of n which split in two conjugate terms so that one part divides $u+i$ while the other one divides $u-i$. Thus, choosing $\alpha := \gcd(n, u+i)$ gives the desired result.

4. Let $\alpha := a+bi$ for some integers a and b. As for assertion 2, we can show that $\gcd(a,n) = \gcd(b,n) = 1$ since n is square free. By assumption, we have $n = a^2+b^2$. If we consider this equation modulo n, we finally get $(ab^{-1})^2 \equiv -1 \pmod{n}$. □

Remark 5.3.4. Note that the assumption $n \not\equiv 2 \pmod{4}$ in the assertion 1 is required. To see this, it suffices to consider $n = 14$ and remark that $5^2 \equiv -3 \pmod{14}$. Since 2 is a prime in $\mathbb{Z}[\omega]$, the existence of a decomposition $n = \alpha\bar{\alpha}$ would imply that $4|14$ which is a contradiction.

We now exhibit some reductions between the above problems. All relations are summarized in Figure 5.1. However, we note that the Karp equivalence between the ROOT problems and the FERMAT ones only hold under additional assumptions on n.

Proposition 5.3.5. *For $d = 2, 3, 4$, we have the following Karp reductions running in polynomial time with respect to* $\log(n)$ *and* $\log(N(\sigma))$.

1. FACT $\equiv_K$ CYCLOFACT^d.

2. MOVA^d $\leq_K$ CYCLOFACT^d.

3. FERMAT^d $\leq_K$ CYCLOFACT^d.

4. *If n is square free and $n \not\equiv 2 \pmod{4}$, then* FERMAT^3 $\equiv_K$ ROOT(-3).

5. *If n is square free, then* FERMAT^4 $\equiv_K$ ROOT(-1).

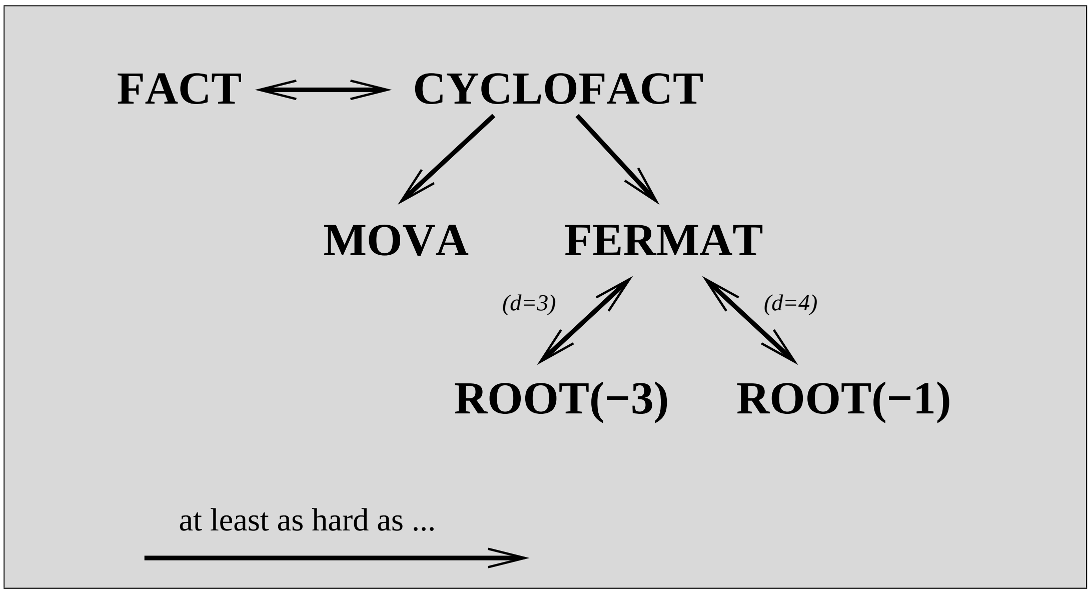

Figure 5.1: Karp reductions related to the MOVA problem

Proof.

1. We first show FACT $\leq_K$ CYCLOFACT^3. By sending n to the oracle solving CYCLOFACT^3, we receive a decomposition of the form

$$n = u \cdot (1-\omega)^{2i} \cdot \prod_{j=1}^{k} \pi_j \cdot \prod_{j=1}^{\ell} q_j,$$

where the π_j's have a rational prime norm $N(\pi_j) = p_j$ and the q_j's are already rational primes. To find the rational prime decomposition of n, it suffices to combine the terms π_j's having the same norm together and to replace $(1-\omega)^{2i}$ by 3^i. By adjusting u according to this process, we finally get $u = 1$ or -1 since n is a classical integer. A very similar reduction can be performed for $d = 4$.

FACT $\geq_K$ CYCLOFACT^d. Let $\sigma \in \mathbb{Z}[\zeta_d]$ be an element to factorize. We compute $n = N(\sigma)$ and send n to the oracle solving FACT. We then get the rational prime factorization $n = \prod_{j=1}^{k} p_j$. When $d = 3$, any $p_j \equiv 1 \pmod 3$ or equal to 3 is not prime in $\mathbb{Z}[\omega]$. For $p_j = 3$, we simply split it using the formula $3 = -\omega^2(1-\omega)^2$. When $p_j \equiv 1 \pmod 3$, we need to compute the prime $\pi_j \in \mathbb{Z}[\omega]$ satisfying $N(\pi_j) = p_j$. This can be done as in Subsection 5.2.2. When $d = 4$, any $p_j \equiv 1 \pmod 4$ or equal to 2 is not prime in $\mathbb{Z}[i]$. Using the equation $2 = -i(1+i)^2$ allows to split 2 and as in Subsection 5.2.3, we can find $\pi_j \in \mathbb{Z}[i]$ such that $N(\pi_j) = p_j$. Once the factorization of $N(\sigma)$ in $\mathbb{Z}[\zeta_d]$ is completed, it remains to test which prime factors of $N(\sigma)$ divide σ. This is determined by computing some gcd. So, we can find all non trivial primes dividing σ and therefore the factorization of σ.

2. We factorize n using an oracle access and get $n = \prod_{i=1}^{k} p_i^{a_i}$, where the p_i's are different primes. Applying Chinese Remainder Theorem, we obtain the decomposition

$$\mathbb{Z}_n^* \simeq \mathbb{Z}_{p_1^{a_1}}^* \oplus \cdots \oplus \mathbb{Z}_{p_k^{a_k}}^*. \tag{5.4}$$

Note that any group $\mathbb{Z}_{p_i^{a_i}}^*$ is cyclic except when $p_i = 2$ and $a_i \geq 3$. In this case, for an integer $a \geq 3$, we have the isomorphism

$$\mathbb{Z}_{2^a}^* \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_{2^{a-2}}. \tag{5.5}$$

A proof of this statement is given in Nathanson [114] (see pp. 96). So, if $8|n$ we decompose $\mathbb{Z}_n^*$ in $k+1$ cyclic subgroups. Otherwise, we content ourselves with the decomposition given in (5.4). By Lemma 5.2.1, any character on $\mathbb{Z}_n^*$ can be decomposed in a product of characters defined on the groups $\mathbb{Z}_{p_i^{a_i}}^*$ for $i = 1, \ldots, k$ respectively. Moreover, since d is a prime power, any character on $\mathbb{Z}_n^*$ of order dividing d can be uniquely decomposed in such a product of characters of order dividing d. As seen in Section 5.1, for any $p_i \neq 2$, one can readily find a computable character χ_i defined on $\mathbb{Z}_{p_i}^*$ which generates the group of all characters defined on $\mathbb{Z}_{p_i}^*$ of order dividing d. Furthermore, we remark that χ_i can be canonically extended on $\mathbb{Z}_{p_i^{a_i}}^*$ so that it also generates the group of characters defined on $\mathbb{Z}_{p_i^{a_i}}^*$ of order dividing d. We now handle the special case $p_1 = 2$ and consider only $a_1 \geq 3$ since the other cases are obvious. Looking at the decomposition given in (5.5), we deduce that two characters on $\mathbb{Z}_{2^a}^*$ are required to generate all characters of order dividing d with $d \neq 3$. We characterize them here and show how they can be computed. First, we use a result given in Nathanson [114] (see pp. 96) stating that 5 is of order 2^{a-2} in $\mathbb{Z}_{2^a}^*$ and that any element $u \in \mathbb{Z}_{2^a}^*$ can be written $u = (-1)^i 5^j$ for some unique $i \in \{0, 1\}$ and $j \in \{0, \ldots, 2^{a-2} - 1\}$. If we consider the application which assigns to any $u \in \mathbb{Z}_{2^a}^*$ the pair $(i, j) \in \mathbb{Z}_2 \oplus \mathbb{Z}_{2^{a-2}}$, we obtain a description of the isomorphism given in (5.5). If $d = 3$, we note that the only character of order dividing d defined on $\mathbb{Z}_{2^a}^*$ is the trivial one. Otherwise, for $d = 2$ or 4, all characters of order dividing 4 defined on $\mathbb{Z}_{2^a}^*$ are generated by two characters χ_0 and χ_1. For the sake of simplicity, we work now with the "logarithmic" version of the characters defined as in Section 5.2. Namely, we denote $\log_\chi := \log_{\zeta_d} \circ \chi$ for any character χ. We define $\log_{\chi_0}$ as the function assigning $i \bmod 2$ and $\log_{\chi_1}$ as the function assigning $j \bmod 4$ to an input $u = (-1)^i 5^j$. Note that $\log_{\chi_0}(u)$ can be easily determined by computing $u \bmod 4$ since $u \equiv (-1)^i \pmod{4}$. Computing $j \bmod 4$ can be achieved by looking for an $e \in \{0, 1, 2, 3\}$ satisfying $(5^e u)^{2^{a-4}} \equiv 1 \pmod{2^a}$. Then, we set $j \bmod 4 = 4 - e$ to obtain the desired result.

We have shown in the above discussion that any character χ defined on $\mathbb{Z}_n^*$ of order d can be uniquely decomposed as a unique product

$$\chi = \chi_0^{b_0}\chi_1^{b_1}\cdots\chi_k^{b_k},$$

where $b_i \in \{0,\ldots,d-1\}$ for any $i = 0,\ldots,d-1$. Moreover, it was shown that any character χ_i is efficiently computable if the factorization is known. It remains to show a way to find the b_i's coefficients. Since χ is uniquely defined by the pairs (α_j, h_j) for $j = 1,\ldots,s$, we can retrieve the b_i's by solving the following system of linear equations

$$\log_\chi(\alpha_j) = \sum_{i=0}^{k} b_i \log_{\chi_i}(\alpha_j) \quad \text{for } j = 1,\ldots,s.$$

This directly leads to the description of the character χ, allowing to compute it efficiently therefrom and to solve the MOVA problem.

3. We factorize n in $\mathbb{Z}[\zeta_d]$ using one oracle access. Then, we remove one half of the primes of the obtained decomposition by carefully keeping the corresponding conjugate. By multiplying the remaining primes, we get α such that $N(\alpha) = n$.

4. **FERMAT^3 $\le_K$ ROOT($-$3)**. We receive an integer n and have to find an $\alpha \in \mathbb{Z}[\omega]$ such that $n = \alpha\bar{\alpha}$. Since n is square free, we know that -3 is a square modulo n by the assertion 2 of Lemma 5.3.3. So, we can send n to the oracle solving ROOT(-3) and get back an integer u such that $u^2 \equiv -3 \pmod{n}$. As for proving the assertion 1 of Lemma 5.3.3, we can retrieve α by computing $\gcd(n, u + 1 + 2\omega)$.

FERMAT^3 $\ge_K$ ROOT($-$3). Here, we are given an integer n such that -3 is a square modulo n and have to find an integer u such that $u^2 \equiv -3 \pmod{n}$. By assertion 1 of Lemma 5.3.3, we know that there exists an $\alpha = a + b\omega \in \mathbb{Z}[\omega]$ such that $n = a^2 - ab + b^2$ since $n \not\equiv 2 \pmod{4}$. We send n to the oracle solving FERMAT^3 so that we receive a and b. As in the proof of the assertion 2 of Lemma 5.3.3, we can show that choosing $u = (2a-b)b^{-1} \bmod n$ leads to the desired result. Note that the assumption for n to be square free is used to show that b is invertible modulo n.

5. **FERMAT^4 $\le_K$ ROOT($-$1)**. Let n of the form $n = N(\alpha)$ for an $\alpha \in \mathbb{Z}[i]$ and square free. By assertion 4 of Lemma 5.3.3, -1 is quadratic residue modulo n. We send n to the oracle solving ROOT(-1) and receive an integer u such that $u^2 \equiv -1 \pmod{n}$. As in the proof of assertion 3 of Lemma 5.3.3, we can choose $\alpha = \gcd(n, u + i)$.

FERMAT$^4 \geq_K$ **ROOT**$(-\mathbf{1})$. Let n such that -1 is a square modulo n and square free. By assertion 3 of Lemma 5.3.3, there exists an $\alpha = a + bi \in \mathbb{Z}[i]$ such that $n = a^2 + b^2$. We can find a and b by sending n to the oracle solving FERMAT^4. Finally, as for proving assertion 4 of Lemma 5.3.3, we can show that choosing $u = ab^{-1} \bmod n$ leads to a square root of -1 modulo n. □

Remark 5.3.6. Note that both factorization problems are related to the problem of recovering the private key of the MOVA scheme from the public key n. For an RSA modulus $n = pq$, both FERMAT problems are related to the special instance where the character is of the form $\chi_{\pi\sigma}$. Hence, we know that solving ROOT(-3) (resp. ROOT(-1)) is equivalent to retrieve the private key of the MOVA scheme instantiated with such a character. Since the ROOT problem with no fixed challenge (contrary to the above ROOT problems with the challenges -1 and -3 respectively) is known to be equivalent (not in the sense of Karp) with the factorization problem, this gives some argument towards the hardness of the MOVA problem.

Before concluding this section, we would like to mention that assertions 1 and 3 of Lemma 5.3.3 can alternatively be proved using lattice theory[3]. For instance, to prove the assertion 3, we consider the lattice

$$L = \{(x, y) \in \mathbb{Z}^2 \mid y \equiv ux \pmod{n}\},$$

where u is a square root of -1 modulo n. Note that any pair $(x, y) \in L$ satisfies $x^2 + y^2 = kn$ for an integer k. Using the Minkowski's convex body theorem (see Appendix A.3), we can show that a non zero pair of L must lie in a disc of radius less than $2n$. Thus, we deduce the existence of a pair $(a, b) \in \mathbb{Z}^2$ such that $a^2 + b^2 = n$. A similar but more complicated proof also applies for the assertion 3 of Lemma 4. These results also apply to some assertions of Proposition 5.3.5. Finally, we would like to say that the above pair (x, y) can be efficiently found (see Nguyễn [115]) using a lattice reduction since the lattice is of dimension 2.

5.4 A Variant without Primes

We develop in this section a MOVA variant instantiated with a particular character which does not need the generation of prime numbers. Although it is very unlikely that this variant will be implemented for some applications, we believe that it is worth to present it. The main reason is that most public-key algorithms require the generation of primes. This part should be mostly seen as an attempt of achieving primeless cryptography and is restricted to the academic interest.

[3]The reader unfamiliar with lattices can have a look at Appendix A.3.

First, we note that the (Jacobi-like) cubic and quartic residue characters χ_α are defined for almost any kind of α. The only restriction is that $1-\omega \nmid \alpha$ for $d=3$ and $1+i \nmid \alpha$ for $d=4$. Hence, as long as α satisfies this property, χ_α is a well defined character of order dividing d defined on $\mathbb{Z}_{N(\alpha)}^*$.

A signer can select a character for the MOVA scheme without picking prime numbers in the following way.

1. Pick $\alpha \in_U \mathbb{Z}[\omega]$ (resp. $\mathbb{Z}[i]$) of a given size uniformly at random.
2. Remove the highest power $(1-\omega)^k$ (resp. $(1+i)^k$) dividing α.
3. Compute $n = \alpha\bar{\alpha}$.
4. The MOVA public key is n and the corresponding secret key α.

Remark 5.4.1. The signer can only apply Setup Variant 1 and 2 here since the other ones require an expert group knowledge. In both variants, we need to generate elements in $\mathbb{Z}_n^*$ from a seed. For this, we need to compute some gcd to check whether the generated elements lie in $\mathbb{Z}_n^*$. If the test is negative, we simply put the element aside and test the next one produced by the pseudorandom generator GenK. When n is almost picked at random the ratio of elements to be thrown away are not negligible at all.

A natural question which immediately arises concerns the security of such a variant. In particular, is it really hard to retrieve α from n? We have proved in Proposition 5.3.5 that if n is square free and $n \not\equiv 2 \pmod 4$ for $d=3$, this problem is equivalent to find a square root of -3 (for $d=3$) and -1 (for $d=4$) modulo n. This gives some argument towards a possible equivalence with the factorization problem. So, we investigate below whether a random modulus n can be hard to factorize with a reasonable size. We will show that an adversary has an non-negligible probability to factorize n even when n is large, but that the expected required time of factoring a random modulus is quite high.

5.4.1 Theoretical Results about Prime Factors

We summarize below some results related to the size of the prime factors of a random integer. The presented material is taken from the article of Knuth and Trabb Pardo [83].

Let n be a positive integer, we write its unique prime decomposition

$$n = n_1 n_2 n_3 \cdots n_\ell,$$

where the n_i's are all (not necessarily different) prime satisfying $n_1 \geq n_2 \geq \ldots \geq n_\ell$. For $x \in \mathbb{R}$ and $k, N \in \mathbb{N}$ we define the function

$$P_k(x, N) := \#\{1 \leq n \leq N \mid n_k \leq N^x\}$$

and the asymptotic probability

$$F_k(x) := \lim_{N \to \infty} \frac{P_k(x, N)}{N}.$$

This corresponds approximately to the probability that $n_k \leq n^x$ for an integer n picked uniformly at random. Evaluating F_1 and F_2 shows that the median value of n_1 is about $n^{0.606}$ and that of n_2 is about $n^{0.211}$ i.e., $F_1(0.606) = F_2(0.211) = 0.50$.

We give in Table 5.1 a sample of some values of the functions F_1^{-1} and F_2^{-1} given in [83]. Experimental tests done with Maple have confirmed the values for F_2.

δ	$F_1^{-1}(\delta)$	$F_2^{-1}(\delta)$
0.01	0.26974	0.00558
0.02	0.29341	0.01110
0.10	0.37851	0.05308
0.50	0.60653	0.21172
0.90	0.90484	0.35899

Table 5.1: Distribution size of the two largest prime factors

Implications for a Random Modulus n. If we pick a modulus n uniformly at random, we see that the probability that the second largest prime factor is small is still too high. For example, for a n of size of $10'000$ bits, the probability that the second largest prime factor is smaller than 110 bits is about 2 %! This is quite bad for the security of such a modulus n, since the elliptic curve method [96] (ECM) allows to find prime factors of 110 bits in a couple of days on a single workstation.

5.4.2 Resistance of a Random Modulus

We study here in more details the problem of estimating the difficulty of factoring an integer n picked uniformly at random. As a reference for the amount of computing time, we will choose the among time that was required for factoring the challenge RSA-155, an RSA modulus of 155 digits.

GNFS versus ECM

We compare the complexity of ECM with the general number field sieve [95] (GNFS), which is known to be the most efficient algorithm for factoring large RSA modulus.

A fundamental difference is that the complexity of GNFS depends on the size of n, while that of ECM depends on the prime factor size of n we would like to extract. Hence, we can say that the complexity for factoring n completely with ECM only depends on the size of its second largest prime factor n_2 plus a primality test. If we consider a probabilistic primality test such as Miller-Rabin, this time is negligible in our context.

We would like first to estimate the size of a prime factor ECM can find using the same amount of time spent by the GNFS for factoring a 1024 bit RSA modulus. To this end, we first use the two theoretical asymptotic complexity formulas for ECM resp. GNFS

$$\begin{aligned}\mathrm{Comp}_{\mathrm{ecm}}(p) &= e^{(\sqrt{2}+o(1))(\log p)^{1/2}(\log\log p)^{1/2}}\\ \mathrm{Comp}_{\mathrm{gnfs}}(n) &= e^{(1.923+o(1))(\log n)^{1/3}(\log\log n)^{2/3}},\end{aligned}$$

where $o(1) \to 0$ when $p, n \to \infty$. Below we approximate these complexity functions by replacing $o(1)$ by 0. We determine two constants c_{ecm}, c_{gnfs} such that $c_{\mathrm{ecm}} \cdot \mathrm{Comp}_{\mathrm{ecm}}(p)$ resp. $c_{\mathrm{gnfs}} \cdot \mathrm{Comp}_{\mathrm{gnfs}}(n)$ gives an expression on the time spent in MIPS by the ECM resp. GNFS method for retrieving the prime factor p resp. for factoring n. The ECM constant has been determined using the fact that retrieving a 40-digit prime factor of the tenth Fermat number required 240 MIPS years[4] (see Brent [27]). The second constant was determined using the amount of computation (8400 MIPS years) needed for factoring RSA-155 (see Cavallar et al. [35]).

These two constants allowed us to deduce that computation complexity of the factorization of RSA 1024 bits with GNFS corresponds to finding a prime factor of size of 311 bits with ECM. In Table 5.2 we give some additional values in bits for which the GNFS resp. ECM should take a similar amount of time.

Random versus RSA Modulus

We estimate here the time required by an adversary to factorize a random modulus of 2048 bits. For this, we consider two kinds of adversary leading to completely different conclusions. Details are explained below.

One-Shot Adversary. First, we imagine that the adversary simply gets one integer n and performs computations until he finally factorizes it. In such a model we will say

[4]A MIPS year is the amount of computation processed during one year at a rate of one Million Instructions Per Second.

GNFS (modulus size)	ECM (prime factor size)
512	160
768	238
1024	311
1536	447
2048	574

Table 5.2: GNFS versus ECM

that the difficulty of factoring can be measured by the expected time the adversary needs to achieve this task. We can easily show that factoring a random n of 2048 bits is harder in average than a 1024 bit RSA modulus. From [83], we know that $F_2(0.2) \approx 0.463$. Then, this means that for more than half of the cases factoring n requires more time than $c_{\text{ecm}} \cdot \text{Comp}_{\text{ecm}}(2^{410})$. Since this last value is more than 1000 times larger than the time required for factoring a 1024 bit RSA modulus with GNFS, a random integer of 2048 bits is sufficiently strong in average.

Multi-Target Adversary. The average time is not always a good security measure. Namely, most of the time we need to be sure that the adversary can factorize one modulus n only with a very small probability. For example, we can think of a scenario where the adversary has access to many moduli n and tries to break one of them. So, if the above probability is not small, he will often find a weak modulus.

We suppose the adversary applies the following strategy. He decides of a time threshold T which corresponds to the time he spends on a modulus n for attempting to factorize it. If he does not succeed to factorize n after a time T, he begins again with a new random target modulus n. He continues this process until he succeeds. The average time in this scenario corresponds to about $T/\Pr[t \leq T]$ where t denotes the time required to factor n. The adversary can find the optimal threshold T_{opt} i.e., the one which minimizes the average time. We assume that the adversary uses ECM for factoring n. This is more adequate than the GNFS, since ECM has a running time depending on the size of the second largest prime factor of n contrary to GNFS. Using GNFS would always need the same amount of time and the search of an optimal threshold T is meaningless.

The average time of such an adversary can be expressed as

$$\frac{c_{\text{ecm}} \cdot \text{Comp}_{\text{ecm}}(N_2)}{\Pr_n[n_2 \leq N_2]},$$

where N_2 is a threshold corresponding to the time T. If we set $n^x = N_2$ the above expression becomes

$$\frac{c_{\text{ecm}} \cdot \text{Comp}_{\text{ecm}}(n^x)}{F_2(x)}.$$

Results for an n of 2048 bits are given in Table 5.3, where $\alpha = 1/x$. Note that time due to the primality test is omitted since it is negligible.

α	av. time in MIPS
6	$0.2343 \cdot 10^{13}$
10	$0.6842 \cdot 10^{7}$
14	$0.9964 \cdot 10^{4}$
18	$0.1670 \cdot 10^{3}$
20	$0.3588 \cdot 10^{2}$
40	$0.1251 \cdot 10^{-1}$

Table 5.3: Average time to factorize a random modulus of 2048 bits.

The best strategy for an adversary is to choose a threshold as small as possible. As illustrated in Table 5.3, this is due to the fact that the probability for n_2 to be smaller than $n^{1/\alpha}$ does not decrease as fast as the complexity of ECM. In fact, the probability for n_2 to be very small is not negligible and the adversary can better wait on a very weak n that can be factorized very rapidly.

To conclude, picking a modulus at random which should be resistant against factorization cannot lead to a secure solution for a reasonable size of n in a multi-target mode. As an illustration, we note that even 1% of the moduli of size of 20'000 bits have a second largest prime factor smaller than 112 bits by Table 5.1. So, the original idea to replace an RSA-like modulus by a random integer and keep a problem relying on the hardness of the factorization problem does not seem to work.

5.5 On the MOVA Key Validation

Here, we come back to a classical RSA modulus $n = pq$, where the primes p and q are chosen according to the character order d, i.e., such that $d|p-1$ and $d|q-1$. Again, we particularly focus to the cases $d = 2, 3, 4$.

Key Validity

The aim of this part is to find some criteria on elements of $\mathbb{Z}_n^*$ to H-generate $\mathbb{Z}_n^*$ for any group H of order d. For the sake of simplicity, we will say "d-generate $\mathbb{Z}_n^*$". Due to the structure of $\mathbb{Z}_n^*$, it will be sufficient to consider two elements so that Lkey $= 2$ for Setup Variants 3 and 4 of the MOVA scheme.

The following proposition shows how we can generally proceed to determine whether two elements d-generate $\mathbb{Z}_n^*$.

Proposition 5.5.1. *Let n and d be integers defined as above and $u, v \in \mathbb{Z}_n^*$. Assume we are given two characters χ_p and χ_q of order d which are defined on $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ respectively. Let consider the function $\varphi : \mathbb{Z}_n^* \to \mathbb{Z}_d \oplus \mathbb{Z}_d$ defined by*

$$\varphi(x) = (\log_{\chi_p}(x \bmod p), \log_{\chi_q}(x \bmod q)).$$

Then, u and v d-generate $\mathbb{Z}_n^*$ if and only if $\varphi(u)$ and $\varphi(v)$ are both elements of order d such that $\langle\varphi(u)\rangle \cap \langle\varphi(v)\rangle = \{(0,0)\}$.

Proof. We note that φ is a surjective group homomorphism since both characters χ_p and χ_q are of order d and $p \neq q$. Namely, $x \bmod q$ and $x \bmod q$ can take all values independently so that the pair $(\chi_p(x \bmod p), \chi_q(x \bmod q))$ generate all possible values. By First Isomorphism Theorem, we have $\mathbb{Z}_n^*/\ker(\varphi) \simeq \mathbb{Z}_d \oplus \mathbb{Z}_d$. Moreover, $\ker(\varphi) = (\mathbb{Z}_n^*)^d$ and we obtain the following commutative diagram

$$\begin{tikzcd}
\mathbb{Z}_n^* \arrow[r, "\varphi"] \arrow[d, "\text{proj}"'] & \mathbb{Z}_d \oplus \mathbb{Z}_d \\
\mathbb{Z}_n^*/(\mathbb{Z}_n^*)^d \arrow[ur, "\simeq"'] &
\end{tikzcd}$$

where proj denotes the canonical projection. By the assertion 7 of Lemma 4.1.3, u and v d-generate $\mathbb{Z}_n^*$ if and only if proj(u) and proj(v) generate $\mathbb{Z}_n^*/(\mathbb{Z}_n^*)^d$. Due to the above commutative diagram, this is equivalent to prove that $\varphi(u)$ and $\varphi(v)$ generate $\mathbb{Z}_d \oplus \mathbb{Z}_d$. It suffices to determine when two elements generate the group $\mathbb{Z}_d \oplus \mathbb{Z}_d$. Any two elements $g, h \in \mathbb{Z}_d \oplus \mathbb{Z}_d$ generate the subgroup $\langle g, h\rangle = \langle g\rangle + \langle h\rangle$. Since the order of any element of $\mathbb{Z}_d \oplus \mathbb{Z}_d$ is at most equal to d, the subgroup $\langle g, h\rangle$ can contain d^2 elements only if g and h are of order d. We conclude by saying that $\langle g, h\rangle = \mathbb{Z}_d \oplus \mathbb{Z}_d$, if g and h additionally generate some subgroups with a trivial intersection. □

In practice, the signer can pick seedK at random until this seed generates two elements GenK(seedK) $\to u, v \in \mathbb{Z}_n^*$ which d-generate $\mathbb{Z}_n^*$. To check the last property, he can use the results of the above proposition. More precisely, an element $(a, b) \in \mathbb{Z}_d \oplus \mathbb{Z}_d$ has order d if and only if $a \in \mathbb{Z}_d^*$ or $b \in \mathbb{Z}_d^*$. The second criterion can be verified by testing whether $\varphi(v) = j\varphi(u)$ for one $j \in \mathbb{Z}_d^*$. In other words, we check that $\varphi(v) \notin \langle\varphi(u)\rangle$. Since v is of order d it is sufficient to test only for $j \in \mathbb{Z}_d^*$ instead of any $j \in \mathbb{Z}_d \backslash \{0\}$.

Example 8. Let $n = pq$ be an RSA modulus with $p \equiv q \equiv 1 \pmod 4$ and $\pi, \sigma \in \mathbb{Z}[i]$ such that $N(\pi) = p$ and $N(\sigma) = q$. The signer can decide whether u and v 4-generate $\mathbb{Z}_n^*$ by checking the following properties.

1. At least one value among $\chi_\pi(u)$, $\chi_\sigma(u)$ is equal to i or $-i$.
2. At least one value among $\chi_\pi(v)$, $\chi_\sigma(v)$ is equal to i or $-i$.
3. $(\chi_\pi(u), \chi_\sigma(u)) \neq (\chi_\pi(v), \chi_\sigma(v))$ and $(\chi_\pi(u), \chi_\sigma(u)) \neq (\chi_\pi(v)^3, \chi_\sigma(v)^3)$.

Very similar results are obtained with $d = 2$ and 3.

Expert Group Knowledge

As shown in Setup Variant 3 and 4 of the MOVA scheme, the signer needs to perform the protocol MGGDproof and NIMGGDproof respectively. Both these protocols require from the signer an expert group knowledge with respect to the set $\{\mathrm{Xkey}_1, \ldots, \mathrm{Xkey}_{\mathrm{Lkey}}\}$. As shown above, we can always reduce Lkey to 2 for the characters we consider here. We have $\mathrm{Xkey}_1 = u$, $\mathrm{Xkey}_2 = v$, $\mathrm{Xgroup} = \mathbb{Z}_n^*$ and $\mathrm{Ygroup} = \mathbb{Z}_d$, where u and v are chosen according to Proposition 5.5.1. In our context, an expert group knowledge of the signer means that for any $x \in \mathbb{Z}_n^*$ he is able to find an $r \in \mathbb{Z}_n^*$ and some coefficients $a, b \in \mathbb{Z}_d$ such that

$$x = r^d u^a v^b \bmod n.$$

We show below how one can solve such an equation provided the knowledge of the factorization of n. Namely, as shown in Section 5.2 all characters defined on $\mathbb{Z}_n^*$ can be found with p and q for $d = 2, 3, 4$. For the sake of simplicity, we consider χ_p and χ_q defined on $\mathbb{Z}_n^*$. We first determine the unique coefficients a and b for a given x. To this end, we compute $\log_{\chi_p}(x)$ and $\log_{\chi_q}(x)$ and solve the equations

$$\begin{aligned}
\log_{\chi_p}(x) &\equiv a \log_{\chi_p}(u) + b \log_{\chi_p}(v) \pmod{d}\\
\log_{\chi_q}(x) &\equiv a \log_{\chi_q}(u) + b \log_{\chi_q}(v) \pmod{d}
\end{aligned}$$

with respect to a and b. By definition of u and v, we are ensured that a unique solution exists. Namely, the two vectors $(\log_{\chi_p}(u), \log_{\chi_q}(u))$, $(\log_{\chi_p}(v), \log_{\chi_q}(v))$ form a basis of $\mathbb{Z}_d \oplus \mathbb{Z}_d$ since

$$\mathbb{Z}_d \oplus \mathbb{Z}_d \simeq \langle \varphi(u) \rangle \oplus \langle \varphi(v) \rangle.$$

It remains to find a dth root of $x/(u^a v^b) \bmod n$. This can be achieved by finding the dth root of this element modulo p and q. We retrieve r by applying Chinese Remainder Theorem. For $d = 2$, we can use the algorithm of Tonelli and Shanks (see [42]) and for $d = 4$, we apply this twice. Extracting cube roots can also be done efficiently modulo p and q by using an algorithm similar to the Shanks and Tonelli algorithm. More details about the computation of a cube root modulo a prime are given in Williams and Zarnke [147].

After having shown that the factorization of n allows to have an expert group knowledge of $\mathbb{Z}_n^*$, we prove the converse statement below. Assume we have an oracle such that for any query $x \in \mathbb{Z}_n^*$, we receive some values $r \in \mathbb{Z}_n^*$, $a, b \in \mathbb{Z}_d$ satisfying $x = r^d u^a v^b \bmod n$. The main technique will consist in picking some $r_1 \in_U \mathbb{Z}_n^*$, $a, b \in_U \mathbb{Z}_d$ uniformly at random and sends $x = r_1^d u^a v^b \bmod n$ to the oracle and attempting to factorize n using the received representation $x = r_2^d u^a v^b \bmod n$. From this, we get two dth root of the same value, namely

$$r_1^d \equiv r_2^d \pmod{n}.$$

It is well known that if $r_1 \not\equiv \pm r_2 \pmod{n}$ and $d = 2$, we retrieve a non trivial factor of n with probability $1/2$ by computing $\gcd(r_1 - r_2, n)$. This extends easily to $d = 4$, since 4th root leads to square root in a straightforward way. For $d = 3$, we proceed in the same way and $\gcd(r_1 - r_2, n)$ leads a non trivial factor of n with a probability of $4/9$. This probability corresponds to the cases where both roots are equal modulo p but not modulo q and vice versa. So, if we repeat the above method until a success occurs, we can factorize n. The expected time of success is quite low and anyway polynomial in $\log(n)$.

Chapter 6

Additional Homomorphisms and Algorithmic Issues

Homomorphism evaluations play a central role in the different components of MOVA and in particular in the signature generation algorithm. Due to the generic nature of MOVA undeniable signature scheme, it is required to specify the choice of group homomorphisms in order to fully determine the different parameters and algorithmic aspects. The main task of this chapter is mostly to contribute to the latter in clarifying the impact of the homomorphism's choice on the efficiency of MOVA. A particular attention will be addressed to the quartic residue symbol for which practical implementations are not widespread. Furthermore, we focus on a homomorphism shortly introduced in Chapter 4 which consists in sending elements in a hidden (relatively small) subgroup followed by a discrete logarithm computation.

We first present the different homomorphisms considered for the efficiency comparisons giving a more detailed treatment to the homomorphism based on the discrete logarithm. The next section is dedicated to provide algorithms for the computation of the quartic residue symbol. In Section 6.3, we briefly explain the different variants for computing the discrete logarithm based homomorphism. Then, we focus on the implementation of this one as well as the quartic residue symbol. We finally conclude by comparing the different homomorphisms by considering signature generation of a 20-bit MOVA signature.

6.1 Homomorphisms

In this section, we describe some instances of the group homomorphism Hom which is the core of the MOVA scheme. As considered instantiations are the characters on $\mathbb{Z}_n^*$, the RSA encryption homomorphism [66, 131], and the discrete logarithm in a hidden subgroup based on Example 7 of Subsection 4.1.2. In the subsequent sections, our focus will be directed towards algorithmic aspects of characters of order 4 and the homomorphism based on the discrete logarithm.

6.1.1 Characters on $\mathbb{Z}_n^*$

As instantiations based on characters were thoroughly described in Chapter 5, we will not dwell again on this. We only consider characters of order 4 and recall how they can be characterized. To this goal, we consider two rational primes p, q such that $p \equiv q \equiv 1 \pmod 4$. Applying the theory presented in Chapter 5, one can efficiently compute some $\pi, \sigma \in \mathbb{Z}[i]$ such that $\pi\bar{\pi} = p$ and $\sigma\bar{\sigma} = q$. Below, we will focus on the quartic residue symbols χ_π and $\chi_{\pi\sigma}$, which can be both trivially defined on $\mathbb{Z}_n^*$, where $n = pq$. Using the latter was notably motivated in some context where two levels of secret are desired (see Subsection 5.2.4).

From the theory of characters (see Section 5.1), we can show that $(\chi_{\pi\sigma}(a))^2 = (a/n)_2$ for any $a \in \mathbb{Z}$. Therefore, without the knowledge of the factorization of n we can easily deduce one bit of $\chi_{\pi\sigma}(a)$. In practice, we will compress this quartic residue symbol to one bit sending $1, i$ to the bit 0 and $-1, -i$ to the bit 1. To decompress, it suffices to compute the Jacobi symbol to retrieve the right quartic residue symbol. Hence, with this quartic residue symbol we have to perform two times more evaluations than with χ_π for the same level of security against an existential forgery. This shows that the signature generation will be anyway less efficient for $\chi_{\pi\sigma}$ than for χ_π.

6.1.2 RSA

Following the long tradition of the RSA based cryptography, an undeniable signature scheme based on RSA [131] was proposed in 1997 by Gennaro et al. [66]. This scheme can be seen as a special case of the MOVA scheme when the homomorphism is the RSA encryption function defined on a modulus of safe primes. So, the signature is generated as for the regular RSA signature scheme. In this chapter, we will only make use of it as a performance benchmark. Namely, we are focusing on homomorphisms which make practical the scalability of the signature size, since this property is the main advantage of MOVA over the other ones.

6.1.3 Discrete Logarithm in a Hidden Subgroup

Another homomorphism suitable for the MOVA scheme is based on the discrete logarithm in a hidden subgroup of $\mathbb{Z}_n^*$.

Let n be such that $n = pq$ with $p = rd + 1$, q, d prime, $\gcd(q-1, d) = 1$, $\gcd(r, d) = 1$ and g generating a subgroup of $\mathbb{Z}_p^*$. We obtain g by choosing a random element $h \in \mathbb{Z}_n^*$ until h satisfies $h^r \bmod p \neq 1$ and we set $g = h^r \bmod p$. Like this we find a homomorphism by "sending" the input in a hidden cyclic subgroup of order d and then computing its discrete logarithm with respect to the generator g,

$$\begin{array}{rccl} \varphi: & \mathbb{Z}_n^* & \longrightarrow & \mathbb{Z}_d \\ & x & \longmapsto & \log_g(x^r \bmod p). \end{array}$$

Below, we develop additional properties such as d-generation of $\mathbb{Z}_n^*$ and expert group knowledge of this group. Namely, these properties need to be considered in some protocols such as Setup Variant 3 and 4 of the MOVA scheme. By the assertion 7 of Lemma 4.1.3, we need to examine the structure of $\mathbb{Z}_n^*/(\mathbb{Z}_n^*)^d$. Since the kernel of φ is $(\mathbb{Z}_n^*)^d$, the quotient group $\mathbb{Z}_n^*/(\mathbb{Z}_n^*)^d$ is isomorphic to $\mathbb{Z}_d$ by First Isomorphism Theorem and we obtain the following commutative diagram

$$\begin{tikzcd} \mathbb{Z}_n^* \arrow[r, "\varphi"] \arrow[d, "\text{proj}"'] & \mathbb{Z}_d \\ \mathbb{Z}_n^*/(\mathbb{Z}_n^*)^d \arrow[ur, "\simeq"'] & \end{tikzcd}$$

where proj denotes the canonical projection. Hence, an element $u \in \mathbb{Z}_n^*$ d-generates $\mathbb{Z}_n^*$ if and only if $\varphi(u)$ generates $\mathbb{Z}_d$. Thus, it suffices to check whether $\varphi(u) \neq 0$ or equivalently $u^r \bmod p \neq 1$ to decide whether an element u d-generates $\mathbb{Z}_n^*$. Let x be a given element of $\mathbb{Z}_n^*$. Expert group knowledge with u in this setting consists in finding some elements $y \in \mathbb{Z}_n^*$ and $a \in \mathbb{Z}_d$ such that

$$x = y^d u^a.$$

Note that a is uniquely defined by x and can be easily deduced from $a\varphi(u) = \varphi(x)$. An element y satisfying the above equation can be also computed by extracting a dth root of $x/u^a \bmod n$. We perform this operation by computing a dth root in $\mathbb{Z}_p^*$ and $\mathbb{Z}_q^*$ and applying Chinese Remainder Theorem. Since $\gcd(d, q-1) = 1$, a dth root in $\mathbb{Z}_q^*$ can be easily computed by raising the element to the power $d^{-1} \bmod q-1$. In $\mathbb{Z}_p^*$, a more sophisticated technique needs to be used since $d|p-1$. To this, we refer to a generalization of the Shank's algorithm [99] which allows to compute a dth root in $\mathbb{Z}_p^*$ using $O(\log^2(p) \log\log(p))$ multiplications modulo p provided that $d = O(\log^2(p))$. To facilitate such a computation, we propose alternatively to choose p in such a way that $r \equiv -1 \pmod{d}$. When this is the case, a dth root of an element $z \in \mathbb{Z}_p^*$ (which is a dth power) can be efficiently found by computing $z^{\frac{r+1}{d}} \bmod p$.

6.2 Quartic Residue Symbol

The results of this section were achieved in a semester project of Yvonne Anne Oswald [122] under the supervision of Serge Vaudenay and myself.

6.2.1 Basic Algorithm

Description

To compute the quartic residue symbol $\chi_\beta(\alpha)$ directly, one has to know the factorization of β into primes over $\mathbb{Z}[i]$ and the computation contains an exponentiation. To avoid this factorization as well as the costly exponentiation we apply the properties of the quartic residue symbol given in Theorem 5.1.16 iteratively. Note that this is similar as for the ordinary method to compute the Jacobi symbol. This basic method is described in Algorithm 6.1.

First we reduce α to an element $\hat{\alpha}$ equivalent to α modulo β which satisfies $N(\hat{\alpha}) < N(\beta)$. From now on, such a reduction of an element α modulo β will be denoted $\mathrm{Red}_\beta(\alpha)$. Note that the obtained $\hat{\alpha} \leftarrow \mathrm{Red}_\beta(\alpha)$ is not necessarily unique. Then, we find the unique representation $\hat{\alpha} = i^j \cdot (1+i)^k \cdot \alpha'$ with α' primary and employ the multiplicativity property and the complementary laws of the quartic residue symbol. Next, we interchange α and β according to the law of reciprocity and start again. Therefore, the size of both α and β decrease progressively. We stop the iteration process when α or β is a unit.

Computation of Related Subfunctions

For this algorithm we have to implement a few functions for calculating basic operations in the ring of Gaussian integers (let $\alpha, \beta \in \mathbb{Z}[i]$):

- Multiplication: $\alpha \cdot \beta$
- Norm: $N(\alpha)$
- Division by $(1+i)^r$
- Primarization: transforms α into its primary associate if possible
- Modular reduction: $\mathrm{Red}_\beta(\alpha)$

The multiplication and the norm are trivially implemented by performing integer multiplications between the appropriate integer components.

The division of α by $(1+i)^r$ can be done by first raising $(1+i)$ to the power of r and then dividing α by the result. We propose a way of achieving the same by only

Algorithm 6.1. Basic algorithm for quartic residue symbol

```
Require: $\alpha, \beta \in \mathbb{Z}[i] \setminus \{0\}$, $\gcd(\alpha, \beta) \sim 1$ and $(1+i) \nmid \beta$
Ensure: $c = \chi_\beta(\alpha)$  $(c = 0 \Leftrightarrow \chi_\beta(\alpha)$ is not defined)
 1: $\alpha \leftarrow \mathrm{Red}_\beta(\alpha)$
 2: if $\alpha = 0$ then $c = 0$ end if
 3: let primary $\alpha_1, \beta_1 \in \mathbb{Z}[i]$ be defined by
    $\alpha = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$ and
    $\beta = (i)^{i_2} \cdot \beta_1$
 4: let $m, n \in \mathbb{Z}$ be defined by $\beta_1 = m + ni$
 5: $t \leftarrow \frac{m^2+n^2-1}{4} i_1 + \frac{m-n-n^2-1}{4} j_1 \bmod 4$
 6: replace $\alpha$ with $\beta_1$, $\beta$ with $\alpha_1$
 7: $t \leftarrow t + \frac{(N(\alpha)-1)(N(\beta)-1)}{8} \bmod 4$
 8: while $N(\alpha) > 1$ do
 9:     $\alpha \leftarrow \mathrm{Red}_\beta(\alpha)$
10:     let primary $\alpha_1$ be defined by $\alpha = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$
11:     let $m, n \in \mathbb{Z}$ be defined by $\beta = m + ni$
12:     $t \leftarrow t + \frac{m^2+n^2-1}{4} i_1 + \frac{m-n-n^2-1}{4} j_1 \bmod 4$
13:     replace $\alpha$ with $\beta$, $\beta$ with $\alpha_1$
14:     $t \leftarrow t + \frac{(N(\alpha)-1)(N(\beta)-1)}{8} \bmod 4$
15: end while
16: if $N(\alpha) \neq 1$ then $c \leftarrow 0$ else $c \leftarrow i^t$ end if
```

using shift operations, additions, and interchanging the imaginary and real part if necessary. Let $\alpha = a + bi$, the following equations

$$\frac{\alpha}{(1+i)} = \frac{a+b}{2} + \frac{b-a}{2} i,$$

and

$$\frac{\alpha}{(1+i)^r} = \frac{i^{3k}\left(\frac{a}{2^k} + \frac{b}{2^k} i\right)}{(1+i)^\ell}, \quad r = 2k + \ell$$

demonstrate our procedure. If $r = 2k,\ k \in \mathbb{N}$ we shift the real and the imaginary parts of α by k to the right and multiply them by -1 and/or interchange them depending on the value of $3k$. If r is odd, there is an additional subtraction and addition to perform.

The primarization function we used consists of a few congruency tests and it also determines the number of times we have to multiply α by i to get the primary associate of α.

The computation of $\mathrm{Red}_\beta(\alpha)$ is first done by approximating $\alpha/\beta = u + vi \in \mathbb{R}[i]$ with some integers u' and v' satisfying

$$|u - u'| \leq \frac{1}{2} \quad \text{and} \quad |v - v'| \leq \frac{1}{2},$$

and then by computing the rest $\alpha - \beta(u' + v'i)$. More details on this are given in Lefèvre [93].

To find the representation of α we proceed as follows. First calculate the norm of α, $N(\alpha)$. Then find j maximal such that $2^j \mid N(\alpha)$. Divide α by $(1+i)^j$ and transform the result into its primary associate.

In the implementation of the algorithm we need to ensure $(1+i) \nmid \beta$ and $\gcd(\alpha, \beta) \sim 1$. The first requirement is taken care of by applying the primarization function on β. If we cannot find a primary associate, β is divisible by $(1+i)$ and we terminate. For the second condition we check in every iteration whether $\mathrm{Red}_\beta(\alpha) \to 0$. This would imply $\gcd(\alpha, \beta) \not\sim 1$ and we terminate.

6.2.2 Algorithm of Damgård and Frandsen

Description

The most expensive operation used in the algorithm described above is $\mathrm{Red}_\beta(\alpha)$. Damgård and Frandsen present in [48] an efficient algorithm for computing the cubic residue symbol in the ring of Eisenstein integers. Their algorithm can be readily transformed into an algorithm for the quartic residue symbol in the ring of Gaussian integers as shown in [48]. This algorithm is depicted in Algorithm 6.2.

There are three main differences to the basic algorithm. Instead of using $\mathrm{Red}_\beta(\alpha)$ to reduce α, they suggest using $\alpha - \beta$. This takes much less time but increases the number of iterations needed. Furthermore they only interchange α and β, if $N(\alpha) < N(\beta)$. It is not necessary to calculate $N(\cdot)$ exactly for this purpose, an approximation $\tilde{N}(\cdot)$ suffices. They demonstrate how one can compute an approximate norm $\tilde{N}(\alpha)$ in linear time. Instead of adding up the squares of the real and the imaginary part of α, one replaces all but the 8 most significant bits of the real and the imaginary part of α with zeroes and computes the norm of the resulting Gaussian number. Their algorithm takes $O(\log^2 N(\alpha\beta))$ time to compute $\chi_\beta(\alpha)$.

6.2.3 Other Algorithms

In addition to the above, we studied papers concerning algorithms for the quartic residue symbol by Weilert. In [145], he presents a fast gcd algorithm for Gaussian integers. Based on this gcd algorithm and using some properties of the Hilbert symbol he demonstrates in [146] how to construct an algorithm for the quartic residue

Algorithm 6.2. Damgård and Frandsen's algorithm

```
Require: $\alpha, \beta \in \mathbb{Z}[i] \setminus \{0\}$, $\gcd(\alpha, \beta) \sim 1$ and $(1+i) \nmid \beta$
Ensure: $c = \chi_\beta(\alpha)$  $(c = 0 \Leftrightarrow \chi_\beta(\alpha)$ is not defined)
 1: let primary $\alpha_1, \beta_1 \in \mathbb{Z}[i]$ be defined by
    $\alpha = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$ and
    $\beta = (i)^{i_2} \cdot \beta_1$
 2: let $m, n \in \mathbb{Z}$ be defined by $\beta_1 = m + ni$
 3: $t \leftarrow \frac{m^2+n^2-1}{4} i_1 + \frac{m-n-n^2-1}{4} j_1 \bmod 4$
 4: replace $\alpha$ with $\alpha_1$, $\beta$ with $\beta_1$
 5: if $\tilde{N}(\alpha) < \tilde{N}(\beta)$ then
 6:     interchange $\alpha$ and $\beta$ and adjust $t$
        $t \leftarrow t + \frac{(N(\alpha)-1)(N(\beta)-1)}{8} \bmod 4$
 7: end if
 8: while $\alpha \neq \beta$ do
 9:     let primary $\alpha_1$ be defined by $\alpha - \beta = (i)^{i_1} \cdot (1+i)^{j_1} \cdot \alpha_1$
10:     let $m, n \in \mathbb{Z}$ be defined by $\beta = m + ni$
11:     $t \leftarrow t + \frac{m^2+n^2-1}{4} i_1 + \frac{m-n-n^2-1}{4} j_1 \bmod 4$
12:     replace $\alpha$ with $\alpha_1$
13:     if $\tilde{N}(\alpha) < \tilde{N}(\beta)$ then
14:         interchange $\alpha$ and $\beta$ and adjust $t$
            $t \leftarrow t + \frac{(N(\alpha)-1)(N(\beta)-1)}{8} \bmod 4$
15:     end if
16: end while
17: if $\alpha \neq 1$ then $c \leftarrow 0$ else $c \leftarrow i^t$ end if
```

symbol. This algorithm involves calculating an Euclidean descent and storing some intermediate results for later use. This algorithm presents a very fast asymptotic complexity which is even better than that of Damgård and Frandsen.

However, as mentioned by Damgård et al. in [48], the fastest algorithms for practical inputs in the case of the Jacobi symbol are based on binary gcd algorithms [102]. Weilert proposed a binary gcd algorithm for Gaussian integers in [144] as well, but did not adapt it to the computation of the quartic residue symbol. Algorithm 6.2 follows this approach and Damgård and Frandsen argued in [48] that this is likely to provide a more efficient algorithm than the asymptotically fast variant of Weilert [146] in practice. Therefore, we have chosen to implement Algorithm 6.2 which takes this binary approach since we need a fast algorithm for practical inputs rather than the best asymptotic complexity.

6.3 Discrete Logarithm in a Hidden Subgroup

One suitable homomorphism for the MOVA scheme is the one mentioned in Subsection 6.1.3. It consists of a modular exponentiation followed by a discrete logarithm computation. The modular exponentiation can be implemented by the classical methods such as the square-and-multiply method. For the discrete logarithm computation we consider three variants which are the use of a precomputed table of all discrete logarithms, the Shanks baby-step giant-step (BSGS) algorithm and Pollard's rho method. The choice of the algorithm will strongly depend on the amount of memory the signer has at disposal, namely the Pollard rho method requires almost no memory while the BSGS method is a time-memory tradeoff. Below we discuss the method of precomputed table and we refer to [100] for a description of the two other methods.

Given p prime, g a generator of a cyclic group G, subgroup of $\mathbb{Z}_p^*$, and $d = |G|$, we construct a table with entries (g^j, j) for $0 \leq j \leq d$. Building this table is a time and memory consuming task, but once the table exists, finding the discrete logarithm consists of a simple look up operation.

There are several ways of constructing such a table. One can use a two dimensional array and sorting it by the first component. Finding the discrete logarithm is then reduced to a binary search. Alternatively, one can use conventional hashing on the first component to store the entries in a hash table, in which case placing an entry and searching for an entry in the table takes constant time. Another advantage is the fact, that we do not need space for g^i. Especially when $p \gg d$, this can save an enormous amount of memory. The only difficulties are finding a suitable hash function and dealing with collisions without losing too much time.

Time complexity of the construction of the table is $O(d)$ multiplications (plus $O(d \log d)$ comparisons to sort). Space complexity is $O(d(\log d + \log p))$ for the sorted table, resp. $O(d \log d)$ for the hash table. The running time to find a discrete logarithm for the sorted table is $O(\log d)$, for the hash table $O(1)$.

6.4 Implementation

The implementation of all algorithms has been written in C using the GNU Multiple Precision Arithmetic Library (GMP) [68] and was done by Yvonne Anne Oswald during her semester project [122]. This library provides highly optimized arithmetic functions on large numbers. Most of the basic computations in $\mathbb{Z}$ have been performed using GMP such as integer multiplication or the modular exponentiation. For all implemented homomorphisms, we focused on the case where the modulus n is of size of 1024 bits.

6.4.1 Quartic Residue Symbol

Our principal optimization effort focused on the two algorithms computing the quartic residue symbol. In particular, we minimized the number of function calls, used some of the more sophisticated GMP functions, reduced the number of `mpz_t` (C data type for a multiple precision integer) variables whenever possible and applied general C optimization techniques such as described in [64,92]. In addition, we used profiling and tried out different compiler optimization levels.

The basic algorithm has been implemented using the above remarks as well as the methods for computing the subfunctions which are explained in Subsection 6.2.1. We proceeded in the same way for the algorithm of Damgård and Frandsen. Additionally, we tested whether the use of an approximative norm allows to obtain effective improvements. We implemented both the standard norm and the norm Damgård and Frandsen suggest. The standard norm consists of only two GMP functions: one multiplication and one combined addition/multiplication whereas the approximate norm involves one bit scan to determine the size of the real part, one shift operation to extract the 8 most significant bits, one multiplication for the squaring of these 8 bits and another shift operation to put the result back to its correct position. We apply the same procedure on the imaginary part and we add the two approximate squarings up. So, we need additional operations to reduce the size of the numbers we have to multiply.

As GMP is a highly optimized library, computing the standard norm takes little time and the additional operations of the approximate norm only amortise if the real and the imaginary part are larger than 2048 bits. This and the fact that the norm of α and β decreases with each iteration convinced us to use the standard norm instead.

6.4.2 Discrete Logarithm

Here, we would like to present how we manage the computation of the discrete logarithm in the case of the precomputed table.

In this suggested instantiation of the MOVA scheme, p is typically of 512 bits and d is a 20-bit prime. Creating a table with d entries of size 532 bits is not convenient (or even impossible sometimes) on a usual desktop computer. Furthermore, as shown in Subsection 6.3, this is not the most efficient variant. Therefore, we decided to use a hash table (key 512 bits, data 20 bits, 2^{20} entries). We found some existing hash table data structures written in C, but they do not fulfil our requirements. They are either too slow, support C types only, do not allow tables that large and/or they store the key as well.

To avoid problems, we did not adapt any of the existing data structures, but implemented a hash table ourselves providing enough storage and a collision handling mechanism suitable for our needs. Our solution is a hash table consisting of an array of unsigned integers. This array is of maximal length 2^{24} to reduce collisions.

An unsigned integer is 32 bits long, so it was possible to store the data for the logarithm as well as using one of the higher order bits as a flag for collisions. Because the key is large and we wanted to avoid any unnecessary computation, we chose to use the 24 least significant bits of the key as the index into the hash table, in case of collision the next 24 bits, etc. By selecting 24 bits instead of the possible 20 bits, we minimize the occurrence of collisions. Tests have shown that most collisions are resolved by choosing the next 24 bits. We tried out other hash functions, but we did not achieve a gain of speed. This way, the size of the table is 64 MB.

To find the correct discrete logarithm for $y \in G$, one has to check if the collision flag at the corresponding array field is set, to decide if one can return the logarithm stored in the field or if one has to continue with the next field.

The implementation of the BSGS was done in a similar way. As the table contains much less entries, collisions hardly ever occur. The implementation of the Pollard rho method did not require any special treatment.

6.5 Results

In this section we present the results of the timing measurements we conducted to determine how well the different algorithms perform. In order to measure the running time precisely, we used functionalities offered by `frequence_cpu.h` by Victor Stinner [141]. The tests have been done on an Intel(R)4 1.4 GHz Desktop Computer with 256 MB RAM. Our results are average values produced by test series of 1000 tests.

6.5.1 Quartic Residue Symbol

We have considered the quartic residue symbol $\chi_\beta(\alpha)$ where α is a Gaussian integer with real and imaginary part of 1024 bits and $\beta = \pi\sigma$ a product of two primes and of size of 512 bits in each component. In such a situation, we have to consider a variant of the Damgård and Frandsen algorithm, we call the mixed algorithm. Namely, since α is much bigger than β it is more efficient in this case to compute first $\hat{\alpha} \leftarrow \mathrm{Red}_\beta(\alpha)$ and apply the Damgård and Frandsen algorithm on $\chi_\beta(\hat{\alpha})$. Timed results and number of iterations are given in Table 6.1. The mixed algorithm is then the most judicious choice for fast implementations. The same phenomenon occurs for the case $\beta = \pi$ as well.

	time in ms	iterations
Basic algorithm	32.12	248.81
Damgård's algorithm	50.63	766.12
Mixed algorithm	24.65	511.92

Table 6.1: Quartic Residue Symbol with $\beta = \pi\sigma$

6.5.2 Signature Generation

Here, we finally compare the time required for generating a MOVA signature with the different homomorphisms. We consider a signature size of 20 bits (except for RSA). We omit the time required by the generation of the values Xsig_i's. Hence, we just have to compare the time required for computing 20 Jacobi symbols $(\cdot/p)_2$ (or $(\cdot/q)_2$), 20 quartic residue symbols with $\beta = \pi\sigma$, 10 quartic residue symbols with $\beta = \pi$, 1 homomorphism based on the discrete logarithm in a hidden subgroup and 1 RSA homomorphism. We recall that for all these homomorphisms, we take a modulus n of size of 1024 bits. Results are given in Table 6.2.

Homomorphism	time in ms
Quartic Residue Symbol ($\beta = \pi\sigma$)	493.01
Quartic Residue Symbol ($\beta = \pi$)	90.32
Jacobi Symbol (ordinary algorithm)	25.22
Jacobi Symbol (`mpz_jacobi`)	2.32
Discrete Logarithm (Precomputed Table)	9.66
Discrete Logarithm (BSGS)	19.47
Discrete Logarithm (Pollard's rho)	74.93
RSA	33.87

Table 6.2: Results Comparison Signature Schemes

We have implemented the Jacobi symbol using a similar algorithm as Algorithm 6.1 and the basic GMP subroutines in order to have a fair comparison with our implementation of the quartic residue symbol. We note that the highly optimized GMP implementation of the Jacobi symbol `mpz_jacobi` provides the fastest signature generation and that the quartic residue symbol χ_π is about 4 times slower than our implementation of the Jacobi symbol. This is mainly due to the fact that all operations are performed in $\mathbb{Z}[i]$ instead of $\mathbb{Z}$. We remark that the variant $\chi_{\pi\sigma}$ is much slower than for χ_π since we have to perform two times more quartic residue computations and that β is two times greater. The variants of the discrete logarithm offer a very competitive homomorphism. In particular, except for the variant using the Pollard rho method this homomorphism is more efficient than an RSA ordinary signature. In particular, the variant with the precomputed table is three times faster than an RSA signature.

Note that these results directly apply to the confirmation protocol since the number of homomorphism evaluations the prover needs to perform is proportional (except for RSA) to that required for the signature generation. For the denial, homomorphism computations represent the principal computational task for the prover as well. However, the smallest prime number p_d of d here influences the number of homomorphism computations in a quadratic way, since this one also depends on the signature size. Since denial protocol is rarely performed in classical applications, optimizations should not focus on this protocol.

Chapter 7

Applications of the MOVA Scheme

Since the invention of the undeniable signatures, several applications of these primitives were mentioned or developed in the literature. In their seminal paper in 1989, Chaum and van Antwerpen [40] motivated their introduction mainly for privacy reasons, arguing that undeniable signatures are much more suitable for signing confidential contracts or sensitive and private information. Licensing of sensitive software is an often cited possible application which was first proposed by Chaum [36] in 1990. More precisely, a software company signs sensitive software using an undeniable signature scheme and restricts the verification of the software authenticity to the customers who paid for a license. In addition to these applications, undeniable signatures were used in the design of digital cash protocols as illustrated in the following articles [25, 39, 128]. Finally, we point out that applications of undeniable signatures were also investigated in the context of electronic auctions [134].

The aim of this chapter is to provide an analysis of some potential applications of the MOVA undeniable signature scheme. Here, we focus on the specific properties provided by this scheme in order to find dedicated applications. In particular, the size of the MOVA signatures allows to consider additional applications taking advantage of very short signatures.

We emphasize that it is sometimes a matter of taste to decide whether one application needs a given property or not. For instance, while some people consider that we can trust some authorities, others would never accept such a fact. This can lead to a decision whether the need of non-repudiation (thus signatures) is necessary or not.

In the next section, we give a high-level approach of different considered applications of MOVA by providing the context and a short analysis of the advantages offered by MOVA. The subsequent section is fully dedicated to an SMS-based lottery application for which we develop a more detailed analysis.

7.1 Potential MOVA Applications

7.1.1 General Properties of MOVA

To begin with, we recall the principal properties of the MOVA scheme:

- the non-repudiation
- the invisibility of the signatures
- the size of MOVA signatures which is fully scalable depending on the security level
- the ability to verify huge sets of signatures with constant communication

Though the aforementioned properties are not specific to the MOVA scheme when treated separately, we are not aware of the existence of another cryptographic primitive which fullfils all of them simultaneously. In particular, the non-repudiation and the size of the signatures are very specific to our scheme. We believe that both of these properties are prone to lead to some interesting applications. Surprisingly, on-line verification which was the main goal of undeniable signatures is not directly used in these properties. It is only required in an indirect way to achieve very short signatures. Focusing on the MOVA scheme, we will not necessarily try to develop applications taking advantage of the invisibility of undeniable signatures. What is more, it is intended to find an application which strictly requires properties of the MOVA scheme and which cannot be achieved by other methods (even standard undeniable signatures). We compare in Table 7.1 the properties achieved by MOVA with different cryptographic primitives which are MAC algorithms, classical digital signatures (universally verifiable), and undeniable signatures developed at a prior time than MOVA.

Primitives	Non-Repu.	Short Size	Invisibility	Batch Ver.
MOVA	✓	✓	✓	✓
Pre. Undeniable Signatures	✓		✓	✓
Classical Signatures	✓			
MAC		✓	✓	

Table 7.1: Comparison with other cryptographic primitives

In this table, the criterion "short size" refers to a very short string, i.e., potentially less than 80 bits. Therefore, even short classical signatures of 160 bits due to Boneh, Lynn, and Shacham [22] or those from Courtois, Finiasz, and Sendrier [44,59] achieving 120 bits with reasonable parameters, are not classified as short.

In what follows, we briefly analyze some potential applications for which MOVA signatures can offer specific advantages over other techniques.

7.1.2 Banknote Protection

Context. We consider a type of banknotes containing a MOVA signature on them. Its role consists in providing some evidence of the authenticity of the banknote. The signature is obtained by signing an identifier (string of characters) associated to the banknote with respect to the secret key of the banknote supplier (typically a central bank). This identifier should preferably have the property of being not reproducible such that a counterfeiter cannot create such a value on another banknote. For instance, we can imagine that this identifier is directly obtained by some unique physical properties of the banknote. If this property cannot be achieved, we cannot thwart copies of genuine banknotes. However, the counterfeiter should be unable to create genuine banknotes with a new identifier, which can facilitate the central bank to track counterfeit banknotes.

Involved Entities. The different entities involved in the use of a banknote are composed of the *central bank*, a *vendor* and a *customer*. The central bank produces the banknotes and add a MOVA signature of the identifier associated to the banknote on this one. Once the banknote is on the market, a customer can use it to pay a vendor. In order to accept the banknote as genuine, the vendor has to interact online with the central bank. If this one confirms the validity of the signature, the vendor then cashes the banknote.

Trusted Central Bank. The central bank is an authority and may be considered as trustful when it confirms the validity of a banknote. Under this quite plausible assumption, we can avoid in practice the use of a confirmation protocol for any banknote verification between the central bank and the vendor. It is sufficient that the central bank checks the validity of the signature by itself and simply sends an acknowledgment of this fact to the vendor. The connection between the vendor and the central bank must be secure. In particular, the central bank has to be authenticated by the vendor.

Non-Repudiation. Even if one can consider that the central bank is trusted when this one confirms the validity of a signature, it is desirable for a customer that the central bank is able to formally prove the invalidity of a counterfeit banknote. Hence, if a customer complains about this fact, a legal authority should be able to require a formal proof of the invalidity of a signature related to the alleged counterfeit banknote.

Signature Size. Below, we present three variants related to the verification of the signatures and analyze the respective signature size for a security level of 2^{-30}.

More precisely, we consider an existential forgery attack and assume that Xgroup is adjusted such that any attacker succeeds with a probability $\mathsf{Succ}^{\text{Lsig-}S\text{-GHI}} \approx d^{-\text{Lsig}}$. As in Subsection 4.4.4, we can deduce that the size s in bits of the signature should satisfy

$$s \approx 32 + \log_2(q_{\text{V}}) + \log_2(q_{\text{S}}),$$

in order to guarantee a security of 2^{-30}.

- **Server Without Protection.** If the verification server of the central bank gives a totally free access to anybody and does not limit the verification with respect to a given banknote (random value), we can imagine that $q_V \approx 2^{40}$ and $q_S = 0$, which leads to a signature size of about 72 bits.

- **Server with Limited Access.** Here, we assume that only a number of limited entities (vendors) have access to the verification server. The communication is secure and the verifier is authenticated by the server. The server does not allow the verifier to verify too many invalid signatures. For instance, after 3 invalid signatures on the same value the server could stop the verification process. This scenario allows to avoid massive queries in order to forge a signature. Therefore, we can imagine signatures of size of about 40 bits.

- **Server with Paying Verification.** We consider a scenario where each signature verification costs a little amount of money for the verifier, e.g., 1 cent. This restriction will also strongly limit the counterfeiter to make many verifications. We can reasonably assume that, he will not pay more than 10 US dollars to have a probability of 2^{-30} of forging a signature. Namely, it would cost to him 10 billion dollars in average to forge one signature. Therefore, we can consider $q_V \approx 2^{10}$ which leads again to signatures of about 40 bits. This can be adapted depending on the size of the value of the banknote as well.

Alternatives to MOVA.

- **Symmetric-key Cryptography.** Using techniques of the symmetric cryptography would allow the central bank using a secret key to generate a secret string from the identifier associated to a banknote e.g., by using a MAC algorithm. This would possibly lead to a very short string as well. This technique would also require an online verification with the central bank. However, there is no way for the central bank to formally prove the invalidity of a signature. Hence, even a legal authority could not decide whether the banknote is a counterfeit one as there is no alternative to completely trust the central bank without disclosing the secret key.

- **Classical Digital Signatures.** Replacing a MOVA signature by a classical one would have the following consequences. The verification of the banknote would be universal, i.e., anybody would be able to check the validity of the signature without any online connection with the central bank. So, a customer is directly convinced on the validity/invalidity of the signature and does not need any interaction with the central bank to get a proof of this statement. The main disadvantage is that the size of the signatures would be clearly longer. This may be inadequate if one wants that humans are able to manually handle the signatures.

Batch Verification. This property can be very useful when the vendor would like to check the validity of several banknotes using one communication session with the server of the central bank.

To summarize, this application mainly requires the non-repudiation property and very short signatures. Looking at Table 7.1 shows that only MOVA can satisfy these requirements.

7.1.3 Sensitive Software Protection

We revisit here the application on the licensing of sensitive software put forth by Chaum [36] and show how additional advantages can be exploited with MOVA, notably the signature size.

Context. A software company signs his software products with MOVA signatures. The MOVA signature is then provided with the software or in a separate way (e-mail, by phone, ...). Then, the client has access to an online verification service of the signature in order to be convinced that the software is genuine and does not contain any backdoor, viruses, etc. This service is only restricted to the customers who paid for a license. For this verification step, the client has to be authentified so that the company can check that this one paid for a license.

This scenario is applicable only when we consider software whose authenticity for the client is crucial (such as software used for sensitive data) since this application does not prevent illegal copies. What is valuable here is the authenticity of the product, i.e., the right to perform a confirmation protocol with the software company.

Personalized Signatures. To make more difficult to fraud, the software company could release different signatures for each customer. The company could for example add a serial number and sign a message composed of the software and the serial number. For some practical reasons, when the software is sold in “hard” (CD, ...), the serial number and the signature are added separately in the software box e.g., on a sticker. In this way, it is easier for the company to produce exactly the same CD and add the personalized part of the product separately (as it is already the case for some software licenses).

Advantages of MOVA. The major advantage of using undeniable signatures comes from the online verification of the software authenticity. Namely, a customer should not be able to transfer a proof of the validity of the MOVA signature, otherwise the authenticity of the product can be verified without the software company, i.e., without paying any license. The short size of MOVA signatures can be very useful when the signature should be entered by hand before the verification. In particular, this is the case when the signatures are personalized.

Alternatives. Classical digital signatures cannot be used here, since one can easily copy the software and the signature allowing a third party to verify the authenticity of the product by himself. MAC algorithms do not offer a convenient alternative in this application. First, the secret key cannot be given to the customer, otherwise he could convince anybody else by releasing this one. If the secret key is only known by the software company, then the verification does not consist in a proof and the customer will need to fully trust the software company during the use of the software. Since this application is dedicated to very sensitive software, it is clearly desirable that the customer can be formally convinced about the authenticity. Finally, we note that conventional cryptography does not seem to achieve non-transferability notion.

7.1.4 Credit Card Number Verification

Context. One of the major issues when using credit cards consists in verifying the validity of the credit card number. A way for preventing the generation of fake credit card numbers, could be to append a MOVA signature to a serial number of the card. So, except by copying a valid card number or forging MOVA signatures, it should not be possible (or difficult depending on the signature size) to generate fake credit card numbers. The verification of the credit card number has to be done online by establishing a connection with the credit card authority. A merchant who sells some goods or services which are provided later to the customer can delay the verification of all credit card numbers at the end of the day.

Major Advantages of MOVA. Since this number must be copied by hand such as on internet, or by phone, it is desirable that this one is as small as possible. An additional advantage is the batch verification of the MOVA signature. So, when the merchant does not need to check the validity of the credit card number immediately, he can verify all the credit card numbers simultaneously at the end of the day using only one verification protocol with the credit card authority. Note that the efficiency gain is rather for the credit card authority than for the merchant.

Alternatives. As usual, one of the major drawback of classical digital signatures is the size. This would not be appropriate for credit card numbers where we need a small number of digits. Replacing the MOVA scheme by a MAC algorithm, will no more allow to verify several signatures at the same time. Moreover, this requires to trust the credit card authority since this one could not provide any proof about validity or invalidity of the credit card number with a MAC.

7.2 SMS Lottery

The aim of this section is to propose a protocol for playing lottery (or any kind of similar games) in which the player receives a receipt by SMS based on a MOVA signature. The role of the signature is to provide a strong evidence that the player's bid was registered by the lottery organization.

The work exposed in this section was realized as part of the master thesis [121] of Florin Oswald.

7.2.1 Scenario

We consider a scenario in which a lottery player would like to play lottery using his mobile phone. We would like that playing lottery is possible even with a mobile phone having only basic features such as sending SMS composed only of text. To participate in a lottery drawing, the player sends the drawing number (or drawing date) and the selected bid by SMS. By default, the lottery ticket remains associated to the phone number but can be associated to another one if one wishes to offer a lottery ticket to someone else. The mobile communication company (service provider) is responsible to charge the price of the "lottery ticket" on the phone account of the player (based on a contract). It also forwards the information with the associated phone number (called the *order*) to the lottery organization.

The lottery organization sends back a MOVA signature of the order. The order and the MOVA signature is the lottery ticket. After the drawing, the player with correct phone number can request his winnings by showing the ticket. Small winnings may automatically be credited by the service provider.

In this scenario, we particularly take advantage of the size of the signatures and non-repudiation of the MOVA scheme. While the former is required in order that a signature fits in a SMS, the latter is crucial in case of dispute about the validity of a ticket corresponding to a large winnings.

In our protocol, we consider the three following entities:

- the player (**P**)
- the service provider (mobile communication company) (**S**)
- the lottery organization (**L**)

The role of each entity is summarized below.

Player. The player simply wants to play at the lottery using SMS. To this, we assume that his mobile phone communications are transmitted throughout the network of the service provider **S**. He only needs a regular contract with this one in order to have access to the lottery game. To play to the lottery, he sends information related to the drawing and his chosen numbers by SMS to the lottery throughout the network of **S**.

Service Provider. This entity is responsible for transmitting the communications[1] between the player **P** and the lottery organization **L** which are required by the lottery protocol. The main role of **S** is also to charge costs of the player on his phone communication bill corresponding to the SMS lottery tickets he bought. An additional task of the service provider consists in paying the small winnings to the player. To this, the winnings can be credited to the player's phone bill.

Lottery Organization. This one is mainly responsible for organizing the lottery and notably the drawing operations. It receives the "SMS lottery orders", checks that the format is correct and generates a receipt using the MOVA scheme. **L** stores all the lottery tickets and related information. After the drawing, the lottery organization checks which tickets have some winnings and transmit related winnings information by SMS to the corresponding players through **S**. As said above, small winnings are directly credited by **S** to the corresponding players, while large winnings should be addressed by other means.

7.2.2 Our Lottery Protocol

This lottery protocol is mainly composed of three sub-protocols which are respectively used for the playing phase, the verification of the lottery ticket, and the

[1]Note that communications between the provider and the lottery organization are likely to be transmitted on another communication channel like the Internet.

winnings distribution. We also consider one additional procedure for a player which claims to have a winning ticket that the lottery organization does not accept as valid.

Before a presentation of the protocols, we assume that the lottery organization possesses a valid pair of keys associated to the MOVA undeniable signature scheme.

Playing Protocol

Here, we describe the so-called "playing protocol" which consists for the player in sending the numbers of its choice for a drawing to the lottery organization. A high-level description of the protocol is depicted in Figure 7.1.

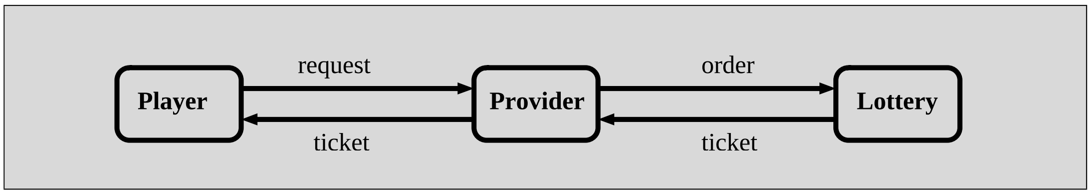

Figure 7.1: High-level description of the playing protocol

The detailed description is provided below.

1. **P** chooses the drawing number DrawN for which he would like to participate and the information related to his bid (chosen numbers) denoted by $\mathsf{Numbers}$. He sends the request $(\mathsf{DrawN}, \mathsf{Numbers})$ by SMS to **S**.

2. **S** appends the phone number ID of **P** and sends the order $(\mathsf{DrawN}, \mathsf{Numbers}, \mathsf{ID})$ to the lottery.

3. **L** checks that $(\mathsf{DrawN}, \mathsf{Numbers})$ has the correct format and that the drawing is in "playing" phase. The lottery organization keeps the information $(\mathsf{DrawN}, \mathsf{Numbers}, \mathsf{ID})$ called "ticket order" in memory. Then, **L** generates a receipt
$$\mathsf{Rec} := \mathrm{MOVA}(\mathsf{DrawN}|\mathsf{Numbers}|\mathsf{ID})$$
by signing with the MOVA scheme. Then, the lottery sends the ticket
$$(\mathsf{DrawN}, \mathsf{Numbers}, \mathsf{ID}, \mathsf{Rec})$$
to **S** with an additional information TiPr about the price of the ticket.

4. **S** forwards the ticket to the player and charges his account according to TiPr.

Remark 7.2.1. If a player wants to buy several tickets for the same drawing, it may be required to identify each ticket separately. Thanks to a counter, one can concatenate an incremented integer to ID which identifies each ticket. For instance, instead of one identifier ID, we would get $\mathsf{ID}|1$, $\mathsf{ID}|2$, etc.

Verification Protocol

In order to formally guarantee the authenticity of the ticket's receipt to the player, this one needs to be given an access to a verification procedure with the lottery organization. The players will be given a *verification Interface* provided by the lottery organization and which allow to verify the validity of a ticket. The verification is restricted to the tickets for which the corresponding ticket order $(\mathsf{DrawN}, \mathsf{Numbers}, \mathsf{ID})$ was previously submitted in the playing protocol.

In practice, this interface is likely to be a software which can be installed on a personal computer (or very advanced mobile phone) and connect to the lottery for requesting a ticket verification. Moreover, since the interest of this lottery application is to work on a very limited mobile phone, we cannot assume that this verification can be achieved from the mobile phone.

The verification protocol is given below.

1. The player **P** sends the ticket $(\mathsf{DrawN}, \mathsf{Numbers}, \mathsf{ID}, \mathsf{Rec})$ through the verification interface to **L**.

2. The lottery organization checks whether $(\mathsf{DrawN}, \mathsf{Numbers}, \mathsf{ID}, \mathsf{Rec})$ lies in his database and verifies whether the MOVA signature Rec is valid or not. Accordingly, **L** interacts with **P** in a confirmation (resp. denial protocol) to show that the ticket is valid (resp. invalid). If the ticket order does not lie in the database, **L** does not launch a verification protocol but sends the message "`ticket not registered`".

Winnings Distribution Protocol

This phase occurs after the drawing has been made and consists for the lottery in dealing with the winnings of lucky players.

1. After the drawing, **L** examines in its stored data which tickets are winning. According to this information, the lottery organization sends the list of all winning tickets with the corresponding amount of the winnings Wi to the service provider.

2. S sends an SMS to each ID corresponding to the winning tickets containing Wi. If Wi is small, the service provider automatically charges the player's phone

account. In case of a big winnings, this is handled by classical means (bank account transfer, etc.)

Dispute

Assume now that a player claims to have a valid winning ticket which is considered as invalid by the lottery organization. So, if both parties (**P** and **L**) do not find any agreement, the player can take legal action against the lottery. In this case, the lottery organization has to perform a MOVA denial protocol to prove the invalidity of the receipt in the presence of a legal authority.

Remark 7.2.2. In such a situation, the signature is used as a strong evidence that the player behaved honestly. However, since the implementation of cryptography in the real world cannot be considered as totally ideal, the signature should not be an absolute mean to determine whether a lottery ticket was honestly obtained. For instance, the ticket might have been generated by a lottery employee who got access to the MOVA secret key. For such reasons, we believe that the final decision should be taken by a legal authority after an investigation of the different facts.

7.2.3 Security Analysis

Forgery Attacks

If we put aside the possible misbehaviour of the service provider and the lottery organization, the main threat concerns a malicious player willing to forge a false (non-paying) winning lottery ticket. Such an adversary can be modeled as a MOVA forger whose goal is to forge valid tickets and especially, the winning ones. Therefore, this kind of attack seems to lie between an existentially forgery and universal forgery attacks in terms of difficulty. While an adversary is not ensured to forge a valid (even non-winning) ticket by succeeding in an existential forgery attack, this one does not strictly need to succeed in a universal forgery attack in order to forge the ticket with the highest winnings, since he may have several identifiers at disposal. This leads us to consider the following adversarial game.

Game$^{\mathsf{lot\text{-}cma}}$. *Let $\mathcal{M}$ be the message space of all possible ticket orders and $\mathcal{F}$ be the forger. First, $\mathcal{F}$ receives the lottery's* MOVA *public key $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$ which was generated by $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$. Then, the game is composed of two phases called* Pre-Draw *phase and* Post-Draw *phase respectively.*

1. **Pre-Draw phase.** *$\mathcal{F}$ has access to a signing oracle* Sign*, a verification oracle* Ver *restricted to the message-signature pairs (m, σ) such that m was queried to* Sign*, and possibly the random oracles (such as* GenS *in the* MOVA *case).*

2. **Post-Draw phase.** *A subset $\mathcal{M}_w \subseteq \mathcal{M}$ composed of N_w different messages is picked uniformly at random and is given to $\mathcal{F}$. The forger $\mathcal{F}$ does not have access to the oracles* Sign *and* Ver *anymore. At the end, $\mathcal{F}$ outputs a messsage-signature pair (m^*, σ^*) and wins the game if $m^* \in \mathcal{M}_w$, the pair is valid, and if m^* has not been queried to the oracle* Sign*.*

The success probability of $\mathcal{F}$ in the above game is denoted by $\mathsf{Succ}_{\mathcal{F}}^{\mathsf{lot\text{-}cma}}$.

Note that the Pre-Draw phase corresponds to the period of time before the drawing. At this moment, the adversary does not know which tickets have an interesting winnings. After the drawing, the adversary does not have access to the signature verifications and cannot buy any tickets for this drawing. Provided that the lottery organization refreshes the MOVA pair of keys for each drawing, this corresponds for the adversary to have no access to the oracles Sign and Ver anymore, as modeled in the Post-Draw phase.

Below, we exhibit a reduction between the above game and the GHI problem using some similar simulation techniques which have been employed in Theorem 4.4.1 to show that MOVA resists existential forgery attacks.

Theorem 7.2.3. *Let $S = \{(\mathrm{Xkey}_1, \mathrm{Ykey}_1), \ldots, (\mathrm{Xkey}_{\mathrm{Lkey}}, \mathrm{Ykey}_{\mathrm{Lkey}})\}$ and N_{tot} denote the cardinality of $\mathcal{M}$. Set $p_w := N_w/N_{\mathrm{tot}}$. Let q_S be an integer satisfying $q_S \cdot p_w \geq 1$. Assume that the signer's public key is valid and* GenS *is a random oracle. Consider the* Lsig-S-GHI *problem with the same parameters as for the* MOVA *scheme, i.e., $G = \mathrm{Xgroup}$, $H = \mathrm{Ygroup}$. Assume that for any solver $\mathcal{B}$ with a given complexity, we have*

$$\mathsf{Succ}_{\mathcal{B}}^{\mathrm{Lsig}\text{-}S\text{-}\mathrm{GHI}} \leq \varepsilon.$$

Then, any forger $\mathcal{F}$ with similar complexity using q_S signing queries wins the lottery game under a chosen-message attack with a probability

$$\mathsf{Succ}_{\mathcal{F}}^{\mathsf{lot\text{-}cma}} \leq \left(\sum_{i=0}^{q_S} \frac{\binom{N_w}{i}\binom{N_{\mathrm{tot}}-N_w}{q_S-i}}{\binom{N_{\mathrm{tot}}}{q_S}} \cdot (1-q)^i \right)^{-1} \cdot \frac{\varepsilon}{q},$$

with $q = 1/(p_w + q_S p_w)$.

Proof. From a forger $\mathcal{F}$ who wins in the lottery game, we construct a simulator $\mathcal{B}$ which solves the Lsig-S-GHI problem. As usual, $\mathcal{B}$ needs to simulate the lottery game environment of $\mathcal{F}$ (different oracles) in order to use $\mathcal{F}$ towards solving the GHIP challenge. First, the simulator $\mathcal{B}$ receives the GHIP challenge $x_1, \ldots, x_{\mathrm{Lsig}}$. He picks N_w different messages uniformly at random in $\mathcal{M}$ and build the set $\mathcal{M}_w$. Then, $\mathcal{B}$ runs the forger and simulates the queries to the random oracle GenS, q_S queries to the signing oracle Sign and q_V queries to the oracle Ver. We can assume that all messages sent to Sign resp. Ver were previously queried to GenS (since the oracle Sign resp. Ver has to make such queries anyway). Additionally, we can assume that the forged pair outputted by $\mathcal{F}$ has been queried to GenS. $\mathcal{B}$ simulates the oracles GenS, Sign, and Ver in the Pre-Draw phase as follows.

GenS. As usual, the random oracle GenS is simulated by maintaining a list of the queries with corresponding answers. A fresh query m is simulated as follows. If $m \notin \mathcal{M}_w$, $\mathcal{B}$ generates a type-1 answer. Otherwise, a type-1 answer is generated with probability $1-q$ and a type-2 answer with probability q. We recall that type-1 answers are of the form

$$\mathrm{Xsig}_i := dr_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Xkey}_j \quad \text{for } i = 1, \dots, \mathrm{Lsig},$$

while type-2 answers are of the form

$$\mathrm{Xsig}_i := dr_i + x_i + \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Xkey}_j \quad \text{for } i = 1, \dots, \mathrm{Lsig}.$$

For each message, $\mathcal{B}$ keeps the coefficients $a_{i,j}$'s and r_i's and answer type in memory.

Sign. For a message m, if the answer to the GenS query of m was of type-1, then $\mathcal{B}$ answers $\mathrm{Ysig}_i := \sum_{j=1}^{\mathrm{Lkey}} a_{i,j}\mathrm{Ykey}_j$ for $i = 1, \dots, \mathrm{Lsig}$. Otherwise, it aborts the simulation. The simulator adds a tag in the stored message m to specify that m was queried to Sign and stores the signature σ as well.

Ver. For any message-signature pair (m, σ) queried to Ver, $\mathcal{B}$ first checks whether m was previously submitted to Sign. If this is not the case, $\mathcal{B}$ answers the tag "`ticket not registered`". Otherwise, the simulator checks whether σ is valid or not and simulates the appropriate protocol accordingly as in Theorem 4.4.1.

In the Post-Draw phase, $\mathcal{B}$ releases $\mathcal{M}_w$ to $\mathcal{F}$. Then, the simulator only needs to simulate the oracle GenS. This can be done perfectly and exactly as in the Pre-Draw phase. At the end, $\mathcal{F}$ outputs a pair (m^*, σ^*).

It remains to compute the probability for $\mathcal{B}$ to have simulated correctly and to solve the GHIP instance. First, we note that the oracles are perfectly simulated except when a type-2 query is sent to the signing oracle. Hence, the probability that $\mathcal{B}$ simulates Sign correctly is given by

$$\gamma(q) := \sum_{i=0}^{q_S} \frac{\binom{N_w}{i}\binom{N_{\mathrm{tot}}-N_w}{q_S-i}}{\binom{N_{\mathrm{tot}}}{q_S}} \cdot (1-q)^i,$$

where N_{tot} is the cardinality of $\mathcal{M}$ and N_w the cardinality of $\mathcal{M}_w$. The proof can be concluded by remarking that the probability for a successful $\mathcal{F}$ to have a simulated GenS answer of m^* of type-2 is equal to q. A heuristic argument leads us to consider $q = 1/(p_w + q_S p_w)$ to maximize the expression $q \cdot \gamma(q)$. $\square$

In what follows, we briefly explain how we can approximate $\gamma(q)$ and how the choice for q in the above theorem has been made. The main reason we get a probability in a so complicated form, comes from the fact that picking q_S different messages in $\mathcal{M}$ uniformly at random and counting the number of messages in $\mathcal{M}_w$ does not formally lead to a binomial distribution. This leads to a hypergeometric distribution of the form

$$\frac{\binom{N_w}{i}\binom{N_{\text{tot}}-N_w}{q_S-i}}{\binom{N_{\text{tot}}}{q_S}}.$$

However, it is well known that if N_{tot} is much larger than q_S, the hypergeometric distribution is very close to a binomial distribution. In this case, probabilities for the different messages to lie in $\mathcal{M}_w$ can be seen as independent. Thus,

$$\gamma(q) \approx (1 - q \cdot p_w)^{q_S},$$

where $p_w := N_w/N_{\text{tot}}$. We set $\gamma_0(q) := (1 - q \cdot p_w)^{q_S}$. By differentiating the function $q \cdot \gamma_0(q)$, we derive that choosing

$$q := \frac{1}{p_w + q_S p_w}$$

leads to a maximized value of the function $q \cdot \gamma_0(q)$. This value can be bounded as follows

$$\frac{1}{p_w + q_S p_w}\left(1 - \frac{1}{1 + q_S}\right)^{q_S} \geq \frac{1}{e(q_S + 1)p_w},$$

where e denotes the natural logarithm base. Note that both terms are very close when q_S is large. This shows that

$$\mathsf{Succ}_{\mathcal{F}}^{\mathsf{lot\text{-}cma}} \approx e(q_S + 1)p_w \cdot \varepsilon \tag{7.1}$$

for q_S reasonably large but enough smaller than N_{tot}.

Additional Threats

Here, we would like to discuss some other threats related to this lottery scenario. They are of less interest for this thesis since they are not related with the different parameters of the MOVA scheme. For this reason, we are not going to make a detailed and exhaustive treatment. The interested reader will find more details in Oswald Master's thesis [121].

Invalid Signatures Released by the Lottery Organization. A major risk for the players comes from the possible release of invalid MOVA signatures by the lottery organization. In practice, the verification interface is likely to be used only by a small number of skeptical players. On the one hand, this verification interface should prevent from players claiming after the drawing to have received an invalid ticket and that the lottery organization cheated. In short, the responsibility for verifying the ticket authenticity is shifted to the player. On the other hand, this verification interface should also prevent the lottery to generate invalid tickets. Namely, before the drawing the value of a ticket is very small and the incentive for the lottery to cheat at this point is quite low. Since the lottery take risks to be detected by some players and have its credibility and reputation affected, it is unlikely that the lottery is going to cheat for a very small amount of money.

In addition to the tickets verification by the player, one can possibly imagine that the service provider is involved in checking the authenticity of the tickets. Since its reputation may also be affected by some complains, this one may decide to proceed in some ticket verifications. What is more, the service provider has the infrastructure at disposal for easily getting access to signature verifications. Namely, the provider certainly communicates with the lottery using computers through a classical network. To summarize, a verification of tickets by the service provider should not be a constraining task. The verification can also be done at the end of each day or just before the drawing using a batch verification of the MOVA scheme.

Non-Cooperation by the Lottery in the Verification Protocol. The malicious behaviour of the lottery organization presented above, can be thwarted provided that this one cooperates for the verification protocol. However, the lottery organization may claim that this ticket order was not registered in its database in order to avoid a verification. To prevent from such a behaviour, the players should be given a procedure allowing them to provide evidence that they previously made a ticket order. One could imagine that in case of non-cooperation by the lottery, the player can contact the service provider which stores all the communications related to the lottery game in order to get a certification that he really submitted this ticket order. In this way, the lottery should not be able to refuse a valid verification request provided that the service provider does not collude with the lottery organization maliciously. Again, since the ticket value at this time is quite small, the interest for the lottery or the service provider to cheat is very limited. Moreover, by doing so both entities take risk to create a damage to their image. Finally, we suggest that the above procedure should not be totally free of charge in case the player made an unjustified request in order to prevent from possible abuses by players.

Privacy of the Players. Identity of players and especially the “lucky” ones should be kept secret by the lottery organization. Here, all data are transmitted through the communication channels of the service provider. So, it is very important that these communications remain confidential and some legal means are required to prevent the service provider from misusing information. For instance, the service provider could try to sell some knowledge about big winners.

7.2.4 Parameter Specifications and the Swiss Lotto Case

Here, we mainly derive some MOVA parameters for our SMS lottery protocol when applied to Swiss Lotto.

In the Swiss Lotto context, the player needs to choose 6 numbers out of 45 ones. A draw consists in picking 6 regular numbers and one additional number. The player wins some money if at least 3 regular numbers were found. Hence, the total number of possible bids is $\binom{45}{6} = 8'145'060$ and the total number of winning bids is

$$\sum_{i=3}^{6} \binom{6}{3}\binom{39}{6-i} = 194'130.$$

Consequently, we get the ratio $p_w = N_w/N_{\text{tot}} = 194'130/8'145'060 \approx 0.0238$. Note that the message space $\mathcal{M}$, we can consider in the adversary model may be larger than 8’145’060 (e.g., several identifiers). However, the ratio p_w between the number of winning tickets and the overall number of tickets remains constant. We also recall that the approximation (7.1) of the reduction factor obtained in Theorem 7.2.3 only depends on q_S and p_w. Some experimental computations showed that at least for the values $2^6 \leq q_S \leq 2^{15}$, $N_{\text{tot}} := 8'145'060$, and $N_w := 194'130$ this approximation is extremely accurate.

As suggested in the previous chapter, we propose to take the Legendre symbol $\text{Hom} = (\cdot/p)$ defined on $\mathbb{Z}_n^*$, where $n = pq$ is an RSA modulus. The size of a modulus of at least 1024 bits is recommended. Provided that the lottery does not use the key for the long term, 1024 bits are sufficient at this time and 1536 bits offer a very good margin of security.

To determine the signature size, we suggest a security of 2^{-30}, i.e.,

$$\mathsf{Succ}_{\mathcal{F}}^{\mathsf{lot\text{-}cma}} \leq 2^{-30}.$$

Since each query to the signing oracle costs a ticket price (about 1 swiss francs), we can assume that the adversary is not going to make too many signing queries so that $q_S = 2^{15} = 32'768$ is sufficient. Applying Theorem 7.2.3 and assuming that $\mathsf{Succ}_{\mathcal{B}}^{\text{Lsig-}S\text{-GHI}} \approx 2^{-\text{Lsig}}$, we deduce that Lsig should be 42 bits long, which can be encoded with 8 alphanumeric characters. To summarize, an adversary spending about 30'000 Swiss francs has only a probability of $2^{-30} \approx 10^{-9}$ to forge a winning ticket. Spending more money does not seem a better strategy since the ratio between q_S (cost for the adversary) and the success probability is constant by the approximation (7.1).

Since invisibility is not an issue in the lottery scenario, we do not need to adjust the parameters to achieve this property. Soundness probabilities of the verification protocols can be obtained exactly as in Subsection 4.4.4. The 4-move protocols lead to better parameters than their 2-move versions, so that we suggest the former. Hence, $\mathrm{Icon} = \mathrm{Iden} = 30$ leads to a soundness probability of 2^{-30} if the commitment binding property is ideal. For the validation of the public key, we propose to use the setup variant 4, which allows to consider $\mathrm{Lkey} = 2$. As in Subsection 4.4.4, we can show that $\mathrm{Ival} = 90$ with $q_{\mathrm{GenM}} = 2^{60}$ corresponds to a security of 2^{-30}.

Chapter 8

Generalized Chaum's Designated Confirmer Signature

The goal of this chapter is to review the original scheme of Chaum [38] as well as the underlying ideas of his construction in a formal and more general setting. Namely, his original article neither presents a formal model nor a security proof. Our principal motivation is that the scheme of Chaum remains at this time one of the most simple and elegant construction of designated confirmer signature scheme. In addition to this, we study the possibility to use an undeniable signature scheme in the construction of a designated confirmer signature, in particular reusing the confirmation and denial protocols.

As far as we know, the only generic constructions of designated confirmer signatures which are based on an undeniable signature scheme are that of Chaum [38] and the one of Okamoto [118]. The security of the latter was only proved in 2001 in [119] and its resistance against existential forgery under an adaptive chosen-message attack holds only against a classical adversary, i.e., anybody but the confirmer. To our best knowledge, the security of the Chaum's construction has not been proved yet. Moreover, the only known security flaw of this scheme was mentioned by Camenisch and Michels in [30]. The authors presented an attack against the invisibility of signatures in the adaptive scenario against the scheme of Michels and Stadler [105] and argued that the same kind of attack holds against the scheme of Chaum. In this attack, the adversary is able to transform a given message-signature pair in a new one such that the latter pair is valid only if the original pair is valid. Hence, the adversary breaks the invisibility of the first signature by sending the second pair to the confirmer for a confirmation (or denial) protocol.

In the first section, we present the generalized construction based on the scheme of Chaum. To this end, we use some cryptographic primitives such as an existentially forgeable classical signature and an existentially forgeable undeniable signature. In the subsequent section, we rely the security properties on that of the underlying primitives in the random oracle model. For instance, resistance against existential forgeries hold provided the existentially forgeable signature resists universal forgery under a no-message attack. We then propose a practical instantiation based on the undeniable signature scheme of Chaum and the plain DSA scheme. We conclude by showing that our construction is consistent with a theoretical result of Okamoto [118] stating that the design of a designated confirmer signature requires cryptographic primitives equivalent to public-key encryption.

8.1 The Generalized Chaum's Construction

To begin with the development of our construction, we recall some notations and the involved parties. We consider three entities that are the signer ($\mathbf{S}$), the confirmer ($\mathbf{C}$) and the verifier ($\mathbf{V}$). They all possess a pair of public/secret key $\mathcal{K}^{\mathbf{U}} := (\mathcal{K}_{\mathrm{p}}^{\mathbf{U}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{U}})$ for $\mathbf{U} \in \{\mathbf{S}, \mathbf{C}, \mathbf{V}\}$. The set of the message space is denoted by $\mathcal{M}$ and the set of the signature space is denoted by Σ. The designated confirmer signature scheme constructed in this section is denoted by Sign.

8.1.1 Building Blocks

In what follows, we present the different building blocks which are used to construct our designated confirmer signature scheme. To simplify the notations, the pair of keys of cryptographic algorithms are directly denoted with respect to their corresponding owner (signer $\mathbf{S}$ or confirmer $\mathbf{C}$) in the designated confirmer signature scheme Sign.

Existentially Forgeable Signature. We consider an existentially forgeable signature ExSign such as the plain RSA or plain DSA[1] scheme. We have a setup which generates the keys associated to this scheme (that of $\mathbf{S}$), $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$ which depends on a security parameter k. Let $\mathcal{M}_{\mathrm{ex}}$ denote the message space and Σ_{ex} denote the signature space of this scheme. We have a probabilistic signature generation algorithm

$$\sigma_{\mathrm{ex}} \leftarrow \mathsf{ExSign}(m_{\mathrm{ex}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}),$$

and a deterministic verification algorithm

$$0 \text{ or } 1 \leftarrow \mathsf{ExVerify}(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}})$$

outputting a bit telling whether $(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}) \in \mathcal{M}_{\mathrm{ex}} \times \Sigma_{\mathrm{ex}}$ is a valid message-signature pair. Both are some polynomial time algorithms. In addition, we have a probabilistic

[1]Plain DSA is DSA [4, 54] without a hash function.

polynomial time algorithm

$$(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}) \leftarrow \mathsf{ExForge}(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}})$$

which existentially forges a valid message-signature pair such that m_{ex} is *uniformly distributed* in $\mathcal{M}_{\mathrm{ex}}$. Moreover, for a given message m_{ex}, the distribution of σ_{ex} generated by ExForge is identical as that generated by ExSign for any $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$.

To prove the security of Sign, we will need to assume that ExSign satisfies universal unforgeability under a no-message attack.

Definition 8.1.1. *We say that the signature scheme* ExSign *resists against a universal forgery under a no-message attack if there exists no probabilistic polynomial time algorithm $\mathcal{B}$ that wins the following game with a non-negligible probability.*

Game: *$\mathcal{B}$ first receives the public key $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$ from $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$ generated randomly and depending on the security parameter k. Then, $\mathcal{B}$ receives a challenged message $m_{\mathrm{ex}} \in_U \mathcal{M}_{\mathrm{ex}}$ picked uniformly at random. At the end, $\mathcal{B}$ wins this game if it outputs a signature σ_{ex} such that $\mathsf{ExVerify}(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}) = 1$.*

Our definition of universal forgery is slightly weaker than usual as in [129], where a successful adversary should be able to forge a valid signature to every challenged message of the message space. In a situation such as plain RSA where messages can be blinded, the two notions are equivalent.

Group Structure. We need $\mathcal{M}_{\mathrm{ex}}$ to form *a group* with an internal operation $\odot$. The inverse of an element $m_{\mathrm{ex}} \in \mathcal{M}_{\mathrm{ex}}$ with respect to this group operation is denoted m_{ex}^{-1}.

Existentially Forgeable Undeniable Signature. We consider an existentially forgeable undeniable signature scheme UnSign whose associated pair of keys is that of the confirmer, i.e., $(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}}) \leftarrow \mathsf{Setup}^{\mathbf{C}}(1^k)$. We denote the message space $\mathcal{M}_{\mathrm{un}}$ and the signature space Σ_{un}. We have two probabilistic polynomial time algorithms

$$\sigma_{\mathrm{un}} \leftarrow \mathsf{UnSign}(m_{\mathrm{un}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}}) \quad \text{and} \quad (m_{\mathrm{un}}, \sigma_{\mathrm{un}}) \leftarrow \mathsf{UnForge}(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}),$$

where the former generates a signature and the latter outputs a valid message-signature pair such that m_{un} is uniformly distributed in $\mathcal{M}_{\mathrm{un}}$. In addition, for a given message m_{un}, the distribution of σ_{un} generated by UnForge is identical as that generated by UnSign for any $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}$. Furthermore, we also have two interactive protocols UnConfirm and UnDeny between $\mathbf{C}$ and $\mathbf{V}$. They correspond to the confirmation and denial protocol of UnSign respectively and are assumed to satisfy classical properties such as completeness, soundness, zero-knowledge, and non-transferability.

Additionally, we will assume that the function $\mathsf{UnSign}(\cdot, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$ *is balanced on the set* Σ_{un} for any secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$. So, the probability for a pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ picked uniformly at random in $\mathcal{M}_{\mathrm{un}} \times \Sigma_{\mathrm{un}}$ to be valid is equal to $\nu := v/|\Sigma_{\mathrm{un}}|$, where v denotes the number of valid signatures related (and independent) to each m_{un}.

Some examples of such undeniable signatures are the RSA based scheme from Gennaro et. al [66], the scheme of Chaum [36] based on the discrete logarithm problem, and the MOVA scheme presented in Chapter 4. All these schemes present this property provided that we remove some hash functions or pseudorandom generators. Furthermore, we note that these obtained signatures schemes are deterministic and therefore cannot satisfy the invisibility property under a chosen-message attack.

Random Hash Function. We consider a hash function $h : \mathcal{M} \rightarrow \mathcal{M}_{\mathrm{ex}}$ which is collision-resistant. We furthermore assume that h is *full-domain* i.e., its range is the full set $\mathcal{M}_{\mathrm{ex}}$. h will be considered as a random oracle.

Random Permutation. We consider a public permutation $C : \mathcal{M}_{\mathrm{ex}} \rightarrow \mathcal{M}_{\mathrm{ex}}$. C will be considered as a random permutation oracle (see [127, 132]) i.e., C is picked uniformly at random among all permutations over $\mathcal{M}_{\mathrm{ex}}$. We assume that we can send queries to the oracle C and the oracle C^{-1}.

Representation Function. We consider a fixed bijection $B : \mathcal{M}_{\mathrm{un}} \times \Sigma_{\mathrm{un}} \rightarrow \mathcal{M}_{\mathrm{ex}}$. In what follows, we will always work with the function $R := C \circ B$ instead of C and B separately. Note that R is then a random bijective function.

8.1.2 The Scheme

The generic construction we develop below is a natural generalization of Chaum's scheme [38]. The signer generates a valid message-signature pair with respect to an existentially forgeable undeniable signature scheme. Next, he mixes this pair with a message digest of the message and finally signs the result in a classical way using ExSign. So, the validity of this designated confirmer signature relies on the validity of the message-signature pair which can only be confirmed by the confirmer. Since ExSign is existentially forgeable, anybody could have produced a signature with an invalid message-signature pair. On the other hand, when the message-signature pair is valid the designated confirmer signature can be produced only by the signer. Putting all together, the help of the confirmer is required in order to deduce the validity or invalidity of a message pair signature with respect to the designated confirmer signature scheme Sign.

Here are the different algorithms of this construction.

Setup Three pairs of keys are generated $(\mathcal{K}_{\mathrm{p}}^{\mathbf{U}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{U}}) \leftarrow \mathsf{Setup}^{\mathbf{U}}(1^k)$ from a security parameter k, where $\mathbf{U} \in \{\mathbf{S}, \mathbf{C}, \mathbf{V}\}$.

Sign Let $m \in \mathcal{M}$ be a given message to sign. The signer runs the algorithm $\mathsf{UnForge}$ to obtain a pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ and computes $h(m)$. He then computes $m_{\mathrm{ex}} := R(m_{\mathrm{un}}, \sigma_{\mathrm{un}}) \odot h(m)$. The designated confirmer signature of m is then $\sigma = (m_{\mathrm{ex}}, \sigma_{\mathrm{ex}})$, where $\sigma_{\mathrm{ex}} \leftarrow \mathsf{ExSign}(m_{\mathrm{ex}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$.

Confirm The verifier and the confirmer first check that $\mathsf{ExVerify}(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}) = 1$. They both compute $m_{\mathrm{ex}} \odot h(m)^{-1}$, apply R^{-1}, and retrieve $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$. Then $\mathbf{V}$ interacts with $\mathbf{C}$ in a proof protocol in which $\mathbf{C}$ proves that $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ is valid using $\mathsf{UnConfirm}$. If this is verified the protocol (verifier) outputs 1.

Deny The verifier and the confirmer first check that $\mathsf{ExVerify}(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}) = 1$ and retrieve $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ as in the confirmation. Then, $\mathbf{V}$ interacts with $\mathbf{C}$ in a proof protocol in which $\mathbf{C}$ proves that $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ is invalid using UnDeny. If this is verified the protocol (verifier) outputs 1.

Remark 8.1.2. Note that the confirmer could also confirm or deny signatures in an anonymous way: he does not need σ_{ex} nor m_{ex} but only m_{un} and σ_{un} which contain no information about the signer or the message. This could be suitable for some applications.

8.2 Security Results

8.2.1 Security Against Existential Forgeries

The following result shows that our construction leads to a scheme resisting existential forgery against an adaptive adversary. This security property is proved according to the model used in Definition 3.3.6.

Theorem 8.2.1. *The scheme* Sign *resists against existential forgery under an adaptive chosen-message attack provided that*

1. *h is a random hash function oracle and C/C^{-1} is a random permutation oracle*
2. ExSign *resists against universal forgery under a no-message attack*
3. *valid $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ pairs are sparse in $\mathcal{M}_{\mathrm{un}} \times \Sigma_{\mathrm{un}}$ (i.e., $\nu \ll 1$)*

even if the adversary is the confirmer $\mathbf{C}$.
More precisely, for any forger $\mathcal{F}$ which wins in the game of existential forgery under an adaptive chosen-message attack against Sign *with success probability*

$$\mathsf{Succ}_{\mathsf{Sign},\mathcal{F}}^{\mathsf{ef-cma}} = \varepsilon$$

using q_h h-queries, q_R R-queries, q_R^* R^{-1}-queries, and q_{S} Sign *queries, we can construct another adversary $\mathcal{B}$ which wins the game of universal forgery under a no-message attack against* ExSign *with success probability*

$$\mathsf{Succ}^{\mathsf{uf-nma}}_{\mathsf{ExSign},\mathcal{B}} \geq \frac{1}{q_R \cdot q_h}\left(\varepsilon - \frac{(q_R + q_R^*)^2}{|\mathcal{M}_{\mathrm{ex}}|} - 2\nu\right)$$

using one run of $\mathcal{F}$.

Proof. In this proof, following Shoup's methodology [138], we provide a sequence of games beginning from the real attack and reach a game allowing to deduce a universal forgery against ExSign. $\mathcal{B}$ is given a challenged public key $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$ generated by $\mathsf{Setup}^{\mathbf{S}}$ and a challenged message $m_{\mathrm{chal}} \in_U \mathcal{M}_{\mathrm{ex}}$ picked uniformly at random for which it has to forge a signature σ_{chal} such that $\mathsf{ExVerify}(m_{\mathrm{chal}}, \sigma_{\mathrm{chal}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}})$ outputs 1 with a non-negligible probability. In what follows, we denote the probability event that $\mathcal{F}$ succeeds in forging a signature in the **Game i** as $\mathsf{S_i}$.

Game 1. Here, we consider the real attack game with the random oracle h and random function oracle R. First, $\mathcal{F}$ receives a challenged public key $\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}$ generated by $\mathsf{Setup}^{\mathbf{S}}$ and for which it will have to existentially forge a message signature pair. Since the adversary $\mathcal{F}$ can be the confirmer, $\mathcal{F}$ also gets the confirmer's key pair $(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$ generated by $\mathsf{Setup}^{\mathbf{C}}$. The adversary makes adaptively and in any order the following queries:

- $\mathcal{F}$ sends q_h messages $m_1, \ldots, m_{q_h} \in \mathcal{M}$ to the random oracle h and receives the corresponding hash values $h_1, \ldots, h_{q_h}$.
- $\mathcal{F}$ sends q_R pairs $(m_{\mathrm{un},1}, \sigma_{\mathrm{un},1}), \ldots, (m_{\mathrm{un},q_R}, \sigma_{\mathrm{un},q_R})$ to the random function oracle R and receives the corresponding values $f_1, \ldots, f_{q_R}$.
- $\mathcal{F}$ sends q_R^* elements $f_1^*, \ldots, f_{q_R^*}^*$ to the random function oracle R^{-1} and receives the corresponding values $(m_{\mathrm{un},1}^*, \sigma_{\mathrm{un},1}^*), \ldots, (m_{\mathrm{un},q_R^*}^*, \sigma_{\mathrm{un},q_R^*}^*)$.
- $\mathcal{F}$ sends q_{S} messages $m_1^{\mathrm{s}}, \ldots, m_{q_{\mathrm{S}}}^{\mathrm{s}}$ to the signing oracle Sign (with respect to the challenged public key) and receives the corresponding signatures $\sigma_1, \ldots, \sigma_{q_{\mathrm{S}}}$. We assume that q_h and q_R includes the queries made by Sign.

After these queries, $\mathcal{F}$ outputs a message m (not queried to the signing oracle) with a correct forged signature σ with success probability $\Pr[\mathsf{S}_1] = \varepsilon$.

Note that the challenged public key $\mathcal{B}$ received in the universal forgery game against ExSign can be given to $\mathcal{F}$ in **Game 1**. Namely, both keys are identically generated so that the simulation is perfect.

Game 2. Here, $\mathcal{B}$ simulates the random oracle h as well as the random function R using two appropriate lists h-List and F-List. It will apply the following rules:

- To a query m_i, $\mathcal{B}$ picks h_i uniformly at random in $\mathcal{M}_{\text{ex}}$ and adds the element (m_i, h_i) in $\mathsf{h\text{-}List}$ if m_i is not already in $\mathsf{h\text{-}List}$. Otherwise, it simply looks in the $\mathsf{h\text{-}List}$ and answers the corresponding h-value.
- To handle the R and R^{-1} oracle queries, it proceeds in a similar way. To a query $(m_{\text{un},i}, \sigma_{\text{un},i})$, it picks f_i uniformly at random in $\mathcal{M}_{\text{ex}}$ and adds the pair $((m_{\text{un},i}, \sigma_{\text{un},i}), f_i)$ in $\mathsf{F\text{-}List}$ if $(m_{\text{un},i}, \sigma_{\text{un},i})$ is not already in $\mathsf{F\text{-}List}$. Otherwise, $\mathcal{B}$ answers the corresponding f_i taken from $\mathsf{F\text{-}List}$. Note that the simulation fails when collisions occur for some distinct f_i since R is a bijective function. It proceeds exactly in the same way for the R^{-1} queries by using the same list $\mathsf{F\text{-}List}$ and picking $(m^*_{\text{un},i}, \sigma^*_{\text{un},i})$ uniformly at random in $\mathcal{M}_{\text{un}} \times \Sigma_{\text{un}}$ when the corresponding query f^*_i is fresh.

Since h is a random oracle and R a random function oracle, we see that the simulation is perfect except when a collision on outputs of R resp. R^{-1} occurs. Let CollF be the event that such a collision occurs in **Game 1** (equivalently in **Game 2**). Obviously, $\Pr[\mathsf{S}_1 \wedge \neg\mathsf{CollF}] = \Pr[\mathsf{S}_2 \wedge \neg\mathsf{CollF}]$, so we can apply Shoup's lemma [138] and obtain

$$|\Pr[\mathsf{S}_2] - \Pr[\mathsf{S}_1]| \leq \Pr[\mathsf{CollF}] \leq \frac{(q_R + q_R^*)^2}{|\mathcal{M}_{\text{ex}}|}.$$

Game 3. This game is identical as **Game 2** except that $\mathcal{B}$ simulates the Sign oracle without $\mathcal{K}_{\text{s}}^{\mathbf{S}}$. Sign must query m_i^{s} to h. This is simulated as explained in **Game 2**. Let h_t be the answer. Sign must also run $\mathsf{UnForge}$ which can be trivially run by $\mathcal{B}$. Let $(m'_{\text{un},i}, \sigma'_{\text{un},i})$ be the forged message-signature pair with respect to the Unsign scheme. It also runs the probabilistic algorithm $\mathsf{ExForge}$ which outputs a valid message-signature pair $(m_{\text{ex},i}, \sigma_{\text{ex},i})$ with respect to ExSign. Sign must also query R with $(m'_{\text{un},i}, \sigma'_{\text{un},i})$ and gets some f_s. $\mathcal{B}$ simulates the value $f_s := R(m'_{\text{un},i}, \sigma'_{\text{un},i})$ by setting

$$f_s := m_{\text{ex},i} \odot (h_t)^{-1}.$$

Note that if $(m'_{\text{un},i}, \sigma'_{\text{un},i})$ or f_s is an element which lies already in $\mathsf{F\text{-}List}$, $\mathcal{B}$ has to abort the simulation. Namely, in the first case it could not choose the output value f_s while in the second case it might fail the simulation if f_s has a preimage which is not a valid message-signature pair in $\mathcal{M}_{\text{un}} \times \Sigma_{\text{un}}$. Since the collisions related to the outputs of R and R^{-1} (even those queried by ExSign) are already cancelled in **Game 2**, such bad events do not happen here. Hence, we notice that the simulation is perfect since $\mathsf{ExForge}$ outputs an $m_{\text{ex},i}$ which is picked uniformly in $\mathcal{M}_{\text{ex}}$ and that $\sigma_{\text{ex},i}$ has the same distribution as a one generated by ExSign (assumption on $\mathsf{ExForge}$). Thus, for any h_t the distribution of f_s is uniform as well and the distribution of the pairs $(m_{\text{ex},i}, \sigma_{\text{ex},i})$ is the same as that produced Sign. We have

$$\Pr[\mathsf{S}_3] = \Pr[\mathsf{S}_2].$$

Game 4. Here, we would like to obtain a game where the output forged message-signature pair $(m, \sigma) = (m, (m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}))$ has the two following properties:

- m was queried to the random oracle h (necessarily not through Sign).
- $f := m_{\mathrm{ex}} \odot h(m)^{-1}$ is an output from a query made to the oracle R (maybe through Sign).

The first condition does not hold with a probability less than $1/|\mathcal{M}_{\mathrm{ex}}|$ since the adversary $\mathcal{F}$ could not do better than guessing the right $h(m)$. The second one does not hold if $\mathcal{F}$ guessed the right f (i.e., with probability up to $1/|\mathcal{M}_{\mathrm{ex}}|$) or if it queried f to R^{-1} oracle and obtained a valid signature pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$, i.e., with probability up to ν since UnSign is balanced. The probability that this condition does not hold is then less than $\max(1/|\mathcal{M}_{\mathrm{ex}}|, \nu)$ which is ν since $1/\nu < |\Sigma_{\mathrm{un}}| < |\mathcal{M}_{\mathrm{ex}}|$. Therefore,

$$|\Pr[\mathsf{S}_4] - \Pr[\mathsf{S}_3]| \leq \frac{1}{|\mathcal{M}_{\mathrm{ex}}|} + \nu \leq 2\nu.$$

Game 5. $\mathcal{B}$ picks $j \in_U \{1, \ldots, q_h\}$, $\ell \in_U \{1, \ldots, q_R\}$ at the beginning and succeeds if m was the jth query to h and $m_{\mathrm{ex}} \odot h(m)^{-1}$ was the output from the ℓth query to R. We denote this event by B. We have

$$\Pr[\mathsf{S}_5 \wedge \mathsf{B}] = \frac{1}{q_h \cdot q_R} \Pr[\mathsf{S}_4].$$

Game 6. Here, $\mathcal{B}$ simulates the output h_j by setting $h_j := f_\ell^{-1} \odot m_{\mathrm{chal}}$. This simulation is perfect because m_{chal} is an element picked uniformly at random and is unused so far. Thus,

$$\Pr[\mathsf{S}_6 \wedge \mathsf{B}] = \Pr[\mathsf{S}_5 \wedge \mathsf{B}].$$

Finally, we notice that $\mathcal{F}$ forged an ExSign signature to the message m_{chal} if it succeeds in the **Game 6** and the event B occurs since $m = m_j$, $f = f_\ell$ and $m_{\mathrm{ex}} = m_{\mathrm{chal}}$ in this case. We then have $\mathsf{Succ}^{\mathsf{uf-nma}}_{\mathsf{ExSign},\mathcal{B}} = \Pr[\mathsf{S}_6 \wedge \mathsf{B}]$. Thus,

$$\mathsf{Succ}^{\mathsf{uf-nma}}_{\mathsf{ExSign},\mathcal{B}} \geq \frac{1}{q_R \cdot q_h} \left(\varepsilon - \frac{(q_R + q_R^*)^2}{|\mathcal{M}_{\mathrm{ex}}|} - 2\nu \right). \qquad \square$$

8.2.2 Invisibility

Lunchtime Attacks

The following result shows that our generalization of the Chaum's scheme is invisible under a lunchtime chosen-message attack. The lunchtime attacks against invisibility are specified in Definition 3.3.3 and Definition 3.3.8.

Theorem 8.2.2. *Assume that h and C are fixed and that $\sigma_{\mathrm{un}} \leftarrow \mathsf{UnSign}(m_{\mathrm{un}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$ is uniformly distributed for any fixed key $\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$ when m_{un} is uniformly distributed. For any invisibility distinguisher $\mathcal{D}$ under a lunchtime chosen-message attack against* Sign *with advantage $\mathsf{Adv}_{\mathsf{Sign},\mathcal{D}}^{\mathsf{inv\text{-}lcma}} = \varepsilon > 0$, there exists an invisibility distinguisher $\mathcal{UD}$ under a lunchtime known-message attack against* UnSign *with advantage*

$$\mathsf{Adv}_{\mathsf{UnSign},\mathcal{UD}}^{\mathsf{inv\text{-}lkma}} = \varepsilon' \geq \varepsilon/2$$

which uses one run of $\mathcal{D}$.

Proof. First $\mathcal{UD}$ is fed with $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}$ issued from $(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}}) \leftarrow \mathsf{Setup}^{\mathbf{C}}(1^k)$. Then, $\mathcal{UD}$ runs $(\mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}) \leftarrow \mathsf{Setup}^{\mathbf{S}}(1^k)$ and transmits $\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$ to $\mathcal{D}$. The answers of the oracle queries from $\mathcal{D}$ will be simulated by $\mathcal{UD}$. Since $\mathcal{D}$ has the signer's secret key $\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}$, it does not need any access to a signing oracle. $\mathcal{UD}$ simulates the oracle queries to the confirmation and denial protocols as follows:

- To a message-signature pair $(m, (m_{\mathrm{ex}}, \sigma_{\mathrm{ex}}))$, $\mathcal{UD}$ checks first that $(m_{\mathrm{ex}}, \sigma_{\mathrm{ex}})$ is a valid pair with respect to ExSign. It retrieves the corresponding $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ and forwards this query to the confirmation (or denial) protocol oracle with respect to UnSign.

At a time, $\mathcal{D}$ sends two messages $m_0, m_1 \in \mathcal{M}$ to $\mathcal{UD}$. $\mathcal{UD}$ receives two random messages $m_{\mathrm{un}}^0, m_{\mathrm{un}}^1 \in \mathcal{M}_{\mathrm{un}}$ and a signature $\sigma_{\mathrm{un}} \in \Sigma_{\mathrm{un}}$ (generated in the following way $\sigma_{\mathrm{un}} \leftarrow \mathsf{UnSign}(m_{\mathrm{un}}^b, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$). Then, $\mathcal{UD}$ picks two random bits $b_1, b_2 \in_U \{0,1\}$, sets

$$m_{\mathrm{ex}} = R(m_{\mathrm{un}}^{b_2}, \sigma_{\mathrm{un}}) \odot h(m_{b_1}),$$

computes $\sigma_{\mathrm{ex}} = \mathsf{ExSign}(m_{\mathrm{ex}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{S}})$ and sends $\sigma = (m_{\mathrm{ex}}, \sigma_{\mathrm{ex}})$ to $\mathcal{D}$. Then, $\mathcal{D}$ answers a bit b'' to $\mathcal{UD}$. Finally, $\mathcal{UD}$ answers a bit $b' = b_1 \oplus b_2 \oplus b''$ (If $\mathcal{D}$ aborts, we pick a random b''.) to its challenger. It remains to compute the advantage $\mathcal{UD}$. To this end, we compute $\Pr[b' = b] = \Pr[b' = b \wedge b_2 = b] + \Pr[b' = b \wedge b_2 \neq b]$. We also have

$$\Pr[b' = b \wedge b_2 \neq b] = \Pr[b'' = b \oplus b_2 \oplus b_1 \wedge b_2 \neq b] = \Pr[b'' = \neg b_1 | b_2 \neq b] \cdot \frac{1}{2}.$$

When $b_2 \neq b$ then $(m_{\mathrm{un}}^{b_2}, \sigma_{\mathrm{un}})$ is uniformly distributed and independent from b_1, hence b'' is independent from b_1. Thus, $\Pr[b' = b \wedge b_2 \neq b] = 1/4$. We also have

$$\Pr[b' = b \wedge b_2 = b] = \Pr[b' = b | b_2 = b] \cdot \Pr[b_2 = b] = \Pr[b'' = b_1 | b_2 = b] \cdot \frac{1}{2}.$$

Without loss of generality, we can assume that $\Pr[b'' = b_1 | b_2 = b] \geq 1/2$ so that $\Pr[b'' = b_1 | b_2 = b] = \varepsilon/2 + 1/2$. Therefore, we obtain

$$\Pr[b' = b] = \frac{1}{2} + \frac{\varepsilon}{4}.$$

Since this probability does not depend on the bit value b, we finally have

$$\varepsilon' = \Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1] = \frac{1}{2} + \frac{\varepsilon}{4} - \left(1 - \left(\frac{1}{2} + \frac{\varepsilon}{4}\right)\right) = \frac{\varepsilon}{2},$$

which concludes the proof. □

Adaptive Attacks

The scheme Sign does not satisfy the stronger adaptive invisibility notion defined in Camenisch and Michels [30], i.e., when the distinguisher can continue to query the oracles after he is given the challenge. Namely, after having received the challenged signature σ, $\mathcal{D}$ could deduce the two pairs $(m_{\mathrm{un}}^0, \sigma_{\mathrm{un}}^0)$, $(m_{\mathrm{un}}^1, \sigma_{\mathrm{un}}^1)$ which would correspond to m_0 and m_1. Then, $\mathcal{D}$ generates a signature σ' on another message m' by using $(m_{\mathrm{un}}^0, \sigma_{\mathrm{un}}^0)$ and queries the pair (m', σ') to the confirmation and denial oracle. Depending on the answer, $\mathcal{D}$ deduces whether $(m_{\mathrm{un}}^0, \sigma_{\mathrm{un}}^0)$ is valid or not. From this, we see that $\mathcal{D}$ wins the invisibility game under an adaptive attack.

The fundamental problem relies on the fact that the adversary can always retrieve the corresponding pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ (as any verifier) from a message-signature pair with respect to Sign. He can then sign a new message m' by reusing the pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ and query the obtained pair to the Confirm or Deny oracle. Assuming that the verifier has to retrieve $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$, the only way to thwart such an attack is to make sure that the adversary cannot generate a new signature with another message m' with the same pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$. This seems to imply that $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ has to depend on m. Moreover, the verifier should not be able to verify how $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ was generated since it would trivially break the invisibility. This leads us to believe that the signer has to encrypt an element with the confirmer's secret key such as in the scheme proposed by Camenisch and Michels [30]. Obviously, the above discussion motivates the fact that we should strongly modify the generalized Chaum's scheme in order to obtain invisibility against an adaptive adversary.

8.2.3 Other Security Properties

The other security properties of our scheme are easier to prove, namely the completeness of the confirmation resp. denial protocol is straightforward. The other properties such as the soundness and non-transferability are inherited from the undeniable signature scheme. The non-coercibility is obtained if the signer deleted intermediate computations from UnForge. In this case, the invisibility of the undeniable signature scheme applies, since this one holds even if the distinguisher is given $\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$. Note that receipt-freeness is not guaranteed.

8.3 A Practical Example

Here, we propose a practical realization of the presented construction quite similar to that of Chaum [38]. First, we consider the Chaum's undeniable signature scheme [36] for UnSign. Let p be a prime integer of 1024 bits and g be a public generator of $\mathbb{Z}_p^*$. Then, $(\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{C}}) = (c, g^c \bmod p) := (c, h)$ for a $c \in_U \mathbb{Z}_{p-1}^*$ picked uniformly at random. We recall that Chaum's undeniable signature of a message $m_{\mathrm{un}} \in \mathbb{Z}_p^*$ is $m_{\mathrm{un}}^c \bmod p$. Hence, UnForge can be implemented by picking a random element $r \in_U \mathbb{Z}_{p-1}$ uniformly at random and outputting the pair

$$(m_{\mathrm{un}}, \sigma_{\mathrm{un}}) := (g^r \bmod p, h^r \bmod p).$$

The random function R applied on $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ can be implemented by computing an AES with a fixed key in a kind of CBC mode on $m_{\mathrm{un}}||\sigma_{\mathrm{un}}$ by

$$B(m_{\mathrm{un}}||\sigma_{\mathrm{un}}) = (x_0||\ldots||x_{15})$$

where $x_i \in \{0,1\}^{128}$ for $i = 0, \ldots, 15$ and $C(x_0||\ldots||x_{15}) = (x_{16}||\ldots||x_{31})$ with

$$x_i = \mathrm{AES}(x_{i-16}) \oplus x_{i-1},$$

for $i = 16, \ldots, 31$. Note that we must choose p close enough to 2^{1024}. The hash function h can be instantiated with SHA-1 by

$$h(m) = \mathrm{trunc}_{2048}(\text{SHA-1}(1||m)||\ldots||\text{SHA-1}(13||m)),$$

where trunc_{2048} outputs the 2048 most significant bits of the input. The group operation $\odot$ can be replaced by the XOR operation $\oplus$ on the set $\{0,1\}^{2048}$. We finally take the plain DSA scheme for ExSign. Let q_1 be a prime integer close to 2^{2048}, a large prime number $q_2 = aq_1 + 1$ and a generator of $\mathbb{Z}_{q_2}^*$ whose a-th power is denoted as g_q. Then, $(\mathcal{K}_{\mathrm{s}}^{\mathbf{S}}, \mathcal{K}_{\mathrm{p}}^{\mathbf{S}}) = (x, g_q^x \bmod q_2)$ for $x \in_U \mathbb{Z}_{q_1}^*$ uniformly at random. Then, $\sigma_{\mathrm{ex}} = (r, s)$, where

$$r = (g_q^k \bmod q_2) \bmod q_1 \quad \text{and} \quad s = \frac{m_{\mathrm{ex}} + xr}{k} \bmod q_1$$

for a random $k \in_U \mathbb{Z}_{q_1}^*$.

8.4 On Feasibility Results Based on Cryptographic Primitives

8.4.1 Discussion

This subsection provides a discussion on the relevance of the primitives used in the generalized Chaum's designated confirmer signature scheme. Namely, we would like to explain why this construction is possible although a previous result due to Okamoto [118] seems at the first glance to provide strong evidence of its impossibility.

The study of relations between the cryptographic primitives always played a central role in cryptography. In particular, it allows to clarify the kind of primitives required to achieve the security of a given construction. Examples of well-known basic primitives are *one-way functions*, *trapdoor one-way functions*, or *trapdoor predicates* which were introduced by Goldwasser and Micali [73]. Here, we will focus on two classes of equivalent primitives, that of one-way functions and that of trapdoor predicates. These two classes contain respectively two major cryptographic primitives, namely the digital signatures resp. the public-key encryption. Rompel [133] proved that one-way functions are equivalent to signatures and Goldwasser and Micali [73] showed the equivalence between trapdoor predicates and public-key encryption. Since then, several cryptographic primitives have been shown to belong to one of these classes, e.g., undeniable signatures exist if and only if digital signatures exist [23].

Soon after their invention, designated confirmer signatures were proved to belong in the public-key encryption class [118]. This showed that despite of their similarities to undeniable signatures these two primitives are not equivalent. Separation between these two classes was first proved by Impagliazzo and Rudich [78] in the black-box case, i.e., when the primitives are considered as black-box. More precisely, they showed that a non-separation in the black-box case would imply to get a proof that $\mathcal{P} \neq \mathcal{NP}$. Since such a proof of this statement is unlikely to appear, the above result is a strong argument for black-box separation. This result was improved by Reingold et al. [130] who proved the above separation unconditionally, i.e., without using an implication towards a proof of $\mathcal{P} \neq \mathcal{NP}$ in the case of a non-separation. Hence, this shows that the construction of a designated confirmer signature requires a primitive equivalent to the public-key encryption. Note that the black-box assumption is not a strong restriction in our context since almost any reductions considered in cryptography are black-box.

Our proposed construction seems only to be based on primitives belonging to the digital signatures class. Actually, this comes from an insufficient precise way to characterize cryptographic primitives. For instance, when we talk about a digital signature scheme, we mean a signature which is resistant to existential forgery under an adaptive chosen-message attack. Similarly an undeniable signature is meant to be implicitly secure in terms of existential forgery attacks and signatures invisibility. In this generalized Chaum's scheme, we have considered a special kind of undeniable signature which is existentially forgeable but remains invisible under a lunchtime known-message attack. In the next subsection, we prove that the existence of such a primitive indeed implies the existence of a public-key encryption semantically secure under a chosen-plaintext attack (IND-CPA). So we prove that undeniable signatures may belong to two different classes depending on the security properties we require. Paradoxically, although this kind of undeniable signature satisfies weaker security properties than usual, it belongs to a stronger class namely that of public-key encryption. Intuitively, this can be explained by the fact that it seems more difficult for an existentially forgeable undeniable signature to remain invisible than for an undeniable signature which is resistant to existential forgery attacks.

8.4.2 UnSign and Public-Key Encryption

We explain here how we can construct an IND-CPA public-key cryptosystem from the existentially forgeable undeniable signature scheme UnSign. We recall that UnSign is assumed to satisfy invisibility under a lunchtime known-message attack (this was required to prove that Sign is invisible under a lunchtime chosen-message attack). For the sake of simplicity, this cryptosystem will encrypt only one bit at a time. We denote the encryption scheme PKE. It is composed of three polynomial time algorithms which are the key generator KGen, the encryption algorithm Enc, and the decryption algorithm Dec. The scheme is inspired from Okamoto [118] and a paper of Abdalla and Warinschi [1].

KGen The key generator KGen generates a pair of key (pk, sk) by calling the key generator of UnSign. It computes $(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}}) \leftarrow \mathsf{Setup}^{\mathbf{C}}(1^k)$ from the security parameter k and sets $(pk, sk) := (\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$.

Enc Let $b \in \{0,1\}$ a bit to encrypt. If $b = 0$, we call the probabilistic algorithm UnForge to generate a valid pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}}) \leftarrow \mathsf{UnForge}(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}})$. The pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ is set to be the ciphertext of b. If $b = 1$, we pick a pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}}) \in_U \mathcal{M}_{\mathrm{un}} \times \Sigma_{\mathrm{un}}$ uniformly at random. The pair $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ is the ciphertext of b in this case.

Dec Let $(m_{\mathrm{un}}, \sigma_{\mathrm{un}})$ be a ciphertext. Using the secret key $sk = \mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$, it suffices to simulate UnConfirm or UnDeny to determine whether this pair is valid or not. If the pair is valid the decrypted ciphertext is 0, else it is 1.

We prove here that PKE is IND-CPA secure provided that UnSign is invisible under a lunchtime known-message attack.

Theorem 8.4.1. *Assume that $\sigma_{\mathrm{un}} \leftarrow \mathsf{UnSign}(m_{\mathrm{un}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$ is uniformly distributed for any fixed key $\mathcal{K}_{\mathrm{s}}^{\mathbf{C}}$ when m_{un} is uniformly distributed. For any adversary $\mathcal{A}$ which wins in an* IND-CPA *game against* PKE *with advantage $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{PKE},\mathcal{A}} = \varepsilon > 0$, there exists an invisibility distinguisher $\mathcal{D}$ under a lunchtime known message attack against* UnSign *with advantage $\mathsf{Adv}^{\mathsf{inv\text{-}lkma}}_{\mathsf{UnSign},\mathcal{D}} = \varepsilon' = \varepsilon$.*

Proof. At the beginning of the invisibility game, $\mathcal{D}$ receives a challenged pair of key $(\mathcal{K}_{\mathrm{p}}^{\mathbf{C}}, \mathcal{K}_{\mathrm{s}}^{\mathbf{C}})$ and playing the role of the challenger in the IND-CPA game forwards the same key pair to $\mathcal{A}$. After a given time, $\mathcal{A}$ will trivially send the two bits

$0, 1$ to $\mathcal{D}$. After a lunchtime, $\mathcal{D}$ receives two challenged messages m_{un}^0, m_{un}^1 with a signature σ_{un}. $\mathcal{D}$ sends the challenged pair $(m_{\text{un}}^0, \sigma_{\text{un}})$ to $\mathcal{A}$. Note that this challenge is perfectly simulated. Then, $\mathcal{A}$ answers a bit b. This bit b is also the answer of $\mathcal{D}$ to its challenger. Thus, the advantage ε' of $\mathcal{D}$ satisfies $\varepsilon' = \varepsilon$. $\square$

Chapter 9

Conclusion and Future Work

In this thesis, we have mainly investigated how to design short undeniable signature schemes and developed a scheme having a fully scalable signature size depending on the security level. To achieve this, we introduced a very general framework based on group homomorphisms as well as some computational problems related to the interpolation of a set of points by a group homomorphism. Additionally, this setting contributes to present a unified view of different well-known computational problems.

Using the above techniques, we have proposed a generic scheme called MOVA which generalizes the Chaum's undeniable signature scheme. By considering group homomorphisms with a small range group and scaling the domain group with respect to the adversary's complexity, we can naturally achieve very short signatures. As far as we know, this is the first signature scheme for which signatures of less than 80 bits can be considered. From our point of view, this represents the most important contribution of this thesis.

As further results, we mention our 2-move confirmation and denial protocols which reach the minimal number of moves for interactive verification protocols. Possible concrete instantiations of MOVA scheme have been studied with a strong focus on the group characters of order 2, 3, and 4. We also analyzed algorithmic aspects related to the computation of characters as well as other group homomorphisms showing that Legendre symbol offers the most efficient signature generation. According to the specific properties of the MOVA scheme and in particular the signature size, we examined some potential applications for which MOVA offers strong advantage over other cryptographic schemes. In a more thorough way, we studied an SMS lottery application where receipts confirming tickets registration are based on a MOVA signature. In a closely related field, we revisited the Chaum's designated confirmer signature and provided a formal security proof on a generalized version.

Besides the above contributions, this work gives rise to some further possible investigations and some new open questions. We believe that the interpolation of group homomorphisms should deserve additional attention in a wider part of public-key cryptography. This tool may be very useful in order to express different cryptographic results under a unified framework. Moreover, this new view might contribute to the design of additional primitives of the public-key cryptography such as classical (short) signature schemes or key-establishment protocols.

Apart from a possible application using two levels of secret, instantiations of MOVA with characters greater than 2 do not provide clear advantages over the Legendre symbols. We believe that the question remains partly open and a challenging task would be to find additional motivations for using such characters.

Appendix A

Algebra

In this appendix, we recall some results of algebra which play an important role in this thesis. An exhaustive treatment of the different subjects presented below would be beyond the scope of this work. So, we restrict ourselves to the principal results.

A.1 The Structure of Finite Abelian Groups

One of the main achievements of group theory is the description of the structure of finite Abelian groups. We give here the main statements related to this. Further details about this development can be obtained in the book of Lang [91].

Definition A.1.1. *Let p be a prime. A finite group is said to be a p-group if its order is a power of p.*

If G is a finite Abelian group and p is a prime, we denote by $G(p)$ the subgroup of all elements $g \in G$ whose order is a power of p. We note that $G(p)$ is a p-group.

Theorem A.1.2. *Let G be a finite Abelian group and $p_1 < p_2 < \cdots < p_n$ be the sequences of all primes dividing the order of G. Then,*

$$G \simeq G(p_1) \oplus G(p_2) \oplus \cdots \oplus G(p_n).$$

Theorem A.1.3. *Let H be a p-group. There exists a unique sequence of integers $0 < e_1 \leq e_2 \leq \cdots \leq e_k$ such that*

$$H \simeq \mathbb{Z}_{p^{e_1}} \oplus \mathbb{Z}_{p^{e_2}} \oplus \cdots \oplus \mathbb{Z}_{p^{e_k}}.$$

Combining Theorem A.1.2 with Theorem A.1.3 leads to the structure of any finite Abelian group.

A.2 Integral Domains

The role of this section is to give some basic definitions and results related to integral domains. For more details about the presented material, we refer the reader to Chapter 1 of the book of Ireland and Rosen [79] (in particular pp. 8–12) and to Chapter 2 of the book of Lang [91].

We recall first the definition of an integral domain as well as notions related to the factorization.

Definition A.2.1. *Let R denote a commutative ring with a unit element. We say that R is an integral domain if there are non zero divisors in the ring, i.e.,*

$$xy = 0 \Rightarrow x = 0 \quad \text{or} \quad y = 0$$

for all $x, y \in R$.

Definition A.2.2. *Let a, b, u, p be elements of an integral domain R.*

1. *If $b \neq 0$, we say that b divides a if $a = bc$ for some $c \in R$. We denote this fact by $b|a$.*

2. *u is called a unit if u divides 1, i.e., if u is invertible.*

3. *a and b are said to be associates if $a = bu$ for some unit u.*

4. *p is called irreducible if for any a, $a|p$ implies that a is either a unit or an associate of p.*

5. *A nonunit p is called a prime if $p \neq 0$ and for any a, b, $p|ab$ implies that $p|a$ or $p|b$.*

We define below some integral domains with special properties and provide important relations between them.

Definition A.2.3. *Let R be an integral domain.*

1. *R is called a principal ideal domain (PID) if every ideal of R is principal, i.e., can be generated by a single element.*

2. *R is said to be an Euclidean domain if there is a function $N : R\backslash\{0\} \longrightarrow \mathbb{N}$ such that for all $a, b \in R$, $b \neq 0$, there exist $c, d \in R$ with the property $a = cb+d$ and either $d = 0$ or $N(d) < N(b)$.*

3. *R is said to be a unique factorization domain (UFD) if there exists a set S of primes in R such that the following statements hold:*

(a) Every prime in R is associate to exactly one prime in S.

(b) Each element r of R can be written as a unique product of the form

$$r = u \prod_{p \in S} p^{a_p},$$

where u is a unit and the a_p's are some nonnegative integers.

Proposition A.2.4. *The following assertions hold.*

1. *An Euclidean domain is a principal ideal domain.*
2. *A principal ideal domain is a unique factorization domain.*
3. *In a unique factorization domain an element is prime if and only if it is irreducible.*

We deduce the fundamental result stating that any element of an Euclidean domain admits a unique factorization.

A.3 Lattices

This section provides some basic results related to lattices. More precisely, we recall the definition of a lattice, the determinant of a lattice, Minkowski's convex body theorem which is fundamental in the theory of lattices and give an upper bound on the discriminant of a lattice produced by the solutions of a linear congruence. The following results were taken from the textbook of Cassels [34] and the PhD thesis of Nguyễn [115]. The former contains a thorough presentation on the lattice theory and the latter provides nice algorithmic applications of lattices to the cryptography.

Definition A.3.1. *A subset L of the space $\mathbb{R}^p$ is called a* lattice *if there exist some linearly independent vectors $v_1, \dots, v_n \in \mathbb{R}^p$ over $\mathbb{R}$ such that*

$$L = \{x = \sum_{i=1}^{n} a_i v_i \mid a_1, \dots, a_n \in \mathbb{Z}\}.$$

The vectors $v_1, \dots, v_n$ are called the basis *of the lattice L and n is the* dimension *of L.*

An equivalent definition of a lattice consists in saying that L is a discrete subgroup of $(\mathbb{R}^p, +)$. We note also that a basis of a lattice is not unique. From now on, we only consider lattices of dimension p, i.e., of full dimension.

An important notion related to a lattice is the determinant.

Definition A.3.2. *Let $L \subset \mathbb{R}^p$ be a p-dimensional (full-dimensional) lattice with basis vectors $v_1, \dots, v_p \in \mathbb{R}^p$. The* determinant *of L is defined by*

$$d(L) = |\det(v_1, \dots, v_p)|,$$

where $(v_1, \dots, v_p)$ denotes the $p \times p$ matrix whose ith column is the vector v_i.

We now state a very remarkable theorem due to Minkowski.

Theorem A.3.3. *(Minkowski's convex body theorem) Let $L \subset \mathbb{R}^p$ be a lattice of dimension p with determinant $d(L)$. Let S be a measurable subset (with respect to the Lebesgue measure μ) of $\mathbb{R}^p$ which is symmetric about 0, convex and such that*

$$\mu(S) > 2^p d(L).$$

Then, S contains at least one point of L different of 0.

Finally, we present a technical result about solutions of a linear congruence and the determinant of the lattice produced by them.

Lemma A.3.4. *Let p, k be positive integers and $a_i \in \mathbb{Z}$ for $i = 1, \dots, p$. The set L composed of the solutions in $\mathbb{Z}^p$ of the congruence*

$$\sum_{j=1}^{p} a_j u_j \equiv 0 \pmod{k}$$

is a lattice of determinant satisfying

$$d(L) \leq k.$$

Bibliography

[1] Michel Abdalla and Bogdan Warinschi. On the Minimal Assumptions of Group Signature Schemes. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *Information and Communications Security, ICICS '04*, volume 3269 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2004.

[2] *Advanced Encryption Standard*, Federal Information Processing Standards Publication 197. U.S. Department of Commerce, National Institute of Standards and Technology, 2001.

[3] Ross J. Anderson, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. The Newton Channel. In Ross J. Anderson, editor, *Information Hiding: 1st International Workshop*, volume 1174 of *Lecture Notes in Computer Science*, pages 151–156. Springer-Verlag, 1996.

[4] ANSI X9.30. *Public Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA)*. American National Standard Institute. American Bankers Association, 1997.

[5] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and Hardness of Approximation Problems. In *33rd Annual IEEE Symposium on Foundations of Computer Science, FOCS '92*, pages 14–23. IEEE Computer Society, 1992.

[6] Gildas Avoine, Jean Monnerat, and Thomas Peyrin. Advances in Alternative Non-adjacent Form Representations. In Anne Canteaut and Kapaleeswaran Viswanathan, editors, *Progress in Cryptology – INDOCRYPT '04*, volume 3348 of *Lecture Notes in Computer Science*, pages 260–274. Springer-Verlag, 2004.

[7] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In *23rd Annual ACM Symposium on Theory of Computing, STOC '91*, pages 21–31. ACM Press, 1991.

[8] Thomas Baignères, Pascal Junod, Yi Lu, Jean Monnerat, and Serge Vaudenay. *A Classical Introduction to Cryptography – Exercise Book*. Springer-Verlag, 2006.

[9] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd Annual IEEE Symposium on Foundations of Computer Science, FOCS '01*, pages 106–115. IEEE Computer Society, 2001.

[10] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. In *44th Annual IEEE Symposium on Foundations of Computer Science, FOCS '03*, pages 384–393. IEEE Computer Society, 2003.

[11] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. Cryptology ePrint Archive, Report 2004/226, 2004. `http://eprint.iacr.org/`. Full version of [10].

[12] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer-Verlag, 2004.

[13] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[14] Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: A Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128. Springer-Verlag, 2006.

[15] Ingrid Biehl, Sacher Paulus, and Tsuyoshi Takagi. Efficient Undeniable Signature Schemes Based on Ideal Arithmetic in Quadratic Orders. *Design, Codes and Cryptography*, 31(2):99–123, 2004.

[16] Lenore Blum, Manuel Blum, and Michael Shub. Comparison of Two Pseudo-Random Number Generators. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO '82*, pages 61–78. Plenum Press, 1983.

[17] Lenore Blum, Manuel Blum, and Michael Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15(2):364–383, 1986. Full version of [16].

[18] Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications. In *20th Annual ACM Symposium on Theory of Computing, STOC '88*, pages 103–112. ACM Press, 1988.

[19] Dan Boneh. The Decision Diffie-Hellman Problem. In Joe P. Buhler, editor, *Algorithmic Number Theory, ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 1998.

[20] Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[21] Dan Boneh and Matthew Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. Full version of [20].

[22] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[23] Joan Boyar, David Chaum, Ivan Damgård, and Torben P. Pedersen. Convertible Undeniable Signatures. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1991.

[24] Joan Boyar, Stuart A. Kurtz, and Mark W. Krentel. A Discrete Logarithm Implementation of Perfect Zero-Knowledge Blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[25] Colin Boyd and Ernest Foo. Off-Line Fair Payment Protocols Using Convertible Signatures. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 271–285. Springer-Verlag, 1998.

[26] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[27] Richard P. Brent. Factorization of the Tenth Fermat Number. *Mathematics of Computation*, 68(225):429–451, 1999.

[28] Emmanuel Bresson, Dario Catalano, and David Pointcheval. A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In Chi Sung Laih, editor, *Advances in Cryptology – ASIACRYPT '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 37–54. Springer-Verlag, 2003.

[29] Lynne M. Butler. A Unimodality Result in the Enumeration of Subgroups of a Finite Abelian Group. *Proceedings of the American Mathematical Society*, 101(4):771–775, 1987.

[30] Jan Camenisch and Markus Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 243–258. Springer-Verlag, 2000.

[31] Jan Camenisch and Victor Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer-Verlag, 2003.

[32] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. In *30th Annual ACM Symposium on Theory of Computing, STOC '98*, pages 209–218. ACM Press, 1998.

[33] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. *Journal of the ACM*, 51(4):557–594, 2004. Full version of [32].

[34] John W. S. Cassels. *An Introduction to the Geometry of Numbers.* Classics in Mathematics. Springer-Verlag, 1997.

[35] Stefania Cavallar, Bruce Dodson, Arjen K. Lenstra, Walter Lioen, Peter L. Montgomery, Brian Murphy, Herman te Riele, Karen Aardal, Jeff Gilchrist, Gérard Guillerm, Paul Leyland, Joël Marchand, François Morain, Alec Muffett, Chris Putnam, Craig Putnam, and Paul Zimmermann. Factorization of a 512-Bit RSA Modulus. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2000.

[36] David Chaum. Zero-Knowledge Undeniable Signatures. In Ivan Damgård, editor, *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer-Verlag, 1990.

[37] David Chaum. Some Weaknesses of “Weaknesses of Undeniable Signature Schemes”. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT ’91*, volume 547 of *Lecture Notes in Computer Science*, pages 554–556. Springer-Verlag, 1991.

[38] David Chaum. Designated Confirmer Signatures. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT ’94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer-Verlag, 1995.

[39] David Chaum and Torben P. Pedersen. Wallet Databases with Observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO ’92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.

[40] David Chaum and Hans van Antwerpen. Undeniable Signatures. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO ’89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–217. Springer-Verlag, 1990.

[41] David Chaum, Eugène van Heijst, and Birgit Pfitzman. Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO ’91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1992.

[42] Henri Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 2000.

[43] Jean-Sébastien Coron. On the Exact Security of Full Domain Hash. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO ’00*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, 2000.

[44] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to Achieve a McEliece-Based Digital Signature Scheme. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT ’01*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer-Verlag, 2001.

[45] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer-Verlag, 2002.

[46] Ivan Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT ’00*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430. Springer-Verlag, 2000.

[47] Ivan Damgård and Gudmund S. Frandsen. Efficient Algorithms for GCD and Cubic Residuosity in the Ring of Eisenstein Integers. In Andrzej Lingas and

Bengt J. Nilsson, editors, *Fundamentals of Computation Theory, FCT '03*, volume 2751 of *Lecture Notes in Computer Science*, pages 109–117. Springer-Verlag, 2003.

[48] Ivan Damgård and Gudmund S. Frandsen. Efficient Algorithms for the gcd and Cubic Residuosity in the Ring of Eisenstein Integer. *Journal of Symbolic Computation*, 39(6):643–652, 2005. Full version of [47].

[49] Ivan Damgård and Torben P. Pedersen. New Convertible Undeniable Signature Schemes. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 372–386. Springer-Verlag, 1996.

[50] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and Sufficient Assumptions for Non-interactive Zero-Knowledge Proofs of Knowledge for All NP Relations. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *Automata, Languages and Programming: 27th International Colloquium, ICALP '00*, volume 1853 of *Lecture Notes in Computer Science*, pages 451–462. Springer-Verlag, 2000.

[51] Yvo Desmedt and Moti Yung. Weaknesses of Undeniable Signature Schemes. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 205–220. Springer-Verlag, 1991.

[52] Giovanni Di Crescenzo. Equivocable and Extractable Commitment Schemes. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communications Network, SCN '02*, volume 2576 of *Lecture Notes in Computer Science*, pages 74–87. Springer-Verlag, 2003.

[53] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[54] *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-2. U.S. Department of Commerce, National Institute of Standards and Technology, 2000.

[55] Cynthia Dwork and Amit Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 442–458. Springer-Verlag, 1998.

[56] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In George R. Blakley and David Chaum, editors,

Advances in Cryptology – CRYPTO '84, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[57] William Feller. *An Introduction to Probability Theory and Its Applications, Volume I, 3rd edition.* Wiley Series in Probability and Mathematical Statistics. John Wiley Sons, 1968.

[58] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[59] Matthieu Finiasz. *Nouvelles constructions utilisant des codes correcteurs d'erreurs en cryptographie à clef publique.* PhD thesis, École Polytechnique, France, 2004.

[60] Marc Fischlin. *Trapdoor Commitment Schemes and Their Applications.* PhD thesis, Johann Wolfgang Goethe-Universität, Frankfurt am Main, Germany, 2001.

[61] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. Interactive Bi-Proof Sytems and Undeniable Signature Schemes. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 243–256. Springer-Verlag, 1991.

[62] Steven D. Galbraith and Wenbo Mao. Invisibility and Anonymity of Undeniable and Confirmer Signatures. In Marc Joye, editor, *Topics in Cryptology – CT-RSA '03*, volume 2612 of *Lecture Notes in Computer Science*, pages 80–97. Springer-Verlag, 2003.

[63] Steven D. Galbraith, Wenbo Mao, and Kenneth G. Paterson. RSA-Based Undeniable Signatures for General Moduli. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA '02*, volume 2271 of *Lecture Notes in Computer Science*, pages 200–217. Springer-Verlag, 2002.

[64] Sachin Garg. How to Optimize C/C++ Source – Performance Programming, 2002. `http://bdn.borland.com/article/0,1410,28278,00.html`.

[65] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. RSA-Based Undeniable Signatures. In Burton S. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 132–149. Springer-Verlag, 1997.

[66] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. RSA-Based Undeniable Signatures. *Journal of Cryptology*, 13(4):397–416, 2000. Full version of [65].

[67] Craig Gentry, David Molnar, and Zulfikar Ramzan. Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs. In Bimal Roy, editor, *Advances in Cryptology – ASIACRYPT '05*, volume 3788 of *Lecture Notes in Computer Science*, pages 662–681. Springer-Verlag, 2005.

[68] The GNU Multiple Precision Arithmetic Library. `http://www.swox.com/gmp/`.

[69] Oded Goldreich. *Foundations of Cryptography, Volume I Basic Tools.* Cambridge University Press, 2001.

[70] Oded Goldreich and Hugo Krawczyk. On the Composition of Zero-Knowledge Proof Systems. In Michael S. Paterson, editor, *Automata, Languages and Programming: 17th International Colloquium, ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 268–282. Springer-Verlag, 1990.

[71] Oded Goldreich and Hugo Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1):169–192, 1996. Full version of [70].

[72] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information. In *14th Annual ACM Symposium on Theory of Computing, STOC '82*, pages 365–377. ACM Press, 1982.

[73] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[74] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *17th Annual ACM Symposium on Theory of Computing, STOC '85*, pages 291–304. ACM Press, 1985.

[75] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Full version of [74].

[76] 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, 1988.

[77] Shafi Goldwasser and Erez Waisbard. Transformation of Digital Signature Schemes into Designated Confirmer Signatures Schemes. In Moni Naor, editor, *Theory of Cryptography, TCC '04*, volume 2951 of *Lecture Notes in Computer Science*, pages 77–100. Springer-Verlag, 2004.

[78] Russell Impagliazzo and Steven Rudich. Limits on the Provable Consequences of One-way Permutations. In *21st Annual ACM Symposium on Theory of Computing, STOC '89*, pages 44–61. ACM Press, 1989.

[79] Kenneth Ireland and Michael Rosen. *A Classical Introduction to Modern Number Theory: 2nd edition*, volume 84 of *Graduate Texts in Mathematics*. Springer-Verlag, 1990.

[80] Markus Jakobsson. Blackmailing using Undeniable Signatures. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 425–427. Springer-Verlag, 1995.

[81] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 1996.

[82] Pascal Junod. On the Optimality of Linear, Differential, and Sequential Distinguishers. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2003.

[83] Donald E. Knuth and Luis Trabb Pardo. Analysis of a Simple Factorization Algorithm. *Theoretical Computer Science*, 3(3):321–348, 1976.

[84] Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". Cryptology ePrint Archive, Report 2004/152, 2004. `http://eprint.iacr.org/`.

[85] Hugo Krawczyk and Tal Rabin. Chameleon Signatures. In *Network and Distributed System Security Symposium, NDSS '00*, pages 143–154. The Internet Society, 2000.

[86] Kaoru Kurosawa and Swee-Huay Heng. 3-Move Undeniable Signature Scheme. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 181–197. Springer-Verlag, 2005.

[87] Fabien Laguillaumie. *Signatures à Vérification Contrôlée Basées sur des Applications Bilinéaires : Conception et Analyse de Sécurité*. PhD thesis, Université de Caen Basse-Normandie, France, 2005.

[88] Fabien Laguillaumie and Damien Vergnaud. Short Undeniable Signatures Without Random Oracles: the Missing Link. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *Progress in Cryptology – INDOCRYPT '05*, volume 3797 of *Lecture Notes in Computer Science*, pages 283–296. Springer-Verlag, 2005.

[89] Fabien Laguillaumie and Damien Vergnaud. Time-Selective Convertible Undeniable Signatures. In Alfred J. Menezes, editor, *Topics in Cryptology – CT–RSA '05*, volume 3376 of *Lecture Notes in Computer Science*, pages 154–171. Springer-Verlag, 2005.

[90] Peter Landrock. A New Concept in Protocols: Verifiable Computational Delegation. In Bruce Christianson, Bruno Crispo, William S. Harbison, and Michael Roe, editors, *Security Protocols: 6th International Workshop*, volume 1550 of *Lecture Notes in Computer Science*, pages 137–145. Springer-Verlag, 1998.

[91] Serge Lang. *Algebra, revised 3rd edition*, volume 211 of *Graduate Texts in Mathematics.* Springer-Verlag, 2002.

[92] Michael. E. Lee. Optimization of Computer Programs in C. `http://www.prism.uvsq.fr/~cedb/local_copies/lee.html`.

[93] Vincent Lefèvre. Entiers de Gauss (sujet d'étude XM'), 1993. `http://www.vinc17.org/math/index.fr.html`.

[94] Franz Lemmermeyer. *Reciprocity Laws: From Euler to Eisenstein.* Springer Monographs in Mathematics. Springer-Verlag, 2000.

[95] Arjen K. Lenstra and Hendrik W. Lenstra, Jr. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics.* Springer-Verlag, 1993.

[96] Hendrik W. Lenstra, Jr. Factoring Integers with Elliptic Curves. *Annals of Mathematics*, 126:649–673, 1984.

[97] Franck Leprévost, Jean Monnerat, Sébastien Varrette, and Serge Vaudenay. Generating Anomalous Elliptic Curves. *Information Processing Letters*, 93(5):225–230, 2005.

[98] Benoît Libert and Jean-Jacques Quisquater. Identity Based Undeniable Signatures. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT–RSA '04*, volume 2964 of *Lecture Notes in Computer Science*, pages 112–125. Springer-Verlag, 2004.

[99] Scott C. Lindhurst. *Computing Roots in Finite Fields and Groups with a Jaunt through Sums of Digits*. PhD thesis, University of Wisconsin – Madison, USA, 1997.

[100] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[101] Ralph C. Merkle. Secure Communications Over Insecure Channels. *Communications of the ACM*, 21(4):294–299, 1978.

[102] Shawna Meyer Eikenberry and Sorenson Jonathan. P. Efficient Algorithms for Computing the Jacobi Symbol. *Journal of Symbolic Computation*, 26(4):509–523, 1998.

[103] Markus Michels, Holger Petersen, and Patrick Horster. Breaking and Repairing a Convertible Undeniable Signature. In *3rd ACM Conference on Computer and Communications Security*, pages 148–152. ACM Press, 1996.

[104] Markus Michels and Markus Stadler. Efficient Convertible Undeniable Signature Schemes. In *Selected Areas in Cryptography – SAC '97*, pages 231–243, 1997.

[105] Markus Michels and Markus Stadler. Generic Constructions for Secure and Efficient Confirmer Signatures Schemes. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 406–421. Springer-Verlag, 1998.

[106] Jean Monnerat, Yvonne Anne Oswald, and Serge Vaudenay. Optimization of the MOVA Undeniable Signature Scheme. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology – MYCRYPT '05*, volume 3715 of *Lecture Notes in Computer Science*, pages 196–209. Springer-Verlag, 2005.

[107] Jean Monnerat and Serge Vaudenay. Generic Homomorphic Undeniable Signatures. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT '04*, volume 3329 of *Lecture Notes in Computer Science*, pages 354–371. Springer-Verlag, 2004.

[108] Jean Monnerat and Serge Vaudenay. On Some Weak Extensions of AES and BES. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *Information and Communications Security, ICICS '04*, volume 3269 of *Lecture Notes in Computer Science*, pages 414–426. Springer-Verlag, 2004.

[109] Jean Monnerat and Serge Vaudenay. Undeniable Signatures Based on Characters: How to Sign with One Bit. In Feng Bao, Robert Deng, and Jianying

Zhou, editors, *Public Key Cryptography – PKC '04*, volume 2947 of *Lecture Notes in Computer Science*, pages 69–85. Springer-Verlag, 2004.

[110] Jean Monnerat and Serge Vaudenay. Chaum's Designated Confirmer Signature Revisited. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *Information Security, ISC '05*, volume 3650 of *Lecture Notes in Computer Science*, pages 164–178. Springer-Verlag, 2005.

[111] Jean Monnerat and Serge Vaudenay. Short 2-Move Undeniable Signatures. In Phong Q. Nguyễn, editor, *VIETCRYPT '06*, volume 4341 of *Lecture Notes in Computer Science*, pages 19–36. Springer-Verlag, 2006.

[112] Jean Monnerat and Serge Vaudenay. Method to Generate, Verify and Deny an Undeniable Signature. PCT/EP2005/001335, February 13, 2004.

[113] Sean Murphy and Matthew J. B. Robshaw. Essential Algebraic Structure within the AES. In Moti Yung, editor, *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2002.

[114] Melvyn B. Nathanson. *Elementary Methods in Number Theory*, volume 195 of *Graduate Texts in Mathematics*. Springer-Verlag, 2000.

[115] Phong Q. Nguyễn. *La Géométrie des Nombres en Cryptologie.* PhD thesis, Université Paris 7 – Denis Diderot, France, 1999.

[116] Wakaha Ogata, Kaoru Kurosawa, and Swee-Huay Heng. The Security of the FDH Variant of Chaum's Undeniable Signature Scheme. In Serge Vaudenay, editor, *Public Key Cryptography – PKC '05*, volume 3386 of *Lecture Notes in Computer Science*, pages 328–345. Springer-Verlag, 2005. Extended version available on: Cryptology ePrint Archive, Report 2004/290, `http://eprint.iacr.org/`.

[117] Tatsuaki Okamoto. A Fast Signature Scheme Based on Congruential Polynomial Operations. *IEEE Transactions on Information Theory*, 36(1):47–53, 1990.

[118] Tatsuaki Okamoto. Designated Confirmer Signatures and Public-key Encryption are Equivalent. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 61–74. Springer-Verlag, 1994.

[119] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In Kwangjo Kim,

editor, *Public Key Cryptography – PKC '01*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118. Springer-Verlag, 2001.

[120] Tatsuaki Okamoto and Jacques Stern. Almost Uniform Density of Power Residues and the Provable Security of ESIGN. In Chi Sung Laih, editor, *Advances in Cryptology – ASIACRYPT '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 287–301. Springer-Verlag, 2003.

[121] Florin Oswald. SMS Lottery – An Application for MOVA. Master's thesis, EPFL, LASEC, Lausanne, Switzerland, 2006.

[122] Yvonne Anne Oswald. Generic Homomorphic Undeniable Signature Scheme: Optimizations. Semester project, EPFL, LASEC, Lausanne, Switzerland, 2005.

[123] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[124] Sylvain Pasini. Secure Communications over Insecure Channels Using an Authenticated Channel. Master's thesis, EPFL, LASEC, Lausanne, Switzerland, 2005. `http://lasecwww.epfl.ch/`.

[125] Rafael Pass. On Deniability in the Common Reference String and Random Oracle Model. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 316–337. Springer-Verlag, 2003.

[126] Torben P. Pedersen. Distributed Provers with Applications to Undeniable Signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 221–242. Springer-Verlag, 1991.

[127] Duong Hieu Phan and David Pointcheval. Chosen-Ciphertext Security without Redundancy. In Chi Sung Laih, editor, *Advances in Cryptology – ASIACRYPT '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2003.

[128] David Pointcheval. Self-Scrambling Anonymizers. In Yair Frankel, editor, *Financial Cryptography, FC '00*, volume 1962 of *Lecture Notes in Computer Science*, pages 259–275. Springer-Verlag, 2001.

[129] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[130] Omer Reingold, Luca Trevisan, and Salil Vadhan. Notions of Reducibility between Cryptographic Primitives. In Moni Naor, editor, *Theory of Cryptography, TCC '04*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 2004.

[131] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[132] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer-Verlag, 2001.

[133] John Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. In *22nd Annual ACM Symposium on Theory of Computing, STOC '90*, pages 387–394. ACM Press, 1990.

[134] Kouichi Sakurai and Shingo Miyazaki. An Anonymous Electronic Bidding Protocol Based on a New Convertible Group Signature Scheme. In Ed Dawson, Andrew Clark, and Colin Boyd, editors, *Information Security and Privacy, ACISP '00*, volume 1841 of *Lecture Notes in Computer Science*, pages 385–399. Springer-Verlag, 2000.

[135] Renate Scheidler. A Public-Key Cryptosystem Using Purely Cubic Fields. *Journal of Cryptology*, 11(2):109–124, 1998.

[136] Renate Scheidler and Hugh C. Williams. A Public-Key Cryptosystem Utilizing Cyclotomic Fields. *Design, Codes and Cryptography*, 6(2):117–131, 1995.

[137] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[138] Victor Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[139] Eric V. Slud. Distribution Inequalities for the Binomial Law. *The Annals of Probability*, 5(3):404–412, 1977.

[140] Nigel P. Smart. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, 12(3):193–196, 1999.

[141] Victor Stinner. `frequence_cpu.c`, 2003. `http://www.haypocal.com/`.

[142] Alan M. Turing. On Computable Numbers with an Application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[143] Serge Vaudenay. *A Classical Introduction to Cryptography: Applications for Communications Security*. Springer-Verlag, 2006.

[144] André Weilert. $(1+i)$-ary GCD Computation in $\mathbb{Z}[i]$ is an Analogue to the Binary GCD Algorithm. *Journal of Symbolic Computation*, 30(5):605–617, 2000.

[145] André Weilert. Asymptotically Fast GCD Computation in $\mathbb{Z}[i]$. In Wieb Bosma, editor, *Algorithmic Number Theory, ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 595–613. Springer-Verlag, 2000.

[146] André Weilert. Fast Computation of the Biquadratic Residue Symbol. *Journal of Number Theory*, 96(1):133–151, 2002.

[147] Hugh C. Williams and C. R. Zarnke. Some Algorithms for Solving a Cubic Congruence Modulo p. *Utilitas Mathematica*, 6:285–306, 1974.

Curriculum Vitæ

Work Experience

Research and Teaching Assistant from October 2002 at EPFL

- Participation in research activities in the Security and Cryptography Laboratory.
- Teaching assistant for the exercise sessions of the course "Cryptography and Security" in 2003, 2004, 2006.
- Teaching assistant for the exercise sessions of the course "Advanced Cryptography" in 2005.
- Supervisor of 8 student projects.

Education

October 2002–	PhD studies in cryptography, supervised by Prof. Serge Vaudenay, Swiss Federal Institute of Technology in Lausanne (EPFL)
August–September 2002	English language course at the Australian College of English, Bondi Junction, Sydney, Australia
April–July 2002	Graduate school project, supervised by Prof. Serge Vaudenay, Swiss Federal Institute of Technology in Lausanne (EPFL)
1997–March 2002	Master studies in mathematics, Swiss Federal Institute of Technology in Zurich (ETHZ)
1994–1997	Lycée cantonal de Porrentruy (high school), scientific section

Scientific Work

Book

- T. Baignères, P. Junod, Y. Lu, J. Monnerat, and S. Vaudenay, *A Classical Introduction to Cryptography – Exercise Book*, Springer, 2005.

Papers with Peer Review

- J. Monnerat and S. Vaudenay, *Short 2-Move Undeniable Signatures*, VIETCRYPT '06, LNCS **4341**, pp. 19–36, Springer, 2006.
- J. Monnerat, Y. A. Oswald, and S. Vaudenay, *Optimization of the MOVA Undeniable Signature Scheme*, Progress in Cryptology – MYCRYPT '05, LNCS **3715**, pp. 196–209, Springer, 2005.
- J. Monnerat and S. Vaudenay, *Chaum's Designated Confirmer Signature Revisited*, ISC '05, LNCS **3650**, pp. 164–178, Springer, 2005.
- F. Leprévost, J. Monnerat, S. Varrette, and S. Vaudenay, *Generating Anomalous Elliptic Curves*, Information Processing Letters **93**(5), pp. 225–230, Elsevier, 2005.
- G. Avoine, J. Monnerat, and T. Peyrin, *Advances in Alternative Non-adjacent Form Representations*, Progress in Cryptology – INDOCRYPT '04, LNCS **3348**, pp. 260–274, Springer, 2004.
- J. Monnerat and S. Vaudenay, *Generic Homomorphic Undeniable Signatures*, Advances in Cryptology –ASIACRYPT '04, LNCS **3329**, pp. 354–371, Springer, 2004.
- J. Monnerat and S. Vaudenay, *On some Weak Extensions of AES and BES*, ICICS '04, LNCS **3269**, pp. 414–426, Springer, 2004.
- J. Monnerat and S. Vaudenay, *Undeniable Signatures Based on Characters: How to Sign with One Bit*, PKC '04, LNCS **2947**, pp. 69–85, Springer, 2004.

Technical Report

- J. Monnerat, *Computation of the Discrete Logarithm on Elliptic Curves of Trace One*, Technical Report LASEC-REPORT-2002-001 200249, EPFL, `http://infoscience.epfl.ch/getfile.py?recid=52470&ln=en`

Other Activities

- General co-chair (with Serge Vaudenay) of the international workshop "Theory and Practice in Public Key Cryptography – PKC 2005" in Les Diablerets (Switzerland), 2005.
- External reviewer for 25 conferences and one journal.

Awards

Participation in the International Mathematical Olympiad at Mar del Plata (Argentina), 1997.

2nd rank at the swiss selection for participation in the International Mathematical Olympiad in Bern, 1997.

5th rank at the final of the international mathematical and logical games contest in Paris, 1994.

19th rank at the final of the international mathematical and logical games contest in Paris, 1992.

Patent

J. Monnerat and S. Vaudenay, *Method to Generate, Verify and Deny an Undeniable Signature*, PCT/EP2005/001335, February 13, 2004.

Languages

French	Mother tongue
English	Working language
German	Written and spoken fluently