Contributions to Secret Sharing and Other Distributed Cryptosystems

by

Alexandre Ruiz Rodriguez

A THESIS PRESENTED TO THE
UNIVERSITAT POLITÈCNICA DE CATALUNYA
IN FULFILLMENT OF THE THESIS REQUIREMENT FOR THE DEGREE OF
DOCTOR OF MATHEMATICS

Defense at July 2013 in Barcelona, Spain

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Date: **14th June 2013**

Author: **Alexandre Ruiz Rodriguez**

Advisors: **Javier Herranz Sotoca**
Germán Sáez i Moreno

Title: **Contributions to Secret Sharing and Other Distributed Cryptosystems**

Department: **Matemàtica Aplicada IV**

Research Group: **Matemàtica Aplicada a la Criptografia - MAK**

Degree: **Ph.D.** Convocation: **July** Year: **2013**

To my children

Nana korobi ya oki

(If you fall down seven times, get up eight)

Abstract

The present thesis deals with primitives related to the field of distributed cryptography. First, we study signcryption schemes, which provide at the same time the functionalities of encryption and signature, where the unsigncryption operation is distributed. We consider this primitive from a theoretical point of view and set a security framework for it. Then, we present two signcryption schemes with threshold unsigncryption, with different properties. Furthermore, we use their authenticity property to apply them in the development of a different primitive: digital signatures with distributed verification. The second block of the thesis deals with the primitive of multi-secret sharing schemes. After stating some efficiency limitations of multi-secret sharing schemes in an information-theoretic scenario, we present several multi-secret sharing schemes with provable computational security. Finally, we use the results in multi-secret sharing schemes to generalize the traditional framework of distributed cryptography (with a single policy of authorized subsets) into a multi-policy setting, and we present both a multi-policy distributed decryption scheme and a multi-policy distributed signature scheme. Additionally, we give a short outlook on how to apply the presented multi-secret sharing schemes in the design of other multi-policy cryptosystems, like the signcryption schemes considered in this thesis.

For all the schemes proposed throughout the thesis, we follow the same formal structure. After defining the protocols of the primitive and the corresponding security model, we propose the new scheme and formally prove its security, by showing a reduction to some computationally hard mathematical problem.

Acknowledgements

An old chinese proverb taught by Confucius says "I hear and I forget. I see and I remember. I do and I understand". Doing this work has been an enriching experience for me and would not have been possible without the support of different persons I would like to thank in the following.

Primer de tot voldria començar agraint els meus directors de tesi, Javier Herranz i Germán Sáez, per la seva dedicació i temps que han invertit en guiar-me durant la realització d'aquesta tesi. Gracies a ells he pogut aprendre una barbaritat sobre temes força diferents, ja sigui des d'un punt de vista acadèmic o personal. Com ara per exemple, a l'hora d'explicar i donar una estructura a totes aquelles idees que tenia desorganitzades. Així mateix voldria agrair-los el seu optimisme quan arribàvem a un carreró sense sortida i també la seva paciència, ja que m'han permès fer-ho a distancia mentre estava involucrat en un altre treball en paral·lel.

Voldria fer una esment especial per la Paz Morillo i en Jorge L. Villar per haver-me donat l'oportunitat d'entrar al grup de recerca de Matemàtica Aplicada a la Criptografia de la Universitat Politècnica de Catalunya. Igualment agrair als membres del tribunal per acceptar examinar-me d'aquesta tesi.

Fuera del ambito investigador me gustaría dar las gracias a mi familia, en especial a mis abuelos por el cariño que siempre me dieron y a mis padres por animarme a perseguir mis sueños. Tampoco quedría olvidarme de mi profesora de Matemáticas en la escuela por hacer que me picara el gusanillo con las Matemáticas.

Zu guter Letzt möchte ich meiner Frau danken, Birgit Thomae, für ihre unendliche Geduld und ihre unschätzbare Hilfe bezüglich aller Bereiche meines Lebens. Ehrlich

gesagt wäre ohne dich nichts von all dem möglich gewesen. Diese Doktorarbeit ist auch für dich, auch wenn du sie nicht lesen wirst. Und zum Schluß will ich auch nicht meine geliebten Kinder vergessen die mir eine städige Quelle der Kraft sind und die mir täglich zeigen was es heißt, ein kämpferischer Geist zu sein, der darum kämpft seine Ziele zu erreichen. Kämpft weiter und ändert euch um nichts in der Welt!

Contents

List of Tables

List of Figures

Introduction

Nowadays people are involved in different digital activities in their professional life as well as in their private time. Many paper items, such as money and tickets, are more and more being replaced with digital objects. Cryptography plays a crucial role in this transformation, because it provides security in the communication between the different players using a digital channel. Depending on the specific situation, some security requirements in the communication can include privacy (or confidenciality), authentication, integrity or non-repudation.

Classic cryptography focused only on sending messages secretly. A user Alice encrypted a message and sent this ciphertext to another user Bob. At this time, the applications were mainly in diplomatic or war missions and consequently the confidenciality of these messages was extremly important. *Encryption schemes* [77, 73] have been used in different ways for a long time to achieve this goal. With the widespread development of computer communications in the last century, cryptographic community starts working in new primitives and consequently other security requirements start being more important. The appearance of public key cryptography in the seventies by Diffie and Hellman [30] introduced *digital signatures* [77], which provide authenticity, message integrity and non-repudation at the same time. Primitives which provide confidenciality and authentication at the same time are called *signcryption schemes* and were introduced by Zheng [93] at the end of the nineties.

Distributed cryptography spreads the operation of a cryptosystem among a group of parties involved in this process in a fault-tolerant way. Distributed cryptosystems consider the threshold failure model with n servers, of which up to t are faulty. They are called threshold cryptosystems [29]. *Secret sharing schemes* [82, 8] are used to share the secret among different parties in a way that the cooperation of some authorized subset of users is needed to recover this secret. In fact, distributed cryptosystems are based on secret sharing schemes. Opposite to traditional non-distributed cryptosystems, where the secret information is centralized in one single party, distributed cryptosystems are more secure and reliable because the secret information is distributed among a set of parties.

Our main goal in this thesis is to study distributed cryptosystems, where the distributed part relies on the "receiver" operation. Surprisingly, they have received

very few attention from the cryptographic literature up to now. In this thesis we present distributed cryptosystems, prove their security and also obtain secret sharing results to make them more efficient from a computational perspective.

Summary of the Contributions

This thesis combines techniques of unconditional and computational security. First, we study signcryption schemes, where the distributed operation relies on the unsigncryption protocol, from a theoretical point of view and set a security framework for them. Then, we present two signcryption schemes with threshold unsigncryption in different security models and apply them to build digital signatures with distributed verification.

The second contribution of this thesis is an information-theoretical result in secret sharing and several computational secure secret sharing schemes (one publicly verifiable secret sharing scheme and three multi-secret sharing schemes).

Finally, we take one of the proposed multi-secret sharing schemes and included it in the distributed part of cryptosystems to build efficiently an encryption scheme with distributed decryption and a distributed signature scheme in a multi-user setting. Additionally, we give a short outlook how to apply the presented multi-secret sharing schemes to the signcryption schemes proposed throughout this thesis.

Structure of this Thesis

In this paragraph we outline the topics related to our research and show how they are organized in the different chapters and sections of this thesis.

The aim of Chapter 1 is to give a brief overview of cryptography and more specifically to introduce in a self-contained way the basic notions related to our research. The primitives and concepts described here will be used throughout this thesis to present our results. In this way, we discuss the concepts of unconditional and computational security as well as the different security levels. We also describe some complexity assumptions on which the security of the proposed schemes will be based, present some paradigms such as the random oracle model and introduce public key cryptography. Finally, we define in a formal way some cryptographic primitives such as public key (or symmetric) encryption schemes and digital signatures, and explain the meaning of distributed cryptography together with the framework for secret sharing schemes.

Chapter 2 is about signcryption schemes. Here we present signcryption schemes with distributed unsigncryption, where the family of authorized subsets is threshold, for a sender A and a set of users B which perform the unsigncryption phase. After describing a security model against insider attackers in a multi-user setting for these kind of schemes, we proof in Section 2.2 that generic constructions like threshold

versions of Sign_then_Encrypt or Encrypt_then_Sign are not secure in this model. In Sections 2.3 and 2.4 we outline two different signcryption schemes with threshold unsigncryption, which are secure in this model, and compare their efficiency in next section. Moreover, we can see that a special property in the unsigncryption protocol is suitable to apply the proposed schemes to auction systems. The results related to this topic are published in [41] and [44].

If the confidenciality part of the proposed signcryption schemes with threshold unsigncryption is not taken into account and the only requirement is related to their authenticity property, then we should talk, as explained in Section 2.6, about signature schemes with distributed verification. Parallel to signcryption schemes with threshold unsigncryption, in digital signatures with distributed verification all players of some authorized subset can verify the validity of the signature, whereas a subset which is not in the corresponding access structure does not obtain any information about its validity. The results on digital signatures were presented in [42] and [43].

In Chapter 3, we move to the world of secret sharing. The distributed parts of the schemes proposed in the second chapter for both signcryptions and signatures use secret sharing schemes to share a secret over a set of participants. These tasks are done by a trusted entity, called Dealer, and the question is what happens if the participants do not trust him. One possibility could be that these participants take over the dealer role running some protocols (with slight modifications) by themselves. The other one would be to use verifiable secret sharing schemes, where the participants can verify the validity of the shares distributed by the dealer. An example of publicly verifiable secret sharing schemes, which could be applied in these cases, is presented in Section 3.1. This scheme has a verification procedure much simpler than other proposals because it uses some homomorphic properties instead of classical additional zero knowledge proofs. The publication of such a scheme can be found in [78].

Multi-secret sharing schemes, MSSSs in short, are an extension of secret sharing schemes, where multiple secrets are distributed among different parties, each one according to a (possibly different) access structure. One of the main results in the third chapter is a proof in Section 3.2 consisting in a lower bound for the length of each secret share in a MSSS. There we prove that multi-threshold secret sharing schemes enjoying information-theoretic security, in the weaker sense proposed by Masucci, must have shares which are at least as long as the secret. Since the final goal of the proposed MSSSs is the design in the next chapter of multi-policy distributed schemes with shorter secret shares of information than those provided by the trivial solution (whose length grows linearly with the number of secrets), this result is quite negative, and forces us to propose MSSSs in a computational scenario. We stress here that computationally secure MSSSs will be enough for our purposes in this thesis, because the security of multi-policy distributed cryptosystems can be at most computational anyway.

We propose in Sections 3.3 and 3.4 two different computational secure multi-threshold secret sharing schemes in the standard model and compare both schemes to each other in terms of efficiency and security in Section 3.5. A third multi-threshold secret sharing scheme together with its security proof is presented in Section 3.6. This new proposal is very similar to the second one but more efficient because the underlying cryptographic primitive is replaced by an idealized version. Although all these schemes were proposed within a threshold framework, we show in Section 3.7 how to extend them to more general access structures and make them secure against active adversaries. The results of the third chapter related to MSSSs are published in [45] and [46].

Chapter 4 is dedicated to multi-policy distributed cryptosystems. Here we generalize the standard scenario of distributed (public key) cryptography, where a single access structure of authorized subsets of users is used for all the executions of the secret task. In a multi-policy distributed cryptosystem there exists a list of possible acces structures (or policies), such that a specific acces structure is chosen for every execution of the cryptographic operation.

In Section 4.1 we propose a secure multi-policy distributed decryption scheme, where a single user chooses a decryption policy to encrypt a message and the users of an authorized subset for this specific policy cooperate to decrypt the encrypted message. Section 4.2 deals with a secure multi-policy distributed signature scheme, where the signers of an authorized subset choose ad-hoc a signing policy and cooperate to sign a message, whereas the validity of the signature can be checked by anyone using the index of this policy. Both schemes are defined for threshold policies and use the MSSS proposed in Section 3.6 as building block to generate the secret and public keys. These two multi-policy schemes have been published in [45].

After some discussions about the relationship between multi-policy distributed cryptosystems and attribute-based cryptography, we explain in Section 4.4 how to use the same type of solutions we applied in this chapter to the schemes proposed in the second chapter in order to build multi-policy distributed unsigncryptions and multi-policy signatures with distributed verification. We finally list other distributed cryptographic primitives, which are working with only one access structure and could be subject of the same application.

Chapter 1

Preliminaries

1.1 History

The general purpose of cryptography is to enable secure communication between two or more people over an insecure channel where third parties, called adversaries, could be present. The first known use of cryptography dates back to Egypt's Old Kingdom 2500 BC when non-standard hieroglyphs are found in monuments. Cryptography has a long tradition in different cultures, which were using classical cipher types. Ancient greeks used transposition chipers, i.e. reordering the letters in a message, whereas roman emperors used shift ciphers, in which each letter of the message was replaced by a letter some fixed number of positions further down the alphabet, to communicate with their generals. For example we could shift the letters three letters down the alphabet producing the ciphetext "QRW WRGDB" from the original message "NOT TODAY". This particular case is known as Caesar chiper because it was purportedly used by Julius Caesar.

Classical cryptosystems used during this period of time were in fact *symmetric key encryption* schemes where the same (necessarily secret) key was used to both encrypt messages and decrypt ciphertexts. At this time the cryptosystems were heuristic and consequently it was easy to gain access to the contents of hidden information using statistical techniques. Cryptanalysis led to the invention of frequency analysis for breaking cryptosystems where the same letter in the ciphertext replaces each of the corresponding ones in the message, e.g. where an "a" is always replaced by a "t".

The era of modern cryptography really begins with Shannon [83] in 1948, who defined an *information-theoretic* scenario, where an adversary, who knows the ciphertext, does not obtain any information (without knowledge of the key) about the message even assuming he has unlimited computing power. This leads to the definition of *perfect secrecy* [84], the property that the distribution of the ciphertext (over the choice of the key) is exactly the same, no matter what message was encrypted.

Information-theoretic security is often used interchangeably with *unconditional security*. Shannon proved that this level of security is achieved by a scheme if and only if (1) the length of the key is equal or bigger than the length of its plaintext and (2) a new key is chosen for every message to be encrypted. This is a quite negative and inefficient result. The only information-theoretically secure cryptosystem is the symmetric encryption scheme called one-time pad, whose random key (with length at least as long as the plaintext) is used only one time. In his work he uses the concept of entropy $H(X)$ of a random variable X, which is a measure of the uncertainty of the random variable.

One of the main drawbacks of symmetric encryption is the requirement that both parties have access to the same secret key, because they must agree on a secret key before they wish to communicate. To solve this old problem of key exchange over an insecure channel, Whitfield Diffie and Martin Hellman [30] introduced *public key cryptography* in 1976. The central idea there is the use of *one-way functions* (easy to compute on every input, but hard to invert given the image of a random input) which allows to publish the encryption key. In this case we talk about public key encryption (or asymmetric key encryption). More in detail, this means that anyone has access to the encryption key, and can encrypt messages. However only the receiving party has access to the decryption secret key and thus is the only one capable of reading the encrypted messages. One of the first public key encryption schemes was the RSA introduced by Ron Rivest, Adi Shamir and Leonard Addleman [77] in 1978, which implements the idea from Diffie and Hellman.

The arrival of public key cryptography opened new research directions and at this time the study of security moved faster from a information-theoretic scenario to a computational one. In the eighties the notions of computational security were formalised.

1.2 Computational Security

In a *computational* scenario, an adversary has access to limited computational resources, opposite to the information-theoretic scenario where unlimited resources are assumed. Let us first define the concepts of polynomial and negligible functions, before starting to explain how security is defined against computationally bounded adversaries.

Definition 1.2.1 *A function $p : \mathbb{N} \to \mathbb{R}$ is polynomial in n if, for every $n_0 \in \mathbb{N}$, there exists a value $c > 0$ such that $|p(n)| < n^c$, for all $n \geq n_0$.*

Definition 1.2.2 *A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is negligible in n if, for every value $c > 0$, there exists a $n_0 \in \mathbb{N}$ such that $|\varepsilon(n)| < \frac{1}{n^c}$, for all $n \geq n_0$.*

Modern cryptography is heavily based on mathematical theory and computer science practice. First, a security model is considered for the cryptographic primitive (encryption, signature, signcryption, ...). To prove that the scheme is computational secure in this model, a proof by reduction is carried out: if an adversary breaks the security model for this scheme, then another (hard) mathematical problem is broken. Consequently, computational security is achieved by crytosystems by reducing them to the hardness of some mathematical problems, as e.g., the discrete logarithm problem, the problem of factoring a number produced by the product of two large prime numbers or in the recent years different number theoretic problems involving elliptic curves. In other words, computational security is based on complexity assumptions, which state that some problems cannot be solved in polynomial time. It is important to understand that this approach only provides a security proof relative to some other problem, not an absolute proof of security.

1.2.1 Hardness Assumptions

Next we describe both the mathematical problems and computational assumptions related to the hardness of these mathematical problems, which will be used throughout this thesis.

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p, such that p is λ bits long for a security parameter $\lambda \in \mathbb{N}$. The *Computational Diffie-Hellman (CDH) problem*, consists of computing the value g^{ab} on input the values (g, g^a, g^b), for random elements $a, b \in \mathbb{Z}_p^*$. The Computational Diffie-Hellman Assumption states that the CDH problem is hard to solve. A bit more formally, for any polynomial-time algorithm $\mathcal{A}^{CDH}$ that receives as input $\mathbb{G}, g, g^a, g^b$, for random elements $a, b \in \mathbb{Z}_p^*$, we can define as $\mathsf{Adv}_{\mathcal{A}^{CDH}}(\lambda)$ the probability that $\mathcal{A}^{CDH}$ outputs the value g^{ab}. The *Computational Diffie-Hellman Assumption* states that $\mathsf{Adv}_{\mathcal{A}^{CDH}}(\lambda)$ is negligible in λ. Sometimes the Computational Diffie-Hellman problem is simply called Diffie-Hellman (DH).

Many variants of the Computational Diffie-Hellman problem have been considered. The most significant variant is the *Decisional Diffie-Hellman (DDH) problem*, which is to decide whether the four group elements (g, g^a, g^b, h) are all random or they are a valid Diffie-Hellman tuple, that is $h = g^{ab}$. Groups where the CDH problem is hard to solve but the DDH problem is easy are called *Gap Diffie-Hellman (GDH) groups*. See [51, 13, 72] for more details on GDH groups. Up to now, the only known GDH groups are related to bilinear pairings on elliptic curves.

The Computational Diffie-Hellman problem is easier to solve than the *Discrete Logarithm (DL) problem*: the input is $(\mathbb{G}, g, y)$, where $y \in \mathbb{G}$, and the goal for a solver $\mathcal{A}^{DL}$ is to find the integer $x \in \mathbb{Z}_q^*$ such that $y = g^x$. We can define $\mathsf{Adv}_{\mathcal{A}^{DL}}(\lambda)$ and the *Discrete Logarithm Assumption* analogously to the Diffie-Hellman case.

A group $\mathbb{G} = \langle g \rangle$ as defined above is said to be *bilinear* if there exist another

group $\mathbb{G}_T$ with the same order p and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, known as bilinear *pairing*, satisfying the following properties:

1. Computability: $e(\cdot, \cdot)$ can be efficiently computed (in time polynomial in λ),

2. Non-Degeneracy: $e(g, g)$ is a generator of $\mathbb{G}_T$,

3. Bilinearity: for any two elements $a, b \in \mathbb{Z}_p$, we have $e(g^a, g^b) = e(g, g)^{ab}$.

The *Decisional Bilinear Diffie-Hellman (DBDH) problem* consists of distinguishing tuples of the form $(g, g^a, g^b, g^c, e(g, g)^{abc})$ from tuples of the form (g, g^a, g^b, g^c, T), for random $a, b, c \in \mathbb{Z}_p^*$ and random $T \in \mathbb{G}_T$. For any polynomial-time solver $\mathcal{A}^{DBDH}$ of this problem, we can define its advantage as $\mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) =$

$$\left|\Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 0] \; - \; \Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, T) = 0]\right|$$

The *Decisional Bilinear Diffie-Hellman Assumption* states that $\mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda)$ is negligible in λ.

1.3 Public Key Encryption Schemes

In an encryption scheme, the message or plaintext is encrypted using an *encryption algorithm*, which outputs a ciphertext. An authorized party, is able to decode the ciphertext using a *decryption algorithm*. This kind of schemes usually needs a *key generation algorithm*, to randomly produce the encryption and decryption keys used in the respective algorithms. When these keys are the same, then we talk about symmetric encryption (or private key encryption) schemes. On the contrary, public key encryption schemes use different keys, one public in the encryption algorithm and the secret one in the decryption algorithm.

Let's define a *public key encryption scheme*, in short PKE, in a formal way: a public key encryption scheme $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ consists of a (randomized) key-generation protocol $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ which takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs both a secret key $\mathsf{sk} \in \mathcal{K}_1$ and a public key $\mathsf{pk} \in \mathcal{K}_2$ in the respective sets $\mathcal{K}_1$ and $\mathcal{K}_2$ of possible keys, then a (randomized) encryption protocol $c \leftarrow \Pi.\mathsf{Enc}(m, \mathsf{pk})$ which encrypts the plaintext or message $m \in \mathcal{M}$ and finally a decryption protocol $\tilde{m} \leftarrow \Pi.\mathsf{Dec}(c, \mathsf{sk})$ used to recover the message for a given ciphertext $c \in \mathcal{C}$. For correctness, $\Pi.\mathsf{Dec}(\Pi.\mathsf{Enc}(m, \mathsf{pk}), \mathsf{sk}) = m$ must hold, for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Pi.\mathsf{KG}(1^\lambda)$.

1.3.1 Security Model

The security of public key encryption schemes is defined in a computational scenario, where different security levels can be achieved by the cryptosystems. For example, a public encryption scheme can be proved to be secure against an adversay who tries to recover the original message given the ciphertext, although there exists an attack which finds a valid ciphertext for some message. These levels can be separated into two different directions depending on the resources of the adversary and their goals.

Let's define the computational security of a public key encryption scheme Π under chosen-ciphertext attacks [5, 52] using the following IND-CCA game $\mathcal{G}$ between a challenger and an adversary $\mathcal{A}_\Pi$.

1. The challenger chooses a bit $\beta \in \{0, 1\}$ at random.
2. The challenger runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ and sends the value pk to $\mathcal{A}_\Pi$.
3. [**Decryption queries**] The adversary $\mathcal{A}_\Pi$ can make adaptive decryption queries for ciphertexts $c \in \mathcal{C}$ of his choice. As an answer, $\mathcal{A}_\Pi$ receives the plaintext $m \leftarrow \Pi.\mathsf{Dec}(c, \mathsf{sk})$.

   Note that encryption queries make no sense because $\mathcal{A}_\Pi$ can reply such queries by himself.
4. [**Challenge**] Let $\mathcal{M}$ be the set of the messages, $\mathcal{A}_\Pi$ broadcasts two different messages $m_0 \neq m_1$ with the same length such that $m_0, m_1 \in \mathcal{M}$. The challenger runs $c^\star \leftarrow \Pi.\mathsf{Enc}(m_\beta, \mathsf{pk})$ and sends the result $c^\star$ back to $\mathcal{A}_\Pi$.
5. $\mathcal{A}_\Pi$ can make more decryption queries, with the restriction that the ciphertext $c^\star$ cannot be queried.
6. Finally, $\mathcal{A}_\Pi$ outputs a bit β'.

The advantage of $\mathcal{A}_\Pi$ in breaking the security of the scheme Π is defined as

$$\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|$$

We say that the public encryption scheme Π enjoys IND-CCA security if the advantage $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda)$ is a negligible function in λ, for any polynomial-time adversary $\mathcal{A}_\Pi$.

If decryption queries are not allowed in the above game then we are reducing the security to a weaker level, IND-CPA, where the adversary has only the capability to choose arbitrary plaintexts to be encrypted, obtaining the corresponding ciphertexts

in this process. Moreover, an extension of the game $\mathcal{G}$ is possible in a multi-user scenario using the ideas from [3].

To avoid security problems within this model, one could modify the encryption scheme embedding some form of structured, randomized padding into the message before encrypting it. This padding can be done by hash functions, which map the messages to their hash values.

Definition 1.3.1 *A cryptographic hash function $H : \{0,1\}^* \to Y$ is a function that compresses an input of arbitrary length to a result with a fixed length, and verifies the one-wayness and (strong) collision-resistance properties:*

1. *For any given data $x \in \{0,1\}^*$ it is easy to calculate the hash value $H(x)$.*
2. *For a given hash value $y \in Y$ it is computationally difficult to find a data $x \in \{0,1\}^*$ such that $H(x) = y$.*
3. *It is computationally unlikely to find two different inputs $x, x' \in \{0,1\}^*$ with the same hash value $H(x) = H(x')$.*

A cryptographic hash function should behave as much as possible like a random function while still being deterministic and efficiently computable. Hash functions play an important role in cryptography not only due to the padding but also because they allow to define an idealized scenario, where security is much easier to prove.

1.3.2 The Random Oracle Model

As explained above the usual way to prove the security in modern cryptographic systems is to reduce the security problem of the system to a related computational problem. In this case schemes are proven secure using only complexity assumptions and are said to be secure in the *standard model*. In order to make the issue of reducing the security property of some schemes (as confidentiality for cryptosystems or unforgeability for signature schemes) to the hardness of well known computational problems easier, Bellare and Rogaway introduced in [6] a new paradigm, where cryptographic primitives are replaced by idealized versions. The *random oracle model*, in short ROM, replaces a cryptographic hash function with a genuinely random function. In this model, hash functions are seen as oracles (a theoretical black box) that respond to every new query with a (truly) random response chosen uniformly from its output domain, except that if the same query is asked twice then the output must be identical.

This paradigm has been criticized in several works [20, 71, 4] because hash functions are deterministic and hence not probabilistic as in the random oracle model. Moreover, this assumption is useful but not achievable in real systems. Therefore,

proofs in the random oracle model are only heuristic arguments, and thus security proofs in the standard model are preferable, when analyzing the security of cryptographic protocols. However, it has been accepted by the cryptographic community because it allows to derive security proofs of many cryptosystems more easily and gives very strong evidence of their security.

1.3.3 Paillier's Public Key Cryptosystem

A known public key cryptosystem was presented in 1999 by Pascal Paillier [73]. This scheme has an additively homomorphic encryption, which allows specific types of computations to be carried out on ciphertext and obtain an encrypted result which is the ciphertext of the result of operations performed on the plaintext. Let's show this scheme more in detail.

Key Generation: $\Pi.\mathsf{KG}(1^\lambda)$.
Given a security parameter λ, two different large primes p, q with length of λ bits are chosen. The values $N = p \cdot q$ and $\eta = lcm(p-1, q-1)$, where η is the result of the Carmichael's function $\eta(N)$, are computed. Finally, an element $g \in \mathbb{Z}^*_{N^2}$ with order a multiple of N is chosen. The public output of this protocol is $\mathsf{pk} = (N, g)$, and the secret output privately stored is $\mathsf{sk} = \eta$. The secret key can also be seen as the tuple (p, q).

Encryption: $\Pi.\mathsf{Enc}(m, N, g)$
For a random element $r \in \mathbb{Z}^*_N$, the encryption of a message m is defined by the bijective function

$$\begin{array}{rccll} \varepsilon_g : & \mathbb{Z}_N \times \mathbb{Z}^*_N & \longrightarrow & \mathbb{Z}^*_{N^2} & \\ & (m, r) & \longmapsto & g^m \cdot r^N & \mod N^2 \end{array}$$

This protocol returns the ciphertext $c := \varepsilon_g(m, r) = g^m \cdot r^N \mod N^2$.

Decryption: $\Pi.\mathsf{Dec}(c, \eta)$
Let $S_N = \{u \in \mathbb{Z}^*_{N^2} \mid u = 1 \mod N\}$ be a multiplicative subgroup of $\mathbb{Z}^*_{N^2}$, the logarithmic function L is defined as $L(u) = \frac{u-1}{N}$, for $u \in S_N$. This protocol returns the message $m = \dfrac{L(c^\eta \mod N^2)}{L(g^\eta \mod N^2)} \mod N$.

Note that Paillier's encryption protocol is additively homomorphic over the plaintexts because of the property $\varepsilon_g(m_1, r_1) \cdot \varepsilon_g(m_2, r_2) = \varepsilon_g(m_1 + m_2, r_1 \cdot r_2)$. In other words, $\Pi.\mathsf{Enc}(m_1 + m_2, \mathsf{pk}) = \Pi.\mathsf{Enc}(m_1, \mathsf{pk}) \cdot \Pi.\mathsf{Enc}(m_2, \mathsf{pk})$ and $\Pi.\mathsf{Dec}(c_1 \cdot c_2, \mathsf{sk}) = \Pi.\mathsf{Dec}(c_1, \mathsf{sk}) + \Pi.\mathsf{Dec}(c_2, \mathsf{sk})$. In relation to the efficiency, a value $g \in \mathbb{Z}^*_{N^2}$ of order N (e.g., $g = 1 + N$) can be taken without affecting the security of the encryption scheme.

Security of Paillier's cryptosystem is based on the intractable *Decisional Composite Residuosity Assumption* (DCRA): Let p, q be two different l-bit primes such that $N = p \cdot q$ and $g \in \mathbb{Z}^*_{N^2}$ with order N. The following two probability distributions are polynomially indistinguishable

$$\begin{cases} D_{residue} = (N, g, \rho^N), & \rho \leftarrow \mathbb{Z}^*_N \\ D_{random} = (N, g, y), & y \leftarrow \mathbb{Z}^*_{N^2} \end{cases}$$

i.e., there does not exist any probabilistic polynomial time distinguisher for N^{th} residues [1] [2] modulo N^2.

Because of the mentioned homomorphic properties, the Paillier's cryptosystem does not achieve IND-CCA security. In general, in homomorphic encryption schemes it is possible for the adversary to modify the challenge ciphertext and submit it to the decryption oracle in order to break the IND-CCA game. Despite this intrinsic feature, it is possible to transform the original cryptosystem using the techniques of Fujisaki-Okamoto [34] to achieve IND-CCA security in the random oracle model. In this case, obviously the homomorphic property of the scheme disappears to achieve this level of security.

1.4 Other Primitives with Computational Security

We explain here another cryptographic primitive, which will be used as key ingredient in the designs of the schemes proposed throughout this thesis, together with their security models in a computational scenario.

1.4.1 Symmetric Encryption with Semantic Security

A symmetric encryption scheme $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ consists of a key-generation protocol $K \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ which takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs a secret key $K \in \mathcal{K}$, then an encryption protocol $c \leftarrow \Pi.\mathsf{Enc}(m, K)$ and a decryption protocol $\tilde{m} \leftarrow \Pi.\mathsf{Dec}(c, K)$. For correctness, $\Pi.\mathsf{Dec}(\Pi.\mathsf{Enc}(m, K), K) = m$ must hold, for any $K \leftarrow \Pi.\mathsf{KG}(1^\lambda)$.

Security of symmetric encryption schemes was used only in an information-theoretic scenario in the past but the come out of formal definitions for the security models, as

[1] An element $x \in \mathbb{Z}^*_{N^2}$ is said to be an N^{th} residue if there exists another element $y \in \mathbb{Z}^*_{N^2}$ such that $y = x^N \mod N^2$.

[2] Since N divides the order of $\mathbb{Z}^*_{N^2}$ ($|\mathbb{Z}^*_{N^2}| = \phi(N^2) = N \cdot \phi(N)$) it follows that the set of N^{th} residues is a subgroup of $\mathbb{Z}^*_{N^2}$ with cardinality $\phi(N)$, where $\phi(N) = (p-1) \cdot (q-1)$ is the Euler's totient function.

explained in Subsection 1.3.1, made it possible to define also its security in computational sense. Semantic security of a symmetric encryption scheme Π under chosen-ciphertext attacks [5, 52], in the multi-user setting [3], is defined by the following $\mathsf{IND\text{-}CCA}$ game $\mathcal{G}$ between a challenger and an adversary $\mathcal{A}_\Pi$.

1. The challenger chooses at random $\beta \in \{0, 1\}$.

2. $\mathcal{A}_\Pi$ chooses the number k of keys in the game.

3. The challenger runs k times the protocol $K_i \leftarrow \Pi.\mathsf{KG}(1^\lambda)$, to produce k secret keys $K_1, \ldots, K_k$.

4. [**Encryption queries**] The adversary $\mathcal{A}_\Pi$ can make, at any time, encryption queries (i, m) of its choice, where $i \in \{1, \ldots, k\}$. As the answer, $\mathcal{A}_\Pi$ receives the ciphertext $c \leftarrow \Pi.\mathsf{Enc}(m, K_i)$.

5. [**Decryption queries**] The adversary $\mathcal{A}_\Pi$ can make, at any time, decryption queries (i, c) of its choice, where $i \in \{1, \ldots, k\}$. As the answer, $\mathcal{A}_\Pi$ receives the plaintext $m \leftarrow \Pi.\mathsf{Dec}(c, K_i)$.

6. [**Challenges**] In an adaptive way, $\mathcal{A}_\Pi$ chooses tuples $(i_j, m_j^{(0)}, m_j^{(1)})$, where $i_j \in \{1, \ldots, k\}$ and $m_j^{(0)} \neq m_j^{(1)}$ have the same length, for all $j = 1, \ldots, q_c$, and q_c is the number of challenges. The challenger runs $c_j^* \leftarrow \Pi.\mathsf{Enc}(m_j^{(\beta)}, K_{i_j})$ and sends c_j^* back to $\mathcal{A}_\Pi$, for $j = 1, \ldots, q_c$.

7. $\mathcal{A}_\Pi$ can make more encryption and decryption queries, provided the challenge pairs (i_j, c_j^*) are not asked as a decryption query.

8. Finally, $\mathcal{A}_\Pi$ outputs a bit β'.

For such an adversary $\mathcal{A}_\Pi$ playing this game and making at most q_e encryption queries, q_d decryption queries and q_c challenge queries, we say $\mathcal{A}_\Pi$ is a (k, q_e, q_d, q_c)-adversary against Π. The advantage of $\mathcal{A}_\Pi$ in breaking the $\mathsf{IND\text{-}CCA}$ security of the scheme Π is defined as

$$\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|$$

The symmetric encryption scheme Π is said to enjoy $\mathsf{IND\text{-}CCA}$ security in the multi-user setting if $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda)$ is a negligible function in λ, for any (k, q_e, q_d, q_c)-adversary $\mathcal{A}_\Pi$ that runs in polynomial-time (in particular, all values k, q_e, q_d, q_c must be polynomial in λ).

1.4.2 Digital Signature Schemes

The appearance of the paradigm of *public key cryptography* introduced in 1976 by Diffie and Hellman [30] involved a new mechanism to provide authentication and non-repudation. These two properties together with message integrity for the signed message are achieved in *digital signature schemes*.

A signature scheme $\Theta = (\Theta.\mathsf{KG}, \Theta.\mathsf{Sign}, \Theta.\mathsf{Vfy})$ consists of three probabilistic polynomial time protocols. $\Theta.\mathsf{KG}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$ is the key generation protocol, which takes as input a security parameter $\lambda \in \mathbb{N}$ and outputs a secret signing key sk and a public verification key vk. The signing protocol $\Theta.\mathsf{Sign}(\mathsf{sk}, m) \to \theta$ takes as input the signing key and a message m, and outputs a signature θ. Finally, the verification protocol $\Theta.\mathsf{Vfy}(\mathsf{vk}, m, \theta) \to 1$ or 0 takes as input the verification key, a message and a signature, and outputs 1 if the signature is valid, or 0 otherwise.

The *correctness* property is held in a signature scheme when a signature generated with the signing protocol is always accepted by the verifier. That is, if $\Theta.\mathsf{KG}(1^\lambda) = (\mathsf{sk}, \mathsf{pk})$ and $\Theta.\mathsf{Sign}(\mathsf{sk}, m) = \theta$ then $\Theta.\mathsf{Vfy}(\mathsf{vk}, m, \theta) = 1$. In this case, it is said that (m, θ) is a valid message/signature pair.

In parallel to encryption schemes, digital signatures can also have different levels of security. These levels are achieved according to the capabilities of the adversary and its final goals. In fact, the intuitive idea behind it was formalized by Goldwasser, Micali and Rivest in [37] and explains that only the owner of a secret key should be able to compute valide signatures with respect to the matching public key.

An adversary can have different capabilities to break a signature scheme. For example, the adversay knows only the public key of the user, the adversary has access to valid signatures of a list of messages which were not chosen by him or the adversary has access to valid signatures for messages that he can adaptively choose.

Regardings to his final goal, different levels of success for an adversary can be taken into account. That is, find a valid signature for some message, find a valid signature for a fixed message, find an efficient algorithm which emulates the signing algorithm of a user or compute the secret key of a user.

Note that the maximum level of security for such a scheme consists of resisting attacks from an adversary with the most powerful capabilities (*adaptive chosen message attack*) but with the less ambitious goal (*existential forgery*, for some message). A signature scheme is unforgeable when any attacker has negligible probability of success in forging a valid signature for some message, even if he knows valid signatures for messages different from the pair message/signature which is tried to be forged, that he can adaptively choose.

More in detail, we consider an adversary F_Θ who first receives a verification key vk obtained from $\Theta.\mathsf{KG}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$. He can make at most q_S signature queries for messages m_i of his choice, obtaining as answer valid signatures $\Theta.\mathsf{Sign}(\mathsf{sk}, m_i) \to \theta_i$, and finally outputs a pair (m', θ'). We say that the adversary succeeds if $\Theta.\mathsf{Vfy}(\mathsf{vk}, m', \theta') \to$

1 and $(m', \theta') \neq (m_i, \theta_i)$ for all $i = 1, \ldots, q_S$.

We denote $\mathcal{F}_\Theta$'s success probability as $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$. The signature scheme Θ is *strongly unforgeable* if $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$ is a negligible function of the security parameter $\lambda \in \mathbb{N}$, for any polynomial-time attacker F_Θ against Θ. Here negligible means that $\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$ decreases (when λ increases, asymptotically) faster than the inverse of any polynomial. If a signature scheme is strongly unforgeable only against adversaries who can make at most $q_S = 1$ signature query, then the scheme is a secure *one-time* signature scheme.

An example of strongly unforgeable signature scheme can be found in [15]. The scheme therein is proved secure, in the standard model, under the Computational Diffie-Hellman Assumption (defined in Subsection 1.2.1). Further examples of secure one-time signature schemes can be found in [68].

1.5 Distributed Cryptography

Distributed cryptography spreads the operation of the schemes among a group of servers (or parties) in a fault-tolerant way. First proposals of distributed cryptography can be found in [16, 29]

Situations where the secret information is distributed among a set of users are very common in real-life applications, where giving too much power to a centralized single user may be delicate for security (because corruption of this user can compromise the whole system) and for reliability (because a technical problem at this user can lead to important delays in the life of the system) reasons. In fact, the secret key of an individual user in the standard scheme is distributed in shares by means of a secret sharing scheme. A typical example is key escrow: a trusted entity stores encrypted versions of the secret key material of all the users in a system or community. If a user loses his secret key, he can ask for it to the entity. Also if a judge decides that the secret communications involving a malicious user must be revealed, he can ask the secret key of that user to the key escrow entity. A good solution is to distribute the power of this trusted entity among a set of entities, through a distributed process.

Threshold cryptography deals with situations where the power to do a secret cryptographic task is shared among a group of n users and where the cooperation of at least t of them is necessary to successfully finish the task. A clear example of this situation could be in a bank where there is a vault which must be opened every day. If the bank has n employees, but they do not trust the combination to any individual employee, then it is more secure to share this vault among these employees. Hence, a threshold cryptosystem could be applied where any $t \leq n$ of them can gain access to the vault, but $t - 1$ employees or less can not do so.

1.5.1 Secret Sharing Schemes

These kind of schemes were created in paralel by Blakley [8] and Shamir [82] in the late seventies and are an important building block in distributed cryptography. Such schemes allow to distribute shares of the secret to each of the participants, where a structure over the set of participants is fixed from the beginning to establish which participants are able to determine the secret.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a finite set of n participants and Γ a set of subsets of $\mathcal{P}$, that is $\Gamma \subset 2^{\mathcal{P}}$; the subsets in Γ are those subsets of participants that should be able to compute the secret. In this case, Γ is called *access structure* and the subsets in Γ are known as *authorized subsets*. This access structure must satisfy the *monotone increasing* property; that is, if $A \in \Gamma$ and $A \subseteq B \subseteq \mathcal{P}$, then $B \in \Gamma$. Moreover, a subset $A \in \Gamma$ is said a *minimal authorized subset* if $B \notin \Gamma$, for all $B \subset A$. The set of minimal authorized subsets of Γ is denoted Γ_0 and is called the *basis* of Γ. In this case we have $\Gamma = \{B \subseteq \mathcal{P} : \exists A \in \Gamma_0 \text{ verifying } A \subseteq B\}$ and we say that Γ is the closure of Γ_0; that is, $\Gamma = cl(\Gamma_0)$.

The value of a secret is chosen in a space of possible secrets by a special participant called the *dealer* and denoted by D. When D wants to share the secret in the *distribution phase* among the participants in $\mathcal{P}$, he uses the access structure Γ to give each participant some partial information called a *share*. These shares should be sent to the participants secretly, so no participant knows the share given to another one. At a later time, a subset of participants $A \subseteq \mathcal{P}$ will pool their shares in the *reconstruction phase* in an attempt to compute the secret (alternatively, they could give their shares to a trusted authority which will perform the computation for them). If $A \in \Gamma$, then the participants of the authorized subset A are able to compute efficiently the value of the secret as a function of the shares they collectively hold; on the contrary, if $A \notin \Gamma$ (A is a corruptible or unauthorized subset) then they can determine no information about the value of the secret although they pool their shares. In the above situation we speak about a *secret sharing scheme*, SSS in short, which realizes the access structure Γ.

The above two requirements, *correctness* and *secrecy* (or privacy), can be formalized either in an information-theoretic sense or in a computational sense, depending on the resources available to the possible attackers. Unconditional security is the standard way to define security in those schemes because they are usually used in a theoretical level to build another protocols in theoretical cryptography, as e.g. multiparty computation or oblivious transfer, where it is important to execute concurrently several instances of the inherent secret sharing scheme. In this case, the security of the new protocols is assured if the underlying secret sharing scheme is unconditional secure. In this thesis we will see that computational security for secret sharing schemes is enough because we will use them as building block to build another primitives whose security can be at most computational.

A tool used in the information-theoretic scenario is the *entropy* of a random variable. Namely, if we use notation S for the random variable associated to the secret, SH_i for the random variable associated to the share of player $P_i \in \mathcal{P}$, and more generally SH_A for the (vector) random variable associated to the shares of players in $A \subseteq \mathcal{P}$, the two required properties defined above for a *perfect* secret sharing scheme become: (1) $H(S|\mathsf{SH}_A) = 0$ for any subset $A \in \Gamma$, and (2) $H(S|\mathsf{SH}_A) = H(S)$ for any subset $A \notin \Gamma$. It is well known that any secret sharing scheme enjoying security in the information-theoretic scenario (where attackers have unlimited computational resources) must produce shares that are, at least, as long as the secret: $|H(\mathsf{SH}_i)| \geq |H(S)|$, for all $P_i \in \mathcal{P}$. In the case that the bound is achieved, $|H(\mathsf{SH}_i)| = |H(S)|$, we talk about *ideal* secret sharing schemes. In secret sharing schemes with security in the computational scenario (where attackers are modeled as polynomial-time algorithms), shares can be shorter than the secret [55]. We define the *information rate* as the quotient between the length of the secret and the maximum length of the shares. Note that ideal secret sharing schemes have an information rate of 1.

A well known *threshold* secret sharing scheme was proposed by Shamir in [82], where subsets that can recover the secret are those with at least t members (t is the threshold); in other words, the (t, n)-threshold access structure is realized by $\Gamma = \{A \subseteq \mathcal{P} : |A| \geq t\}$. The set of possible secrets is a finite field $\mathbb{K}$. To share a secret $s \in \mathbb{K}$ the dealer D chooses at random a polynomial $f(x) \in \mathbb{K}[X]$ of degree $t-1$ with evaluation point $f(0) = s$ and sends to every participant $P_i \in \mathcal{P}$ the share $s_i = f(i)$ via a secure channel. When t or more participants of $A \in \Gamma$ want to recover the secret, they compute $s = f(0) = \sum_{P_i \in A} \lambda_i^A \cdot f(i)$ using the Lagrange interpolation coefficients $\lambda_i^A = \prod_{1 \leq h \leq t, h \neq i} \frac{h}{h-i}$, whereas any set of less than t shares gives no information at all about the secret s.

The role of the dealer in (Shamir) threshold secret sharing schemes can be replaced by all the participants in $\mathcal{P}$. They can distribute a secret working jointly as follows: the secret s to be distributed is not an input of the distribution protocol, because it will be generated by players in $\mathcal{P}$ "on the fly". Every $P_i \in \mathcal{P}$ chooses a random polynomial $f_i(x) \in \mathbb{K}[X]$ of degree $t-1$. The secret s is implicitly defined as $s = \sum_{P_i \in \mathcal{P}} f_i(0)$. Every $P_i \in \mathcal{P}$ sends the value $f_i(k)$ to the other participants $P_k \in \mathcal{P}$. At this point, the participant $P_k \in \mathcal{P}$ is able to compute his own secret value $s_k = \sum_{P_i \in \mathcal{P}} f_i(k)$, which is a polynomial share of the secret s. In the reconstruction protocol the players of an authorized subset $A \in \Gamma_j$ use the values $\{s_i\}_{P_i \in A}$ to interpolate the polynomial $F(x) = \sum_{P_i \in \mathcal{P}} f_i(x)$ in $x = 0$, recovering in this way the secret $s = F(0)$.

An important class of access structures are the *vector space access structures* introduced by Brickell [17]. A vector space secret sharing scheme realizes such a structure Γ, defined in a finite field $\mathbb{K}$, if there exists a map $\psi : \mathcal{P} \cup \{D\} \to \mathbb{K}^t$ with

$t \in \mathbb{N}$, such that $A \in \Gamma$ if and only if $\psi(D) \in \langle \psi(i) \rangle_{P_i \in A}$. If the dealer wants to distribute the secret value $s \in \mathbb{K}$, he chooses randomly a vector $v \in \mathbb{K}^t$, such that $s = v \cdot \psi(D)$ and distributes the shares $s_i = v \cdot \psi(i)$ amount the participants $P_i \in \mathcal{P}$. When the participants of an authorized subset $A \in \Gamma$ want to recover the secret, then they check the values $\{\lambda_i^A\}_{P_i \in A} \in \mathbb{K}^t$, obtained by definition in $\psi(D) = \sum_{P_i \in A} \lambda_i^A \cdot \psi(i)$ and compute the secret value using their distributed shares $\{s_i\}_{P_i \in A}$ as follows.

$$\sum_{P_i \in A} \lambda_i^A \cdot s_i = \sum_{P_i \in A} \lambda_i^A \cdot v\psi(i) = v \cdot \sum_{P_i \in A} \lambda_i^A \psi(i) = v \cdot \psi(D) = s.$$

Note that the Shamir's threshold SSS, with threshold t, can be seen as a particular case of vector space SSS, by defining $\psi(D) = (1, 0, \ldots, 0)$ and $\psi(i) = (1, i, i^2, \ldots, i^{t-1})$ for every player $P_i \in \mathcal{P}$.

In the case of vector space access structures the members of $\mathcal{P}$ can do the tasks of a trusted dealer by themselves, as proposed in [47], without any interaction with him: firstly, every participant $P_i \in \mathcal{P}$ chooses a random $v_i \in \mathbb{K}^t$ and sends the values $s_{ij} = v_i \cdot \psi(j)$ to the other participants $P_j \in \mathcal{P}$. Secondly, every participant $P_j \in \mathcal{P}$ sums the n values $\{s_{ij}\}_{P_i \in \mathcal{P}}$ he has received giving as a result $\sum_{P_i \in \mathcal{P}} s_{ij} = \sum_{P_i \in \mathcal{P}} v_i \cdot \psi(j) = v \cdot \psi(j)$, where $v = \sum_{P_i \in \mathcal{P}} v_i$. In this way every participant $P_j \in \mathcal{P}$ has the share $v \cdot \psi(j)$ for a random vector v.

Multi-secret sharing scheme, MSSS for short, has been studied in different works [9, 67] and is an extension of secret sharing scheme. They are very useful for situations where different runnings of a secret task (like signature or decryption) may have different levels of importance or secrecy. Now, ℓ different secrets are distributed among the players in set $\mathcal{P}$, each one according to a (possibly different) access structure. The trivial solution to this problem is to run ℓ independent instances of a standard secret sharing scheme, one for each secret and access structure. In this solution, the length of the secret share to be stored by each player is linear in ℓ. More about MSSS will be said in Section 3.2.

Chapter 2

Signcryption Schemes with Threshold Unsigncryption

Signcryption is a paradigm in public key cryptography that simultaneously fulfils the function of digital signature, providing authenticity, and public key encryption, providing confidentiality, in a logically single step and with a cost significantly lower (between 50%-90% more efficient) than with the trivial solution of signing and encrypting each message separately.

In many applications, confidentiality and authenticity are needed together. Such applications include secure email (S/MIME), secure shell (SSH), and secure web browsing (HTTPS). By *confidentiality*, we mean that only the intended recipient of a signcrypted message should be able to read its contents. That is, upon seeing a signcrypted message, an attacker should learn nothing about the original message, other than perhaps its length. By *authenticity*, we mean that the recipient of a signcrypted message can verify the sender's identity, i.e., an attacker should not be able to send a message, claiming to be someone else. Note also that signature schemes always provide *non-repudiation*, since anyone can verify a signature using only the sender's public key. This is not the case for signcryption, since the confidentiality property implies that only the recipient should be able to read the content of a signcrypted message sent to him. However, it is possible to achieve non-repudiation by other means.

In 1997, Yuliang Zheng launched the study of signcryption (also known as authenticated encryption) by giving a pair of signcryption schemes in [93]. Since the introduction of this cryptographic primitive several generic constructions [1], combining signature and encryption schemes, and specific proposals have appeared. A survey about this topic can be found in [28].

Signcryption schemes consist of a *key generation* protocol, a *signcryption* protocol run by the sender of the message (which uses his secret key and the public key of the

receiver to hide and authenticate the message) and an *unsigncryption* protocol run by the receiver (which uses his secret key and the public key of the sender to recover the message and verify its authenticity).

Most of the papers dealing with signcryption consider individual entities to perform the secret tasks of signcryption and unsigncryption. In many real-life situations, centralizing a secret task is not desirable due to both security and reliability reasons (a security / technical problem at a single entity can cause important threats / delays to the system). In these cases, a common approach is to decentralize the secret task(s) by considering a group of n entities, in such a way that the cooperation of at least t of them is necessary to successfully finish the task. This approach is known as (t, n)-*threshold cryptography*. In the scenario of signcryption, there are two secret tasks, so threshold cryptography could be applied to the signcryption protocol [64], to the unsigncryption protocol [54], or to both of them. For the first concept, designing a threshold signcryption variant of an ordinary signcryption scheme may be reasonably easy, for example if the signcryption scheme works in the Discrete Logarithm framework. This is similar to what happens with the transition from a standard signature scheme to a threshold signature scheme.

Among the above three possibilities, here we focus on the situation where the unsigncryption task is distributed among a set of entities through a (t, n)-threshold process. Such schemes are known as *threshold unsigncryption* schemes. For simplicity we consider that the signcryption protocol is run by an individual entity (see however [44] for a discussion on fully threshold signcryption). We want to stress that the primitive of threshold unsigncryption is not just of theoretical interest; it has applications in real-life scenarios. For example, in a digital auction system, bidders may send their authenticated private bids, encrypted with the public key of a set of servers. In this way, even if some dishonest servers (less than t) collude, they will not be able to obtain information about the bids and influence the result of the auction. At the end of the auction, a large enough number of servers will cooperate to decrypt the bids and determine the winner of the auction and the price to pay.

The first works that focused on threshold unsigncryption [54, 92, 59, 60, 90] failed to achieve the desired security properties for this kind of schemes: existential unforgeability under chosen message attacks, and plaintext indistinguishability under chosen-ciphertext attacks (CCA), in a multi-user setting where the adversary can be insider and can corrupt up to $t - 1$ members of the target receiver entity. These security properties, along with the syntactic definition of threshold unsigncryption schemes, are detailed in Section 2.1. The security weaknesses of the above-mentioned threshold unsigncryption schemes were pointed out in [41, 85]. We showed in [41] (and we include this in Section 2.2 of this chapter, for completeness) that even generic constructions of threshold unsigncryption schemes, obtained by combining a fully secure standard signature scheme and a fully secure threshold decryption scheme, do not

achieve the maximum level of security. This is in contrast to what happens in the traditional scenario of signcryption, with a single receiver entity.

For this reason, one of the main goals in this area of threshold unsigncryption is to design new threshold unsigncryption schemes which are provably secure in the desired security model. The first two such schemes are very recent: one scheme was proposed in [85] in an identity-based scenario. We propose and analyze in Section 2.3 of this chapter a scheme which works in the traditional PKI setting. This scheme is a slightly modified version of the one, we published in [41]. Both our scheme [41] and the scheme in [85] are proved secure in the random oracle model.

To overcome the drawback with the random oracle model, we propose and analyze in Section 2.4 a new threshold unsigncryption scheme. It was published in [44] and is the first one in the literature to achieve, in the standard model, the required security properties. The design of this second scheme is quite modular: it employs two signature schemes and the ideas by Canetti-Halevi-Katz to achieve CCA security from identity-based selectively secure encryption [21].

The two schemes that we present in this chapter, in Sections 2.3 and 2.4, have an additional property which may be of independent interest: the unsigncryption protocol of the schemes can be split into two parts. The first part, verifying the validity and the authorship of the ciphertext, can be done by anyone, because the required inputs are the ciphertext and the public key of the sender. The second part, decrypting the (valid) ciphertext, can be done without using the public key of the sender. To the best of our knowledge, these are the first fully secure signcryption schemes in the literature that enjoy this property, considering both individual and threshold (un)signcryption. This 'splitting' property seems to be very promising for applications requiring authentication and confidentiality, but also some level of anonymity or privacy in some of their phases. As an illustrative example, we explain in Section 2.5.2 the case of an electronic auction system.

Finally, two different signature schemes with distributed verification are described in Section 2.6 for threshold access structures. These schemes are motivated on the signcryption schemes defined in Sections 2.3 and 2.4. Both signature schemes are IND-CCA secure, one in the random oracle model and the other in the standard model.

2.1 Modeling Threshold Unsigncryption Schemes

In a signcryption scheme, a user A sends a message to an intended receiver B, in a confidential and authenticated way: only B can obtain the original message, and B is convinced that the message comes from A. In a scenario where the role of B is distributed among a set of users, the cooperation of some authorized subset of users will be necessary to perform the unsigncryption phase. Each user in the set

B will have a share of the secret information of B, and will use it to perform his part of the unsigncryption process. In this chapter we will focus on threshold families of authorized subsets: the cooperation of at least t users in B will be necessary to successfully run the unsigncryption protocol. Both our formal definitions and our schemes can be extended to more general families of authorized subsets, by replacing threshold secret sharing techniques (i.e. Shamir's scheme [82]) with more general linear secret sharing schemes.

2.1.1 Syntactic Definition

A signcryption scheme with threshold unsigncryption $\Sigma = (\Sigma.\mathsf{St}, \Sigma.\mathsf{KG}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Uns})$ consists of four probabilistic polynomial-time algorithms:

- The randomized *setup* algorithm $\Sigma.\mathsf{St}$ takes a security parameter $\lambda \in \mathbb{N}$ and outputs some public parameters params that will be common to all the users in the system: the mathematical groups, generators, hash functions, etc. We write $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ to denote an execution of this algorithm.

- The *key generation* algorithm $\Sigma.\mathsf{KG}$ is different for an individual sender A than for a collective B of receivers. A single user A will get a pair $(\mathsf{sk}_A, \mathsf{pk}_A)$ of secret and public keys. In contrast, for a collective $B = \{B_1, \ldots, B_n\}$ of n users, the output will be a single public key pk_B for the group, and then a threshold secret share $\mathsf{sk}_{B,j}$ for each user B_j, for $j = 1, \ldots, n$, and for some threshold t such that $1 \leq t \leq n$. The key generation process for the collective B can be either run by a trusted third party, or by the users in B themselves. We will write $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, A, \text{'single'})$ and $(\{\mathsf{sk}_{B,j}\}_{1 \leq j \leq n}, \mathsf{pk}_B) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, B, n, t, \text{'collective'})$ to refer to these two key generation protocols.

- The *signcryption* algorithm $\Sigma.\mathsf{Sign}$ takes as input params, a message M, the public key pk_B of the intended receiver group B, and the secret key sk_A of the sender. The output is a ciphertext C. We denote an execution of this algorithm as $C \leftarrow \Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A)$.

- The *threshold unsigncryption* algorithm $\Sigma.\mathsf{Uns}$ is an interactive protocol run by some subset of users $B' \subset B$. The common inputs are params, a ciphertext C and the public key pk_A of the sender, whereas each user $B_j \in B'$ has as secret input his secret share $\mathsf{sk}_{B,j}$. The output is a message $\widetilde{M}$, which can eventually be the special symbol $\perp$, meaning that the ciphertext C is invalid. We write $\widetilde{M} \leftarrow \Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'})$ to refer to an execution of this protocol.

For correctness, the condition $\Sigma.\mathsf{Uns}(\mathsf{params}, \Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A), \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'}) = M$ is required, whenever B' contains at least t honest users and the values $\mathsf{params}, \mathsf{sk}_A, \mathsf{pk}_A, \{\mathsf{sk}_{B,j}\}_{1 \leq j \leq n}, \mathsf{pk}_B$ have been obtained by properly executing the protocols $\Sigma.\mathsf{St}$ and $\Sigma.\mathsf{KG}$.

A different property that can be required is that of *robustness*, which informally means that dishonest receivers in B who do not follow the threshold unsigncryption protocol correctly can be detected and discarded, without affecting the correct completion of the protocol.

2.1.2 Security Model

A correct signcryption scheme must satisfy the security properties that are required for encryption and signatures: confidentiality and unforgeability. In the threshold setting for unsigncryption, confidentiality must hold even if an attacker corrupts $t-1$ members of a collective of receivers. We consider a multi-user setting where an adversary is allowed to corrupt the maximum possible number of users (all except the target one), and where he can make both signcryption and unsigncryption queries for different users, messages and ciphertexts. In particular, unforgeability must hold even if the adversary knows the secret keys of all the possible collectives of receivers, and confidentiality must hold even if the adversary knows the secret keys of all the possible senders. In other words, we require *insider security*.

Note that we are considering only *static* adversaries, who choose the corrupted users at the beginning of the attack, in order to simplify the notation and thus allow a better understanding of the proposed schemes. In order to resist more powerful *adaptive* attacks, where the users may be corrupted at different stages of the system, our schemes should be combined with well-known techniques, as those in [19, 50, 63].

Unforgeability.

Unforgeability under chosen message attacks is the standard security notion for signature schemes and in general for any cryptographic primitive which pretends to provide some kind of authentication or non-repudiation. The idea is that an attacker who does not know the secret key of a user A and who can ask A for some valid signatures (or, in our case, signcryptions) for messages of his choice must not be able to produce a different valid signature (signcryption) on behalf of A. For a security parameter $\lambda \in \mathbb{N}$, this notion is formalized by describing the following game that an attacker $\mathcal{A}_{\mathsf{UNF}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{UNF}}$.
2. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target user $A^\star$. The challenger runs $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star}) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, A^\star, \text{`single'})$, keeps $\mathsf{sk}_{A^\star}$ private and gives $\mathsf{pk}_{A^\star}$ to $\mathcal{A}_{\mathsf{UNF}}$.

3. [**Queries**] The attacker $\mathcal{A}_{\mathsf{UNF}}$ can make adaptive queries to a signcryption oracle for sender $A^\star$: $\mathcal{A}_{\mathsf{UNF}}$ sends a tuple (M, pk_B) for some collective B of his choice, and obtains as answer $C \leftarrow \Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_{A^\star})$.

   Note that other kinds of queries (such as unsigncryption queries or signcryption queries for senders different from $A^\star$) make no sense because $\mathcal{A}_{\mathsf{UNF}}$ can reply such queries by himself.

4. [**Forgery**] Eventually, $\mathcal{A}_{\mathsf{UNF}}$ outputs a tuple $(\mathsf{pk}_{A^\star}, B^\star, \mathsf{pk}_{B^\star}, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star}, C^\star)$.

We say that $\mathcal{A}_{\mathsf{UNF}}$ wins the game if:

- the output of the protocol $\Sigma.\mathsf{Uns}(\mathsf{params}, C^\star, \mathsf{pk}_{A^\star}, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$ is a message $M^\star \neq \perp$,
- the tuple $(\mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star}, C^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ through a signcryption query.

The *advantage* of such an adversary $\mathcal{A}_{\mathsf{UNF}}$ in breaking the unforgeability of the signcryption scheme is defined as

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda) = \Pr[\mathcal{A}_{\mathsf{UNF}} \text{ wins}].$$

A signcryption scheme Σ with threshold unsigncryption is unforgeable if, for any polynomial time adversary $\mathcal{A}_{\mathsf{UNF}}$, the value $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$ is negligible with respect to the security parameter λ.

Indistinguishability.

The confidentiality requirement for a signcryption scheme Σ with (t, n)-threshold unsigncryption (i.e. the fact that a signcryption on the message m addressed to B leaks no information on m to an attacker who only knows $t-1$ secret shares of sk_B) is ensured if the scheme enjoys the property of *indistinguishability under chosen ciphertext attacks* ($\mathsf{IND\text{-}CCA}$ security, for short). For a security parameter $\lambda \in \mathbb{N}$, this property is defined by considering the following game that an attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

2. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses a target set $B^\star$ of n users and a subset $\widetilde{B} \subset B^\star$ of $t-1$ users, to be corrupted. The challenger runs $(\{\mathsf{sk}_{B^\star,j}\}_{1 \leq j \leq n}, \mathsf{pk}_{B^\star}) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, B^\star, n, t, \text{'collective'})$ and gives to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the values $\mathsf{pk}_{B^\star}$ and $\{\mathsf{sk}_{B^\star,j}\}_{B_j \in \widetilde{B}}$. Without loss of generality, we can assume $B^\star = \{B_1, \ldots, B_n\}$ and $\widetilde{B} = \{B_1, \ldots, B_{t-1}\}$.

   Note that we are considering only *static* adversaries who choose the subset $\widetilde{B}$ of corrupted users at the beginning of the attack.

3. [**Queries**] $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make adaptive queries to a threshold unsigncryption oracle for the target set $B^\star$: $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ sends a tuple (pk_A, C) for some public key pk_A of his choice. The challenger runs $\widetilde{M} \leftarrow \Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$. The attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ must be given all the information that is broadcast during the execution of this protocol $\Sigma.\mathsf{Uns}$, including $\widetilde{M}$.

   Other kinds of queries (such as unsigncryption queries for other collectives $B \neq B^\star$ or signcryption queries) make no sense because $\mathcal{A}_{\mathsf{UNF}}$ can reply such queries by himself.

4. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses two messages M_0, M_1 of the same length, and a sender $A^\star$ along with $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star})$.

5. [**Challenge**] The challenger picks a random bit $d \in \{0, 1\}$, runs $C^\star \leftarrow \Sigma.\mathsf{Sign}(\mathsf{params}, M_d, \mathsf{pk}_{B^\star}, \mathsf{sk}_{A^\star})$ and gives $C^\star$ to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

6. Step 3 is repeated, with the restriction that the tuple $(\mathsf{pk}_{A^\star}, C^\star, B^\star)$ cannot be queried to the threshold unsigncryption oracle.

7. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a bit d' as his guess of the bit d.

The advantage of such a (static) adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ in breaking the IND-CCA security of the signcryption scheme is defined as

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) = \left| \Pr[d' = d] - \frac{1}{2} \right|.$$

A signcryption scheme Σ with (t, n)-threshold unsigncryption is IND-CCA secure if $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ is negligible with respect to the security parameter λ, for any polynomial time (static) adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

2.2 Existing Threshold Unsigncryption Schemes Are Not Secure

There are very few papers proposing explicit signcryption schemes with threshold unsigncryption. We are only aware of two proposals [54, 92] in the traditional PKI setting, and three proposals [59, 60, 90] in the identity-based setting.

It turns out that none of these schemes achieves the full level of security described in the previous section. The security weakness is always related to the indistinguishability property. For the two schemes in the PKI setting (which do not contain any formal security definitions or analysis), simple IND-CCA attacks can be mounted without assuming multiple users or insider attackers. The schemes proposed for the

identity-based scenario are analyzed more formally, but IND-CCA attacks against them exist anyway. Specifically, the scheme described in [90] has the same security problems than the two schemes in the PKI setting: the verification step is performed at the end of the protocol, once all the receivers have broadcasted their partial decryption shares. This means that an attacker can take the challenge ciphertext $C^\star$ and modify only the "signature" part of it, obtaining an invalid ciphertext $C \neq C^\star$ that will be queried to the threshold unsigncryption oracle. As answer, the attacker will obtain the final value $\perp$, but also the partial decryption values broadcast by all the receivers, which will allow the attacker to decrypt the (valid) challenge ciphertext.

The scheme in [59] does not resist insider attacks: knowing the secret key of the sender $A^\star$, one can immediately obtain the plaintext from the challenge ciphertext. Finally, the scheme presented in [60] is also insecure against insider attacks. This scheme together with such an attack can be found in Appendix of [41] as an illustrative example.

2.2.1 What about Generic Constructions?

Since the existing threshold unsigncryption schemes are not fully secure, one could wonder if such fully secure schemes actually exist. The first attempt could / should be to think of possible generic constructions like the threshold versions of the well-known approaches Sign_then_Encrypt and Encrypt_then_Sign, that have been deeply analyzed in [1] for the case of ordinary signcryption. There, it is proved that both generic constructions achieve full security (against insider attackers in a multi-user setting) if the underlying signature and encryption schemes have full security. Thus, one could expect that the same happens in the scenario with threshold unsigncryption. But unfortunately this is not the case, as we argue below.

Let $\Omega = (\Omega.\mathsf{KG}, \Omega.\mathsf{Sign}, \Omega.\mathsf{Vfy})$ be a signature scheme, and $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{ThrDec})$ be a public encryption scheme with threshold decryption. For the keys of the generic signcryption schemes with threshold unsigncryption, an ndividual sender will run $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Omega.\mathsf{KG}$ and a collective of receivers B will run $(\{\mathsf{sk}_{B,j}\}_{1\leq j\leq n}, \mathsf{pk}_B) \leftarrow \Pi.\mathsf{KG}$.

Let us consider for example the ThresholdEncrypt_then_Sign approach. To signcrypt a message m for the collective B, a sender A first computes $c \leftarrow \Pi.\mathsf{Enc}(\mathsf{pk}_B, m||\mathsf{pk}_A)$ and then signs $c||\mathsf{pk}_B$ to obtain $\omega \leftarrow \Omega.\mathsf{Sign}(\mathsf{sk}_A, c||\mathsf{pk}_B)$. The final ciphertext is $C = (c, \omega)$. To unsigncrypt such a ciphertext, members of B first verify the correctness of signature ω by running $\Omega.\mathsf{Vfy}(\mathsf{pk}_A, c||\mathsf{pk}_B, \omega)$. If the signature is not correct, the symbol $\perp$ is output. Otherwise, a subset $\tilde{B} \subset B$ of at least t members of B run $\Pi.\mathsf{ThrDec}(\{\mathsf{sk}_{B,j}\}_{B_j\in\tilde{B}}, c)$ to recover the message $m||\mathsf{pk}_A$. If the public key pk_A corresponds with that of the sender A, then m is the output of the protocol. If not, the output is $\perp$.

The IND-CCA security of this generic construction can be broken by an insider attacker $\mathcal{A}_{\text{IND-CCA}}$ in a multi-user scenario. $\mathcal{A}_{\text{IND-CCA}}$ receives a challenge ciphertext $C^\star = (c^\star, \omega^\star)$ for a challenge sender $A^\star$ and a challenge collective $B^\star$ of receivers. After that, $\mathcal{A}_{\text{IND-CCA}}$ can generate keys $(\mathsf{sk}_A, \mathsf{pk}_A)$ for another user $A \neq A^\star$, compute a valid signature ω for $c^\star||\mathsf{pk}_{B^\star}$ using sk_A, and send $C = (c^\star, \omega)$ as a threshold unsigncryption query for sender A and collective $B^\star$ of receivers. As answer to this query, since the signature ω is valid, $\mathcal{A}_{\text{IND-CCA}}$ must receive all the information that the members of $B^\star$ would broadcast in the execution of the threshold decryption of $c^\star$. Even if the final output of this query is $\perp$, because the public key pk_A does not match the public key $\mathsf{pk}_{A^\star}$ which is encrypted in $c^\star$, the attacker $\mathcal{A}_{\text{IND-CCA}}$ has obtained enough information to recover the whole plaintext encrypted in $c^\star$, and therefore succeeds in breaking the indistinguishability of the scheme. We stress that this same attack is valid against relaxed IND-CCA (see [22]), because the decryption of C (which is $\perp$) is different from the decryption of $C^\star$.

Regarding the Sign_then_ThresholdEncrypt approach, the attack is even simpler. Once $\mathcal{A}_{\text{IND-CCA}}$ gets a challenge ciphertext $C^\star = c^\star$ for $A^\star$ and $B^\star$, where $c^\star$ is an encryption under Π of $(m, \omega^\star, \mathsf{pk}_{A^\star})$ and $\omega^\star$ is a signature on $m||\mathsf{pk}_{B^\star}$, all that $\mathcal{A}_{\text{IND-CCA}}$ has to do is to make an unsigncryption query for the tuple $(C^\star, \mathsf{pk}_A, \mathsf{pk}_{B^\star})$, where $A \neq A^\star$. Even if the output of the protocol is again $\perp$, the attacker $\mathcal{A}_{\text{IND-CCA}}$ gets all the partial information broadcast by the members of $B^\star$ in the execution of the threshold decryption of $c^\star$, which allows $\mathcal{A}_{\text{IND-CCA}}$ to directly obtain the plaintext m.

Note that we are affirming that generic constructions do not achieve strong security. That means, against IND-CCA adversaries who can ask adaptive threshold unsigncryption queries. If we remove this point in the indistinguishability game of page 24 then we should talk about weak security, which is verified by the generic constructions.

2.3 A First Threshold Unsigncryption Scheme

This section is dedicated to the description and analysis of our first new signcryption scheme with (t, n)-threshold unsigncryption, achieving full security in the random oracle model. Our approach has been to take a secure public key encryption scheme with threshold decryption and modify it in order to accommodate also the authentication process. In particular, we have considered the scheme TDH1 of Shoup and Gennaro [88]. The idea of that scheme, to encrypt a message m for a collective B with public key pk_B, is to first compute a hashed ElGamal encryption (R, c) of m. That is, assuming that we have fixed a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, along with a hash function H_0, the sender computes $R = g^r$ and $c = m \oplus H_0((pk_B)^r)$. After that, he adds to the ciphertext another element $\bar{g} \in \mathbb{G}$ and the value $\bar{R} = \bar{g}^r$, and finally a zero-knowledge proof that $\mathsf{DiscLog}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$. Members of B will

start the real decryption process only if the proof of knowledge is valid.

Our signcryption scheme follows the same principle, but the sender A will compute now a zero-knowledge proof that both $\mathsf{DiscLog}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$ hold and he knows sk_A such that $\mathsf{pk}_A = g^{\mathsf{sk}_A}$. We will prove that the resulting signcryption scheme (with threshold unsigncryption) enjoys the strong notions of unforgeability and indistinguishability. We consider for simplicity a scenario where the receivers follow the threshold unsigncryption protocol correctly. A simple modification of our scheme, by including appropriate non-interactive zero-knowledge proofs of the equality of two discrete logarithms, allows to provide robustness to the scheme against the action of malicious receivers. The protocols of the scheme are described below.

Setup: $\Sigma.\mathsf{St}(1^\lambda)$.
Given a security parameter λ, a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, such that q is λ bits long, is chosen. A length ℓ, which must be polynomial in λ, is defined for the maximum number of bits of the messages to be sent by the system. Three hash functions $H_0 : \{0,1\}^* \to \{0,1\}^\ell$, $H_1 : \{0,1\}^* \to \mathbb{G}$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ are chosen. The output of the protocol is $\mathsf{params} = (q, \mathbb{G}, g, H_0, \ell, H_1, H_2)$.

Key Generation: $\Sigma.\mathsf{KG}(\mathsf{params}, A, \text{'single'})$ and $\Sigma.\mathsf{KG}(\mathsf{params}, B, n, t, \text{'collective'})$.
For an individual user A, the secret key sk_A is a random element in $\mathbb{Z}_q^*$, whereas the corresponding public key is $\mathsf{pk}_A = g^{\mathsf{sk}_A}$. The public output of this protocol is pk_A, and the secret output that is privately stored by A is sk_A.

For a collective $B = \{B_1, \ldots, B_n\}$ of n users, the common public key is computed as $\mathsf{pk}_B = g^{\mathsf{sk}_B}$ for some random value $\mathsf{sk}_B \in \mathbb{Z}_q^*$ that will remain unknown to the members of B. Each user $B_j \in B$ will receive a (t,n)-threshold share $\mathsf{sk}_{B,j}$ of sk_B, computed by using Shamir's secret sharing scheme [82]. This means that, for every subset $B' \subset B$ containing exactly t users, there exist values $\lambda_j^{B'} \in \mathbb{Z}_q^*$ such that $\mathsf{sk}_B = \sum_{B_j \in B'} \lambda_j^{B'} \mathsf{sk}_{B,j}$. The public output of this protocol is pk_B, whereas each user $B_j \in B$ receives a secret output $\mathsf{sk}_{B,j}$.

The key generation process for a collective B can be performed by a trusted dealer, or by the members of B themselves, by using some well-known techniques [36].

Both solutions permit that the values $D_{B,j} = g^{\mathsf{sk}_{B,j}}$ are made public, for $j = 1, \ldots, n$. These values would be necessary to provide robustness to the threshold unsigncryption process.

We assume that both pk_A and pk_B include descriptions of the identities of A and members of B.

Signcryption: $\Sigma.\mathsf{Sign}(\mathsf{params}, m, \mathsf{pk}_B, \mathsf{sk}_A)$.

1. Choose at random $r \in \mathbb{Z}_q^*$ and compute $R = g^r$.
2. Compute $k = H_0(R, \mathsf{pk}_B, (\mathsf{pk}_B)^r)$ and $c = m \oplus k$.

3. Choose at random $\alpha_1, \alpha_2 \in \mathbb{Z}_q^*$ and compute $Y_1 = g^{\alpha_1}$ and $Y_2 = g^{\alpha_2}$.
4. Compute $\bar{g} = H_1(c, R, Y_1, Y_2, \mathsf{pk}_A, \mathsf{pk}_B) \in \mathbb{G}$, and then $\bar{R} = \bar{g}^r$ and $\bar{Y}_1 = \bar{g}^{\alpha_1}$.
5. Compute $h = H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B)$.
6. Compute $s_1 = \alpha_1 - h \cdot r \mod q$.
7. Compute $s_2 = \alpha_2 - h \cdot \mathsf{sk}_A \mod q$.
8. Return the signcryption $C = (c, R, \bar{R}, h, s_1, s_2)$.

Threshold Unsigncryption: $\Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'})$.
Let $B' \subset B$ be a subset of users in B that want to cooperate to unsigncrypt a signcryption $C = (c, R, \bar{R}, h, s_1, s_2)$. They proceed as follows.

1. Each $B_j \in B'$ computes $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \mathsf{pk}_A, \mathsf{pk}_B)$.
2. Each $B_j \in B'$ checks if the following equality holds:

$$h \;=\; H_2(c, R, \bar{g}, \bar{R}, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \bar{g}^{s_1} \cdot \bar{R}^h, \mathsf{pk}_A, \mathsf{pk}_B)$$

3. If the equality does not hold, B_j broadcasts $(j, \perp)$.
4. Otherwise, $B_j \in B'$ broadcasts the value $T_j = R^{\mathsf{sk}_{B,j}}$.

   If robustness was required, then B_j should also provide a non-interactive zero-knowledge proof that $\mathsf{DiscLog}_g(D_{B,j}) = \mathsf{DiscLog}_R(T_j)$.
5. If there are not t valid shares, then stop and output $\perp$. From t valid values T_j, different from $(j, \perp)$, recover the value R^{sk_B} by interpolation in the exponent: $R^{\mathsf{sk}_B} = \prod_{B_j \in B'} T_j^{\lambda_j^{B'}}$, where $\lambda_j^{B'} \in \mathbb{Z}_q$ are the Lagrange interpolation coefficients.
6. Compute $k = H_0(R, \mathsf{pk}_B, R^{\mathsf{sk}_B})$.
7. Return the value $m = c \oplus k$.

2.3.1 Security Analysis

The unforgeability of our signcryption scheme will be reduced to the hardness of the *Discrete Logarithm (DL) problem*, whereas its indistinguishability will be reduced to the hardness of the *Computational Diffie-Hellman (CDH) problem*. See Subsection 1.2.1 for a description of these problems.

Unforgeability.

We are going to prove that our scheme enjoys unforgeability as long as the Discrete Logarithm problem is hard to solve. The proof is in the random oracle model for the hash function H_2.

Theorem 2.3.1 *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{UNF}$ against the unforgeability of the new signcryption scheme, in the random oracle model, there exists a solver $\mathcal{A}^{DL}$ of the Discrete Logarithm problem such that*

$$Adv_{\mathcal{A}^{DL}}(\lambda) \;\geq\; \mathcal{O}\left(Adv_{\mathcal{A}_{UNF}}(\lambda)^2\right).$$

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{UNF}}$ that has advantage $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$ in breaking the unforgeability of our scheme, and assuming that the hash function H_2 behaves as a random oracle, we are going to construct an algorithm $\mathcal{A}^{DL}$ that solves the Discrete Logarithm problem in $\mathbb{G}$. Below figure shows how $\mathcal{A}^{DL}$ executes the adversary $\mathcal{A}_{\mathsf{UNF}}$ as a subroutine to solve an instance of this problem.

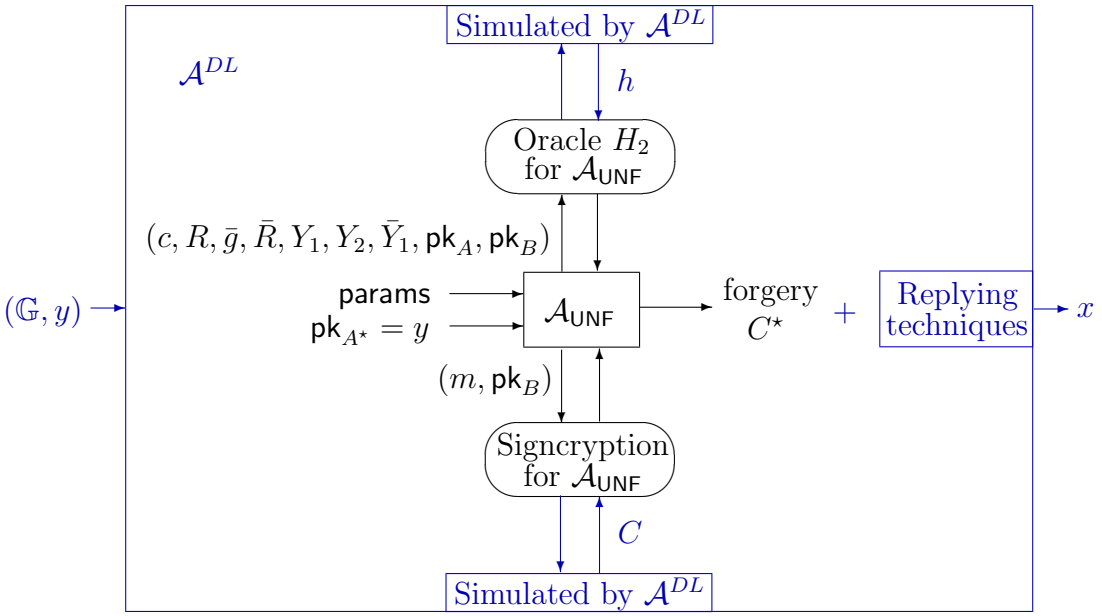

Figure 2.1: $\mathcal{A}^{DL}$ simulates the environment of $\mathcal{A}_{\mathsf{UNF}}$.

Let $(\mathbb{G}, y)$ be the instance of the Discrete Logarithm problem in $\mathbb{G} = \langle g \rangle$ that $\mathcal{A}^{DL}$ receives. The goal of $\mathcal{A}^{DL}$ is to find the integer $x \in \mathbb{Z}_q$ such that $y = g^x$. The algorithm $\mathcal{A}^{DL}$ initializes the attacker $\mathcal{A}_{\mathsf{UNF}}$ by giving $\mathsf{params} = (q, \mathbb{G}, g, H_0, \ell, H_1, H_2)$ to him. Here the hash functions $H_0 : \{0,1\}^* \to \{0,1\}^\ell$ and $H_1 : \{0,1\}^* \to \mathbb{G}$ are

arbitrarily chosen by $\mathcal{A}^{DL}$. However, H_2 is modeled as a random oracle and so $\mathcal{A}^{DL}$ will maintain a table TAB_2 to answer the hash queries from $\mathcal{A}_{\mathsf{UNF}}$.

Key generation. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target sender $A^\star$ and requests the execution of the key generation protocol for this user. $\mathcal{A}^{DL}$ defines the public key of $A^\star$ as $\mathsf{pk}_{A^\star} = y$ and sends it to $\mathcal{A}_{\mathsf{UNF}}$. Note that the corresponding secret key $\mathsf{sk}_{A^\star}$, which is unknown to $\mathcal{A}^{DL}$, is precisely the solution to the given instance of the Discrete Logarithm problem.

Hash queries. Since H_2 is assumed to behave as a random function, $\mathcal{A}_{\mathsf{UNF}}$ can make queries $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B)$ to the random oracle model for H_2. $\mathcal{A}^{DL}$ maintains a table TAB_2 to reply to these queries. TAB_2 contains two columns, one for the inputs and one for the corresponding outputs h of H_2. To reply the query $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B)$, the algorithm $\mathcal{A}^{DL}$ checks if this input is already in TAB_2. If so, the matching output h is answered. If not, a random value $h \in \mathbb{Z}_q$ is chosen and answered to $\mathcal{A}_{\mathsf{UNF}}$, and the entry $H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_A, \mathsf{pk}_B) = h$ is added to TAB_2.

Signcryption queries. $\mathcal{A}_{\mathsf{UNF}}$ can make signcryption queries for the sender $A^\star$, for pairs (m, pk_B) of his choice, where m is a message and B is a collective of receivers with public key pk_B. To reply to such queries, $\mathcal{A}^{DL}$ chooses at random a value $r \in \mathbb{Z}_q^*$ and computes $R = g^r$, $k = H_0(R, \mathsf{pk}_B, (\mathsf{pk}_B)^r)$ and $c = m \oplus k$. Then, $\mathcal{A}^{DL}$ must simulate a valid proof of knowledge to complete the rest of the ciphertext. To do this, $\mathcal{A}^{DL}$ acts as follows:

1. Choose at random $h, s_1, s_2 \in \mathbb{Z}_q$ and compute the values $Y_1 = g^{s_1} \cdot R^h$ and $Y_2 = g^{s_2} \cdot (\mathsf{pk}_{A^\star})^h$.

2. Compute $\bar{g} = H_1(c, R, Y_1, Y_2, \mathsf{pk}_{A^\star}, \mathsf{pk}_B)$, and then the values $\bar{R} = \bar{g}^r$ and $\bar{Y}_1 = \bar{g}^{s_1} \cdot \bar{R}^h$.

3. If the input $(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_{A^\star}, \mathsf{pk}_B)$ is already in TAB_2 (which happens with negligible probability), go back to Step 1.

4. Otherwise, 'falsely' add the relation $h = H_2(c, R, \bar{g}, \bar{R}, Y_1, Y_2, \bar{Y}_1, \mathsf{pk}_{A^\star}, \mathsf{pk}_B)$ to TAB_2.

The final signcryption that $\mathcal{A}^{DL}$ sends to $\mathcal{A}_{\mathsf{UNF}}$ is $C = (c, R, \bar{R}, h, s_1, s_2)$.

Forgery. At some point, $\mathcal{A}_{\mathsf{UNF}}$ outputs a successful forgery; that is, a public key $\mathsf{pk}_{B^\star}$ and a signcryption $C^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$ such that:

- the protocol $\Sigma.\mathsf{Uns}(\mathsf{params}, C^\star, \mathsf{pk}_{A^\star}, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$ outputs $m^\star \neq \perp$,
- $(\mathsf{pk}_{A^\star}, m^\star, \mathsf{pk}_{B^\star}, C^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ during a signcryption query.

Since the forgery is valid, we must have $h^\star = H_2(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$, where $Y_1^\star = g^{s_1^\star} \cdot (R^\star)^{h^\star}$, $Y_2^\star = g^{s_2^\star} \cdot (\mathsf{pk}_{A^\star})^{h^\star}$ and $\bar{Y}_1^\star = (\bar{g}^\star)^{s_1^\star} \cdot (\bar{R}^\star)^{h^\star}$.

Furthermore, since the forgery is different from the ciphertexts obtained during the signcryption queries, we can be sure that the input $\mathsf{query}^\star = (c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ for H_2 has not been 'falsely' added by $\mathcal{A}^{DL}$ to TAB_2.

Replying the attack. Now the idea is to use the reply techniques introduced by Pointcheval and Stern in [76]. Without going into the details, $\mathcal{A}^{DL}$ will repeat the execution of the attacker $\mathcal{A}_{\mathsf{UNF}}$, with the same randomness but changing the values output by the random oracle H_2 from the query $\mathsf{query}^\star$ on.

With non-negligible probability (quadratic on the probability $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$ of the first successful forgery), the whole process run by $\mathcal{A}^{DL}$ would lead to two different successful forgeries $C^\star$ and $C'^\star$, for the same values of $c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star$, $\mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star}$ (the input values for H_2), but with different H_2 outputs $h^\star \neq h'^\star$, and therefore (possibly different) values $s_1^\star, s_2^\star, s_1'^\star, s_2'^\star$.

We thus have

$$g^{s_2^\star} \cdot (\mathsf{pk}_{A^\star})^{h^\star} = Y_2^\star = g^{s_2'^\star} \cdot (\mathsf{pk}_{A^\star})^{h'^\star},$$

which leads to the relation $y = \mathsf{pk}_{A^\star} = \left(g^{s_2^\star - s_2'^\star}\right)^{1/(h'^\star - h^\star)}$.

Summing up, $\mathcal{A}^{DL}$ can output the value $x = \frac{s_2^\star - s_2'^\star}{h'^\star - h^\star} \mod q$ as the solution to the given instance of the Discrete Logarithm problem. □

Indistinguishability.

We reduce the IND-CCA security of the scheme to the hardness of solving the CDH problem. The proof is in the random oracle model for the three hash functions H_0, H_1, H_2. The conclusion is that, under the Computational Diffie Hellman Assumption for our group $\mathbb{G} = \langle g \rangle$, the new signcryption scheme has IND-CCA security.

Theorem 2.3.2 *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ against the IND-CCA security of the new signcryption scheme, in the random oracle model, there exists a solver $\mathcal{A}^{CDH}$ of the Computational Diffie-Hellman problem such that*

$$\mathsf{Adv}_{\mathcal{A}^{CDH}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)/2.$$

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ that has advantage $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ in breaking the IND-CCA security of our scheme, and assuming that hash functions H_0, H_1, H_2 behave as random oracles, we are going to construct an algorithm $\mathcal{A}^{CDH}$ that solves the Computational Diffie-Hellman problem.

$\mathcal{A}^{CDH}$ receives as input $\mathbb{G}, g^a, g^b$, where $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order q. The goal of $\mathcal{A}^{CDH}$ is to compute g^{ab}. $\mathcal{A}^{CDH}$ initializes the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ by giving $\mathsf{params} = (q, \mathbb{G}, g, H_0, \ell, H_1, H_2)$ to him. Here the hash functions H_0, H_1 and H_2 will be modeled as random oracles; therefore, $\mathcal{A}^{CDH}$ will maintain three tables TAB_0, TAB_1 and TAB_2 to answer the hash queries from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Let $B^\star = \{B_1, \ldots, B_n\}$ be the target collective, and $\widetilde{B} = \{B_1, \ldots, B_{t-1}\} \subset B^\star$ be the subset of corrupted members of $B^\star$. The algorithm $\mathcal{A}^{CDH}$ defines the public key of $B^\star$ as $\mathsf{pk}_{B^\star} = g^b$. This means that $\mathsf{sk}_{B^\star}$ is implicitly defined as b. For the corrupted members of $B^\star$, the shares $\{\mathsf{sk}_{B^\star,j}\}_{B_j \in \widetilde{B}}$ are chosen randomly and independently in $\mathbb{Z}_q$. Using interpolation in the exponent, all the values $D_{B^\star,j} = g^{\mathsf{sk}_{B^\star,j}}$ can be computed, for all the members $B_j \in B^\star$, corrupted or not.

Hash queries. $\mathcal{A}^{CDH}$ creates and maintains three tables TAB_0, TAB_1 and TAB_2 to reply the hash queries from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$. All the hash queries are processed by $\mathcal{A}^{CDH}$ in the same way: given the input for a hash query, the algorithm $\mathcal{A}^{CDH}$ checks if there already exists an entry in the corresponding table for that input. If this is the case, the existing output is answered. If this is not the case, a new output is chosen at random and answered to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$, and the new relation between input and output is added to the corresponding table.

For the particular case of H_1 queries, the corresponding outputs $\bar{g}$ are chosen as random powers of g^b. That is, $\mathcal{A}^{CDH}$ chooses at random a fresh value $\beta \in \mathbb{Z}_q^*$ and computes the new output of H_1 as $\bar{g} = (g^b)^\beta$. The value β is stored as an additional value of the new entry in table TAB_1.

Whenever $\mathcal{A}^{CDH}$ receives a H_0 query whose two first elements are g^a and g^b, the third element of the query is added to a different output table $\mathsf{TAB}^\star$, which will be the final output of $\mathcal{A}^{CDH}$.

Unsigncryption queries. For an unsigncryption query (pk_A, C) sent for the target collective $B^\star$, where $C = (c, R, \bar{R}, h, s_1, s_2)$, the first thing to do is to check the validity of the zero-knowledge proof (h, s_1, s_2); that is, to check if $h = H_2(c, R, \bar{g}, \bar{R}, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \bar{g}^{s_1} \cdot \bar{R}^h, \mathsf{pk}_A, \mathsf{pk}_{B^\star})$, where $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \mathsf{pk}_A, \mathsf{pk}_{B^\star}) = (g^b)^\beta$, for some value β known by $\mathcal{A}^{CDH}$. If this equation does not hold, then the answer to the query is $\perp$.

Otherwise, $\mathcal{A}^{CDH}$ has to give to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the values $R^{\mathsf{sk}_{B^\star,j}}$, for all $B_j \in B^\star$. For the corrupted members B_j, $j = 1, \ldots, t-1$, such values can be easily computed by $\mathcal{A}^{CDH}$, because it knows $\mathsf{sk}_{B^\star,j}$. Note now that the value $R^{\mathsf{sk}_{B^\star}}$ can be computed by

$\mathcal{A}^{CDH}$ as $\bar{R}^{1/\beta}$. In effect, since the zero-knowledge proof is valid, this means that $\mathsf{DiscLog}_g(R) = \mathsf{DiscLog}_{\bar{g}}(\bar{R})$, where $\bar{g} = g^{b\beta}$, and so $R^{b\beta} = \bar{R}$. Now, knowing $R^{\mathsf{sk}_{B^\star}}$ and $R^{\mathsf{sk}_{B^\star,j}}$ for $j = 1, \ldots, t-1$, the algorithm $\mathcal{A}^{CDH}$ can compute the rest of values $R^{\mathsf{sk}_{B^\star,j}}$, for $j = t, t+1, \ldots, n$, by interpolation in the exponent. Once this is done, the rest of the unsigncryption process can be easily completed by $\mathcal{A}^{CDH}$, who obtains a message m and sends all this information to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Challenge. At some point, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs two messages m_0, m_1 of the same length, along with a key pair $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star})$ for a sender $A^\star$. To produce the challenge ciphertext $C^\star$, the algorithm $\mathcal{A}^{CDH}$ defines $R^\star = g^a$ and then chooses at random the values $c^\star \in \{0,1\}^\ell$, $h^\star, s_1^\star, s_2^\star \in \mathbb{Z}_q$ and $\beta^\star \in \mathbb{Z}_q^*$. After that, $\mathcal{A}^{CDH}$ defines $\bar{g}^\star = g^{\beta^\star}$, $\bar{R}^\star = (g^a)^{\beta^\star}$, $Y_1^\star = g^{s_1^\star} \cdot (R^\star)^{h^\star}$, $Y_2^\star = g^{s_2^\star} \cdot (\mathsf{pk}_{A^\star})^{h^\star}$ and $\bar{Y}_1^\star = \bar{g}^{s_1^\star} \cdot (\bar{R}^\star)^{h^\star}$.

If either the input $(c^\star, R^\star, Y_1^\star, Y_2^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ already exists in TAB_1, or the input $(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ already exists in TAB_2, the algorithm $\mathcal{A}^{CDH}$ goes back to choose at random other values for $c^\star, h^\star$, etc. Finally, the relation $\bar{g}^\star = H_1(c^\star, R^\star, Y_1^\star, Y_2^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ is added to TAB_1 and the relation $h^\star = H_2(c^\star, R^\star, \bar{g}^\star, \bar{R}^\star, Y_1^\star, Y_2^\star, \bar{Y}_1^\star, \mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star})$ is added to TAB_2. The challenge ciphertext that $\mathcal{A}^{CDH}$ sends to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ is $C^\star = (c^\star, R^\star, \bar{R}^\star, h^\star, s_1^\star, s_2^\star)$.

More unsigncryption queries. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make more hash and unsigncryption queries, which are answered exactly in the same way as described before the challenge phase. The only delicate point is that $\mathcal{A}^{CDH}$ could not answer to a valid unsigncryption query $C = (c, R, \bar{R}, h, s_1, s_2)$ for which the value of $\bar{g} = H_1(c, R, g^{s_1} \cdot R^h, g^{s_2} \cdot (\mathsf{pk}_A)^h, \mathsf{pk}_A, \mathsf{pk}_{B^\star}) = \bar{g}^\star$, because this value does not have the necessary form $(g^b)^\beta$. But this happens only if the two inputs of H_1, in both the challenge ciphertext and in this queried ciphertext, are the same. Since both zero-knowledge proofs are valid, we would also have that the value of $\bar{R}$ is equal in both cases, and therefore the values of $h, s_1, s_2, \mathsf{pk}_A$ would also be equal. The conclusion is that the unsigncryption query C would be exactly the challenge ciphertext, and this query is prohibited to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Final analysis. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a guess bit b'. We are assuming that $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ succeeds with probability significantly greater than $1/2$ (random guess). Since H_0 is assumed to behave as a random function, this can happen only if $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ has asked to the random oracle H_0 the input corresponding to the challenge $C^\star$. This input is (g^a, g^b, g^{ab}). Therefore, with non-negligible probability $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)/2$, the value g^{ab} is in the table $\mathsf{TAB}^\star$ constructed by $\mathcal{A}^{CDH}$, and therefore the output of $\mathcal{A}^{CDH}$ contains the correct answer for the given instance of the CDH problem. As the authors of [88] indicate, we could use the Diffie-Hellman self-corrector described

in [86] to transform this algorithm $\mathcal{A}^{CDH}$ into an algorithm that only outputs the single and correct solution to the CDH problem. □

2.4 A Scheme in the Standard Model

The security of the scheme in the previous section has been proved in the random oracle model, which is an heuristic model, not a realistic one. Therefore, schemes enjoying security in the standard model are much preferable. We design and analyze in this section the first signcryption scheme with (t, n)-threshold unsigncryption enjoying full security in the standard model.

The rationale for the design of this second scheme is the following one. Boneh, Boyen and Halevi showed in [12] how to design threshold decryption schemes with CCA security in the standard model, by adapting the strategy proposed by Canetti, Halevi and Katz in [21]. They propose to generate a key-pair $(\tilde{\mathsf{sk}}, \tilde{\mathsf{vk}})$ for some strongly secure one-time signature scheme to encrypt a message M. Then $\tilde{\mathsf{vk}}$ is used to derive an identity id, and the message M is encrypted for identity id, by using a selectively-secure identity-based encryption scheme such as the one presented in [11]. The resulting ciphertext $\tilde{C}$ is signed with $\tilde{\mathsf{sk}}$, leading to a signature $\tilde{\theta}$. Both $\tilde{\mathsf{vk}}$ and $\tilde{\theta}$ are added to $\tilde{C}$. The set of receivers share the master secret key of the identity-based encryption scheme. To decrypt, they first verify that the signature $\tilde{\theta}$ on $\tilde{C}$ is correct under key $\tilde{\mathsf{vk}}$; if this is the case, they can cooperate to derive the secret key for identity id and then decrypt $\tilde{C}$ to recover the plaintext M.

To implement the primitive of signcryption with threshold unsigncryption, the idea is that the sender A signs the message $\tilde{C}||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}}$ with a strongly secure signature scheme, obtaining θ, and then the (one-time) signature $\tilde{\theta}$ is computed on $\tilde{C}||\mathsf{pk}_A||\mathsf{pk}_B||\theta$. The final signcryption is $C = (\tilde{C}, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$. With this technique, the receivers will be convinced of the authorship of sender A because even insider attacks can be prevented.

Although a more generic construction could have been described by using in a black-box way the primitives of (one-time) signature schemes and identity-based encryption with threshold key generation, it turns out that the only realization of the later primitive in the standard model is the specific scheme in [12], using bilinear pairings. For this reason, and for the sake of clarity in the presentation, it has been decided to describe the new signcryption scheme directly instantiated with the pairing-based scheme in [12]. The protocols of the scheme are detailed below.

Setup: $\Sigma.\mathsf{St}(1^\lambda)$.
Given $\lambda \in \mathbb{N}$, a cyclic bilinear group $\mathbb{G} = \langle g \rangle$ of prime order p, such that p is λ bits long, is chosen. This means that there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ for some

group $\mathbb{G}_T$. Let $H : \{0,1\}^* \to \mathbb{Z}_p^*$ be a collision-resistant hash function. Two more generators $h, g_2 \in \mathbb{G}$ are randomly selected.

Let $\Theta = (\Theta.\mathsf{KG}, \Theta.\mathsf{Sign}, \Theta.\mathsf{Vfy})$ be a strongly unforgeable signature scheme, and let $\tilde{\Theta} = (\tilde{\Theta}.\mathsf{KG}, \tilde{\Theta}.\mathsf{Sign}, \tilde{\Theta}.\mathsf{Vfy})$ be a strongly secure one-time signature scheme. Note that we could take $\tilde{\Theta} = \Theta$.

The output of the protocol is $\mathsf{params} = (p, \mathbb{G}, g, \mathbb{G}_T, e, H, h, g_2, \Theta, \tilde{\Theta})$.

Key Generation: $\Sigma.\mathsf{KG}(\mathsf{params}, A, \text{'single'})$ and $\Sigma.\mathsf{KG}(\mathsf{params}, B, n, t, \text{'collective'})$. For an individual user A, the key generation protocol of the signature scheme Θ is executed, and the resulting signing and verification keys are defined as the secret and public keys for user A. That is, $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Theta.\mathsf{KG}(1^\lambda)$.

For a collective $B = \{B_1, \ldots, B_n\}$ of n users, the common public key is computed as $\mathsf{pk}_B = g^{\alpha_B}$ for some random value $\alpha_B \in \mathbb{Z}_p^*$ that will remain unknown to the members of B. This value α_B is distributed in shares $\{\alpha_{B,j}\}_{B_j \in B}$ through Shamir's (t,n)-threshold secret sharing scheme [82]. In particular, for every subset $B' \subset B$ containing at least t users, there exist values $\lambda_j^{B'} \in \mathbb{Z}_q^*$ such that $\alpha_B = \sum_{B_j \in B'} \lambda_j^{B'} \alpha_{B,j}$. The public output of this protocol is pk_B, whereas each user $B_j \in B$ privately receives and stores his share $\mathsf{sk}_{B,j} = g_2^{\alpha_{B,j}}$ of the secret key $\mathsf{sk}_B = g_2^{\alpha_B}$. Again, the key generation process for a collective B can be performed by a trusted dealer, or by the members of B themselves, by using the techniques in [36].

Both solutions allow the publication of the values $D_{B,j} = g^{\alpha_{B,j}}$, for $j = 1, \ldots, n$. These values would be necessary if robustness of the threshold unsigncryption process was required.

Signcryption: $\Sigma.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A)$, where $M \in \mathbb{G}_T$.

1. Run $(\tilde{\mathsf{sk}}, \tilde{\mathsf{vk}}) \leftarrow \tilde{\Theta}.\mathsf{KG}(1^\lambda)$ to obtain an ephemeral pair of signing and verification keys for the one-time signature scheme $\tilde{\Theta}$.
2. Derive $\mathsf{id} = H(\tilde{\mathsf{vk}})$, which is an element in $\mathbb{Z}_p^*$.
3. Choose at random $s \in \mathbb{Z}_p^*$.
4. Compute $C_1 = g^s$, $C_2 = M \cdot e(\mathsf{pk}_B, g_2)^s$ and $C_3 = \left(\mathsf{pk}_B^{\mathsf{id}} \cdot h\right)^s$.
5. Use sk_A to compute a signature θ on the message $C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}}$ for the scheme Θ. That is, $\theta \leftarrow \Theta.\mathsf{Sign}(\mathsf{sk}_A,\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}})$.
6. Use the ephemeral secret key $\tilde{\mathsf{sk}}$ to compute a signature $\tilde{\theta}$ on the message $C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta$ for the scheme $\tilde{\Theta}$. That is, $\tilde{\theta} \leftarrow \tilde{\Theta}.\mathsf{Sign}(\tilde{\mathsf{sk}},\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta)$.
7. Return the signcryption $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$.

Threshold Unsigncryption: $\Sigma.\mathsf{Uns}(\mathsf{params}, C, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'})$.
Let $B' \subset B$ be a subset of users in B that want to cooperate to unsigncrypt a signcryption $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$ sent by user A. They proceed as follows.

1. Each $B_j \in B'$ runs $\tilde{\Theta}.\mathsf{Vfy}(\tilde{\mathsf{vk}},\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta\ ,\tilde{\theta})$. If the output is 0, he broadcasts $(j, \perp)$.

2. Each $B_j \in B'$ runs $\Theta.\mathsf{Vfy}(\mathsf{pk}_A,\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\tilde{\mathsf{vk}}\ ,\theta)$. If the output is 0, he broadcasts $(j, \perp)$.

3. Each $B_j \in B'$ derives $\mathsf{id} = H(\tilde{\mathsf{vk}})$ and checks if $e(C_3, g) = e(\mathsf{pk}_B^{\mathsf{id}} \cdot h, C_1)$. If this equality does not hold, B_j broadcasts $(j, \perp)$.

4. Each $B_j \in B'$ chooses $r_j \in \mathbb{Z}_p$ at random and broadcasts the tuple $(j, \omega_{0,j}, \omega_{1,j})$, where
$$\omega_{0,j} = \mathsf{sk}_{B,j} \cdot (\mathsf{pk}_B^{\mathsf{id}} \cdot h)^{r_j} \quad \text{and} \quad \omega_{1,j} = g^{r_j}$$
If robustness was required, the correctness of this tuple could be publicly verified by checking if $e(\omega_{0,j}, g) = e(D_{B,j}, g_2) \cdot e(\mathsf{pk}_B^{\mathsf{id}} \cdot h, \omega_{1,j})$.

5. If there are not t valid shares, then stop and output $\perp$. From t valid tuples $\{(j, \omega_{0,j}, \omega_{1,j})\}_{B_j \in B'}$, one can consider the Lagrange interpolation coefficients $\lambda_j^{B'} \in \mathbb{Z}_q$ such that $\mathsf{sk}_B = \prod_{B_j \in B'} \mathsf{sk}_{B,j}^{\lambda_j^{B'}}$.

6. Compute $\omega_0 = \prod_{B_j \in B'} \omega_{0,j}^{\lambda_j^{B'}}$ and $\omega_1 = \prod_{B_j \in B'} \omega_{1,j}^{\lambda_j^{B'}}$.
[Note that $\omega_0 = \mathsf{sk}_B \cdot (\mathsf{pk}_B^{\mathsf{id}} \cdot h)^{\tilde{r}}$ and $\omega_1 = g^{\tilde{r}}$, being $\tilde{r} = \sum_{B_j \in B'} \lambda_j^{B'} r_j$.]

7. Return the message $M = C_2 \cdot \frac{e(C_3, \omega_1)}{e(C_1, \omega_0)}$.

It is important to point out that the threshold unsigncryption protocol is non-interactive, in the sense that each receiver B_j can do his secret part of the unsigncryption task independently of the other receivers.

2.4.1 Security Analysis

The unforgeability of this signcryption scheme will be reduced to the strongly unforgeability of the underlying signature schemes used in the scheme, whereas its indistinguishability will be reduced to the hardness of the *Decisional Bilinear Diffie-Hellman (DBDH) problem*. See Subsections 1.4.2 and 1.2.1 for a description of these problems.

Unforgeability.

We are going to prove in the following that the scheme Σ enjoys unforgeability as long as the signature schemes Θ and $\tilde{\Theta}$ are strongly unforgeable.

Theorem 2.4.1 *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{UNF}$ against the unforgeability of the new signcryption scheme, making Q signcryption queries, there exists an attacker $\mathcal{F}_\Theta$ against Θ or an attacker $\mathcal{F}_{\tilde{\Theta}}$ against $\tilde{\Theta}$, such that*

$$\mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda) + Q \cdot \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_{UNF}}(\lambda).$$

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{UNF}}$ against the unforgeability of the scheme, we are going to construct a forger $\mathcal{F}_\Theta$ against the signature scheme Θ.

$\mathcal{F}_\Theta$ receives as input a verification key vk obtained from an execution $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Theta.\mathsf{KG}(1^\lambda)$, and has access to a signing oracle $\Theta.\mathsf{Sign}(\mathsf{sk}, \cdot)$ for messages of its choice. The algorithm $\mathcal{F}_\Theta$ runs the setup protocol $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and initializes the attacker $\mathcal{A}_{\mathsf{UNF}}$ by giving params to it.

Key generation. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target sender $A^\star$ and requests the execution of the key generation protocol for this user. $\mathcal{F}_\Theta$ defines the public key of $A^\star$ as $\mathsf{pk}_{A^\star} = \mathsf{vk}$ and sends it to $\mathcal{A}_{\mathsf{UNF}}$.

Signcryption queries. $\mathcal{A}_{\mathsf{UNF}}$ can make signcryption queries for the sender $A^\star$, for pairs (M_i, pk_{B_i}) of its choice, where M_i is a message and B_i is a collective of receivers with public key pk_{B_i}. To reply such queries, $\mathcal{F}_\Theta$ runs steps 1-4 of the signcryption protocol $\Sigma.\mathsf{Sign}(\mathsf{params}, M_i, \mathsf{pk}_{B_i}, \mathsf{sk}_{A^\star})$, obtaining consistent values $\tilde{\mathsf{sk}}_i, \tilde{\mathsf{vk}}_i, C_{1,i}, C_{2,i}, C_{3,i}$. After that, $\mathcal{F}_\Theta$ queries its signing oracle with message $m_i = C_{1,i}||C_{2,i}||C_{3,i}||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B_i}||\tilde{\mathsf{vk}}_i$, and obtains as answer a valid signature θ_i for the signature scheme Θ and public key $\mathsf{pk}_{A^\star}$.

Then, $\mathcal{F}_\Theta$ can run step 6 of the signcryption protocol: $\tilde{\theta}_i \leftarrow \tilde{\Theta}.\mathsf{Sign}(\tilde{\mathsf{sk}}_i,\ C_{1,i}||C_{2,i}||C_{3,i}||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B_i}||\theta_i)$. The final signcryption that $\mathcal{F}_\Theta$ sends to $\mathcal{A}_{\mathsf{UNF}}$ is $C_i = (C_{1,i}, C_{2,i}, C_{3,i}, \tilde{\mathsf{vk}}_i, \theta_i, \tilde{\theta}_i)$.

Forgery. At some point, and with probability $\varepsilon = \mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda)$, the attacker $\mathcal{A}_{\mathsf{UNF}}$ outputs a successful forgery; that is, a public key $\mathsf{pk}_{B^\star}$ and a signcryption $C^\star = (C_1^\star, C_2^\star, C_3^\star, \tilde{\mathsf{vk}}^\star, \theta^\star, \tilde{\theta}^\star)$ such that:

- the protocol $\Sigma.\mathsf{Uns}(\mathsf{params}, C^\star, \mathsf{pk}_{A^\star}, B^\star, \{\mathsf{sk}_{B^\star,j}\}_{B_j \in B^\star})$ outputs $M^\star \neq \perp$,
- $(\mathsf{pk}_{A^\star}, \mathsf{pk}_{B^\star}, C^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ during a signcryption query.

Let us define $m^\star = C_1^\star||C_2^\star||C_3^\star||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B^\star}||\tilde{\mathsf{vk}}^\star$. We can distinguish two cases.

First, with probability ε_1 we can have $(m^\star, \theta^\star) \neq (m_i, \theta_i)$, for all messages m_i that $\mathcal{F}_\Theta$ has queried to its signing oracle. Then $\mathcal{F}_\Theta$ has obtained a valid and new signature $(m^\star, \theta^\star)$ for the scheme Θ and public key $\mathsf{pk}_{A^\star}$. Therefore, $\varepsilon_1 \leq \mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda)$.

Otherwise, with probability $\varepsilon_2 = \varepsilon - \varepsilon_1$, we can have $(m^\star, \theta^\star) = (m_i, \theta_i)$ for some of the Q messages m_i that $\mathcal{F}_\Theta$ has queried to its signing oracle. In this case, since the forgery by $\mathcal{A}_{\mathsf{UNF}}$ is valid, the only possibility is $\tilde{\theta}^\star \neq \tilde{\theta}_i$. In this case, it is easy to construct a forger $\mathcal{F}_{\tilde{\Theta}}$ against the strong one-time unforgeability of $\tilde{\Theta}$: this forger receives as input a target verification key $\tilde{\mathsf{vk}}'$, then guesses the correct signcryption query i, uses its only access to a signing oracle to obtain the corresponding signature $\tilde{\theta}_i$ for this query, and uses other ephemeral key pairs $(\tilde{\mathsf{sk}}, \tilde{\mathsf{vk}})$ to reply the rest of signcryption queries. If the guess of i is correct (which happens with probability $1/Q$), then this second kind of forgery by $\mathcal{A}_{\mathsf{UNF}}$ leads to a valid forgery by $\mathcal{F}_{\tilde{\Theta}}$ against scheme $\tilde{\Theta}$. Therefore, we have $\mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda) \geq \varepsilon_2/Q$.

Summing up, we have $\mathsf{Adv}_{\mathcal{A}_{\mathsf{UNF}}}(\lambda) = \varepsilon = \varepsilon_1 + \varepsilon_2 \leq \mathsf{Adv}_{\mathcal{F}_\Theta}(\lambda) + Q \cdot \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda)$, as desired. □

Indistinguishability.

We reduce the IND-CCA security of the scheme to the hardness of solving the DBDH problem in groups $\mathbb{G}, \mathbb{G}_T$ and to the security of the underlying signature scheme $\tilde{\Theta}$, which we assume to be one-time strongly secure. The proof is in the standard model.

Theorem 2.4.2 *Let λ be an integer. For any polynomial-time attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ against the IND-CCA security of the new signcryption scheme, there exists a solver $\mathcal{A}^{DBDH}$ of the Decisional Bilinear Diffie-Hellman problem or an attacker $F_{\tilde{\Theta}}$ against $\tilde{\Theta}$ such that $\mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda) \geq \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$.*

Proof. Assuming the existence of an adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ that has advantage $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ in breaking the IND-CCA security of our scheme, we construct an algorithm $\mathcal{A}^{DBDH}$ that solves the Decisional Bilinear Diffie-Hellman problem in groups $\mathbb{G}, \mathbb{G}_T$.

$\mathcal{A}^{DBDH}$ receives as input g^a, g^b, g^c, R, where R is either $e(g,g)^{abc}$ or a random element in $\mathbb{G}_T$. The goal of $\mathcal{A}^{DBDH}$ is to distinguish between these two cases.

$\mathcal{A}^{DBDH}$ runs the key generation protocol for the signature scheme $\tilde{\Theta}$, obtaining $(\tilde{\mathsf{sk}}^\star, \tilde{\mathsf{vk}}^\star) \leftarrow \tilde{\Theta}.\mathsf{KG}(1^\lambda)$. Then $\mathcal{A}^{DBDH}$ chooses at random a suitable hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ and a suitable signature scheme Θ. The value $\mathsf{id}^\star = H(\tilde{\mathsf{vk}}^\star)$ is computed. $\mathcal{A}^{DBDH}$ defines $g_2 = g^a$, chooses at random $\gamma \in \mathbb{Z}_p^*$ and defines $h = (g^b)^{-\mathsf{id}^\star} \cdot g^\gamma$. Then $\mathcal{A}^{DBDH}$ initializes the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ by giving $\mathsf{params} = (p, \mathbb{G}, g, \mathbb{G}_T, e, H, h, g_2, \Theta, \tilde{\Theta})$ to it.

Key generation. Let $B^\star = \{B_1, \ldots, B_n\}$ be the target collective and $\widetilde{B} = \{B_1, \ldots, B_{t-1}\} \subset B^\star$ be the subset of corrupted members of $B^\star$, chosen by $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$. The algorithm $\mathcal{A}^{DBDH}$ defines the public key of $B^\star$ as $\mathsf{pk}_{B^\star} = g^b$. This means that the secret value $\alpha_{B^\star}$ is implicitly defined as b. For the corrupted members of $B^\star$, the shares $\{\mathsf{sk}_{B^\star,j}\}_{B_j \in \widetilde{B}}$ are computed by first choosing random and independent values $\alpha_{B^\star,j} \in \mathbb{Z}_p$ and then computing $\mathsf{sk}_{B^\star,j} = g_2^{\alpha_{B^\star,j}}$. Let $f \in \mathbb{Z}_p[X]$ be the implicit polynomial, with degree $t-1$, that satisfies $f(0) = b = \alpha_{B^\star}$ and $f(j) = \alpha_{B^\star,j}$ for $j = 1, \ldots, t-1$.

Using interpolation in the exponent and the values $\mathsf{pk}_{B^\star} = g^{\alpha_{B^\star}} = g^b$ and $\{\alpha_{B^\star,j}\}_{B_j \in \widetilde{B}}$, all the values $D_{B^\star,j} = g^{\alpha_{B^\star,j}}$ could be obtained (if robustness was required) for all the members $B_j \in B^\star$.

Unsigncryption queries. Let (pk_A, C) be an unsigncryption query sent for the target collective $B^\star$, where $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$. If $\tilde{\mathsf{vk}} = \tilde{\mathsf{vk}}^\star$ and $1 \leftarrow \tilde{\Theta}.\mathsf{Vfy}(\tilde{\mathsf{vk}}, C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta\,, \tilde{\theta})$, then $\mathcal{A}^{DBDH}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}^{DBDH}$ runs steps 1-3 (which are public verifications) of the unsigncryption protocol.

If (pk_A, C) is a valid signcryption and $\mathcal{A}^{DBDH}$ has not aborted, we have $\tilde{\mathsf{vk}} \neq \tilde{\mathsf{vk}}^\star$ and $\mathsf{id} = H(\tilde{\mathsf{vk}})$. Since the hash function is assumed to be collision-resistant, we have $\mathsf{id} \neq \mathsf{id}^\star$ as well. Now $\mathcal{A}^{DBDH}$ is required to simulate the values that would be broadcast in an execution of the rest of the protocol. This means simulating consistent tuples $(j, \omega_{0,j}, \omega_{1,j})$ for any $B_j \in B^\star$, where

$$\omega_{0,j} = \mathsf{sk}_{B^\star,j} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{r_j} \quad \text{and} \quad \omega_{1,j} = g^{r_j}$$

for some (randomly uniform) value $r_j \in \mathbb{Z}_p$. For the corrupted members B_j, $j = 1, \ldots, t-1$, such values can be easily computed by $\mathcal{A}^{DBDH}$, because it knows $\mathsf{sk}_{B^\star,j}$.

For any non-corrupted member B_i, $i = t, \ldots, n$, let $\lambda_0, \lambda_1, \ldots, \lambda_{t-1} \in \mathbb{Z}_p$ be the Lagrange interpolation coefficients corresponding to the set $\{0, 1, \ldots, t-1\}$ and interpolation point i. These coefficients can be publicly computed because they are independent of the (unknown) polynomial f. We have $f(i) = \lambda_0 f(0) + \sum_{j=1}^{t-1} \lambda_j f(j)$. Now $\mathcal{A}^{DBDH}$ can choose a random $\tilde{r}_i \in \mathbb{Z}_p$ and define

$$\omega_{0,i} = g_2^{\frac{-\gamma\lambda_0}{\mathsf{id}-\mathsf{id}^\star}} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{\tilde{r}_i} \cdot \prod_{j=1}^{t-1} \mathsf{sk}_{B^\star,j}^{\lambda_j} \quad \text{and} \quad \omega_{1,i} = g_2^{\frac{-\lambda_0}{\mathsf{id}-\mathsf{id}^\star}} \cdot g^{\tilde{r}_i}$$

It is not difficult to see that these two values $(\omega_{0,i}, \omega_{1,i})$ have the form

$$\omega_{0,i} = g_2^{f(i)} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{r_i} = \mathsf{sk}_{B^\star,i} \cdot (\mathsf{pk}_{B^\star}^{\mathsf{id}} \cdot h)^{r_i} \quad \text{and} \quad \omega_{1,i} = g^{r_i},$$

being $r_i = \tilde{r}_i - \frac{a\lambda_0}{\mathsf{id}-\mathsf{id}^\star}$ an implicit but randomly uniform value in $\mathbb{Z}_p$.

Summing up, $\mathcal{A}^{DBDH}$ can simulate valid tuples $(j, \omega_{0,j}, \omega_{1,j})$ for any $B_j \in B^\star$. Once this is done, the rest of the unsigncryption process can be easily completed by $\mathcal{A}^{DBDH}$, who obtains a message M and sends all this information to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Challenge. At some point, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs two messages M_0, M_1 of the same length, along with a key pair $(\mathsf{sk}_{A^\star}, \mathsf{pk}_{A^\star})$ for a sender $A^\star$. To produce the challenge ciphertext $C^\star$, the algorithm $\mathcal{A}^{DBDH}$ chooses at random a bit $d \in \{0,1\}$, and proceeds as follows.

1. Define $C_1^\star = g^c$, $C_2^\star = M_d \cdot R$ and $C_3^\star = (g^c)^\gamma = \ldots = (\mathsf{pk}_{B^\star}^{\mathsf{id}^\star} \cdot h)^c$.

   Note that $(C_1^\star, C_2^\star, C_3^\star)$ is a consistent encryption of M_b for identity $\mathsf{id}^\star$ when $R = e(g,g)^{abc}$. On the other hand, when $R \in \mathbb{G}_T$ is random, the distribution of $(C_1^\star, C_2^\star, C_3^\star)$ is independent of the bit d.

2. Run $\theta^\star \leftarrow \Theta.\mathsf{Sign}(\mathsf{sk}_{A^\star},\ C_1^\star||C_2^\star||C_3^\star||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B^\star}||\tilde{\mathsf{vk}}^\star)$.

3. Run $\tilde{\theta}^\star \leftarrow \tilde{\Theta}.\mathsf{Sign}(\tilde{\mathsf{sk}}^\star,\ C_1^\star||C_2^\star||C_3^\star||\mathsf{pk}_{A^\star}||\mathsf{pk}_{B^\star}||\theta^\star)$.

4. Send to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the challenge signcryption $C^\star = (C_1^\star, C_2^\star, C_3^\star, \tilde{\mathsf{vk}}^\star, \theta^\star, \tilde{\theta}^\star)$.

More unsigncryption queries. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make more unsigncryption queries $(\mathsf{pk}_A, C) \neq (\mathsf{pk}_{A^\star}, C^\star)$ for the target collective $B^\star$, where $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$, as long as the challenge signcryption is not queried. If $\tilde{\mathsf{vk}} \neq \tilde{\mathsf{vk}}^\star$, then these queries are handled in the same way as explained above.

Otherwise, if $\tilde{\mathsf{vk}} = \tilde{\mathsf{vk}}^\star$ and $1 \leftarrow \tilde{\Theta}.\mathsf{Vfy}(\tilde{\mathsf{vk}},\ C_1||C_2||C_3||\mathsf{pk}_A||\mathsf{pk}_B||\theta\ ,\tilde{\theta})$, then $\mathcal{A}^{DBDH}$ aborts and outputs a random bit.

Final analysis. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a guess bit d'. If $d' = d$, then $\mathcal{A}^{DBDH}$ outputs 0 as its answer to the given instance of the DBDH problem. If $d' \neq d$, then $\mathcal{A}^{DBDH}$ outputs 1.

Let us denote as δ the probability that $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ makes an unsigncryption query for a valid signcryption $C = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$ such that $\tilde{\mathsf{vk}} = \tilde{\mathsf{vk}}^\star$. In other words, δ is the probability that $\mathcal{A}^{DBDH}$ aborts before $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs its guess bit d'. Using a similar argument as in the unforgeability proof, it is easy to see that, in this case, one can construct a forger $\mathcal{F}_{\tilde{\Theta}}$ against the one-time signature scheme $\tilde{\Theta}$: the input of $\mathcal{F}_{\tilde{\Theta}}$ is $\tilde{\mathsf{vk}}^\star$, the only access to the signing oracle is used to compute the challenge signcryption, and any valid unsigncryption query coming from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ which involves $\tilde{\mathsf{vk}}^\star$ leads to a valid strong forgery of the signature scheme $\tilde{\Theta}$. Therefore, we have $\delta \leq \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda)$.

Let us now compute the probabilities that the output of the constructed solver $\mathcal{A}^{DBDH}$ of the DBDH problem is 0 in the two possible cases. When $R = e(g,g)^{abc}$,

then the challenge signcryption is consistent, and consequently we have probability $\Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] = \delta \cdot \frac{1}{2} + (1-\delta) \cdot (\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) + \frac{1}{2})$.

Otherwise, when $R = T$ is a random element in $\mathbb{G}_T$, the challenge signcryption is independent of the bit d, so the probability that $d' = d$ is $1/2$, and we have $\Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, T) = 0] = \delta \cdot \frac{1}{2} + (1-\delta) \cdot \frac{1}{2}$.

Now we have $\mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) =$

$$\left|\Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] \; - \; \Pr[\mathcal{A}^{DBDH}(g, g^a, g^b, g^c, T) = 0]\right| =$$

$$= (1-\delta)\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) = \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) - \delta \cdot \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda).$$

Putting all together, we have, as desired:

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) = \mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \delta \cdot \mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) \leq$$

$$\leq \mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \delta = \mathsf{Adv}_{\mathcal{A}^{DBDH}}(\lambda) + \mathsf{Adv}_{\mathcal{F}_{\tilde{\Theta}}}(\lambda).$$

□

2.5 Efficiency of the Schemes and Properties

In this section we discuss some topics related to both threshold unsigncryption schemes that we have just proposed.

2.5.1 Computational Complexity and Overhead

The two schemes proposed in Sections 2.3 and 2.4 are the first PKI-based threshold unsigncryption schemes which achieve a high enough level of security. In particular, generic constructions obtained by composing a fully secure signature scheme and a fully secure threshold decryption scheme do not achieve this level of security, as we have shown in Section 2.2.

Therefore, our first goal was to show that the maximum level of security for threshold unsigncryption schemes can indeed be achieved. This is what we have done with our two proposals. Regarding efficiency, there are no previous schemes with the same level of security to compare with, so it is not possible to say if the two new schemes are efficient or not. Table 2.1 summarizes the computational and communication costs of our schemes (without considering robustness). The costs of these two schemes should be considered as a benchmark for any future proposal of threshold unsigncryption scheme.

To measure the efficiency of the second scheme, in Section 2.4, we have taken as the signature scheme Θ the scheme in [15], and as the one-time signature scheme $\tilde{\Theta}$

Scheme	cost of Signcryption	$\|C\|$	cost of Unsigncryption (*per* receiver)	Security model
Section 2.3	6 Exp	6λ	8 Exp	ROM
Section 2.4	12 Exp + 1 Pa	12λ	11 Exp + 6 Pa	Standard

Table 2.1: Efficiency of our two threshold unsigncryption schemes.

the scheme in Appendix B of [68]. In the table, λ denotes the security parameter of the scheme; i.e., an element in the group $\mathbb{G}$ can be represented by λ bits. In the case of our first scheme, we have considered that the length of the plaintexts is $\ell = \lambda$, for simplicity.

We denote the size in bits of a ciphertext C as $|C|$. For the computational costs, we just consider exponentiations (denoted as Exp) and bilinear pairing computations (denoted as Pa, only for the second scheme). The rest of operations (xor, modular addition and multiplication, hash computations) are not considered because they are very cheap; they do not affect the real efficiency of the schemes. Roughly speaking, we can say that the scheme in Section 2.4, whose security is proven in the standard model, is twice less efficient than the scheme in Section 2.3, whose security is proved in the random oracle model (ROM).

2.5.2 Splitting the Unsigncryption Protocol for Auctions Systems

If we go back to the description of the Threshold Unsigncryption protocol of the two new schemes, in Sections 2.3 and 2.4, we can easily distinguish two parts in those protocols. Steps 1-3 correspond to the (public) verification procedure; these authentication steps can be run by any (individual) party, not necessarily inside the set B of intended receivers. In other words, no secret information is needed as input to run these three steps; the only inputs are the ciphertext and the public key of the sender. Then, Steps 4-7 correspond to the (secret) decryption procedure, which in this case requires the participation of at least t members of the set B of intended receivers. The important point here is that the identity or public key pk_A of the sender is not used at all for the execution of Steps 4-7. In some sense, the public key pk_A of the sender could be removed from the process once the ciphertext has been accepted as valid, in Step 3. After that, the identity of the sender would be unknown during the rest of the unsigncryption process.

In this way, the unsigncryption part of our two new schemes could be split into two parts. The first one could be run by an entity $T \notin B$, who would discard invalid ciphertexts and remove (or store privately) the identities of the senders. In the second

part, only valid (and anonymized) ciphertexts would reach the set B of receivers, who would jointly decrypt the ciphertext to recover the original plaintexts, maybe without knowing at any moment who are the senders of the messages.

As far as we know, these are the first signcryption schemes (with either individual or threshold unsigncryption) enjoying this property, which may be of interest in some applications requiring some level of anonymity or privacy, such as electronic auctions.

In an electronic auction system, participants send their confidential and authenticated bids for a product. At the end of the process, some (distributed) entity B detects the highest bid and identifies the author of that bid, who wins the right to buy the product for that price. Identities of the authors of the remaining bids should remain hidden. To increase the confidentiality of the process, entity B can consist of a set of n entities, working in a (t, n)-threshold fashion.

Let us assume that such an auction system is implemented by using a signcryption scheme where the unsigncryption protocol can be split into two phases, in such a way that the decryption part is anonymized. An external authority (or machine) T, different from B, can be in charge of the first part of the unsigncryption process: T receives the ciphertexts from the participants in the auction, verifies that the participants are in the list of admitted participants, and runs the verification part of the unsigncryption. Invalid ciphertexts are discarded, and valid ciphertexts are anonymized and forwarded to the auction decryption entity B. Entity T must privately store a table (pk_A, C), relating the public keys of the participants with their ciphertexts. Optionally, the anonymized ciphertexts that are forwarded to B (or a hashed version of them) can be published so that all the participants in the auction can verify that their bids have been taken into account.

The decryption process is then run by entity B, in an individual or threshold fashion, and the highest bid among the resulting (anonymous) bids is selected. The winning bid and its corresponding ciphertext C are announce by B, and then T can search in its table and recover the identity of the author of the winning bid. Assuming the honesty of entity T, the anonymity of the participants that do not win the auction is clearly preserved, even in front of the decryption authority B. Since the role of T can be easily implemented by a secure piece of hardware, trusting entity T is not a very strong assumption.

This splitting property could also be applied in electronic voting systems, where signcryptions contain now the ballots of the persons who have obtained legal authorisation to take part in the electronic vote. These systems could work in a similar way as explained above to ensure the anonymity of the voters.

2.6 Applications to Digital Signatures with Distributed Verification

The usual implementations of digital signature schemes allow *universal verification*; that means that everybody can verify the validity of the signature using the verification protocol, which requires as inputs the message, the signature and the public key of the signer. In fact, in some applications the universal verification property can be too strong if the signer wants some level of anonymity and privacy. Maybe the signer does not want that everybody can check the authenticity of his signature because of the possible consequences of the signed document. For example, when supporting a public declaration to go on strike, a company worker might wish that the other employees of his level are able to verify his signature, but that the boss of his company is not. Or when infiltrating a rumour to the yellow press, the person who spreads this rumour might ask for the collaboration of a certain subset of journalists so that the verification is successful and that the rumour converts into a headline. The person who spreads the rumour might even wish to stay anonymous, by only passing the information to those who belong to a certain trustworthy set of people, but without revealing his identity. Examples can be found in signatures on medical records, in statements of witness in a judgement, economic information and most personal/business transactions which contains information personally sensitive to the signers.

With the intention of giving the signer a certain degree of anonymity, in this section we propose a different approach to restrict the verification in a digital signature scheme. Taking into account the liability of the signed document, the signer prefers to restrict the signature capacity in two different ways. First, a set of verifiers is established; second, a family (access structure) of a subset of verifiers who are authorized to verify is defined. In this way, two properties must be satisfied after the signer has obtained and published the signature. On the one hand, if the verifiers of an authorized subset collaborate, they can verify the validity of the signature and be convinced that it comes from the real signer, since the scheme is unforgeable and nobody except the signer could have produced a valid signature. On the other hand, a group outside the verifiers of an authorized subset (an unauthorized subset) can not obtain any information about the validity of a signature for a certain message, although they collaborate with each other. We refer to these schemes as *signatures with distributed verification*.

Let us define now this kind of signature schemes for a single signer A and a set B of verifiers, where a linear secret sharing scheme on the access structure Γ_B is used to share the secret. A signature scheme with distributed verification $\Theta = (\Theta.\mathsf{St}, \Theta.\mathsf{KG}, \Theta.\mathsf{Sign}, \Theta.\mathsf{Vfy})$ consists of four probabilistic polynomial-time algorithms:

- The randomized *setup* algorithm $\Theta.\mathsf{St}$ takes a security parameter $\lambda \in \mathbb{N}$ and outputs some public parameters params that will be common to all the users in the system. We write $\mathsf{params} \leftarrow \Theta.\mathsf{St}(1^\lambda)$ to denote an execution of this algorithm.

- The *key generation* algorithm $\Theta.\mathsf{KG}$ is different for an individual signer A than for a collective B of verifiers. A single user A will get a pair $(\mathsf{sk}_A, \mathsf{pk}_A)$ of secret and public keys. In contrast, for a set $B = \{B_1, \ldots, B_n\}$ of n verifiers, the output will be a single public key pk_B for the group, and then a secret share $\mathsf{sk}_{B,j}$ for each user B_j, for $j = 1, \ldots, n$, and for the access structure Γ_B. The key generation process for the collective B can be either run by a trusted third party, or by the users in B themselves. We will write $(\mathsf{sk}_A, \mathsf{pk}_A) \leftarrow \Theta.\mathsf{KG}(\mathsf{params}, A, \text{'single'})$ and $(\{\mathsf{sk}_{B,j}\}_{1 \le j \le n}, \mathsf{pk}_B) \leftarrow \Theta.\mathsf{KG}(\mathsf{params}, B, \Gamma_B, \text{'collective'})$ to refer to these two key generation protocols.

- The *signing* algorithm $\Theta.\mathsf{Sign}$ takes as input params, a message M, the public key pk_B of the intended verifier group B, and the secret key sk_A of the signer. The output is a signature θ. We denote an execution of this algorithm as $\theta \leftarrow \Theta.\mathsf{Sign}(\mathsf{params}, M, \mathsf{pk}_B, \mathsf{sk}_A)$.

- The *distributed verification* algorithm $\Theta.\mathsf{Vfy}$ is an interactive protocol run by some subset of users $B' \subset B$. The common inputs are params, a message M, a signature θ and the public key pk_A of the signer, whereas each user $B_j \in B'$ has as secret input his secret share $\mathsf{sk}_{B,j}$. The output will be 1 if $\theta(M)$ is a valid signature of M and 0 otherwise. We write $1/0 \leftarrow \Theta.\mathsf{Vfy}(\mathsf{params}, M, \theta, \mathsf{pk}_A, B', \{\mathsf{sk}_{B,j}\}_{B_j \in B'})$ to refer to an execution of this protocol.

This kind of signatures must verify *Correctness*: whenever the signature with distributed verification is generated by following the protocols correctly, the result of the verification is always 1 if the subset of verifiers is authorized to perform a verification.

Both *Unforgeability* and *Privacy* properties are held by this kind of signatures. Among all the proposed definitions of unforgeability we consider the strongest one, i.e. existential unforgeability against chosen message attacks (See Subsection 1.4.2). Related to the privacy (sometimes called *Invisibility* or *Indistinguishability*), the idea behind it is that a subset which is not in the corresponding access structure does not obtain any information about if the signature is valid or not. Or in other words, this property satisfies that if a signature θ is sent from A to B, an adversary can not distinguish if θ is a signature of the message M, even in the case that this message M has been chosen by the adversary, or a signature of any other message.

2.6.1 Relation with Threshold Unsigncryption

This kind of signatures are a cryptographic primitive very similar, but weaker, to signcryption schemes with threshold unsigncryption because now it is not necessary that the message remains secret. The key point here is to replace the step where the message is hidden using a key one time, in the encryption part of the signcryption scheme, by a *message authentication code* [1] (MAC) of the message using the same key one time, in the signature scheme.

We use the signcryption schemes with threshold unsigncryption presented in sections 2.3 and 2.4 to build two new signature schemes with threshold verification, whose security is based on the same computational problems and achieve the same level of security (either in the random oracle model or in the standard model).

More in detail, in the scheme defined in 2.3 the value $c = m \oplus k$ is now obtained using a hash funcion $H' : \{0,1\}^* \to \{0,1\}^\kappa$ as $c = H'(k, m)$. In this case, the two first elements (c, R) of the signature $\theta(m) = (c, R, \bar{R}, h, s_1, s_2)$ for the message m is the MAC. For the other scheme defined in 2.4, the value $C_2 = M \cdot e(\mathsf{pk}_B, g_2)^s$ is obtained using a hash function $H'' : \{0,1\}^* \times \mathbb{G}_T \to \{0,1\}^*$ for a group $\mathbb{G}_T$ as $C_2 = H''(M, e(\mathsf{pk}_B, g_2)^s)$. Now the MAC is the tuple (C_1, C_2, C_3) with the three first elements of the signature $\sigma(m) = (C_1, C_2, C_3, \tilde{\mathsf{vk}}, \theta, \tilde{\theta})$ for the message m. These results are published in [42] and [43]. A detailed description of these schemes and their security proofs can be found there.

2.6.2 Related Work

Different kind of signatures which restrict the (universal) verification in some way already exist. Some of the signatures which give certain degree of anonymity to the signer are the following.

Designated Verifier Signatures, denoted by DVS, were introduced in [49] and are intended to a specific verifier, who is the only one able to check their validity. An extension is the case of *Multi-Designated Verifier Signatures* [57], MDVS in short, where a set of different verifiers exists. A generalization of these signatures to *Strong Designated Verifier Signatures*, denoted by SDVS, where the indistinguishability (or invisibility) property is taken into account, is also available. In an *Undeniable Signature* [24], denoted by US, the signature can not be verified without the signer's cooperation. *Designated Confirmer Signatures*, DCS in short, appeared in [23] and use a confirmation/disavowal protocol with the assistance of a semitrusted third party to overcome the concept that signers might be unavailable or unwilling to cooperate and hence signatures would no longer be verifiable, as e.g. in undeniable signatures.

[1] A MAC algorithm accepts as input a symmetric secret key and an arbitrary-length message to be authenticated, and outputs a tag value used to authenticate.

Comparing some of the above signatures with our primitive, threshold (t, n)-signatures with distributed verification, we can see that if we use our primitive but now setting $n = 1$ and $t = 1$, then we have only one verifier which can generate himself his keys without any problem. In this particular case we obtain a strong designated verifier signature, SDVS. In the case of $(1, n)$-signatures with distributed verification, for $n > 1$, every receiver of the signature can verify it by himself. That is, we obtain the same functionality that a MDVS but with the following differences: In the signatures with distributed verification there is a key generation protocol, where every verifier from a group receives a share of the secret key related to the unique public key for this group, but in the MDVS every verifier generates his pair of secret/public key individually. Furthermore, the group of verifiers and the access structures are fixed from the begining in the signatures with distributed verification, whereas in the MDVS, the signer chooses (ad-hoc) the participants who can verify the signature. This point allows that in the signatures with distributed verification the length of the signature and the computational cost of the signature protocol are independent of the number of verifiers.

Chapter 3

New Results for Secret Sharing

In the design of distributed public key cryptosystems, a key ingredient are *secret sharing schemes*. Most of the secret sharing schemes proposed and analyzed so far enjoy *unconditional* (or *information-theoretic*) security, which means that the value of the shared secret is perfectly hidden to an (even computationally unbounded) adversary who controls any non-authorized subset of users. The reason for that is because secret sharing schemes are usually used to build other cryptographic primitives whose security is proved only when the underlying secret sharing scheme is unconditional secure. Since the main application of secret sharing schemes in this thesis is their use in the design of distributed cryptosystems (that can enjoy computational security, at most), it seems that requiring unconditional security for the underlying secret sharing schemes might be innecessarily restrictive for us.

This chapter presents how one or more secrets can be shared over a set of participants. More specifically, verifiable secret sharing schemes and multi-secret sharing schemes are studied and different proposals, whose security is proved in a computational framework, are presented for them.

A publicly verifiable secret sharing scheme is a sharing scheme, where not only the participants but anybody can verify the validity of the shares. The first section proposes a simple publicly verifiable secret sharing scheme based on the homomorphic properties of the Paillier's encryption scheme, whose security is based on the decisional composite residuosity assumption. The process of verification in this scheme is much simpler than in the other known schemes. Furthermore, this process is made non-interactive without using any additional *zero-knowledge proof* (show a statement to be true without revealing anything other than the veracity of the statement to be proven).

In the following sections we work with multi-secret sharing schemes, MSSS in short, which are protocols that share multiple secrets (instead of one as in traditional schemes) among a set of participants. These kind of schemes have been studied by the

cryptographic community mostly from a theoretical perspective: different models and definitions have been proposed, for both unconditional (information-theoretic) and computational scenario. Works focusing on MSSS in the computational scenario are mostly practical [18, 39, 62] and they lack a formal security analysis of the proposed schemes. In the information-theoretic scenario, there are some results [9, 67] that are analogous to what happens in the case of standard secret sharing. That is, multi-secret sharing schemes with unconditional security produce very long shares for some important access structures. We will show in Section 3.2, that this situation also happens for multi-threshold secret sharing schemes when we relax the notion of unconditional security and then we will move to the scenario of computational security to perform this issue.

We propose in Sections 3.3, 3.4 and 3.6 three different multi-secret sharing schemes in the computational scenario, denoted by Ω_1, Ω_2 and Ω_3 respectively, which are inspired by previous work in this area. Namely, on the one hand, Ω_1 is obtained by running ℓ parallel instances of (a simple modification of) the secret sharing scheme of Krawczyk [55]; on the other hand, Ω_2 and Ω_3 can be thought of as a generalization of some previous MSSSs in the computational scenario [39, 62]. We provide a formal security analysis for all three schemes. As far as we know, this is the first rigorous security analysis for multi-secret sharing schemes in the computational setting. We show that they have provable security: the first two schemes Ω_1 and Ω_2 use an underlying symmetric encryption scheme Π, and we prove the exact relation between the security of Π and the security of each of these MSSSs. This fact allows us to properly compare in Section 3.5 the two schemes Ω_1 and Ω_2 in terms of efficiency and security, and to recommend the use of one MSSS or the other depending on the specific situation in which such a scheme must be used. Last proposed scheme Ω_3 is a slight modification of Ω_2 and [62]. The scheme Ω_3 uses hash functions to make the protocols more efficient and is presented in Section 3.6 with a formal proof of computational security in the random oracle model. Although we describe and analyze all these MSSSs for the case of threshold access structures, it can be easily extended to more general access structures (see Section 3.7).

3.1 Publicly Verifiable Secret Sharing from Paillier's Cryptosystem

In a secret sharing (SS) scheme there are several participants and a dealer D who shares them a secret. Verifiable secret sharing (VSS) was proposed in [25] to solve the problem of dishonest dealers and dishonest participants who try to deceive other participants. In a publicly verifiable secret sharing (PVSS) not only the participants

can verify the validity of their shares but also anyone can do it from the public information. Note that in a PVSS no private channels between the D and the participants are assumed.

The originality of the scheme that we propose lies in the holder of the secret key. When we compare with other PVSS like [33], [81] or [89] we can see that every participant has a secret key and when the dealer wants to send the shares or other information, he uses their respective public keys. On the contrary, in our scheme the dealer has a secret key and when he wants to broadcast the shares, he uses some information from the participants which they have send him previously.

As long as we are using only one secret/public key, verification procedure is much simpler than in the other PVSS proposals. This is the main advantage compared to other similar schemes that were known before this scheme was presented. However, every participant should hold a public/secret key pair for a signature scheme in order to allow D to prove that each participant received his share. Potential applications of this scheme are the same that for any PVSS, like e.g. multi-party computation, key-escrow cryptosystems and threshold cryptography.

The building blocks of our scheme are the Shamir (t, n)-threshold scheme and the homomorphic Paillier's encryption scheme [73]. However, the proposed scheme can be adapted for more general access structures (vector space access structures) and other homomorphic encryption schemes.

3.1.1 A Non-Interactive PVSS Scheme

We now proceed to describe a secure (t, n)-threshold PVSS scheme $\Omega = (\Omega.\mathsf{Stp}, \Omega.\mathsf{Dist}, \Omega.\mathsf{Ver}, \Omega.\mathsf{Rec})$. We work with a dealer D who shares the secret, the set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n participants and a verifier V who verifies the honestity of D. We can suppose that V is any person. Taking into account the ideas from [81], we have four importants steps in a PVSS:

1. The *setup* algorithm $\Omega.\mathsf{Stp}$ takes as input a security parameter $\lambda \in \mathbb{N}$, the set of players $\mathcal{P}$ and the threshold access structure Γ and returns the system public parameters params. We write $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}\left(1^{\lambda}, \mathcal{P}, \Gamma\right)$ to denote an execution of this algortihm.

2. The *distribution of the shares* $\Omega.\mathsf{Dist}$ takes as input params and the secret s to be distributed. The output of this algorithm is the set of messages $\{m_i\}_{P_i \in \mathcal{P}}$, that remains secret to every participant, and the following public information: D shares the secret s among the participants in $\mathcal{P}$ by publishing the set of encrypted shares $\{d_i\}_{P_i \in \mathcal{P}}$. Moreover, this algorithm also outputs additionally public information $\mathsf{out}_{\mathsf{pub}}$ to be used in the next steps. We write $(\mathsf{out}_{\mathsf{pub}}, \{d_i\}_{P_i \in \mathcal{P}}, \{m_i\}_{P_i \in \mathcal{P}}) \leftarrow \Omega.\mathsf{Dist}\,(\mathsf{params}, s)$.

3. The *verification of the shares* $\Omega.\mathsf{Ver}$ takes as input params, the public information $\mathsf{out}_{\mathsf{pub}}$ and all the encrypted shares, and outputs 1 if the verification is successful and 0 when it fails. In this last case the whole protocol is aborted. Here V verifies non-interactively that the published information is consistent and that every authorized subset of (honest) participants will recover the same secret. We write $1/0 \leftarrow \Omega.\mathsf{Ver}\,(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, \{d_i\}_{P_i \in \mathcal{P}})$.

4. The *reconstruction of a secret* $\Omega.\mathsf{Rec}$ takes as input $\mathsf{params}, \mathsf{out}_{\mathsf{pub}}$ and also the messages $\{m_i\}_{P_i \in A}$ and encrypted shares $\{d_i\}_{P_i \in A}$ of the participants in some subset $A \subset \mathcal{P}$. The output of this algorithm is a possible value $\tilde{s}$ for the secret. In this step every participant $P_i \in A$ opens his commitment to the others, so that the other participants in A can compute the share of P_i and verify its validity (participants whose validity fails are excluded). If enough honest participants remain in A, every participant in A can recover the secret by himself. We write $\tilde{s} \leftarrow \Omega.\mathsf{Rec}\,(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, \{d_i\}_{P_i \in A}, \{m_i\}_{P_i \in \mathcal{A}})$.

We have two main protocols in this PVSS. The *distribution protocol*, which is specified in both steps 2 and 3, and the *reconstruction protocol*, which is specified in step 4.

Basic secret sharing schemes require the notions of *correctness* and *secrecy*, defined in Subsection 1.5.1, but PVSS schemes require an additional property. This new property is called *verifiability* and checks the following two points: First, all qualified subsets of honest participants will reconstruct the same secret if D passes the verification step. Second, only correct shares will be accepted by other participants in the reconstruction protocol.

Let $N = p \cdot q$ be the same that in the public key of Paillier's Cryptosystem for primes p and q, we share with the Shamir (t, n)-threshold scheme in the ring $\mathbb{Z}_N$, where the secret $a_0 \in \mathbb{Z}_N$ is hidden in a random polynomial $a(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$ with coefficients in $\mathbb{Z}_N$. Then D gives the value $(x_i, s_i) \in \{1, 2, \ldots, n\} \times \mathbb{Z}_N$ to P_i, where $s_i = a(x_i) \bmod N$. For simplicity, we assume in the following calculations that $x_i = i$. When a subset of t participants $P_{i_1}, \ldots, P_{i_t}$ want to reconstruct the secret a_0, then they must solve the system of t linear equations in the t unknowns $a_0, \ldots, a_{t-1}$ as follows:

$$\begin{pmatrix} 1 & x_{i_1} & x_{i_1}^2 & \cdots & x_{i_1}^{t-1} \\ 1 & x_{i_2} & x_{i_2}^2 & \cdots & x_{i_2}^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i_t} & x_{i_t}^2 & \cdots & x_{i_t}^{t-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} s_{i_1} \\ s_{i_2} \\ \vdots \\ s_{i_t} \end{pmatrix}$$

with Vandermonde determinant $det(A) = \prod_{1 \le k < j \le t}(x_{i_j} - x_{i_k})$.

In order to be able to recover the secret we want that $det(A) \in \mathbb{Z}_N^*$. In order to guarantee that $det(A) \in \mathbb{Z}_N^*$ holds, it is sufficient that D chooses p, q such that $n << p, q$. And we can force that D chooses these suitable p, q by proving that $i \nmid N$, for all i such that $1 \leq i \leq n$.

From now on we suppose that $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, t-1\}$. We describe below the algorithms of the PVSS scheme Ω, which uses the underlying Paillier's probabilistic encryption scheme $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ [1] (see Subsection 1.3.3).

1. **Setup**: $\Omega.\mathsf{Stp}\left(1^\lambda, \mathcal{P}, t\right)$.
   Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of n participants and let t be the threshold value. Given a security parameter λ, D runs $\Pi.\mathsf{KG}(1^\lambda)$ obtaining as result the public key $\mathsf{pk} = (N, g)$ and the secret key $\mathsf{sk} = \eta$. The output of this step is $\mathsf{params} = (N, g)$.

2. **Distribution of the shares**: $\Omega.\mathsf{Dist}\left(\mathsf{params}, s\right)$.
   Let $s \in \mathbb{Z}_N$ be the secret to be shared.

   (a) Every P_i selects a random pair $(m_i, r_i) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ which remains private, then publishes the encrypted value $c_i = Enc(m_i, r_i)$.

   (b) D selects a random polynomial $a(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1} \in \mathbb{Z}_N[X]$ which must verify $a_0 = s$ and sets $s_i = a(x_i) \mod N$, where every P_i has the value x_i.

   (c) D recovers the message $m_i = Dec(c_i)$ and the randomness $r_i = \left(\frac{c_i}{g^{m_i}}\right)^{N^{-1}} \mod N$. Then he broadcasts $d_i = s_i + m_i \mod N$, which allows every participant P_i to compute its own share s_i.

   (d) Finally, D selects random values $r'_j \in \mathbb{Z}_N^*$, which remain private, and broadcasts $A_j = Enc(a_j, r'_j)$ and $t_i = r'_0 \cdot {r'_1}^i \cdots {r'_{t-1}}^{i^{t-1}} \cdot r_i \mod N$. Note that, $\mathsf{out_{pub}} = (\{c_i\}_{1 \leq i \leq n}, \{A_j\}_{0 \leq j \leq t-1}, \{t_i\}_{1 \leq i \leq n})$.

3. **Verification of the shares**: $\Omega.\mathsf{Ver}\left(\mathsf{params}, \mathsf{out_{pub}}, \{d_i\}_{P_i \in \mathcal{P}}\right)$.

   (a) For every participant i, V uses the homomorphic property of Π to check $Enc(d_i, t_i) = c_i \cdot A_0 \cdot {A_1}^i \cdots A_{t-1}^{i^{t-1}}$.

   (b) V returns 1 if all the above equations hold and 0 otherwise.

[1] To simplify the notation we use in the following $Enc(m, r)$ to denote the encryption process $\Pi.\mathsf{Enc}(m, \mathsf{pk}) = \varepsilon_g(m, r)$, for a pair $(m, r) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ of message m and random variable r, and $Dec(c)$ to denote the decryption process $\Pi.\mathsf{Dec}(c, \mathsf{sk})$ run by D over the ciphertext c.

4. **Reconstruction of a secret**. $\Omega.\mathsf{Rec}\,(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, \{d_i\}_{P_i \in A}, \{m_i\}_{P_i \in A})$.
   Let $A \subset \mathcal{P}$ be a subset of participants in $\mathcal{P}$ that want to recover the secret s. They proceed as follows.

   (a) Decryption of the shares: every $P_j \in A$ sends $(m_j, r_j) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ to the other participants in A. Every $P_j \in A$ checks $c_k = Enc(m_k, r_k)$ for all $P_k \in A$ and defines $B_j \subseteq A$ as the subset of participants that pass the test. Finally, every P_j computes $s_k = d_k - m_k$ for all $P_k \in B_j$.

   (b) Pooling the shares: every $P_j \in A$ checks if $|B_j| > t$. In this case, P_j use the secret values $\{s_k\}_{P_k \in B_j}$ to recover $s = a_0 = \sum_{k=1}^{t} \lambda_k^A \cdot s_k \mod N$, where λ_k^A are the Lagrange interpolation coefficients.

Usually in a PVSS scheme the dealer uses a public encryption scheme to encrypt the shares, which are decrypted by every participant with its own secret key. This is not the case in the proposed scheme Ω. Note that the dealer uses here implicitly a symmetric encryption scheme (one-time pad) to encrypt the shares s_i. The symmetric keys m_i are generated at the beginning of $\Omega.\mathsf{Dist}$ by every participant P_i, who encrypts it using the public Paillier's encryption scheme Π and sends after that to the dealer.

We assume the existence of a broadcast authentication protocol for our scheme. We can use e.g. the BiBa broadcast authentication protocol, which allows all receivers to verify the origin of the data. See [75] for more information. But it is sufficient to take a broadcast channel without authentication and a digital signature scheme for the participants to sign their c_i in step 2(a) in order to assure that the honest participants receive their shares.

Correctness of the scheme means that a honest D always passes the verification procedure, and honest participants are always able to recover the unique secret shared by an honest D. These requirements can be easily checked for the above scheme.

3.1.2 Unconditional Verifiability

We see in the next two points that if D passes the verification step then all the players must be honest in this PVSS, i.e, on the one hand, the dealer must be honest in the distribution protocol and, on the other hand, the participants must be honest in the reconstruction protocol.

Distribution protocol.

In the following we are going to prove that when D is dishonest he can convince nobody. In other words, when V checks the third point in the protocol and he is convinced from his result, then all sets in Γ reconstruct the same secret.

$Rec(A, S_A)$ denotes the result of the reconstruction protocol executed by an authorized subset $A \subseteq \mathcal{P}$ such that $|A| = t$, where $S_A = (s_i)_{i \in A}$ are the shares hold by participants in A.

Definition 3.1.1 *We say that D decives if there exists different $A_1, A_2 \in \Gamma$ such that $Rec(A_1, S_{A_1}) \neq Rec(A_2, S_{A_2})$. In the case that D can not deceive, we say that D is honest.*

Theorem 3.1.2 *If V verifies the step 3 in the scheme, then D must be honest.*

Proof. In order to prove that all the authorized subsets in Γ reconstruct the same secret, it is sufficient to prove that the secret that users of any minimal authorized subset reconstruct with their shares is the same secret that D has given V through Paillier's encryption.

If every participant of any minimal authorized subset $A = \{P_1, \ldots, P_t\} \in \Gamma$ of t participants (if A have more than t participants then we work only with t of them because we need only t participants to recover the secret) has his share $s_i = a_0 + a_1 i + \cdots + a_{t-1} i^{t-1} \mod N$ and V has the elements $A_j = Enc(a'_j, r'_j)$ then we only need to prove that $a'_j = a_j, \forall j$ where $0 \leq j \leq t-1$.

Since V has verified $c_i \cdot A_0 \cdot {A_1}^i \cdots A_{t-1}^{i^{t-1}} = Enc(d_i, t_i)$ and $Dec(c_i \cdot A_0 \cdot {A_1}^i \cdots A_{t-1}^{i^{t-1}}) = Dec(c_i) + Dec(A_0) + i \cdot Dec(A_1) + \cdots + i^{t-1} \cdot Dec(A_{t-1}) = m_i + a'_0 + a'_1 i + \cdots + a'_{t-1} i^{t-1} \mod N$, then $a'_0 + a'_1 i + \cdots + a'_{t-1} i^{t-1} = d_i - m_i = s_i \mod N$. Since

$$\begin{cases} s_1 = a'_0 + a'_1 + \cdots + a'_{t-1} \\ \vdots \\ s_t = a'_0 + a'_1 t + \cdots + a'_{t-1} t^{t-1} \end{cases}$$

has a unique solution then $a'_j = a_j, \forall j$. □

So the dealer is always commited to give the real secret s and it is guaranteed that any authorized set of participants obtain the same secret s in the reconstruction protocol. Therefore we do not need any additional proof other than the information broadcast by D at steps 2 and 3.

From now on we can suppose that the dealer is always honest with the other players.

Reconstruction protocol.

When t participants meet each other in order to reconstruct the original secret $s = a_0$, every participant must open his own commitment c_j, for every $0 \leq j \leq t-1$, that means that all of these t participants know the elements (m_j, r_j) of the other ones,

and then these participants together can compute the shares and recover the secret using Lagrange interpolation.

In the following we are going to remark that none of the participants among these t can trick the other participants in this protocol, i.e, if some participant gives a different information then the other participants, without interaction, know that he is trying to cheat them.

Definition 3.1.3 *Let $A \in \Gamma$ and $P_1, P_2 \in A$. We say that P_1, who has the pair $(m_1, r_1) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$, cheats P_2 when P_1 gives other pair (m_1', r_1') to P_2 and P_2 is convinced that $m_1 = m_1' \mod N$ and $r_1 = r_1' \mod N$ when he verifies $c_1 = Enc(m_1, r_1)$.*

Remark. Let $A \in \Gamma$ and $P_1, P_2 \in A$. P_1 can not cheat P_2 because for every c_1 there is only one $m_1 \in \mathbb{Z}_N$ and $r_1 \in \mathbb{Z}_N^*$ such that $c_1 = Enc(m_1, r_1)$ and, therefore, P_1 must open c_1 with (m_1, r_1). Furthermore, for given m_1 and d_1, s_1 is uniquely determined.

When the participants open their commitments c_j it is very easy to show that their respective shares are correct and consequently to exclude the participants who try to deceive. So, we do not need here any additional proof. We only need that the participants open their commitments and if somebody does not want to open then he is excluded from $\mathcal{P}$.

3.1.3 Computational Secrecy

Our goal now is to see that any not authorized subset not only cannot reconstruct the secret but also does not obtain any information about the secret. The proposed scheme achieves correctness and verifiability in an information-theoretic scenario. On the contrary, secrecy is only achieved under some computational assumption because we are using an underlying computationally secure cryptosystem, Paillier's probabilistic encryption scheme Π. For this reason, we must lower the security level of our proposed scheme and go to a computational scenario. We prove then that our PVSS scheme is semantically secure against passive adversaries using the Decisional Composite Residuosity Assumption (DCRA), defined in Subsection 1.3.3.

In the standard IND-CPA game an adversary $\mathcal{A}_\Pi$, who knows the public key pk of the encryption scheme Π and encrypts messages of its choice, selects two different messages $m_0, m_1 \in \mathcal{M}$. Then, on input $z_b = Enc(m_b, r)$, where $r \in \mathcal{R}$ is a random element and $b \in \{0, 1\}$ is a random bit, the adversary tries to guess b by outputting a bit b'. The advantage of adversary $\mathcal{A}_\Pi$ is defined as $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$, where λ is the security parameter. We say that the encryption scheme Π is IND-CPA secure if this advantage is a negligible function in λ, for any polynomial-time adversary $\mathcal{A}_\Pi$.

The computational security of a PVSS scheme is defined by the following game $\mathcal{G}$ between a challenger and an adversary $\mathcal{A}_\Omega$.

1. The adversary $\mathcal{A}_\Omega$ publishes the set of n players $\mathcal{P}$ and the threshold access structure $\Gamma = (t, n)$.
2. The challenger runs $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}\left(1^\lambda, \mathcal{P}, \Gamma\right)$ and sends params to $\mathcal{A}_\Omega$.
3. The adversary $\mathcal{A}_\Omega$ manipulates a subset $B \subset \mathcal{P}$ of corrupted players and broadcasts it. In the worst case, $|B| = t - 1$.
4. Let $\mathcal{S}$ be the set of the secrets, $\mathcal{A}_\Omega$ broadcasts two different secrets $\bar{s}_0 \neq \bar{s}_1$ such that $\bar{s}_0, \bar{s}_1 \in \mathcal{S}$.
5. [**Challenge**] The challenger chooses a bit $b \in \{0, 1\}$ at random and runs $(\mathsf{out}_{\mathsf{pub}}, \{d_i\}_{P_i \in \mathcal{P}}, \{m_i\}_{P_i \in \mathcal{P}}) \leftarrow \Omega.\mathsf{Dist}\left(\mathsf{params}, \bar{s}_b\right)$, playing the roles of both the dealer and the players. The challenger sends to $\mathcal{A}_\Omega$ the public values and also the secret information of the corrupted players $P_i \in B$ on running this algorithm.
6. Finally, $\mathcal{A}_\Omega$ outputs a bit b'.

The advantage of $\mathcal{A}_\Omega$ in breaking the security of the PVSS scheme Ω is defined as $\mathsf{Adv}_{\mathcal{A}_\Omega}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$. Similarly to the encryption scheme, we say that the scheme Ω enjoys semantic security if the advantage $\mathsf{Adv}_{\mathcal{A}_\Omega}(\lambda)$ is a negligible function in λ, for any polynomial-time adversary $\mathcal{A}_\Omega$.

We are going to reduce now the computational security of the proposed PVSS scheme Ω to the $\mathsf{IND\text{-}CPA}$ security of the underlying homomorphic Paillier's encryption scheme Π.

Theorem 3.1.4 *For any adversary $\mathcal{A}_\Omega$ against the proposed PVSS scheme Ω, there exists an adversary $\mathcal{A}_\Pi$ against the encryption scheme Π with the same advantage.*

Proof. Let $\mathcal{A}_\Omega$ be an adversary against the semantic security (game $\mathcal{G}$) of the proposed PVSS scheme Ω. We are going to construct an adversary $\mathcal{A}_\Pi$ against the $\mathsf{IND\text{-}CPA}$ security of the Paillier's encryption scheme Π, which will use $\mathcal{A}_\Omega$ as subroutine. The input of adversary $\mathcal{A}_\Pi$ is the tuple (N, g) and its goal to win some advantage $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda)$ from two different encrypted messages.

Note that the secret information of the corrupted participants $P_i \in B$ corresponds to the tuple (m_i, r_i, s_i) for every of them. Whereas public information is $\mathsf{params} = (N, g)$, $\mathsf{out}_{\mathsf{pub}} = (\{c_i\}_{1 \leq i \leq n}, \{A_j\}_{0 \leq j \leq t-1}, \{t_i\}_{1 \leq i \leq n})$ and $\{d_i\}_{P_i \in \mathcal{P}}$. In our case the set of secrets $\mathcal{S} = \mathbb{Z}_N$ is the same as the set of messages $\mathcal{M} = \mathbb{Z}_N$.

For the participants $\mathcal{P}$ and the threshold access structure $\Gamma = (t, n)$ chosen by $\mathcal{A}_\Omega$, the adversary $\mathcal{A}_\Pi$ simulates the algorithm $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}\left(1^\lambda, \mathcal{P}, t\right)$ just sending

the same values N and g from its own input to $\mathcal{A}_\Omega$. For the two different secrets $\bar{s}_0, \bar{s}_1 \in \mathbb{Z}_N$ chosen by $\mathcal{A}_\Omega$, the adversary $\mathcal{A}_\Pi$ takes them and gives the messages $\bar{m}_0 = \bar{s}_0 \in \mathbb{Z}_N$ and $\bar{m}_1 = \bar{s}_1 \in \mathbb{Z}_N$ to its own challenger. The challenger of $\mathcal{A}_\Pi$ chooses a random bit $b \in \{0, 1\}$, a random value $\rho \leftarrow \mathbb{Z}_N^*$ and sends back the encryption $z_b = Enc(\bar{m}_b, \rho)$ of $\bar{m}_b$ to $\mathcal{A}_\Pi$.

The goal from adversary $\mathcal{A}_\Pi$ is to guess the bit b. $\mathcal{A}_\Pi$ chooses random $d_i \leftarrow \mathbb{Z}_N$, $t_i \leftarrow \mathbb{Z}_N^*$, for all $i \in \{1, .., n\}$ and also $r_i \leftarrow \mathbb{Z}_N^*$, $s_i \leftarrow \mathbb{Z}_N$, for all the corrupted participants $P_i \in B$. In the worst case, we can assume that $B = \{1, \ldots, t-1\}$ without loss of generality. After that, $\mathcal{A}_\Pi$ constructs the other elements m_i, A_j, c_i in input of $\mathcal{A}_\Omega$ as following:

- $m_i = d_i - s_i$ such that $1 \leq i \leq t-1$.
- To construct the value A_0 the adversary $\mathcal{A}_\Pi$ defines $A_0 = z_b$. Now, we are going to construct the remaining A_j, where $1 \leq j \leq t-1$ using A_0 and (r_i, s_i), for every $P_i \in B$: there exists a unique interpolating polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \in \mathbb{Z}_N[x]$ such that $f(i) = s_i$ and $f(0) = \bar{s}_b$ (unknown). Thus all the coefficients can be uniquely determined for some efficiently computable constants ν_{ij} (that only depend on B) as $a_j = \nu_{0j} \cdot \bar{s}_b + \sum_{i=1}^{t-1} \nu_{ij} \cdot s_i$. Therefore, $g^{a_j} = (g^{\bar{s}_b})^{\nu_{0j}} \cdot \prod_{i=1}^{t-1} (g^{s_i})^{\nu_{ij}}$ which enables us to compute $A_j = (g^s \cdot \rho^N)^{\nu_{0j}} \cdot \prod_{i=1}^{t-1} (g^{s_i} \cdot r_i^N)^{\nu_{ij}} = A_0^{\nu_{0j}} \cdot \prod_{i=1}^{t-1} Enc(s_i, r_i)^{\nu_{ij}}$, for all j such that $1 \leq j \leq t-1$.
- $c_i = \dfrac{Enc(d_i, t_i)}{A_0 \cdot {A_1}^i \cdots A_{t-1}^{i^{t-1}}}$, for all $i \in \{1, \ldots, n\}$. For $i \in \{1, \ldots, t-1\}$, these c_i can be alternatively computed using $c_i = Enc(m_i, r_i)$.

So adversary $\mathcal{A}_\Pi$ gives $(\{c_i\}_{1 \leq i \leq n}, \{A_j\}_{0 \leq j \leq t-1}, \{t_i\}_{1 \leq i \leq n}, \{d_i\}_{1 \leq i \leq n})$ to $\mathcal{A}_\Omega$ simulating its view for the distribution of secret $s_b \in \mathbb{Z}_N$. At this point $\mathcal{A}_\Omega$ guesses the bit b' that is also used as the output of $\mathcal{A}_\Pi$. It is straightforward to prove that the probability that $\mathcal{A}_\Omega$ wins the game $\mathcal{G}$ is exactly the same as that $\mathcal{A}_\Pi$ guesses the bit $b' = b$. Then $\mathcal{A}_\Pi$ has exactly the same advantage as $\mathcal{A}_\Omega$, and runs in the same time as $\mathcal{A}_\Omega$ plus the running time of the distribution protocol. □

The above result proves that the proposed PVSS scheme is semantically secure because the underlying Paillier's encryption scheme is IND-CPA secure under DCRA assumption.

3.1.4 Achieving Robustness

Up to now we have explained how to detect dishonest users and how to complete the protocols in case of sufficient honest users. The PVSS scheme with slight modifications

can be extended to ensure that the protocols are always completed in case of dishonest users. This property is called *robustness*.

Let $\mathcal{P}$ be the set of n participants. We need t of them to recover the secret and we have, in the worst case, $t-1$ corrupt participants. Therefore, we need to impose $n \geq 2t-1$ to achieve robustness, whereas before it was necessary only that $n \geq t$. We describe now the additional steps based on the original protocols to detect and reject the dishonest users.

In the step 2(a) every participant must broadcast c_i. If these c_i are not suitable, e.g. are not in $\mathbb{Z}_{N^2}^*$ or the participants broacast nothing, then these participants are sent off from the protocol. Hence we have some $\mathcal{P}_1 \subseteq \mathcal{P}$, which contains the participants who were not sent off in the step 2(a).

In steps 2(c) and 2(d) D must broadcast d_i and A_j, t_i respectively. In step 3(a) every participant checks if $c_i \cdot A_0 \cdot {A_1}^i \cdots A_{t-1}^{i^{t-1}} = Enc(d_i, t_i)$. If one of these three steps has not satisfactorily occurred then the honest participants accuse D using the broadcast channel and they drop out of the protocol.

From now on we can suppose that D is honest. In the step 4(a) every participant of $\mathcal{P}_1$ broadcasts his own (m_j, r_j) and checks if $c_j = Enc(m_j, r_j)$ from the other participants of $\mathcal{P}_1$. This leads to define $\mathcal{P}_2 = \{P_j \mid P_j \in \mathcal{P}_1, P_j \text{ broadcast his own } (m_j, r_j), c_j = Enc(m_j, r_j)\}$. Then the participants of $\mathcal{P}_1$ who are not in $\mathcal{P}_2$ are sent off and the participants of $\mathcal{P}_2$ can recover the secret using t shares, which they have computed, and interpolation. With regard to the steps which have not been mentioned, they remain the same.

In order to prove that the participants of $\mathcal{P}_2$ can recover the secret, we prove the following theorem using the same notation as above.

Theorem 3.1.5 *Let A be an authorized subset such that $A \subseteq \mathcal{P}_1$ and $|A| \geq n_1 - (n - 2t+1)$, where $|P_1| = n_1$ and $n \geq 2t-1$. If there is an active adversary who controls at most $t-1$ participants then during the execution protocols one of the following holds:*

1. *All honest participants realize that D is corrupt and nobody can recover the secret.*

2. *All honest participants recover the secret.*

Proof. Following the protocol above we note that $n_1 \geq n - (t-1) \geq t$ and that we have at most $(t-1) - (n-n_1)$ corrupt participants to uncover when the step 2(a) is executed.

If D is corrupt then all the honest participants accuse D and drop out of the protocol. That means that at most $t-1$ participants (the corrupts) remain in the protocol and can not recover any secret.

If D is not corrupt then we follow with step 4. As there are at most $(t-1)-(n-n_1)$ corrupt participants in A and, by hypothesis, $|A| \geq n_1 - (n - 2t + 1)$ then there is

at least t honest participants in A. Hence, since all honest participants of A are in $\mathcal{P}_2$ we get that $|\mathcal{P}_2| \geq t$ and all honest participants can recover the secret with any t different shares. □

3.2 Multi-Secret Sharing Schemes

In a multi-secret sharing scheme (MSSS), ℓ different secrets are distributed among the players in some set $\mathcal{P} = \{P_1, \ldots, P_n\}$, each one according to a (possibly different) access structure. The trivial solution to design a MSSS is to run ℓ independent standard secret sharing schemes, one for each secret and access structure. In this case, the length of the secret share to be stored by each player grows linearly with ℓ.

Multi-secret sharing schemes have been studied, *per se*, in different works. As far as we know, no specific application of a MSSS into a more general scenario or cryptographic protocol has been explicitly proposed. Most of the works on MSSSs have focused on unconditionally secure MSSSs. Blundo et al. [9] introduced a strong definition for the unconditional security of a MSSS, and gave some lower bounds on the length of the secret shares to be stored in a MSSS enjoying that level of security. Masucci proposed in [67] a weaker (althoug still information-theoretic) notion of security for MSSSs, and also gave some lower bounds on the length of secret shares for schemes enjoying the two notions. For some particular families of access structures, which include the threshold case where each access structure is defined by a threshold value, the results in [9, 67] imply that the length of each secret share in a MSSS with the strong level of unconditional security must be linear in ℓ. Therefore, the optimal solution in this strong scenario is equivalent to running ℓ independent instances of a standard secret sharing scheme, one for each secret and access structure.

The first result in this section is a proof that this is also the case for MSSSs enjoying security in the weaker (but still information-theoretic) sense proposed by Masucci. That is, we show that for some lists of access structures (in particular, when all of them are threshold ones), the length of each secret share in a MSSS for these access structures will be linear in ℓ, if the MSSS enjoys weaker unconditional security. In other words, we prove that multi-threshold secret sharing schemes enjoying information-theoretic security must have shares which are as long as the secret. This result is quite negative because cryptographic primitives using these MSSSs in their key generation protocols have then secret shares of information at least as long as those provided by the trivial solution, whose length grows linearly in ℓ. For this reason, we will move to the weaker setting of MSSSs with computational security.

3.2.1 A New Result in the Information-Theoretic Scenario

Blundo *et al.* introduced in [9] the notion of *multi-secret sharing schemes*: ℓ secrets $s_1, \ldots, s_\ell \in \mathcal{K}$ are distributed at the same time between a set $\mathcal{P}$ of n players, according to ℓ access structures $\Gamma_1, \ldots, \Gamma_\ell \subset 2^{\mathcal{P}}$. Again, sh_i denotes the share of secret information received by each player P_i in the distribution phase. The reconstruction phase takes as input a subset of shares and an index $j \in \{1, 2, \ldots, \ell\}$, and the expected output is the secret s_j. In [9], two requirements are defined for multi-secret sharing schemes, one related to correctness and one related to information-theoretic privacy.

1. *Correctness.* If the reconstruction phase takes as input a subset of shares $\{\mathsf{sh}_i\}_{P_i \in A}$ and an index j, and $A \in \Gamma_j$, then the recovered secret is actually s_j. In other words, $H(S_j|\mathsf{SH}_A) = 0$ for any subset $A \in \Gamma_j$.

2. *Strong information-theoretic security.* From the knowledge of a non-authorized subset of shares $\{\mathsf{sh}_i\}_{P_i \in B}$, with $B \notin \Gamma_j$, and of some secrets, different from s_j, the information obtained on the secret s_j is the same as if the shares $\{\mathsf{sh}_i\}_{P_i \in B}$ were not known. In the entropy language: for any subset $B \notin \Gamma_j$ and any subset $T \subset \{S_1, \ldots, S_\ell\} \backslash \{S_j\}$, it holds $H(S_j|\mathsf{SH}_B, T) = H(S_j|T)$.

This strong security requirement has an impact on the efficiency of multi-secret sharing schemes. Blundo *et al.* give in [9] lower bounds for the size of the shares sh_i in such a strongly secure multi-secret sharing scheme. In particular, for the case of *multi-threshold* secret sharing schemes[2], where $\Gamma_j = \{A \subset \mathcal{P} : |A| \geq t_j\}$ and $1 \leq t_1 < t_2 < \ldots < t_\ell \leq n$, Blundo *et al.* proved that the entropy $H(\mathsf{SH}_i)$ of each individual share sh_i must be greater than or equal to the entropy $H(S)$ of the global secret $S = (S_1, \ldots, S_\ell)$, in any multi-threshold secret sharing scheme satisfying this strong security condition. This means that running ℓ independent instances of Shamir's threshold secret sharing scheme gives an optimal multi-threshold secret sharing scheme.

Other works [48, 67] consider a weaker (but maybe more realistic in actual applications of secret sharing) security notion for multi-secret sharing schemes, which does not consider the possibility that the adversary obtains some other subset T of secrets.

- *Weak information-theoretic security.* No information at all on the secret s_j can be obtained from a non-authorized subset of shares $\{\mathsf{sh}_i\}_{P_i \in B}$, with $B \notin \Gamma_j$. In the entropy language: for any subset $B \notin \Gamma_j$, it holds $H(S_j|\mathsf{SH}_B) = H(S_j)$.

[2] We stress that the name of multi-threshold secret sharing has been previously used in other works [48, 67] for a different kind of access structures.

Masucci gives in [67] lower bounds for the size of the shares sh_i in weakly information-theoretically secure multi-secret sharing schemes. However, these bounds do not lead to any result for the family of multi-threshold access structures that we consider in this thesis. Therefore, according to the results that we have up to now, it may still be possible to design a multi-threshold secret sharing scheme which enjoys weak information-theoretic security, and where the share of some participant is shorter than the secret. However, we prove below in Corollary 3.2.6 that this cannot be the case.

Before proving it, we define here the notion of *entropy* (introduced in Subsection 1.5.1) and some results, which will be used in Theorem 3.2.5. Let X be a random variable that takes values in a finite set X. For any $x \in \mathsf{X}$, let $p(x) = \Pr[X = x]$ be the probability that X takes the value x. The entropy $H(X)$ of X is defined as

$$H(X) = -\sum_{x \in \mathsf{X}} p(x) \cdot \log(p(x)),$$

where $0 \cdot \log 0$ should be treated as being equal to zero. The entropy $H(X)$ measures the uncertainty on the value taken by the random variable X. It always satisfies $0 \leq H(X) \leq \log |\mathsf{X}|$. The minimum value $H(X) = 0$ is achieved if and only if there exists $x_0 \in \mathsf{X}$ such that $p(x_0) = 1$, and the maximum value $H(X) = \log |\mathsf{X}|$ is achieved if and only if the probability is distributed uniformly (that is, $p(x) = 1/|\mathsf{X}|$ for all $x \in \mathsf{X}$).

Given two random variables X, Y, their *joint entropy* is defined as

$$H(X,Y) = -\sum_{(x,y) \in \mathsf{X} \times \mathsf{Y}} p(x,y) \cdot \log(p(x,y)),$$

where $p(x,y) = \Pr[X = x \text{ and } Y = y]$.

If we denote $p(x|y) = \Pr[X = x \mid Y = y]$, then the *conditional entropy* $H(X|Y)$ is defined as

$$H(X|Y) = -\sum_{x \in \mathsf{X}} \sum_{y \in \mathsf{Y}} p(y)p(x|y) \cdot \log(p(x|y)),$$

and it satisfies $H(X|Y) = H(X,Y) - H(Y)$.

Similarly, we can define $H(X|\, Y, Z)$ or $H(X, Y\, |Z)$, for random variables X, Y, Z. We put now the following well-known results about the entropy of random variables. They are more or less easy to deduce from the previous definitions, so we only include one of the proofs, as an illustrative example.

Lemma 3.2.1 *For all random variables X, Y, it holds:*

(i) $H(X) \geq H(X|Y) \geq 0$, *and*

(ii) $H(X) + H(Y) \geq H(X,Y)$.

Lemma 3.2.2 *For all random variables X, Y, Z, if $H(X|Y) = 0$, then $H(Z|X) \geq H(Z|Y)$.*

Proof. First of all, it is easy to see that $H(Z|X) \geq H(Z|\ X,Y)$ and $H(Z,X\ |Y) \geq H(Z|Y)$, for any random variables X, Y, Z. Now we have

$$H(Z|X) \geq H(Z|\ X,Y) = H(Z,X,Y) - H(X,Y) = H(Z,X,Y) - \big(H(Y) + H(X|Y)\big) \overset{(*)}{=}$$

$$= H(Z,X,Y) - H(Y) = H(Z,X\ |Y) \geq H(Z|Y),$$

where we have used in $(*)$ the fact that $H(X|Y) = 0$. □

Lemma 3.2.3 *For all random variables X, Y, if $H(X|Y) = H(X)$, then $H(X,Y) = H(X) + H(Y)$.*

Lemma 3.2.4 *For all random variables X, Y, if $H(X|Y) = 0$, then:*

(i) $H(X,Y) = H(Y)$, *and*

(ii) $H(Z|\ X,Y) = H(Z|Y)$, *for all random variable Z.*

Theorem 3.2.5 *Let $\Gamma_1, \ldots, \Gamma_\ell \subset 2^{\mathcal{P}}$ be ℓ access structures, and let $P_i \in \mathcal{P}$. Assume there exist subsets of players $B_1 \subset B_2 \subset \ldots \subset B_\ell \subset \mathcal{P} - \{P_i\}$ satisfying, for all $j = 1, \ldots, \ell$, the following three conditions:*

(i) $B_j \in \Gamma_{j-1}$. *(Here, for index $j = 1$, we assume $\Gamma_0 = 2^{\mathcal{P}}$.)*

(ii) $B_j \notin \Gamma_j$,

(iii) $B_j \cup \{P_i\} \in \Gamma_j$,

Then, for any multi-secret sharing scheme for $\Gamma_1, \ldots, \Gamma_\ell$ with weak unconditional security, it holds $H(\mathsf{SH}_i) \geq \sum_{j=1}^{\ell} H(S_j)$.

Proof. We proceed iteratively to deduce a sequence of equalities and inequalities:

$$H(\mathsf{SH}_i) + \sum_{j=1}^{\ell} H(\mathsf{SH}_{B_j}) = H(\mathsf{SH}_i) + H(\mathsf{SH}_{B_1}) + \sum_{j=2}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(1)}{\geq}$$

$$H(\mathsf{SH}_i, \mathsf{SH}_{B_1}) + \sum_{j=2}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(2)}{=} H(\mathsf{SH}_i, \mathsf{SH}_{B_1}, S_1) + \sum_{j=2}^{\ell} H(\mathsf{SH}_{B_j}) =$$

$$H(\mathsf{SH}_{B_1}, S_1) + H(\mathsf{SH}_i|\mathsf{SH}_{B_1}, S_1) + \sum_{j=2}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(3)}{=}$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(\mathsf{SH}_i|\mathsf{SH}_{B_1}, S_1) + \sum_{j=2}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(4)}{\geq}$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(\mathsf{SH}_i|\mathsf{SH}_{B_2}, S_1) + \sum_{j=2}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(5)}{=}$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(\mathsf{SH}_i|\mathsf{SH}_{B_2}) + H(\mathsf{SH}_{B_2}) + \sum_{j=3}^{\ell} H(\mathsf{SH}_{B_j}) =$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(\mathsf{SH}_i, \mathsf{SH}_{B_2}) + \sum_{j=3}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(2)}{=}$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(\mathsf{SH}_i, \mathsf{SH}_{B_2}, S_2) + \sum_{j=3}^{\ell} H(\mathsf{SH}_{B_j}) =$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(\mathsf{SH}_{B_2}, S_2) + H(\mathsf{SH}_i|\mathsf{SH}_{B_2}, S_2) + \sum_{j=3}^{\ell} H(\mathsf{SH}_{B_j}) \overset{(3)}{=}$$

$$H(S_1) + H(\mathsf{SH}_{B_1}) + H(S_2) + H(\mathsf{SH}_{B_2}) + H(\mathsf{SH}_i|\mathsf{SH}_{B_2}, S_2) + \sum_{j=3}^{\ell} H(\mathsf{SH}_{B_j}) \geq$$

$$\ldots \geq \sum_{j=1}^{\ell-1} \big(H(S_j) + H(\mathsf{SH}_{B_j})\big) + H(\mathsf{SH}_i|\mathsf{SH}_{B_{\ell-1}}, S_{\ell-1}) + H(\mathsf{SH}_{B_\ell}) \overset{(4)}{\geq}$$

$$\sum_{j=1}^{\ell-1} \big(H(S_j) + H(\mathsf{SH}_{B_j})\big) + H(\mathsf{SH}_i|\mathsf{SH}_{B_\ell}, S_{\ell-1}) + H(\mathsf{SH}_{B_\ell}) \overset{(5)}{=}$$

$$\sum_{j=1}^{\ell-1} \big(H(S_j) + H(\mathsf{SH}_{B_j})\big) + H(\mathsf{SH}_i|\mathsf{SH}_{B_\ell}) + H(\mathsf{SH}_{B_\ell}) =$$

$$\sum_{j=1}^{\ell-1} \big(H(S_j) + H(\mathsf{SH}_{B_j})\big) + H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}) \overset{(2)}{=}$$

$$\sum_{j=1}^{\ell-1} \big(H(S_j) + H(\mathsf{SH}_{B_j})\big) + H(\mathsf{SH}_i, \mathsf{SH}_{B_\ell}, S_\ell) =$$

$$\sum_{j=1}^{\ell-1} \big(H(S_j) + H(\mathsf{SH}_{B_j})\big) + H(\mathsf{SH}_{B_\ell}, S_\ell) + H(\mathsf{SH}_i|\mathsf{SH}_{B_\ell}, S_\ell) \overset{(3)}{=}$$

$$\sum_{j=1}^{\ell-1}\left(H(S_j)+H(\mathsf{SH}_{B_j})\right)+H(\mathsf{SH}_{B_\ell})+H(S_\ell)+H(\mathsf{SH}_i|\mathsf{SH}_{B_\ell},S_\ell)\overset{(6)}{\geq}$$

$$\sum_{j=1}^{\ell-1}\left(H(S_j)+H(\mathsf{SH}_{B_j})\right)+H(\mathsf{SH}_{B_\ell})+H(S_\ell)=\sum_{j=1}^{\ell}\left(H(S_j)+H(\mathsf{SH}_{B_j})\right).$$

From the first and last terms of this sequence of equalities and inequalities, we deduce the desired result $H(\mathsf{SH}_i)\geq\sum_{j=1}^{\ell}H(S_j)$. In the sequence, we have used the previous lemmas from entropy theory, in the following way:

- inequality $\overset{(1)}{\geq}$ is deduced from Lemma 3.2.1 (ii),
- equalities $\overset{(2)}{=}$ are deduced from Lemma 3.2.4 (i), because $B_j\cup\{P_i\}\in\Gamma_j$, for all $j=1,\ldots,\ell$,
- equalities $\overset{(3)}{=}$ are deduced from Lemma 3.2.3, because $B_j\notin\Gamma_j$, for all $j=1,\ldots,\ell$,
- inequalities $\overset{(4)}{\geq}$ are deduced from Lemma 3.2.2, applied to SH_{B_j} and $\mathsf{SH}_{B_{j+1}}$, because $B_j\subset B_{j+1}$,
- equalities $\overset{(5)}{=}$ are deduced from Lemma 3.2.4 (ii), because $B_j\in\Gamma_{j-1}$, for all $j=2,\ldots,\ell$,
- finally, inequality $\overset{(6)}{\geq}$ is deduced from Lemma 3.2.1 (i).

□

Corollary 3.2.6 *For any multi-threshold secret sharing scheme for thresholds* $1\leq t_1<t_2<\ldots<t_\ell\leq n$ *that enjoys weak information-theoretic security, it holds* $H(\mathsf{SH}_i)\geq\sum_{j=1}^{\ell}H(S_j)$, *for any player* $P_i\in\mathcal{P}$.

Proof. Just apply the previous theorem to a sequence of subsets $B_1\subset B_2\subset\ldots B_\ell\subset\mathcal{P}-\{P_i\}$ such that $|B_j|=t_j-1$, for all $j=1,2,\ldots,\ell$. □

This means that the optimal multi-threshold secret sharing scheme with weak unconditional security, in terms of the ratio between the length of shares and the

length of the global secret, is equivalent to running ℓ independent instances of Shamir's secret sharing scheme.

We stress that the three conditions in the statement of Theorem 3.2.5 imply that all the access structures must be different. If there was some repeated access structure, but the non-repeated ones did still satisfy these three conditions, then some variations of the theorem could be easily proved. For instance, if $\Gamma_1 = \Gamma_2$ but the rest of access structures satisfy the conditions, then we can ensure that $H(\mathsf{SH}_i) \geq \sum_{j=2}^{\ell} H(S_j)$ for all player $P_i \in \mathcal{P}$.

As an explicit example where Theorem 3.2.5 cannot be applied, and where $H(\mathsf{SH}_i) < \sum_{j=1}^{\ell} H(S_j)$, let us consider the case of ℓ threshold access structures with the same threshold: $\Gamma_j = T(t,n)$ for all $j \in \{1,\ldots,\ell\}$. We can share the global secret $\vec{s} = (s_1,\ldots,s_\ell) \in (\mathbb{Z}_p)^\ell$ by following the ideas proposed in [27], provided $\ell \leq t(n-t)$. For some big prime number p, there are ℓ values $x_{j,0} \in \mathbb{Z}_p$, for $j \in \{1,\ldots,\ell\}$, assigned to the secrets, and there are $n-t$ values $x_{i,k}$ assigned to player P_i, for $k \in \{1,\ldots,n-t\}$. All these values must be pairwise different and public. To distribute the secret $\vec{s}$, a random polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree $t(n-t)-1$ is chosen, such that $f(x_{j,0}) = s_j$ for all $j \in \{1,\ldots,\ell\}$. Player P_i receives the share $\mathsf{sh}_i = (f(x_{i,1}),\ldots,f(x_{i,n-t})) \in (\mathbb{Z}_p)^{n-t}$. If t players work together, they can interpolate the polynomial $f(x)$ at any point and recover any of the secrets, because they hold $t(n-t)$ evaluations of the polynomial, which has degree $t(n-t)-1$. If less than t players cooperate, they obtain no information on any secret s_j, for $j \in \{1,\ldots,\ell\}$, and so the scheme enjoys weak unconditional security. If $n-t < \ell$, then the length of each sh_i is strictly smaller than the length of the global secret $\vec{s}$.

Anyway, for the applications of multi-secret sharing that we have in mind, we are looking for efficient ways of sharing ℓ secrets for ℓ different access structures, in particular for the threshold case. The result in Theorem 3.2.5, although very interesting from a theoretical point of view, is quite negative for our interests, and we thus move to the scenario of computationally secure multi-secret sharing. This is not a big problem, taking into account that our final goal is to use multi-secret sharing schemes as an ingredient to implement cryptographic primitives (multi-policy distributed decryption and signatures) whose security is going to be at most computational, in any case.

3.2.2 Computational Security for Multi-Secret Sharing Schemes

A multi-secret sharing scheme $\Omega = (\Omega.\mathsf{Stp}, \Omega.\mathsf{Dist}, \Omega.\mathsf{Rec})$ consists of three protocols. The setup protocol takes as input a security parameter $\lambda \in \mathbb{N}$, the set of

players $\mathcal{P}$ and the ℓ different access structures $\Gamma_1, \ldots, \Gamma_\ell$, and outputs some public and common parameters params for the scheme (such as mathematical groups, hash functions, etc.). We implicitly assume that params also contains the descriptions of $\mathcal{P}$ and the access structures. We denote an execution of this protocol as $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}\left(1^\lambda, \mathcal{P}, \{\Gamma_j\}_{1\leq j\leq \ell}\right)$.

The distribution protocol takes as input the parameters params and the global secret $\vec{s} = (s_1, \ldots, s_\ell)$ to be distributed, and produces the set of shares $\{\mathsf{sh}_i\}_{P_i\in\mathcal{P}}$ and possibly some public output $\mathsf{out}_{\mathsf{pub}}$. We write $(\mathsf{out}_{\mathsf{pub}}, \{\mathsf{sh}_i\}_{P_i\in\mathcal{P}}) \leftarrow \Omega.\mathsf{Dist}\left(\mathsf{params}, \vec{s}\right)$.

The reconstruction protocol takes as input params, $\mathsf{out}_{\mathsf{pub}}$, an index $j \in \{1, \ldots, \ell\}$, and the shares $\{\mathsf{sh}_i\}_{P_i\in A}$ of the players in some subset $A \subset \mathcal{P}$, and outputs a possible value $\tilde{s}_j$ for the j-th secret. We write $\tilde{s}_j \leftarrow \Omega.\mathsf{Rec}\left(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, j, \{\mathsf{sh}_i\}_{P_i\in A}\right)$.

For correctness, we require that, for any index $j \in \{1, \ldots, \ell\}$ and any subset $A \in \Gamma_j$, it holds

$$\Omega.\mathsf{Rec}\left(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, j, \{\mathsf{sh}_i\}_{P_i\in A}\right) = s_j,$$

if the setup protocol has produced the values $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}\left(1^\lambda, \mathcal{P}, \{\Gamma_j\}_{1\leq j\leq \ell}\right)$ and $(\mathsf{out}_{\mathsf{pub}}, \{\mathsf{sh}_i\}_{P_i\in\mathcal{P}}) \leftarrow \Omega.\mathsf{Dist}\left(\mathsf{params}, \vec{s}\right)$ is a distribution of the global secret $\vec{s} = (s_1, \ldots, s_j, \ldots, s_\ell)$.

The computational security of a multi-secret sharing scheme is defined by the following game $\mathcal{G}$ between a challenger and an adversary $\mathcal{A}_\Omega$.

1. The adversary $\mathcal{A}_\Omega$ publishes the set of players $\mathcal{P}$ and the ℓ access structures $\Gamma_1, \ldots, \Gamma_\ell \subset 2^{\mathcal{P}}$.

2. The challenger runs $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}\left(1^\lambda, \mathcal{P}, \{\Gamma_j\}_{1\leq j\leq \ell}\right)$ and sends params to $\mathcal{A}_\Omega$.

3. The adversary $\mathcal{A}_\Omega$ broadcasts a subset $\widetilde{B} \subset \mathcal{P}$ of corrupted players.

4. $\mathcal{A}_\Omega$ broadcasts two different global secrets $\vec{s}^{(0)} \neq \vec{s}^{(1)}$ with the following restriction:
$$s_j^{(0)} = s_j^{(1)}, \quad \forall j \in \{1, \ldots, \ell\} \text{ s.t. } \widetilde{B} \in \Gamma_j.$$

5. [**Challenge**] The challenger chooses at random $b \in \{0, 1\}$, runs the protocol $(\mathsf{out}_{\mathsf{pub}}, \{\mathsf{sh}_i\}_{P_i\in\mathcal{P}}) \leftarrow \Omega.\mathsf{Dist}\left(\mathsf{params}, \vec{s}^{(b)}\right)$, and sends $\left(\mathsf{out}_{\mathsf{pub}}, \{\mathsf{sh}_i\}_{P_i\in\widetilde{B}}\right)$ to $\mathcal{A}_\Omega$.

6. Finally, $\mathcal{A}_\Omega$ outputs a bit b'.

The advantage of $\mathcal{A}_\Omega$ in breaking the security of the multi-secret sharing scheme Ω is defined as

$$\mathsf{Adv}_{\mathcal{A}_\Omega}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

The scheme Ω is said to enjoy computational security if $\mathsf{Adv}_{\mathcal{A}_\Omega}(\lambda)$ is a negligible function in λ, for any polynomial-time adversary $\mathcal{A}_\Omega$.

3.3 A First Computationally Secure Multi-Secret Sharing Scheme

We propose and analyze in this section a computationally secure multi-secret sharing scheme, Ω_1, which essentially consists in repeating ℓ times, in parallel, (a modification of) the secret sharing scheme of Krawczyk [55]. We consider, for simplicity, the case where all the access structures are threshold ones: $\Gamma_j = T(t_j, n)$, for all $j \in \{1, \ldots, \ell\}$. The protocols of the scheme are detailed below.

Setup: $\Omega_1.\mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$.
Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of n users and let $1 \leq t_1 \leq t_2 \leq \ldots \leq t_\ell \leq n$ be the ℓ thresholds that define the access structures $\{\Gamma_j\}_{1 \leq j \leq \ell}$. A secure symmetric encryption scheme $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ with key space $\mathcal{K}$, plaintext space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is chosen, such that $\mathcal{M}$ contains the space of secrets to be shared. Let q be a prime number, $q > n$, such that $\mathcal{K} \subset \mathbb{Z}_q$. Each player P_i is assigned the value i. The public parameters are $\mathsf{params} = (q, \Pi)$.

Distribution of the Shares: $\Omega_1.\mathsf{Dist}(\mathsf{params}, \vec{s})$.
Let $\vec{s} = (s_1, \ldots, s_\ell) \in \mathcal{M}^\ell$ be the global secret to be distributed. For simplicity, we assume the distribution is done by an external dealer.

1. For $j = 1, \ldots, \ell$, run $K_j \leftarrow \Pi.\mathsf{KG}(1^\lambda)$.
2. For $j = 1, \ldots, \ell$, compute $c_j \leftarrow \Pi.\mathsf{Enc}(s_j\ ,\ K_j)$.
3. For $j = 1, \ldots, \ell$, use Shamir's secret sharing scheme to distribute the secret key $K_j \in \mathcal{K} \subset \mathbb{Z}_q$ according to Γ_j. That is, choose a random polynomial $f_j(x) \in \mathbb{Z}_q[x]$ of degree $t_j - 1$ such that $f_j(0) = K_j$, and compute the shares $K_j^{(i)} = f_j(i)$, for $i = 1, 2, \ldots, n$.
4. Each player P_i receives, through a secure channel, his secret share $\mathsf{sh}_i = (K_1^{(i)}, \ldots, K_\ell^{(i)}) \in (\mathbb{Z}_q)^\ell$.
5. The public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{c_j\}_{j \in \{1,\ldots,\ell\}}$.

Reconstruction of a Secret: $\Omega_1.\mathsf{Rec}(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, j, \{\mathsf{sh}_i\}_{P_i \in A})$.
When the players of an authorized subset $A \in \Gamma_j$ (i.e. $|A| \geq t_j$) want to recover the secret s_j, they run the following steps.

1. They use their secret values $\{K_j^{(i)}\}_{P_i \in A}$ to interpolate the polynomial $f_j(x)$ and recover the value $f_j(0) = K_j$.

2. They take c_j from $\mathsf{out}_{\mathsf{pub}}$ and run $s_j \leftarrow \Pi.\mathsf{Dec}(c_j, K_j)$ to recover the desired secret s_j.

The correctness of the scheme Ω_1 holds trivially.

3.3.1 Security Analysis

In this section we are going to reduce the computational security of the MSSS Ω_1 to the security of the underlying symmetric encryption scheme Π in the multi-user setting. The security of a symmetric encryption scheme Π is described by the game $\mathcal{G}_1$ defined in Subsection 1.4.1, whereas the security of the MSSS Ω_1 is described by the game $\mathcal{G}_2$ defined in Subsection 3.2.2.

Theorem 3.3.1 *For any adversary $\mathcal{A}_{\Omega_1}$ against the threshold MSS scheme Ω_1 that chooses ℓ threshold access structures, corrupts $t^\star$ players in a set $\mathcal{P}$ and chooses global secrets $\vec{s}^{(0)} \neq \vec{s}^{(1)}$, there exists a (k, q_e, q_d, q_c)-adversary $\mathcal{A}_\Pi$ against the encryption scheme Π, with advantage $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \mathsf{Adv}_{\mathcal{A}_{\Omega_1}}(\lambda)$ and parameters $k = \ell - |\mathbb{J}^\star|$, $q_e = q_d = 0$ and $q_c = \ell - |\mathbb{J}^\star|$, where $\mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } \vec{s}_j^{(0)} = \vec{s}_j^{(1)}\}$.*

Proof. Let $\mathcal{A}_{\Omega_1}$ be an adversary against the computational security of the multi-secret sharing scheme Ω_1. We are going to construct an adversary $\mathcal{A}_\Pi$ against the CCA security (game $\mathcal{G}_1$) of the symmetric encryption scheme Π, which will use $\mathcal{A}_{\Omega_1}$ as a sub-routine. When the game $\mathcal{G}_1$ starts, the challenger of this game chooses a secret bit $\beta \in \{0, 1\}$ at random. The goal of adversary $\mathcal{A}_\Pi$ will be to output a bit $\beta' \in \{0, 1\}$ such that $\beta' = \beta$ with probability significantly greater than $1/2$. To achieve it, $\mathcal{A}_\Pi$ initializes the adversary $\mathcal{A}_{\Omega_1}$ in a running of game $\mathcal{G}_2$.

$\mathcal{A}_{\Omega_1}$ starts the game $\mathcal{G}_2$ by choosing the set of users $\mathcal{P} = \{P_i\}_{1 \leq i \leq n}$ and ℓ access structures $\{\Gamma_j\}_{1 \leq j \leq \ell}$ with $\Gamma_j = T(t_j, n)$. Then, $\mathcal{A}_\Pi$ has to simulate a running of the protocol $\mathsf{params} \leftarrow \Omega_1.\mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$. To do this, $\mathcal{A}_\Pi$ simply chooses a prime number $q > n$, such that the key space $\mathcal{K}$ of the target symmetric encryption scheme Π satisfies $\mathcal{K} \subset \mathbb{Z}_q$, and sends $\mathsf{params} = (q, \Pi)$ to $\mathcal{A}_{\Omega_1}$.

At some point, $\mathcal{A}_{\Omega_1}$ chooses a subset $\widetilde{B} \subset \mathcal{P}$ of corrupted players, with $|\widetilde{B}| = t^\star$, and also chooses two different multi-secrets $\vec{s}^{(0)} \neq \vec{s}^{(1)}$, such that $s_j^{(0)} = s_j^{(1)}$ for all j satisfying $t^\star \geq t_j$. Let us define the subset of indices $\mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } \vec{s}_j^{(0)} = \vec{s}_j^{(1)}\}$. By definition, since $\vec{s}^{(0)} \neq \vec{s}^{(1)}$, we have that $\mathbb{J}^\star \subsetneq \{1, \ldots, \ell\}$ and, furthermore, if $t^\star \geq t_j$, then $j \in \mathbb{J}^\star$.

$\mathcal{A}_\Pi$ runs $\tilde{K}_j \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ for all $j \in \mathbb{J}^\star$. For these indices $j \in \mathbb{J}^\star$, $\mathcal{A}_\Pi$ chooses random polynomials $f_j(x) \in \mathbb{Z}_q[x]$ of degree $t_j - 1$ such that $f_j(0) = \tilde{K}_j$ and uses these polynomials to compute the shares $\tilde{K}_j^{(i)} = f_j(i)$, for all the corrupted participants $P_i \in \widetilde{B}$. Then $\mathcal{A}_\Pi$ defines $k = \ell - |\mathbb{J}^\star|$ (step 2 of game $\mathcal{G}_1$) and the challenger of this

game $\mathcal{G}_1$ runs k times the protocol $K_j \leftarrow \Pi.\mathsf{KG}(1^\lambda)$, for all $j \notin \mathbb{J}^\star$. For each corrupted participant $P_i \in \widetilde{B}$ and each index $j \notin \mathbb{J}^\star$, $\mathcal{A}_\Pi$ chooses at random $\tilde{K}_j^{(i)} \in \mathbb{Z}_q$. The secret key of each corrupted player $P_i \in \widetilde{B}$ is defined as $\mathsf{sh}_i = (\tilde{K}_1^{(i)}, \ldots, \tilde{K}_\ell^{(i)}) \in (\mathbb{Z}_q)^\ell$. Since $\widetilde{B} \notin \Gamma_j$ for all index $j \notin \mathbb{J}^\star$, the values $\{\tilde{K}_j^{(i)}\}_{P_i \in \widetilde{B}}$ are perfectly possible shares of the (unknown) secret K_j, and the shares $\{\mathsf{sh}_i\}_{P_i \in \widetilde{B}}$ of the corrupted players are consistent.

For the indices $j \in \mathbb{J}^\star$, $\mathcal{A}_\Pi$ computes the values $c_j = \Pi.\mathsf{Enc}(s_j^{(0)}, \tilde{K}_j)$. For the rest of indices $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$, the adversary $\mathcal{A}_\Pi$ sends the challenge queries $(j, s_j^{(0)}, s_j^{(1)})$ to its challenger (game $\mathcal{G}_1$). As the answers, $\mathcal{A}_\Pi$ receives $c_j = \Pi.\mathsf{Enc}(s_j^{(\beta)}, K_j)$ for all $j \notin \mathbb{J}^\star$. The number of challenge queries made by $\mathcal{A}_\Pi$ is $q_c = \ell - |\mathbb{J}^\star|$.

The public output is defined as $\mathsf{out}_{\mathsf{pub}} = \{c_j\}_{j \in \{1,\ldots,\ell\}}$. In this way, the adversary $\mathcal{A}_\Pi$ is perfectly simulating an execution of the distribution protocol $(\mathsf{out}_{\mathsf{pub}}, \{\mathsf{sh}_i\}_{P_i \in \mathcal{P}}) \leftarrow \Omega_1.\mathsf{Dist}\left(\mathsf{params}, \vec{s}^{(b)}\right)$, where $\vec{s}^{(b)} = (s_1^{(b)}, \ldots, s_\ell^{(b)})$ and $b = \beta$. Now $\mathcal{A}_\Pi$ sends $\mathsf{out}_{\mathsf{pub}}$ and the shares $\{\mathsf{sh}_i\}_{P_i \in \widetilde{B}}$ of the corrupted players to $\mathcal{A}_{\Omega_1}$.

As expected, the adversary $\mathcal{A}_{\Omega_1}$ broadcasts a bit $b' \in \{0, 1\}$ as his final output. $\mathcal{A}_\Pi$ outputs the same bit $\beta' = b'$ to conclude the game $\mathcal{G}_1$. Note that $\mathcal{A}_\Pi$ has not made neither encryption queries nor decryption queries. That is, $q_e = 0$ and $q_d = 0$.

The probability that $\mathcal{A}_{\Omega_1}$ wins game $\mathcal{G}_2$ (that is, the probability that $b' = b$) is $1/2 + \mathsf{Adv}_{\mathcal{A}_{\Omega_1}}(\lambda)$, by hypothesis. The probability that $\mathcal{A}_\Pi$ wins the game $\mathcal{G}_1$ (that is, the probability that $\beta' = \beta$) is exactly the same, since $\beta' = b'$ and $\beta = b$. Summing up, we have $1/2 + \mathsf{Adv}_{\mathcal{A}_{\Omega_1}}(\lambda) = 1/2 + \mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda)$, which leads to the desired result $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \mathsf{Adv}_{\mathcal{A}_{\Omega_1}}(\lambda)$. □

3.4 A Second Computationally Secure Multi-Secret Sharing Scheme

The MSSS described in the previous section is provably secure in the standard model and has the property that the shares of the players (roughly, in $\mathcal{K}^\ell$) are shorter than the global secret (roughly, in $\mathcal{M}^\ell$), because the key-space $\mathcal{K}$ is usually much smaller than the plaintext-space $\mathcal{M}$, in a symmetric encryption scheme. However, the length of each share sh_i still depends linearly on the number ℓ of secrets.

We propose and analyze in this section an alternative multi-secret sharing scheme, Ω_2, also for threshold access structures (for simplicity) and also with provable security in the standard model. This new scheme generalizes in some way some previous schemes [39, 62], which however did not have a formal security analysis / proof. In this second MSS scheme, the length of each share sh_i will be constant, independent

of ℓ. The protocols of the scheme are the following.

Setup: $\Omega_2.\mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$.
Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of n users and let $1 \leq t_1 \leq t_2 \leq \ldots \leq t_\ell \leq n$ be the ℓ thresholds that define the access structures $\{\Gamma_j\}_{1 \leq j \leq \ell}$. A prime number $p > n$ is chosen, such that p is λ bits long. A secure symmetric encryption scheme $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$, whose plaintext space $\mathcal{M}$ contains $\mathbb{Z}_p$, is chosen. Each player P_i is assigned the value i. The public parameters are $\mathsf{params} = (p, \Pi)$.

Distribution of the Shares: $\Omega_2.\mathsf{Dist}(\mathsf{params}, \vec{s})$.
Let $\vec{s} = (s_1, \ldots, s_\ell) \in (\mathbb{Z}_p)^\ell$ be the global secret to be distributed. For simplicity, we assume that the distribution is done by an external dealer.

1. For $i = 1, 2, \ldots, n$, run $K_i \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ and define the secret key of P_i to be $\mathsf{sh}_i = K_i$.
2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.
3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $c_{ij} = \Pi.\mathsf{Enc}(f_j(i)\ ,\ K_i)$.
4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{c_{ij}\}_{P_i \in \mathcal{P}, j \in \{1, \ldots, \ell\}}$.

Reconstruction of a Secret: $\Omega_2.\mathsf{Rec}(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, j, \{\mathsf{sh}_i\}_{P_i \in A})$.
If the players of an authorized subset $A \in \Gamma_j$ (i.e. $|A| \geq t_j$) want to recover the secret s_j, they act as follows.

1. Each player $P_i \in A$ computes $f_j(i) = \Pi.\mathsf{Dec}(c_{ij}, \mathsf{sh}_i)$, by decrypting the ciphertext c_{ij}, in $\mathsf{out}_{\mathsf{pub}}$, with secret key $\mathsf{sh}_i = K_i$.
2. Use the values $\{f_j(i)\}_{P_i \in A}$ to interpolate the polynomial $f_j(x)$ and recover the secret $s_j = f_j(0)$.

Note that the correctness of the proposed scheme Ω_2 holds directly via interpolation.

3.4.1 Security Analysis

In this section we are going to reduce the computational security of the MSS scheme Ω_2 to the security of the symmetric encryption scheme Π in the multi-user setting. Following the same notation as in the previous proposal, we will use $\mathcal{G}_1$ for the game (defined in Subsection 1.4.1) that describes the security of the symmetric encryption scheme Π and $\mathcal{G}_2$ for the game (defined in Subsection 3.2.2) that describes the security of the MSSS Ω_2.

Theorem 3.4.1 *For any adversary $\mathcal{A}_{\Omega_2}$ against the threshold MSS scheme Ω_2 that chooses ℓ access structures and corrupts $t^\star$ players in a set $\mathcal{P}$ of n players, there exists a (k, q_e, q_d, q_c)-adversary $\mathcal{A}_\Pi$ against the encryption scheme Π, with $k = n - t^\star$, $q_d = 0$ and $q_e + q_c = \ell(n - t^\star)$, such that $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \mathsf{Adv}_{\mathcal{A}_{\Omega_2}}(\lambda)$.*

Proof. Let $\mathcal{A}_{\Omega_2}$ be an adversary against the computational security of the multi-secret sharing scheme Ω_2. As shown in below figure, we are going to construct an adversary $\mathcal{A}_\Pi$ against the CCA security (game $\mathcal{G}_1$) of the symmetric encryption scheme Π, which will execute adversary $\mathcal{A}_{\Omega_2}$ as a sub-routine. When the game $\mathcal{G}_1$ starts, the challenger of this game chooses a secret bit $\beta \in \{0, 1\}$ at random. The adversary $\mathcal{A}_\Pi$, at this point, initializes the adversary $\mathcal{A}_{\Omega_2}$ in a running of game $\mathcal{G}_2$.

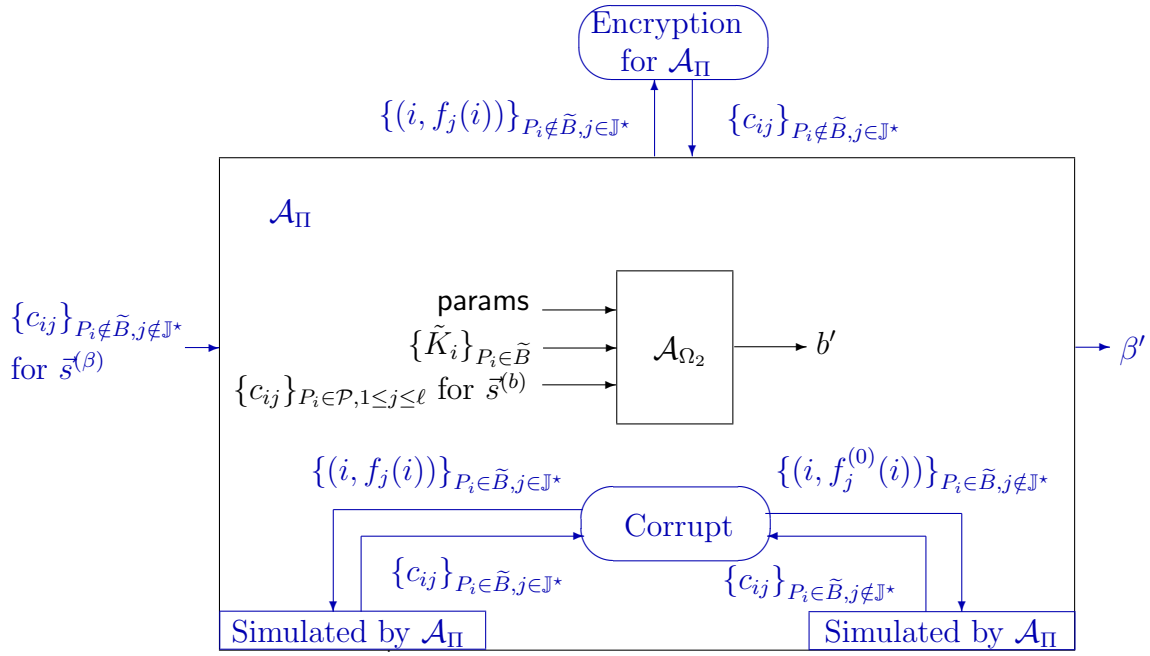

Figure 3.1: $\mathcal{A}_\Pi$ simulates the environment of $\mathcal{A}_{\Omega_2}$.

$\mathcal{A}_{\Omega_2}$ starts the game $\mathcal{G}_2$ by choosing the set of users $\mathcal{P} = \{P_i\}_{1 \le i \le n}$ and ℓ access structures $\{\Gamma_j\}_{1 \le j \le \ell}$ with $\Gamma_j = T(t_j, n)$. Then, $\mathcal{A}_\Pi$ has to simulate a running of the protocol $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$. To do this, $\mathcal{A}_\Pi$ simply chooses a big prime number $p > n$ with λ bits, and sends $\mathsf{params} = (p, \Pi)$ to $\mathcal{A}_{\Omega_2}$, where Π is the symmetric encryption scheme that $\mathcal{A}_\Pi$ is trying to break, and whose plaintext space contains $\mathbb{Z}_p$.

$\mathcal{A}_{\Omega_2}$ chooses a subset $\widetilde{B} \subset \mathcal{P}$ of corrupted players, with $|\widetilde{B}| = t^\star$. For clarity of presentation and without loss of generality, we assume $\widetilde{B} = \{P_1, \ldots, P_{t^\star}\}$. Also, $\mathcal{A}_{\Omega_2}$

outputs two different multi-secrets $\vec{s}^{(0)} \neq \vec{s}^{(1)}$, such that $s_j^{(0)} = s_j^{(1)}$ for all index j such that $t^\star \geq t_j$. Let us define the subset of indices $\mathbb{J}^\star = \{j \in \{1,\ldots,\ell\} \text{ s.t. } \vec{s}_j^{(0)} = \vec{s}_j^{(1)}\}$. Note that $\mathbb{J}^\star \subsetneq \{1,\ldots,\ell\}$ and, furthermore, if $t^\star \geq t_j$, then $j \in \mathbb{J}^\star$.

$\mathcal{A}_\Pi$ runs $\tilde{K}_i \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ for the corrupted players $P_i \in \widetilde{B} = \{P_1,\ldots,P_{t^\star}\}$. Then $\mathcal{A}_\Pi$ defines $k = n - t^\star$ (step 2 of game $\mathcal{G}_1$) and the challenger of this game $\mathcal{G}_1$ runs k times the protocol $K_i \leftarrow \Pi.\mathsf{KG}(1^\lambda)$, for the non-corrupted players $P_i \notin \widetilde{B}$, that is for players $P_{t^\star+1},\ldots,P_n$.

For all index $j \in \mathbb{J}^\star$, $\mathcal{A}_\Pi$ chooses random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$ and such that $f_j(0) = s_j^{(0)} = s_j^{(1)}$. For those values of $j \in \mathbb{J}^\star$, $\mathcal{A}_\Pi$ computes the values $c_{ij} = \Pi.\mathsf{Enc}(f_j(i), K_i)$ in two different ways: for the corrupted players $P_i \in \widetilde{B}$, by using the encryption protocol and knowledge of $\tilde{K}_i$; for the remaining and non-corrupted players $P_i \notin \widetilde{B}$, by sending the encryption query $(i, f_j(i))$ to its encryption oracle. This means that $\mathcal{A}_\Pi$ has made $q_e = |\mathbb{J}^\star| \cdot (n - t^\star)$ encryption queries.

For the rest of indices $j \in \{1,\ldots,\ell\}$, $j \notin \mathbb{J}^\star$, the adversary $\mathcal{A}_\Pi$ chooses random pairs of polynomials $f_j^{(0)}(x), f_j^{(1)}(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$, such that $f_j^{(0)}(0) = s_j^{(0)}$, $f_j^{(1)}(0) = s_j^{(1)}$, and $f_j^{(0)}(i) = f_j^{(1)}(i)$ for all corrupted players $P_i \in \widetilde{B}$. This can be done because $|\widetilde{B}| = t^\star \leq t_j - 1$, for all these indices $j \notin \mathbb{J}^\star$. Now, for the $t^\star$ corrupted players $P_i \in \widetilde{B}$, $\mathcal{A}_\Pi$ computes the ciphertexts $c_{ij} = \Pi.\mathsf{Enc}(f_j^{(0)}(i), \tilde{K}_i)$ by himself. For the $n - t^\star$ non-corrupted players $P_i \notin \widetilde{B}$, $\mathcal{A}_\Pi$ sends the challenge queries $(i, f_j^{(0)}(i), f_j^{(1)}(i))$ to its challenger (game $\mathcal{G}_1$). As the answers, $\mathcal{A}_\Pi$ receives $c_{ij} = \Pi.\mathsf{Enc}(f_j^{(\beta)}(i), K_i)$ for all $j \notin \mathbb{J}^\star$ and all $P_i \notin \widetilde{B}$. The number of challenge queries made by $\mathcal{A}_\Pi$ is $q_c = (\ell - |\mathbb{J}^\star|) \cdot (n - t^\star)$.

This completes the public output $\mathsf{out}_{\mathsf{pub}} = \{c_{ij}\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$. In this way, $\mathcal{A}_\Pi$ is perfectly simulating an execution of the distribution protocol $(\mathsf{out}_{\mathsf{pub}}, \{\mathsf{sh}_i\}_{P_i \in \mathcal{P}}) \leftarrow \Omega.\mathsf{Dist}\left(\mathsf{params}, \vec{s}^{(b)}\right)$, where $\vec{s}^{(b)} = (s_1^{(b)},\ldots,s_\ell^{(b)})$ and $b = \beta$. Now $\mathcal{A}_\Pi$ sends $\mathsf{out}_{\mathsf{pub}}$ to $\mathcal{A}_{\Omega_2}$, along with the secret shares $\{\mathsf{sh}_i\}_{P_i \in \widetilde{B}}$ of the corrupted players, defined as $\mathsf{sh}_i = \tilde{K}_i$.

As expected, the adversary $\mathcal{A}_{\Omega_2}$ broadcasts a bit $b' \in \{0,1\}$ as his final output. $\mathcal{A}_\Pi$ outputs the same bit $\beta' = b'$ to conclude the game $\mathcal{G}_1$. Note that $\mathcal{A}_\Pi$ has not made any decryption query, and so $q_d = 0$.

The probability that $\mathcal{A}_{\Omega_2}$ wins game $\mathcal{G}_2$ (that is, the probability that $b' = b$) is $1/2 + \mathsf{Adv}_{\mathcal{A}_{\Omega_2}}(\lambda)$, by hypothesis. The probability that $\mathcal{A}_\Pi$ wins the game $\mathcal{G}_1$ (that is, the probability that $\beta' = \beta$) is exactly the same, since $\beta' = b'$ and $\beta = b$. Summing up, we have $1/2 + \mathsf{Adv}_{\mathcal{A}_{\Omega_2}}(\lambda) = 1/2 + \mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda)$, which leads to the desired result $\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) = \mathsf{Adv}_{\mathcal{A}_{\Omega_2}}(\lambda)$. □

3.5 Comparing both Schemes in the Standard Model

In this section we are going to compare the schemes presented in Sections 3.3 and 3.4. We start with Table 3.1, where we compare the lengths of shares and public outputs in three different MSS schemes (Ω_1, Ω_2 and running parallel instances of Shamir's standard secret sharing scheme), assuming we share ℓ different secrets, each one in the plaintext space $\mathcal{M}$ of a symmetric encryption scheme Π, according to ℓ different threshold access structures. We do not consider other MSS schemes in the literature, either because they can be thought of as particular cases of Ω_1 or Ω_2, or because they are quite inefficient for the threshold case. For instance, in the MSS scheme by Cachin [18], the length of $\mathsf{out}_{\mathsf{pub}}$ depends on the number of minimally authorized subsets in each access structure; when the access structure is a threshold $T(t,n)$ one, this number corresponds to the very big value $\binom{n}{t}$, which makes Cachin's MSS scheme clearly worse than Ω_2 in this case. Cachin's MSS could be a good alternative for situations where all the access structures have few minimally authorized subsets.

	space of secrets $\vec{s}$	length of $\mathsf{out}_{\mathsf{pub}}$	length of sh_i
ℓ parallel instances of Shamir	$\mathcal{M}^\ell$	-	$\ell \cdot \lvert\mathcal{M}\rvert$
Scheme Ω_1 (Section 3.3)	$\mathcal{M}^\ell$	$\ell \cdot \lvert\mathcal{C}\rvert$	$\ell \cdot \lvert\mathcal{K}\rvert$
Scheme Ω_2 (Section 3.4)	$\mathcal{M}^\ell$	$n \cdot \ell \cdot \lvert\mathcal{C}\rvert$	$\lvert\mathcal{K}\rvert$

Table 3.1: Basic comparison between some MSSS.

Length of sh_i and $\mathsf{out}_{\mathsf{pub}}$.

Usually, the ratio betwen the length of a share sh_i and the length of the secret $\vec{s}$ is considered as the most important aspect for the efficiency of a (multi-)secret sharing scheme. For the case of multi-secret sharing schemes for different threshold access structures, it is well known that $|\mathsf{sh}_i| \geq |\vec{s}|$ must hold in any scheme with unconditional security [67]. This is reflected in the fist row of Table 3.1, where we have included an unconditionally secure MSS scheme as a benchmark.

Regarding the two computationally secure MSS schemes, we first note that the length of each sh_i can be smaller than the length of $\vec{s}$, because $\mathcal{K}$ can be (much) smaller than $\mathcal{M}$ in a symmetric encryption scheme. Ω_2 achieves a better ratio $\frac{|\mathsf{sh}_i|}{|\vec{s}|} = \frac{|\mathcal{K}|}{\ell|\mathcal{M}|}$. The price to pay is that the information $\mathsf{out}_{\mathsf{pub}}$ that must be published is quite big, $|\mathsf{out}_{\mathsf{pub}}| = n \cdot \ell \cdot |\mathcal{C}|$. On the other hand, scheme Ω_1 has a worse ratio $\frac{|\mathsf{sh}_i|}{|\vec{s}|} = \frac{|\mathcal{K}|}{|\mathcal{M}|}$, but it needs a shorter public output, $|\mathsf{out}_{\mathsf{pub}}| = \ell \cdot |\mathcal{C}|$.

This trade-off between the lengths of sh_i and $\mathsf{out}_{\mathsf{pub}}$ in Ω_1 and Ω_2 is due to the fact that the two solutions are in some way dual of each other. In Ω_1 we encrypt the ℓ secrets $s_1, \ldots, s_\ell$, each one with a different symmetric key, we publish the ℓ resulting ciphertexts and we use standard secret sharing to distribute the ℓ symmetric keys among the players, each one according to the corresponding access structure. In Ω_2, on the contrary, there are n symmetric keys K_i, one for each player, and standard secret sharing is used to distribute the ℓ secrets $s_1, \ldots, s_\ell$, each one according to the corresponding Γ_j. This produces ℓ shares for each player P_i, which are encrypted with K_i. The $n \cdot \ell$ resulting ciphertexts are published in $\mathsf{out}_{\mathsf{pub}}$.

Depending on the specific situation in which the multi-secret sharing scheme has to be used, Ω_1 or Ω_2 may be preferable. If the players have very limited space for secret storage (due to either physical or security reasons), and the number of access structures ℓ is moderately big, then Ω_2 is clearly preferable. In some situations, however, a public bulletin board where $\mathsf{out}_{\mathsf{pub}}$ is available makes no sense in the application, and in practice all this information $\mathsf{out}_{\mathsf{pub}}$ must be locally (but not privately) stored by each player P_i, as well. For instance, we can imagine mobile applications where players are smart-phones or smart-cards that run a part of a secret operation (with their secret shares sh_i) in a restricted environment where there is no connection to the Internet. In such a case, the total information that each player / device P_i has to store (privately or not) would have length $|\mathsf{sh}_i| + |\mathsf{out}_{\mathsf{pub}}|$, and therefore Ω_1 may be preferable in these situations.

Exact security, and its consequences.

Leaving aside the unconditionally secure MSS scheme (ℓ parallel instances of Shamir), which suffers from very long secret shares, we have that both Ω_1 and Ω_2 enjoy computational security; they are, roughly speaking, as secure as the underlying symmetric encryption scheme Π. However, the security analysis that we have provided in Theorems 3.3.1 and 3.4.1 shows some differences for the values k, q_e, q_c in the two cases. These differences are very important when one designs and implements cryptographic schemes with a desired level of exact security. In the upcoming Subsection 3.5.1 we illustrate this fact with a detailed example.

The conclusion is that the security reduction seems to be better for Ω_1 than for Ω_2: for a same desired level of security for the multi-secret sharing scheme, we can use a more efficient symmetric encryption scheme Π when implementing Ω_1 than when implementing Ω_2. This may have some (important) consequences in the final efficiency properties of the resulting instantiations of Ω_1 and Ω_2.

Removing the trusted dealer.

We have described the two new MSS schemes Ω_1 and Ω_2 assuming the existence of a trusted entity, a dealer. In particular, in the protocols for the Distribution of the Shares, the trusted dealer takes as input the secret $\vec{s}$ to be shared, computes the shares and sends them privately to the players. This dealer knows all the secrets, and so the security of all the system strongly depends on its resistance to be attacked or corrupted. It may be desirable to have a fully distributed version of this protocol, that can be run by the players $\mathcal{P} = \{P_1, \ldots, P_n\}$ themselves, without the participation of any trusted dealer. In fact, slight modifications would be necessary in the distribution and reconstruction protocols but not in the setup protocol, which would remain without any change. For simplicity, we assume that the adversary is honest-but-curious, and so data is not corrupted.

Regarding the second MSS scheme, Ω_2, this can be easily done. In this case, the global secret $\vec{s} = (s_1, \ldots, s_\ell) \in (\mathbb{Z}_p)^\ell$ to be distributed is not an input for $\Omega_2.\mathsf{Dist}$. Instead, the ℓ secrets $s_1, \ldots, s_\ell$ are implicitly defined by the random choices of the players, during the execution of this distributed protocol. The idea is that each player $P_k \in \mathcal{P}$ will generate his secret share $\mathsf{sh}_k = K_k \leftarrow \Pi.\mathsf{KG}(1^\lambda)$ and then, for each index $j \in \{1, \ldots, \ell\}$, will choose a random polynomial $f_j^{(k)}(x) \in \mathbb{Z}_p[x]$ with degree $t_j - 1$. P_k will privately send the value $f_j^{(k)}(i)$ to player P_i. Implicitly, the j-th secret will be defined as $s_j = \sum_{P_k \in \mathcal{P}} f_j^{(k)}(0)$. Each player P_i can compute his own secret value $s_{ij} = \sum_{P_k \in \mathcal{P}} f_j^{(k)}(i)$, which is a a polynomial (Shamir) share of the secret s_j, and then encrypt it to obtain the ciphertexts $c_{ij} = \Pi.\mathsf{Enc}(s_{ij}\ ,\ K_i)$, that are all included in $\mathsf{out}_{\mathsf{pub}}$. In the reconstruction protocol $\Omega_2.\mathsf{Rec}$ the players of an authorized subset $A \in \Gamma_j$ use the values $\{s_{ij}\}_{P_i \in A}$ to interpolate the polynomial $F_j(x) = \sum_{P_i \in \mathcal{P}} f_j^{(i)}(x)$ in $x = 0$, recovering in this way the j-th secret $s_j = F_j(0)$. In the Appendix of [45] can be found a similar scheme which uses the above steps to create a multi-threshold secret sharing scheme without dealer.

Regarding the first MSS scheme, Ω_1, we could try to apply the idea in the previous paragraph so that ℓ secrets $K_1, \ldots, K_\ell$ are implicitly and jointly generated by the own players, at the same time that each player P_i obtains a Shamir share $K_j^{(i)}$ of each secret $K_j \in \mathbb{Z}_q$. Once this is done, and for each index j, the players must run a distributed version of the protocol $\Pi.\mathsf{Enc}$ for some (maybe random) plaintext s_j and secret encryption key K_j, that is shared among the players. The resulting ciphertext c_j will be added to $\mathsf{out}_{\mathsf{pub}}$. Such a distributed (or multi-party) version of a symmetric encryption protocol is not easy at all (see [66, 91] for some constructions).

Summing up, if some of our new MSS schemes needs to be implemented without any trusted dealer, then Ω_2 is possibly better, because it admits a much simpler fully

distributed version.

3.5.1 A Specific Example

We want to share $\ell = 2^5 = 32$ different secret keys of RSA (with $2048 = 2^{11}$ bits), each one according to a different threshold access structure, defined on a set of $n = 2^{10}$ players, and with a security level of 100 bits (that is, we want a MSS scheme Ω such that $\mathsf{Adv}_{\mathcal{A}_\Omega}(\lambda) \leq 2^{-100}$, for any adversary $\mathcal{A}_\Omega$). Each secret s_j will thus belong to $\{0,1\}^{2^{11}}$, and we look for a symmetric encryption scheme Π with plaintext space $\mathcal{M} = \{0,1\}^{2^{11}}$.

We take Π as the result of applying some mode of operation (such as CTR or CBC, with d blocks) to a pseudorandom permutation $F : \{0,1\}^\kappa \times \{0,1\}^r \to \{0,1\}^r$. For simplicity, we will assume $\kappa = r$. The properties of the resulting encryption scheme Π are as follows: the key space is $\mathcal{K} = \{0,1\}^r$ (the same as in F), the ciphertext space is $\mathcal{C} = \{0,1\}^r$ (the same as in F), and the plaintext space is $\mathcal{M} = \{0,1\}^{r \cdot d}$ (that is, d times bigger than that of F). We are interested in parameters r, d such that $r \cdot d = 2^{11}$, so we will use $d = 2^{11}/r$.

Combining Theorem 4.7.2 in [7], for the particular case of the CTR mode of operation, with the generic results on security of encryption in the k-user setting [3], we have

$$\mathsf{Adv}_{\mathcal{A}_\Pi}(\lambda) \leq k \cdot \left(\mathsf{Adv}_{\mathcal{A}_F(q')}(\lambda) + \frac{(q')^2}{2^{r+1}} \right), \tag{3.1}$$

where $\mathsf{Adv}_{\mathcal{A}_F(q')}(\lambda)$ is an upper bound for the advantage of any algorithm trying to break, through q' queries, the pseudorandomness of F, and $\mathcal{A}_\Pi$ is a (k, q_e, q_d, q_c)-adversary against Π. The theorem is only valid when $q_d = 0$ (which is the case for both Ω_1 and Ω_2) and for $q' = d(q_e + q_c)$. Let us assume, for simplicity, that the best known attack against the pseudorandomness of F is the birthday attack; this means $\mathsf{Adv}_{\mathcal{A}_F(q')}(\lambda) \leq \frac{(q')^2}{2^r}$.

If we choose to use Ω_1, we can apply our Theorem 3.3.1, with $k = \ell - |\mathbb{J}^\star| \leq \ell = 2^5$, $q_e = q_d = 0$ and $q_c = \ell - |\mathbb{J}^\star| \leq \ell = 2^5$. Combining the result in that theorem with Equation (3.1), with $q' = d \cdot q_c \leq d \cdot 2^5$, we have

$$\mathsf{Adv}_{\mathcal{A}_{\Omega_1}}(\lambda) \leq 2^5 \cdot \left(\frac{d^2 \cdot 2^{10}}{2^r} + \frac{d^2 \cdot 2^{10}}{2^{r+1}} \right) \leq \frac{2^{15} d^2}{2^{r-1}}.$$

Using $d = 2^{11}/r$ and the fact that we want $\mathsf{Adv}_{\mathcal{A}_{\Omega_1}}(\lambda) \leq 2^{-100}$, we reach the restriction $r^2 \cdot 2^{r-1} \geq 2^{137}$, which is achieved by the value $r = 126$. Therefore, we can securely implement Ω_1 by using an underlying symmetric encryption scheme where plaintexts are 2^{11} bits long, but where keys and ciphertexts are 126 bits long.

On the other hand, if we choose to use Ω_2, we can apply our Theorem 3.4.1, with $k = n - t^\star \leq n = 2^{10}$, $q_e + q_c = \ell(n - t^\star) \leq \ell \cdot n = 2^{15}$. Combining the result in that

theorem with Equation (3.1), as we have done in the previous paragraph, we have

$$\mathsf{Adv}_{\mathcal{A}_{\Omega_2}}(\lambda) \leq 2^{10} \cdot \left(\frac{d^2 \cdot 2^{30}}{2^r} + \frac{d^2 \cdot 2^{30}}{2^{r+1}} \right) \leq \frac{2^{40} d^2}{2^{r-1}}.$$

Using $d = 2^{11}/r$ and the desired inequality $\mathsf{Adv}_{\mathcal{A}_{\Omega_2}}(\lambda) \leq 2^{-100}$, we conclude that $r^2 \cdot 2^{r-1} \geq 2^{162}$ is enough, which is achieved by the value $r = 149$. We can thus implement Ω_2 and obtain a security level of 100 bits, by using an underlying symmetric encryption scheme where plaintexts are again 2^{11} bits long, and where keys and ciphertexts are 149 bits long.

Now if we adapt Table 3.1 to this specific example with $\ell = 32$ and $n = 1024$, we obtain

	space of secrets $\vec{s}$	length of $\mathsf{out}_{\mathsf{pub}}$	length of sh_i
ℓ parallel instances of Shamir	$\{0,1\}^{2048 \cdot 32}$	-	65.536 bits
Scheme Ω_1 (Section 3.3)	$\{0,1\}^{2048 \cdot 32}$	4.032 bits	4.032 bits
Scheme Ω_2 (Section 3.4)	$\{0,1\}^{2048 \cdot 32}$	4.882.432 bits	149 bits

Table 3.2: Comparison between some MSSS for a specific example.

The price to pay, if one chooses Ω_2 because of their shorter secret shares, is a huge public output. Again, as discussed at the beginning of this section, one MSS scheme or another can be preferable depending on the specific application.

3.6 A Multi-Secret Sharing Scheme in the Random Oracle Model

Opposite to both previous proposals, whose security was in the standard model, we introduce here a computationally secure multi-threshold secret sharing scheme with provable security in the random oracle model.

The new MSSS, denoted by Ω_3, is a variant of the previous scheme Ω_2 defined in Section 3.4, but replacing the underlying symmetric encryption scheme $\Pi = (\Pi.\mathsf{KG}, \Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ of Ω_2 by a secure one-way hash function H. Consequently, Ω_3 is slightly more efficient than Ω_2 because hash functions are more efficient than encrypt and decrypt, but with security in the random oracle model. The protocols of Ω_3 are the following:

Setup: $\Omega_3.\mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$.
Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of n users and let $1 \leq t_1 < t_2 < \ldots < t_\ell \leq n$ be the ℓ different thresholds that define the access structures $\{\Gamma_j\}_{1 \leq j \leq \ell}$. A prime

number $p > n$ is chosen, such that p is λ bits long. A secure one-way hash function $H : \mathbb{N} \times \mathbb{Z}_p^\star \to \mathbb{Z}_p$ is also chosen. Each player P_i is assigned the value i. The public parameters are $\mathsf{params} = (p, H)$.

Distribution of the Shares: $\Omega_3.\mathsf{Dist}(\mathsf{params}, \vec{s})$.
Assumed that the secret to be distributed is $\vec{s} = (s_1, \ldots, s_\ell) \in (\mathbb{Z}_p)^\ell$. For simplicity, we assume the distribution is done by an external dealer; see Section 3.5 for a discussion on how the own members of $\mathcal{P}$ could run this protocol.

1. Choose random values $\mathsf{sh}_i \in \mathbb{Z}_p^\star$, pairwisse different for $i = 1, 2, \ldots, n$, as the secret shares.
2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.
3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $h_{ij} = H(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$.
4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{r_{ij}\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$.

Reconstruction of the Secrets: $\Omega_3.\mathsf{Rec}(\mathsf{params}, \mathsf{out}_{\mathsf{pub}}, j, \{\mathsf{sh}_i\}_{P_i \in A})$.
When the players of an authorized subset $A \in \Gamma_j$ (i.e. $|A| \geq t_j$) want to recover the secret s_j, they must cooperate performing the following steps.

1. Each player $P_i \in A$ computes his pseudo secret share as $h_{ij} = H(j, \mathsf{sh}_i)$.
2. Take the values $\{r_{ij}\}_{P_i \in A}$ from $\mathsf{out}_{\mathsf{pub}}$ and compute $f_j(i) = r_{ij} + h_{ij} \bmod p$, for every $P_i \in A$.
3. Use the values $\{f_j(i)\}_{P_i \in A}$ to interpolate the polynomial $f_j(x)$ and recover the j-secret $s_j = f_j(0)$.

Note that the correctness of the proposed scheme Ω_3 holds directly via interpolation.

3.6.1 Security Analysis

The proposed scheme is computational secure, assuming that the hash function H behaves as a random oracle [6].

Theorem 3.6.1 *For any adversary $\mathcal{A}_{MSS}$ against the described threshold MSSS that makes at most q_H queries to the random oracle for H, we have* $\mathsf{Adv}_{\mathcal{A}_{MSS}}(\lambda) \leq \frac{q_H(q_H+n)}{2^{\lambda+1}} + o\left(\left(\frac{q_H(q_H+n)}{2^{\lambda+1}}\right)^2\right)$.

Proof. Let $\mathcal{A}_{MSS}$ be an adversary against the computational security of the multi-threshold secret sharing scheme. We act as the challeger of the security game described in Subsection 3.2.2. $\mathcal{A}_{MSS}$ starts the game by choosing the set of users $\mathcal{P} = \{P_i\}_{1\leq i\leq n}$ and the access structures $\{\Gamma_j\}_{1\leq j\leq \ell}$ with $\Gamma_j = T(t_j, n)$. We then run $\mathsf{params} \leftarrow \Omega.\mathsf{Stp}(1^\lambda, \mathcal{P}, t_1, \ldots, t_\ell)$ and send $\mathsf{params} = (p, H)$ to $\mathcal{A}_{MSS}$, who chooses a subset $\widetilde{B} \subset \mathcal{P}$ of corrupted players with $|\widetilde{B}| = t^\star$. Let $\mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } t_j \leq t^\star\}$.

We choose random pairwise different elements $\mathsf{sh}_i \in \mathbb{Z}_p$, for $P_i \in \mathcal{P}$. If $\mathcal{A}_{MSS}$ makes a hash query (j, x) to the random oracle such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $x \in \{\mathsf{sh}_i\}_{P_i\in\widetilde{B}}$, then we abort the game. Otherwise, the query is answered by choosing a random element $h \in \mathbb{Z}_p$, storing the relation $H(j, x) = h$ in a hash table, and sending back the output h to $\mathcal{A}_{MSS}$. If a hash query (j, x) by $\mathcal{A}_{MSS}$ is already in the hash table, the stored value h is sent back to $\mathcal{A}_{MSS}$.

Challenge. At some point, $\mathcal{A}_{MSS}$ outputs two different multi-secrets $\vec{s}^{(0)} \neq \vec{s}^{(1)}$, such that $s_j^{(0)} = s_j^{(1)}$ for all $j \in \mathbb{J}^\star$. We choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$ and such that $f_j(0) = s_j^{(0)}$, for all $j \in \mathbb{J}^\star$. For those values of $j \in \mathbb{J}^\star$, we compute (via the hash-table procedure) the values $h_{ij} = H(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$, for all $P_i \in \mathcal{P}$.

For the rest of values $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$, we choose random pairs of polynomials $f_j^{(0)}(x), f_j^{(1)}(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$, such that $f_j^{(0)}(0) = s_j^{(0)}$, $f_j^{(1)}(0) = s_j^{(1)}$, and $f_j^{(0)}(i) = f_j^{(1)}(i)$ for all corrupted players $P_i \in \widetilde{B}$. For indices i such that $P_i \in \widetilde{B}$, we compute (via the hash-table procedure) the values $h_{ij} = H(j, \mathsf{sh}_i)$ and $r_{ij} = f_j^{(0)}(i) - h_{ij} \bmod p$. For indices i such that $P_i \notin \widetilde{B}$, we choose at random $r_{ij} \in \mathbb{Z}_p$.

We give to $\mathcal{A}_{MSS}$ the shares $\{\mathsf{sh}_i\}_{P_i\in\widetilde{B}}$ of the corrupted players, as well as the public output of the protocol $\mathsf{out}_{\mathsf{pub}} = \{r_{ij}\}_{P_i\in\mathcal{P}, j\in\{1,\ldots,\ell\}}$.

For indices $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and indices i such that $P_i \notin \widetilde{B}$, let us define $h_{ij}^{(0)} = r_{ij} - f_j^{(0)}(i) \bmod p$ and $h_{ij}^{(1)} = r_{ij} - f_j^{(1)}(i) \bmod p$. We can choose at random a bit $\beta \in \{0, 1\}$ and include in the hash-table the values $H(j, \mathsf{sh}_i) = h_{ij}^{(\beta)}$, for all i, j such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $P_i \notin \widetilde{B}$. In this way, we are perfectly simulating an execution of the distribution of shares for the secret $\vec{s}^{(\beta)} = (s_1^{(\beta)}, \ldots, s_\ell^{(\beta)})$. The key point here is that, as long as $\mathcal{A}_{MSS}$ does not make any hash query $H(j, \mathsf{sh}_i)$ for indices i, j such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and $P_i \notin \widetilde{B}$, the information that $\mathcal{A}_{MSS}$ gets is the same as if the shared secret was $\vec{s}^{(1-\beta)}$.

Final analysis. Therefore, to compute the probability that $\mathcal{A}_{MSS}$ guesses the correct shared secret, we distinguish between two cases, depending on whether $\mathcal{A}_{MSS}$ makes a hash query $H(j, \mathsf{sh}_i)$ for indices i, j such that $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$ and

$P_i \notin \widetilde{B}$. If this is the case, which happens with probability δ, then we assume the best case for $\mathcal{A}_{MSS}$: he always guesses the correct secret in that case. On the other hand, if $\mathcal{A}_{MSS}$ does not make such a hash query, which happens with probability $1-\delta$, then the probability that $\mathcal{A}_{MSS}$ guesses is exactly $1/2$. Summing up, the probability that $\mathcal{A}_{MSS}$ guesses the correct secret is at most $\delta + 1/2(1-\delta)$. Therefore, the advantage of $\mathcal{A}_{MSS}$ is

$$\mathsf{Adv}_{\mathcal{A}_{MSS}}(\lambda) = \left|\Pr[\mathcal{A}_{MSS} \text{ guesses}] - \frac{1}{2}\right| \leq \frac{1}{2}\delta(\lambda).$$

The probability $\delta(\lambda)$ that some of q_H randomly chosen elements falls in a perfectly hidden subset $\{\mathsf{sh}_i\}_{P_i \notin \widetilde{B}}$ of $n - t^\star$ random elements of $\mathbb{Z}_p$ can be bounded as

$$\delta(\lambda) < 1 - \left(\frac{p-(n+q_H-t^\star-1)}{p}\right)^{q_H} = \frac{q_H(q_h+n)}{p} + o\left(\left(\frac{q_H(q_h+n)}{p}\right)^2\right).$$

Using that $p > 2^\lambda$, we obtain the desired result $\mathsf{Adv}_{\mathcal{A}_{MSS}}(\lambda) \leq \frac{q_H(q_H+n)}{2^{\lambda+1}} + o\left(\left(\frac{q_H(q_H+n)}{2^{\lambda+1}}\right)^2\right)$. □

In cryptography, the number of hash queries is usually estimated as $q_H \leq 2^{60}$. The number n of players will be much smaller than that, in real situations. Therefore, the multi-threshold secret sharing scheme described here achieves a 80-bit security level as long as $\lambda \geq 200$. In this case, the prime number p used in the scheme, which is λ bits long, verifies $p \geq 2^{200}$.

3.7 Some Extensions

All proposed MSSSs presented in this chapter can be generalized from different points of view to be used in specific situations. In Section 3.5 we explained how the role of the trusted dealer could be done by the participants themselves for both schemes Ω_1 and Ω_2. This property is also satisfied by the scheme Ω_3 following the same ideas presented for the scheme Ω_2. Next we show the following two extensions.

Active Adversaries.

The idea to make the MSSSs secure against active adversaries (who can send incorrect values during the protocol) is to consider verifiable secret sharing techniques ([31, 74]) as described for instance in [36]. Slight modifications of the scheme above are necessary to achieve this goal. For instance, in scheme Ω_2 of Subsection 3.4 every

participant $P_i \in \mathcal{P}$ must publish, in Step 2 of the distribution protocol, the commitments $A_{ij}^{(u)} = g^{a_{ij}^{(u)}}$ to the coefficients of his polynomials, with $j \in \{1, \ldots, \ell\}$ and $u \in \{1, \ldots, t_j - 1\}$. Later, these commitments are used to detect incorrect values that are sent in Step 3 of the distribution protocol or broadcast in Step 2 of the reconstruction protocol.

More General Access Structures.

We have described and analyzed different MSSSs throughout this chapter. The first two proposals were proven in the standard model (Sections 3.3 and 3.4) and the third one in the random oracle model (Section 3.6), for the particular case where the access structures $\Gamma_1, \ldots, \Gamma_\ell$ are all threshold ones. However, all these protocols can be easily extended to the case of more general access structures Γ_j, as long as they admit a linear and ideal secret sharing scheme (also known as *vector space secret sharing scheme* [17], which were introduced in Section 1.5). To achieve this goal, the proposed MSSSs can be easily modified just replacing polynomials with vectors, and polynomial evaluations with scalar products.

Advancing some results of Chapter 4, the same will happen with the multi-policy distributed cryptosystems which are using these sharing schemes as building block. If the access structures admit a vector space secret sharing scheme, then the key generation protocol can be modified as explained above; later, the encryption / decryption or signature / verification operations can be (slightly) modified to work with these more general access structures and linear secret sharing schemes, because they will involve only linear operations (additions and multiplications with a constant value). Even the version of our protocols that works without any trusted dealer, described above, can be extended using the ideas and techniques from [47].

Chapter 4

Multi-Policy Distributed Cryptosystems

In public key cryptography, some operations (like encrypting a message or verifying a signature) can be done by any user in the system, with access to the public information of the other users. However, the associated *secret* operations (like decrypting a ciphertext or signing a message) can be done only by the user who knows the corresponding secret information.

In some situations, such secret tasks are too important and sensitive to rely on a single user or machine; a good solution then is to use *distributed* (in particular, *threshold*) public key cryptography: the secret information is distributed among a set of users, and the cooperation of several authorized subset (in a fixed access structure) of them is required in order to correctly perform the corresponding secret task. Depending on the considered secret task, this approach leads to different distributed cryptosystems as e.g. distributed decryption schemes or distributed signature schemes.

In this chapter we consider an extension of the standard scenario of distributed (public key) cryptography. In some cases, setting a single access structure of authorized subsets of users for all the executions of the secret task may be unrealistic. For instance, some messages encrypted for a receiver entity $\mathcal{P}$ may be more sensitive or confidential than others, and thus require the cooperation of more or less members of $\mathcal{P}$ in order to be decrypted. With this motivation in mind, we will consider *multi-policy* distributed cryptosystems: in the setup of the system, a list of ℓ possible (and different) acces structures is chosen; later, for each execution of the cryptographic operation, a specific access structure in this list is chosen "on the fly", depending on the desired security level. Only those subsets of players authorized with respect to this specific access structure will be able to perform the secret task, by using their secret shares of information. A trivial way of implementing multi-policy distributed

cryptosystems is by running ℓ independent instances of a standard distributed cryptosystem, one for each of the access structures in the list. This solution has the drawback that the length of the secret information to be stored by each user is linear in ℓ; we look for more efficient solutions.

As it happens with standard distributed cryptosystems, where standard secret sharing schemes are a key ingredient in their design, it is natural that *multi-secret sharing schemes* (MSSSs) are a key tool when designing multi-policy distributed cryptosystems.

In the following two sections we will use the MSSS proposed in Section 3.6 as a key tool to design a new multi-policy distributed decryption scheme and a new multi-policy distributed signature scheme. We prove the security of these two schemes, in the random oracle model, by taking into account formal security models that we introduce in the corresponding sections. Again, and for simplicity, we describe the schemes for the threshold case, but extensions to the case of more general policies are easy to do, as we discuss in Section 3.7. The efficiency of the new multi-policy distributed cryptosystems is essentially the same as the efficiency of the standard distributed cryptosystems (Shoup-Gennaro [88] for decryption and Boldyreva [10] for signatures) by which they are inspired. Namely, the length of secret shares, ciphertexts and signatures, and the cost of encryption, decryption, signature and verification are the same; the only changes are in the size of the public parameters and public key of the set $\mathcal{P}$, which are increased by a factor of $n \cdot \ell$.

In Section 4.3 we discuss the relations between the new primitives of multi-policy distributed cryptosystems and attribute-based cryptosystems. Even if any attribute-based cryptosystem leads to a multi-policy distributed cryptosystem, the obtained scheme has some drawbacks that are not present in the solutions that we propose in previous Sections. For instance, our new MSSS and our multi-policy distributed schemes can be modified so that no external and trusted entity is needed in the life of the system; this is not possible in attribute-based solutions, where a trusted master entity generates and distributes the secret material to the members of the set $\mathcal{P}$.

In the last section a slight overview on different cryptographic primitives, where the proposed multi-threshold secret sharing schemes of this thesis can be applied, is given. One of these applications could be in the signcryption process of signcryptions with threshold unsigncryption or in the verification process of signatures with distributed verification proposed in Chapter 2.

4.1 Multi-Policy Distributed Decryption

In this section, we will have a set of users $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ as the receivers of confidential messages. There will be ℓ different access structures $\Gamma_j \subset 2^{\mathcal{P}}$ defined on $\mathcal{P}$, for $j = 1, \ldots, \ell$. When encrypting, the sender will choose the desired decryption access

structure Γ_j. A ciphertext encrypted for the access structure (or decryption policy) Γ_j will be correctly decrypted only if the users in some subset $A \in \Gamma_j$ cooperate to run the protocol $\mathsf{Decrypt}$. Each user $\mathcal{P}_i \in \mathcal{P}$ will have a share sh_i of secret information, and will use it to perform his part of the decryption process.

After defining in a formal way the syntactic definition and the security model for this primitive, that we call multi-policy distributed decryption, we will present a scheme for this functionality in the case of threshold decryption policies, that uses as a building block the multi-threshold secret sharing scheme of Section 3.6, and we will prove that it satisfies the required security properties. We consider threshold access structures only for simplicity of the presentation; the scheme can be extended to work with more general access structures as explained in Section 3.7.

4.1.1 Syntactic Definition

A multi-policy distributed decryption scheme $\Sigma = (\Sigma.\mathsf{St}, \Sigma.\mathsf{KG}, \Sigma.\mathsf{Enc}, \Sigma.\mathsf{Decrypt})$ consists of four probabilistic polynomial-time algorithms:

- The randomized *setup* algorithm $\Sigma.\mathsf{St}$ takes a security parameter $\lambda \in \mathbb{N}$ and outputs some public parameters params that will be common to all the users in the system: the mathematical groups, generators, hash functions, etc. We write $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ to denote an execution of this algorithm.

- The *key generation* algorithm $\Sigma.\mathsf{KG}$ for a collective $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ of n users and ℓ different access structures $\Gamma_j \subset 2^{\mathcal{P}}$ for $j = 1, \ldots, \ell$ has as public output a public key PK. We implicitly assume that PK contains the description of $\mathcal{P}$ and the ℓ access structures. Each user $\mathcal{P}_i \in \mathcal{P}$ receives a secret share sh_i. This key generation process for the collective $\mathcal{P}$ can be run either by a trusted third party, or by the users in $\mathcal{P}$ themselves. We will write $(\{\mathsf{sh}_i\}_{1 \leq i \leq n}, PK) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ to refer to this key generation protocol.

- The *encryption* algorithm $\Sigma.\mathsf{Enc}$ takes as input params, a message m, the public key PK of the intended receiver group $\mathcal{P}$, and the index xj for the desired decryption policy Γ_j, where $j \in \{1, \ldots, \ell\}$. The outputs are a ciphertext C and the index j of the chosen decryption policy. We denote an execution of this algorithm as $(C, j) \leftarrow \Sigma.\mathsf{Enc}(\mathsf{params}, m, PK, j)$.

- The *joint decryption* algorithm $\Sigma.\mathsf{Decrypt}$ is a distributed protocol run by some subset of users $A \subset \mathcal{P}$. The common inputs are params, PK, a ciphertext C and an index j, whereas each user $\mathcal{P}_i \in A$ has as secret input his secret share sh_i. The output is a message $\widetilde{m}$, which can eventually be the special symbol $\perp$, meaning that the pair (C, j) is invalid. To refer to an execution of this protocol, we write $\widetilde{m} \leftarrow \Sigma.\mathsf{Decrypt}(\mathsf{params}, C, j, PK, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A})$.

For correctness, $\Sigma.\mathsf{Decrypt}(\mathsf{params}, \Sigma.\mathsf{Enc}(\mathsf{params}, m, PK, j), PK, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A}) = m$ is required, whenever $A \in \Gamma_j$ and the values $\mathsf{params}, \{\mathsf{sh}_i\}_{1 \le i \le n}, PK$ have been obtained by properly executing the protocols $\Sigma.\mathsf{St}$ and $\Sigma.\mathsf{KG}$.

4.1.2 Security Model

A correct encryption scheme must satisfy the proper confidentiality property. In the distributed setting that we are considering, confidentiality must hold even if an attacker corrupts many members of the collective of receivers, provided the corrupted members are not authorized to decrypt the challenge ciphertext.

The confidentiality requirement for a multi-policy distributed decryption scheme Σ (i.e. the fact that a ciphertext on the message m addressed to $\mathcal{P}$ for access structure Γ_j leaks no information on m to an attacker who has corrupted a subset of users $\widetilde{B} \subset \mathcal{P}$ such that $\widetilde{B} \notin \Gamma_j$) is ensured if the scheme enjoys the property of *indistinguishability under chosen ciphertext attacks* ($\mathsf{IND\text{-}CCA}$ security, for short). For a security parameter $\lambda \in \mathbb{N}$, this property is defined by considering the following game that an attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Sigma.\mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

2. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses a target set $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ of users, ℓ different decryption policies $\Gamma_j \subset 2^{\mathcal{P}}$, for $j = 1, \ldots, \ell$, and a subset $\widetilde{B} \subset \mathcal{P}$ of users, to be corrupted. The challenger runs $(\{\mathsf{sh}_i\}_{1 \le i \le n}, PK) \leftarrow \Sigma.\mathsf{KG}(\mathsf{params}, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ and gives to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the values PK and $\{\mathsf{sh}_i\}_{\mathcal{P}_i \in \widetilde{B}}$.

   Note that we are considering only *static* adversaries who choose the subset $\widetilde{B}$ of corrupted users at the beginning of the attack. Considering security against *adaptive* adversaries is an interesting problem for future research.

3. [**Queries**] $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make adaptive queries to a distributed decryption oracle for the target set $\mathcal{P}$: $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ sends a tuple (C, j). The challenger runs $\widetilde{m} \leftarrow \Sigma.\mathsf{Decrypt}(\mathsf{params}, C, j, PK, \mathcal{P}, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in \mathcal{P}})$. The attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ must be given all the information that is broadcast during the execution of this protocol $\Sigma.\mathsf{Decrypt}$, including $\widetilde{m}$.

4. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses two messages m_0, m_1 of the same length, and a decryption policy Γ_j, where $j \in \{1, \ldots, \ell\}$ and $\widetilde{B} \notin \Gamma_j$.

5. [**Challenge**] The challenger picks a random bit $d \in \{0, 1\}$, runs the protocol $(C^\star, j) \leftarrow \Sigma.\mathsf{Enc}(\mathsf{params}, m_d, PK, j)$ and gives $(C^\star, j)$ to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

6. Step 3 is repeated, with the restriction that the tuple $(C^\star, j)$ cannot be queried to the distributed decryption oracle.

7. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a bit d' as his guess of the bit d.

The advantage of such a (static) adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ in breaking the IND-CCA security of the multi-policy distributed decryption scheme is defined as

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda) = \left| \Pr[d' = d] - \frac{1}{2} \right|.$$

A multi-policy distributed decryption scheme Σ is IND-CCA secure if $\mathsf{Adv}_{\mathcal{A}_{\mathsf{IND\text{-}CCA}}}(\lambda)$ is negligible with respect to the security parameter λ, for any polynomial time (static) adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

4.1.3 A New Multi-Threshold Decryption Scheme

We propose here a new multi-policy distributed decryption scheme. For simplicity we describe the scheme when all the decryption policies are threshold ones; that is, $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$, where $1 \leq t_1 < t_2 < \ldots < t_\ell \leq n$. The scheme is inspired by the (single) threshold decryption scheme proposed by Shoup and Gennaro in [88]. A key ingredient in the design of the new scheme will be the multi-threshold secret sharing scheme proposed in Section 3.6. The protocols of the multi-threshold decryption scheme Σ work as follows.

Setup: $\Sigma.\mathsf{St}(1^\lambda)$.
Given a security parameter $\lambda \in \mathbb{N}$, a group $\mathbb{G} = \langle g \rangle$ of prime order p, such that p is λ bits long, is chosen. A positive integer $l \in \mathbb{N}$, which must be polynomial in λ, is chosen for the maximum number of bits of the messages to be encrypted. Five hash functions are chosen: $H_0 : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^* \to \{0,1\}^l$, $H_2 : \{0,1\}^* \to \mathbb{G}$, $H_3 : \{0,1\}^* \to \mathbb{Z}_p$, $H_4 : \{0,1\}^* \to \mathbb{Z}_p$. The output of this protocol is $\mathsf{params} = (p, \mathbb{G}, g, l, H_0, H_1, H_2, H_3, H_4)$.

Key Generation: $\Sigma.\mathsf{KG}(\mathsf{params}, \mathcal{P}, t_1, \ldots, t_\ell, n)$.
Let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a set of n users and $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$ the threshold decryption policies defined on $\mathcal{P}$, where $1 \leq t_1 < t_2 < \ldots < t_\ell \leq n$. For $j = 1, \ldots, \ell$, the value $PK_j = g^{s_j}$ is computed, for a random value $s_j \in \mathbb{Z}_p^*$ that will remain unknown to the members of $\mathcal{P}$. These ℓ secret values will correspond to a secret vector $\vec{s} = (s_1, \ldots, s_\ell)$ of the multi-threshold secret sharing scheme described in Section 3.6, that will be shared by running the distribution protocol $\Omega.\mathsf{Dist}(\mathcal{P}, t_1, \ldots, t_\ell, \vec{s})$, with hash function H_0:

1. Choose random values $\mathsf{sh}_i \in \mathbb{Z}_p^\star$, pairwisse different for $i = 1, 2, \ldots, n$, as the secret shares.

2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.

3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $h_{ij} = H_0(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$.

4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{r_{ij}\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$.

The secret share for each player P_i is sh_i, whereas the global public key is $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}})$. In case one wants to provide robustness to the threshold decryption process, the values $D_{ij} = g^{h_{ij}+r_{ij}}$ must be included in PK, for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell$.

Encryption: $\Sigma.\mathsf{Enc}(\mathsf{params}, m, PK, j)$.

1. Choose at random the values $r_u, r_w \in \mathbb{Z}_p^*$.

2. Compute $c = H_1(PK_j^{r_u}, j) \oplus m$.

3. Use the element g to compute $u = g^{r_u}$ and $w = g^{r_w}$.

4. Use the value $\bar{g} = H_2(c, u, w, j)$ to compute $\bar{u} = \bar{g}^{r_u}$ and $\bar{w} = \bar{g}^{r_w}$.

5. Compute $e = H_3(\bar{g}, \bar{u}, \bar{w}, j)$ and $\sigma = r_w + r_u \cdot e \mod p$.

6. Return (C, j) with the ciphertext $C = (c, u, \bar{u}, e, \sigma)$.

Joint Decryption: $\Sigma.\mathsf{Decrypt}(\mathsf{params}, C, j, PK, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A})$
Let $A \subset \mathcal{P}$ be a subset of users in $\mathcal{P}$ that want to cooperate to decrypt a ciphertext $C = (c, u, \bar{u}, e, \sigma)$ according to the threshold decryption policy $T(t_j, n)$. We assume, thus, $|A| \geq t_j$. Players in A proceed as follows.

1. Each $\mathcal{P}_i \in A$ checks if $e = H_3(\bar{g}, \bar{u}, \bar{w}, j)$, where $w = g^f / u^e$, $\bar{g} = H_2(c, u, w, j)$, $\bar{w} = \bar{g}^f / \bar{u}^e$.
   If this equality does not hold, $\mathcal{P}_i$ broadcasts $(i, \perp)$.

2. Otherwise, $\mathcal{P}_i \in A$ chooses $v_{ij} \in \mathbb{Z}_p$ at random, recovers r_{ij} from $\mathsf{out}_{\mathsf{pub}}$, computes $h_{ij} = H_0(j, \mathsf{sh}_i)$ and broadcasts the tuple $(i, u_{ij}, e_{ij}, \sigma_{ij})$, where

$$u_{ij} = u^{h_{ij}+r_{ij}},\ \widehat{u}_{ij} = u^{v_{ij}},\ \widehat{h}_{ij} = g^{v_{ij}},\ e_{ij} = H_4(u_{ij}, \widehat{u}_{ij}, \widehat{h}_{ij})$$

$$\text{and} \quad \sigma_{ij} = v_{ij} + (h_{ij} + r_{ij}) \cdot e_{ij} \mod p$$

   [If robustness is required, the correctness of this tuple can be publicly verified by checking if $e_{ij} = H_4(u_{ij}, \widehat{u}_{ij}, \widehat{h}_{ij})$, where $\widehat{u}_{ij} = u^{\sigma_{ij}} / u_{ij}^{e_{ij}}$, $\widehat{h}_{ij} = g^{\sigma_{ij}} / D_{ij}^{e_{ij}}$. Note that this check ensures that (u, D_{ij}, u_{ij}) is a valid Diffie-Hellman triple.]

3. If there are not t_j valid shares, stop and output $\perp$. Otherwise, from t_j valid tuples $\{(i, u_{ij}, e_{ij}, \sigma_{ij})\}_{\mathcal{P}_i \in A}$, different from $(i, \perp)$, one can consider the Lagrange interpolation coefficients $\lambda_{ij}^A \in \mathbb{Z}_p$ such that $s_j = f_j(0) = \sum_{\mathcal{P}_i \in A} \lambda_{ij}^A \cdot f_j(i)$.

4. Return the message $m = c \oplus H_1(\prod_{\mathcal{P}_i \in A} u_{ij}^{\lambda_{ij}^A}, j)$.

4.1.4 Security Analysis

A first attempt to prove the security of the new multi-threshold decryption scheme would be to reduce its security to the security of the inherent multi-threshold secret sharing scheme, which is described in Section 3.6. However, such a reduction does not work, because in the new decryption scheme, the values $PK_j = g^{s_j}$ are public, for $j = 1, \ldots, \ell$. In this scenario it is trivial to distinguish between two potentially shared secrets $s_j^{(0)} \neq s_j^{(1)}$, chosen by the adversary. Therefore, in order to prove the security of the multi-threshold decryption scheme, we have to construct a whole proof, simulating all the values that an adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ would see in the execution of the different protocols of Σ. We would use the hypothetical existence of such a successful adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ to solve a computationally hard problem.

We next reduce the IND-CCA security of the new multi-policy distributed decryption scheme to the hardness of solving the CDH problem (See a detailed description in Subsection 1.2.1). The proof is in the random oracle model for the five hash functions H_0, H_1, H_2, H_3, H_4. The conclusion is that, under the Computational Diffie-Hellman Assumption for our group $\mathbb{G} = \langle g \rangle$, the new multi-policy distributed decryption scheme enjoys IND-CCA security.

Theorem 4.1.1 *In the random oracle model, the scheme Σ is IND-CCA secure, assuming the Computational Diffie-Hellman problem is hard to solve in $\mathbb{G}$.*

Proof. The proof is by reduction, assuming that hash functions H_0, H_1, H_2, H_3, H_4 are modeled as random oracles. An adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ that can guess the bit d in the game described on Page 86 is used to construct an algorithm $\mathcal{A}^{CDH}$ that solves the CDH problem.

In this simulation $\mathcal{A}^{CDH}$ receives as input (g, g^a, g^b), where $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order p. The goal of $\mathcal{A}^{CDH}$ is to compute g^{ab}. The algorithm $\mathcal{A}^{CDH}$ initializes the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ by giving $\mathsf{params} = (p, \mathbb{G}, g, l, H_0, H_1, H_2, H_3, H_4)$ to him. Since the hash functions H_0, H_1, H_2, H_3, H_4 are supposed to behave as random oracles, $\mathcal{A}^{CDH}$ will create and maintain tables TAB_k, for $k = 0, 1, \ldots, 4$, to answer the hash queries from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$. These answers are produced by $\mathcal{A}^{CDH}$ by first checking if there already exists an entry in the corresponding table for the input of the hash query; if so, $\mathcal{A}^{CDH}$ responds with the existing output; otherwise, $\mathcal{A}^{CDH}$ chooses a new

random value in the corresponding set of valid outputs for that hash function, adds the new relation input-output to the corresponding table, and responds to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ with this output value.

Key distribution. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ chooses the target collective $\mathcal{P}^\star = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, the decryption policies $\Gamma_j = T(t_j, n) \subset 2^B$ for $j = 1, \ldots, \ell$ where $t_1 < t_2 < \cdots < t_\ell$, and also the subset of corrupted members $\widetilde{B} \subset \mathcal{P}^\star$. For simplicity, and without loss of generality, we assume $\widetilde{B} = \{P_1, \ldots, P_{t^\star}\}$, where $1 \leq t^\star \leq n$. This implies that the challenge must be for an index $j \notin \mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } t_j \leq t^\star\}$, so that the corrupted members cannot trivially decrypt the challenge ciphertext from their secret shares.

For the corrupted members, the algorithm $\mathcal{A}^{CDH}$ chooses randomly and independently the shares $\mathsf{sh}_i \in \mathbb{Z}_p$ producing the set $\{\mathsf{sh}_i\}_{\mathcal{P}_i \in \widetilde{B}}$.

For every index $j \in \mathbb{J}^\star$, the algorithm $\mathcal{A}^{CDH}$ chooses at random a secret $s_j \in \mathbb{Z}_p^*$ and a polynomial $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$ such that $f_j(0) = s_j$. It computes (via the hash-table procedure) the values $h_{ij} = H_0(j, \mathsf{sh}_i)$, $r_{ij} = f_j(i) - h_{ij} \bmod p$, for all $P_i \in \widetilde{B}$. For the non-corrupted players, $P_i \notin \widetilde{B}$, the algorithm $\mathcal{A}^{CDH}$ chooses random and independent values $r_{ij} \in \mathbb{Z}_p$, then computes the values $f_j(i)$ by using the chosen polynomial. Finally, $\mathcal{A}^{CDH}$ computes the values $PK_j = g^{s_j}$ and (if necessary) $D_{ij} = g^{f_j(i)}$, for all $P_i \in \mathcal{P}^\star$.

For the rest of indices $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$, the algorithm $\mathcal{A}^{CDH}$ chooses at random $\alpha_j \in \mathbb{Z}_p$ and defines $PK_j = (g^b)^{\alpha_j}$ (which implicitly defines $s_j = b \cdot \alpha_j$). For each $j \notin \mathbb{J}^\star$, $\mathcal{A}^{CDH}$ chooses at random the values r_{ij}, for all $P_i \in \mathcal{P}$. In particular, this means that, for the corrupted members $\mathcal{P}_i \in \widetilde{B}$, we have that the values $r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$ are already determined. Let $f_j(x) \in \mathbb{Z}_p[x]$ be an implicit polynomial of degree $t_j - 1$ such that $f_j(0) = b \cdot \alpha_j$ and $f_j(i) = r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$, for every corrupted player $\mathcal{P}_i \in \widetilde{B}$. Since $|\widetilde{B}| = t^\star < t_j$, the algorithm $\mathcal{A}^{CDH}$ could (if necessary) compute the values $D_{ij} = g^{r_{ij} + H_0(j, \mathsf{sh}_i)}$, for $\mathcal{P}_i \in \widetilde{B}$, and then combine these values with $PK_j = (g^b)^{\alpha_j}$ in order to obtain, by interpolation in the exponent, the rest of values D_{ij}, for non-corrupted players $P_i \notin \widetilde{B}$.

Finally $\mathcal{A}^{CDH}$ sends to the adversary $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ the secret keys $\{\mathsf{sh}_i\}_{\mathcal{P}_i \in \widetilde{B}}$ of the corrupted players, along with the public information $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}})$, where $\mathsf{out}_{\mathsf{pub}} = \{r_{ij}\}_{P_i \in \mathcal{P}^\star, j \in \{1, \ldots, \ell\}}$. If robustness is considered, the values D_{ij} are also sent.

Special hash queries. For the particular case of H_2 queries, the outputs $\bar{g}$ are chosen as random powers of g^b. That is, $\mathcal{A}^{CDH}$ chooses at random a fresh value $\beta \in \mathbb{Z}_p^*$ and computes the new output of H_2 as $\bar{g} = (g^b)^\beta$. The value β is stored as an additional value of the new entry in table TAB_2.

Regarding the simulation of the hash function H_0, the analysis is the same as the one in the proof of Theorem 3.6.1: the simulation is sound as long as the hash queries from $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ do not cause a collision. If the number of hash queries for H_0 is q_0, such a collision happens with probability at most $\frac{q_0^2}{2p} + o\left(\left(\frac{q_0^2}{2p}\right)^2\right)$, which is negligible if $p > 2^{200}$.

Decryption queries. Let (C, j) be a decryption query sent by $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$, for users $\mathcal{P}^\star$ and decryption policy $\Gamma_j = T(t_j, n)$, where $C = (c, u, \bar{u}, e, \sigma)$. We assume $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ is not able to decrypt the ciphertext by itself, and so $t^\star < t_j$.

The first thing $\mathcal{A}^{CDH}$ does is to check the validity of the ciphertext; that is, to check if $e = H_3(\bar{g}, \bar{u}, \bar{w}, j)$, where $w = g^\sigma / u^e$, $\bar{g} = H_2(c, u, w, j) = (g^b)^\beta$ for some value β known by $\mathcal{A}^{CDH}$ and $\bar{w} = \bar{g}^\sigma / \bar{u}^e$. If this equation does not hold, then the answer to the query is $\perp$. Otherwise, $\mathcal{A}^{CDH}$ has to simulate the output of the users $\mathcal{P}_i \in \mathcal{P}^\star$ and recover the message $\widetilde{m}$. This is done as follows.

$\mathcal{A}^{CDH}$ obtains $\{u_{ij}\}_{P_i \in \widetilde{B}}$ trivially, because $u_{ij} = u^{H_0(j, \mathsf{sh}_i) + r_{ij}}$, these corrupted shares sh_i are known to $\mathcal{A}^{CDH}$ and values r_{ij} are included in $\mathsf{out}_{\mathsf{pub}}$. Analogously, the values e_{ij}, σ_{ij} can be consistently computed, for the corrupted players $P_i \in \widetilde{B}$. The difficulties appear when simulating the tuples $(i, u_{ij}, e_{ij}, \sigma_{ij})$ for non-corrupted players $P_i \notin \widetilde{B}$. Again, the technique for doing this will be interpolation in the exponent.

Indeed, since the ciphertext has passed the first validity check, we know that $\mathsf{DiscLog}_g(u) = \mathsf{DiscLog}_{\bar{g}}(\bar{u})$, where $\bar{g} = g^{b \cdot \beta}$ is a value produced by $\mathcal{A}^{CDH}$ in some (previous) hash query for H_2. This means that $u^b = \bar{u}^{1/\beta}$, and so $\mathcal{A}^{CDH}$ can compute the value $u^{b \cdot \alpha_j}$. Again, let $f_j(x)$ be the implicit polynomial of degree $t_j - 1$ such that $f_j(0) = b \cdot \alpha_j$ and $f_j(i) = r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$, for every corrupted player $\mathcal{P}_i \in \widetilde{B}$. The values $\{u_{ij}\}_{P_i \notin \widetilde{B}}$, which are implicitly defined as $u_{ij} = u^{f_j(i)}$, can be computed by interpolation in the exponent, from $u^{f_j(0)} = u^{b \cdot \alpha_j}$ and $\{u_{ij}\}_{P_i \in \widetilde{B}}$.

Regarding the remaining values e_{ij}, σ_{ij}, for non-corrupted players $P_i \notin \widetilde{B}$, they are computed as follows. $\mathcal{A}^{CDH}$ takes random, uniform and independent values $e_{ij}, \sigma_{ij} \in \mathbb{Z}_p$, computes the values $\widehat{u}_{ij} = u^{\sigma_{ij}} / u_{ij}^{e_{ij}}$ and $\widehat{h}_{ij} = g^{\sigma_{ij}} / D_{ij}^{e_{ij}}$, defines the relation $H_4(u_{ij}, \widehat{u}_{ij}, \widehat{h}_{ij}) = e_{ij}$ and adds this relation to TAB_4. The probability that this simulation produces some collision with some other H_4 hash query by $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ is negligible.

The rest of the decryption process can be easily completed by $\mathcal{A}^{CDH}$, who obtains a message $\widetilde{m}$ and sends all the obtained information to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Challenge. At some point, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs two messages m_0, m_1 of the same length, along with a policy $\Gamma_{j^\star}$, where $j^\star \in \{1, \ldots, \ell\}$ and $\widetilde{B} \notin \Gamma_{j^\star}$. Note that the

part of the public key associated to the policy $\Gamma_{j^\star}$ was defined as $PK_{j^\star} = (g^b)^{\alpha_{j^\star}}$. To produce the challenge ciphertext $(C^\star, j^\star)$, the algorithm $\mathcal{A}^{CDH}$ chooses $c^\star \in \{0,1\}^l$ and $\beta^\star, e^\star, \sigma^\star \in \mathbb{Z}_p^*$ at random and defines

$$u^\star = g^a, \bar{g}^\star = g^{\beta^\star}, \bar{u}^\star = (u^\star)^{\beta^\star}, w^\star = g^{\sigma^\star}/(u^\star)^{e^\star}, \bar{w}^\star = (\bar{g}^\star)^{\sigma^\star}/(\bar{u}^\star)^{e^\star}$$

If either the input $(c^\star, u^\star, w^\star, j^\star)$ already exists in TAB_2, or the input $(\bar{g}^\star, \bar{u}^\star, \bar{w}^\star, j^\star)$ already exists in TAB_3, the algorithm $\mathcal{A}^{CDH}$ goes back to choose other random values. Finally, the relation $\bar{g}^\star = H_2(c^\star, u^\star, w^\star, j^\star)$ is added to TAB_2 and the relation $e^\star = H_3(\bar{g}^\star, \bar{u}^\star, \bar{w}^\star, j^\star)$ is added to TAB_3. The challenge ciphertext that $\mathcal{A}^{CDH}$ sends to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ is $(C^\star, j^\star)$ with $C^\star = (c^\star, u^\star, \bar{u}^\star, e^\star, \sigma^\star)$.

More decryption queries. $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ can make more hash and decryption queries (C, j), which are answered exactly in the same way as described before the challenge phase. The only delicate point is that $\mathcal{A}^{CDH}$ could not answer a valid decryption query (C, j), with $C = (c, u, \bar{u}, e, \sigma)$, for which the value of $\bar{g} = H_2(c, u, g^\sigma/u^e, j) = \bar{g}^\star$, because this value does not have the necessary form $(g^b)^\beta$. But this happens only if the two inputs of H_2, in both the challenge ciphertext and in this queried ciphertext, are the same. Since the zero-knowledge proofs $\mathsf{DiscLog}_g(u) = \mathsf{DiscLog}_{\bar{g}}(\bar{u})$ and $\mathsf{DiscLog}_g(w) = \mathsf{DiscLog}_{\bar{g}}(\bar{w})$ are valid, for both the queried ciphertext and the challenge ciphertext, we would have in this case $\bar{u} = \bar{u}^\star$ and $\bar{w} = \bar{w}^\star$. Therefore, it would hold $e = H_3(\bar{g}, \bar{u}, \bar{w}, j) = e^\star$ and consequently $\sigma = \sigma^\star$, since the inputs for H_2 are the same. The conclusion is that the problematic decryption query (C, j) would be exactly equal to the challenge ciphertext $(C^\star, j^\star)$, and this query is not allowed to $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$.

Final analysis. Finally, $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ outputs a guess bit d'. If the probability that $d' = d$ is significantly greater than $1/2$ (random guess), since m_d is perfectly hidden by $H_1(\cdot)$ and H_1 behaves as a random function, the only possibility is that $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ queried the hash function H_1 on the input which corresponds to the challenge ciphertext $(C^\star, j^\star)$, which is $(g^{ab})^{\alpha_{j^\star}}$. Then $\mathcal{A}^{CDH}$ outputs the list TAB_1 of all inputs on which H_1 was queried by $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$, which with overwhelming probability will contain $(g^{ab})^{\alpha_{j^\star}}$. Since $\mathcal{A}^{CDH}$ knows the value $\alpha_{j^\star}$, it can raise all the elements of that list to $(\alpha_{j^\star})^{-1}$. One of the elements of the resulting list will be the solution g^{ab} to the given instance of the CDH problem. As the authors of [88] indicate, the Diffie-Hellman self-corrector in [86] can be used to transform this algorithm $\mathcal{A}^{CDH}$ into an algorithm that outputs the single and correct solution to the CDH problem. □

4.2 Multi-Policy Distributed Signatures

In this section, we deal with the secret task of signing instead of decrypting. Threshold (or distributed) signatures have also received a lot of attention from the cryptographic community [35, 87, 10]; they have applications in scenerios where the cooperation of more than one single entity is necessary to authenticate a message. In our multi-policy setting, we will have a set of users $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ as the possible signers of messages. Depending on the content and the importancy of the message, more or less members of $\mathcal{P}$ can be required to participate in the signing process. In other words, the subset of real signers $A \subset \mathcal{P}$ will choose ad-hoc a signing policy Γ_j among a set of pre-defined different signing policies $\Gamma_1, \ldots, \Gamma_\ell$, such that $A \in \Gamma_j$, and will cooperate to sign the message on behalf of that policy Γ_j. The final verification step will take as inputs the index j, the message, the signature, and the global public key of the set $\mathcal{P}$, in order to check the validity of the signature. Note that the knowledge of identities of the real signers (in subset B) is not necessary to verify a signature, which provides some kind of anonymity (if desired) to the process.

After defining the syntactic definition and the security model for this primitive of multi-policy distributed signatures, we present a scheme for the case of threshold signing policies (that uses as a building block, again, the multi-threshold secret sharing scheme proposed in Section 3.6) and we prove its security. The scheme can be extended to work with more general access structures, as we discussed in Section 3.7.

4.2.1 Syntactic Definition

A multi-policy distributed signature scheme $\Theta = (\Theta.\mathsf{St}, \Theta.\mathsf{KG}, \Theta.\mathsf{Sign}, \Theta.\mathsf{Ver})$ consists of four probabilistic polynomial-time algorithms:

- The randomized *setup* algorithm $\Theta.\mathsf{St}$ takes a security parameter $\lambda \in \mathbb{N}$ and outputs some public parameters params that will be common to all the users in the system: the mathematical groups, generators, hash functions, etc. We write $\mathsf{params} \leftarrow \Theta.\mathsf{St}(1^\lambda)$ to denote an execution of this algorithm.

- The *key generation* algorithm $\Theta.\mathsf{KG}$ for a collective $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ of n users and ℓ different signature policies $\Gamma_j \subset 2^B$ for $j = 1, \ldots, \ell$ has as public output a public key PK that will be used in both the signing and verification steps. We implicitly assume that PK contains the description of $\mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell$. Each user $\mathcal{P}_i \in \mathcal{P}$ receives a secret share sh_i. This key generation process for the collective $\mathcal{P}$ can be run either by a trusted third party, or by the users in $\mathcal{P}$ themselves. We will write $(\{\mathsf{sh}_i\}_{1 \le i \le n}, PK) \leftarrow \Theta.\mathsf{KG}(\mathsf{params}, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ to refer to this key generation protocol.

- The *joint signature* algorithm $\Theta.\mathsf{Sign}$ is a distributed protocol run by some subset of users $A \subset \mathcal{P}$. The common inputs are params, PK, a message m, the secret shares sh_i of the users $\mathcal{P}_i \in A$, and the index j of the desired signature policy Γ_j, where $j \in \{1, \ldots, \ell\}$. The outputs are a signature σ and the index j of the chosen signature policy. We write $(\sigma, j) \leftarrow \Theta.\mathsf{Sign}(\mathsf{params}, PK, m, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A}, j)$ to refer to an execution of this protocol.

- The *verification* algorithm $\Theta.\mathsf{Ver}$ takes as input params, a message m, a signature (σ, j), and the public key PK of the intended receiver group $\mathcal{P}$. The output will be 1 if (σ, j) is a valid signature of m and 0 otherwise. We denote an execution of this algorithm as $\{1, 0\} \leftarrow \Theta.\mathsf{Ver}(\mathsf{params}, m, \sigma, j, PK)$.

For correctness, $\Theta.\mathsf{Ver}(\mathsf{params}, m, \Theta.\mathsf{Sign}(\mathsf{params}, PK, m, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A}, j), PK) = 1$ is required, whenever $A \in \Gamma_j$ and the values $\mathsf{params}, \{\mathsf{sh}_i\}_{1 \leq i \leq n}, PK$ have been obtained by properly executing the protocols $\Theta.\mathsf{St}$ and $\Theta.\mathsf{KG}$.

4.2.2 Security Model

A multi-policy distributed signature scheme must be *robust*. The robustness property holds when the protocols $\Theta.\mathsf{KG}$ and $\Theta.\mathsf{Sign}$ always complete successfully, even under the action of a polynomial-time adversary that is allowed to corrupt an unauthorized set of users.

As any other primitive related to signatures, a multi-policy distributed signature scheme Θ must also be *unforgeable*. That is, any polynomial-time adversary that is allowed to corrupt a subset of users $\widetilde{B} \subset \mathcal{P}$ such that $\widetilde{B} \notin \Gamma_j$ must have negligible probability of success in producing a new valid signature for some message with respect to signing policy Γ_j, even if this adversary has access to a signing oracle for messages and signing policies of his choice. This property is known as *unforgeability against chosen message attacks* (UNF security, for short) and is defined, for a security parameter $\lambda \in \mathbb{N}$, by considering the following game that an attacker $\mathcal{A}_{\mathsf{UNF}}$ plays against a challenger:

1. The challenger runs $\mathsf{params} \leftarrow \Theta.\mathsf{St}(1^\lambda)$ and gives params to $\mathcal{A}_{\mathsf{UNF}}$.

2. $\mathcal{A}_{\mathsf{UNF}}$ chooses a target set $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ of users, ℓ different signature policies $\Gamma_j \subset 2^{\mathcal{P}}$, for $j = 1, \ldots, \ell$, and a subset $\widetilde{B} \subset \mathcal{P}$ of users, to be corrupted. The challenger runs $(\{\mathsf{sh}_i\}_{1 \leq i \leq n}, PK) \leftarrow \Theta.\mathsf{KG}(\mathsf{params}, \mathcal{P}, \Gamma_1, \ldots, \Gamma_\ell)$ and gives to $\mathcal{A}_{\mathsf{UNF}}$ the values PK and $\{\mathsf{sh}_i\}_{\mathcal{P}_i \in \widetilde{B}}$.

   We consider only *static* adversaries who choose the subset $\widetilde{B}$ of corrupted users at the beginning of the attack.

3. [**Queries**] $\mathcal{A}_{\mathsf{UNF}}$ can make adaptive queries to a distributed signing oracle for the target set $\mathcal{P}$: $\mathcal{A}_{\mathsf{UNF}}$ sends a tuple (m, j) for the signature policy Γ_j. The challenger runs the distributed signature algorithm $(\sigma, j) \leftarrow \Theta.\mathsf{Sign}(\mathsf{params}, m, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A}, j)$ for an authorized subset $A \in \Gamma_j$. The attacker $\mathcal{A}_{\mathsf{UNF}}$ must be given all the information that corrupted players (in $\widetilde{B}$) would obtain during the execution of this protocol $\Theta.\mathsf{Sign}$, including the final signature and all the broadcast information.

4. At some point, $\mathcal{A}_{\mathsf{UNF}}$ outputs a forgery $(j^\star, m^\star, \sigma^\star)$. We say that $\mathcal{A}_{\mathsf{UNF}}$ is successful if: (1) $\widetilde{B} \notin \Gamma_j$, (2) $\Theta.\mathsf{Ver}(\mathsf{params}, m^\star, \sigma^\star, j^\star, PK) = 1$, and (3) $(j^\star, m^\star, \sigma^\star)$ has not been obtained by $\mathcal{A}_{\mathsf{UNF}}$ in a signature query (step 3).

The advantage of such a (static) adversary $\mathcal{A}_{\mathsf{UNF}}$ in breaking the UNF security of the multi-policy distributed signature scheme is defined as the probability that $\mathcal{A}_{\mathsf{UNF}}$ is successful in the game above.

A multi-policy distributed signature scheme Θ is UNF secure if the advantage of any such polynomial-time (static) adversary $\mathcal{A}_{\mathsf{UNF}}$ is a negligible function of the security parameter λ.

4.2.3 A New Multi-Threshold Signature Scheme

We propose here a new multi-policy distributed signature scheme, that we describe (for simplicity) for the case where all the signing policies are threshold ones: $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$, where $1 \leq t_1 < t_2 < \ldots < t_\ell \leq n$. See Section 3.7 for an extension of this scheme to the case of more general access structures. The scheme is inspired by the (single) threshold signature scheme proposed by Boldyreva in [10] (which is itself inspired by the individual signature scheme of Boneh-Lynn-Shacham [14]). A key ingredient in the design of the new scheme will be, again, the multi-threshold secret sharing scheme proposed in Section 3.6. We also use the notion of Gap Diffie-Hellman groups (See Page 7 for a detailed description on this).

The protocols of the new multi-threshold signature scheme Θ work as follows.

Setup: $\Theta.\mathsf{St}(1^\lambda)$.
Given a security parameter $\lambda \in \mathbb{N}$, a GDH group $\mathbb{G} = \langle g \rangle$ of prime order p, such that p is λ bits long, is chosen. Two hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_1 : \{0,1\}^* \to \mathbb{G}$ are chosen. The output of this protocol is $\mathsf{params} = (p, \mathbb{G}, g, H_0, H_1)$.

Key Generation: $\Theta.\mathsf{KG}(\mathsf{params}, \mathcal{P}, t_1, \ldots, t_\ell, n)$.
This protocol is exactly the same as the Key Generation protocol of the multi-threshold decryption scheme Σ, in Section 4.1. Let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a set of n users and let $\Gamma_j = T(t_j, n)$ for $j = 1, \ldots, \ell$ be the threshold signing policies defined on $\mathcal{P}$, where $1 \leq t_1 < t_2 < \ldots < t_\ell \leq n$.

For $j = 1, \ldots, \ell$, the value $PK_j = g^{s_j}$ is computed, for a random value $s_j \in \mathbb{Z}_p^*$ that will remain unknown to the members of $\mathcal{P}$. These ℓ secret values will correspond to a secret vector $\vec{s} = (s_1, \ldots, s_\ell)$ of the multi-threshold secret sharing scheme described in Section 3.6, that will be shared by running the distribution protocol $\Omega.\mathsf{Dist}(\mathcal{P}, t_1, \ldots, t_\ell, \vec{s})$, with hash function H_0:

1. Choose random values $\mathsf{sh}_i \in \mathbb{Z}_p^\star$, pairwisse different for $i = 1, 2, \ldots, n$, as the secret shares.

2. Choose random polynomials $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$, for $j = 1, \ldots, \ell$, such that $f_j(0) = s_j$.

3. For $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, \ell$, compute the values $h_{ij} = H_0(j, \mathsf{sh}_i)$ and $r_{ij} = f_j(i) - h_{ij} \bmod p$. Compute the public verification values $D_{ij} = g^{h_{ij}+r_{ij}}$.

4. The secret share sh_i is sent to player P_i via a secure channel, whereas the public output of the protocol is $\mathsf{out}_{\mathsf{pub}} = \{r_{ij}\}_{P_i \in \mathcal{P}, j \in \{1,\ldots,\ell\}}$.

The global public key is $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}}, \{D_{ij}\}_{P_i \in \mathcal{P}, 1 \le j \le \ell})$, whereas the secret share for each player P_i is sh_i.

Joint Signature: $\Theta.\mathsf{Sign}(\mathsf{params}, PK, m, A, \{\mathsf{sh}_i\}_{\mathcal{P}_i \in A}, j)$.
Let $A \subset \mathcal{P}$ be a subset of users in $\mathcal{P}$ that want to cooperate to sign a message m with respect to a signing policy $\Gamma_j = T(t_j, n)$ for which they form an authorized subset, $A \in \Gamma_j$. Members of A proceed as follows:

1. Each $\mathcal{P}_i \in A$ computes $h_{ij} = H_0(j, \mathsf{sh}_i)$, recovers r_{ij} from $\mathsf{out}_{\mathsf{pub}}$ and broadcasts his signature share $\sigma_{ij} = H_1(m, j)^{h_{ij}+r_{ij}} \in \mathbb{G}$.

2. The rest of members of A verify if $(g, D_i, H_1(m, j), \sigma_{ij})$ is a valid Diffie-Hellman tuple. If this is not the case, then $(i, \perp)$ is broadcast.

3. If there are not t_j valid signature shares (i, σ_{ij}), then stop and output $\perp$. Otherwise, from t_j valid signature shares $\{\sigma_{ij}\}_{\mathcal{P}_i \in A}$, one can consider the Lagrange interpolation coefficients $\lambda_{ij}^A \in \mathbb{Z}_p$ such that $s_j = f_j(0) = \sum_{\mathcal{P}_i \in A} \lambda_{ij}^A \cdot f_j(i)$.

4. Return the resulting signature and index (σ, j), where $\sigma = \prod_{\mathcal{P}_i \in A} \sigma_{ij}^{\lambda_{ij}^A}$.

Verification: $\Theta.\mathsf{Ver}(\mathsf{params}, m, \sigma, j, PK)$
In the verification step it is enough to check if $(g, PK_j, H_1(m, j), \sigma)$ is a valid Diffie-Hellman tuple. Return 1 if this is the case, or 0 otherwise.

4.2.4 Security Analysis

The multi-policy threshold signature scheme described above, with a trusted dealer in charge of the key generation phase, is trivially robust: during the joint signature generation phase, cheating players are detected in step 2 and rejected from the protocol. Assuming the remaining players are enough (i.e. they are at least t_j), the signing protocol finishes correctly. One way to ensure this is by requiring that an adversary can corrupt at most $n - t_\ell$ players.

Regarding unforgeability, we now prove that the proposed scheme is UNF secure provided the Computational Diffie-Hellman (CDH) problem is hard in the GDH group $\mathbb{G}$. The proof is in the random oracle model for hash functions H_0, H_1.

Theorem 4.2.1 *In the random oracle model, the scheme Θ is UNF secure, assuming the Computational Diffie-Hellman problem is hard to solve in the GDH group $\mathbb{G}$.*

Proof. The proof is by reduction, assuming that hash functions H_0, H_1 are modeled as random oracles. An adversary $\mathcal{A}_{\mathsf{UNF}}$ that has non-negligible success in forging a new valid signature is used to construct an algorithm $\mathcal{A}^{CDH}$ that solves the CDH problem in $\mathbb{G}$.

$\mathcal{A}^{CDH}$ receives as input (g, g^a, g^b), where $\mathbb{G} = \langle g \rangle$ is a GDH group of prime order p. The goal of $\mathcal{A}^{CDH}$ is to compute g^{ab}. The algorithm $\mathcal{A}^{CDH}$ initializes the attacker $\mathcal{A}_{\mathsf{IND\text{-}CCA}}$ by giving $\mathsf{params} = (p, \mathbb{G}, g, H_0, H_1)$ to him. Since the hash functions H_0, H_1 are supposed to behave as random oracles, $\mathcal{A}^{CDH}$ will create and maintain tables TAB_0 and TAB_1 to answer the hash queries from $\mathcal{A}_{\mathsf{UNF}}$. These answers are produced by $\mathcal{A}^{CDH}$ by first checking if there already exists an entry in the corresponding table for the input of the hash query; if so, $\mathcal{A}^{CDH}$ responds with the existing output; otherwise, $\mathcal{A}^{CDH}$ chooses a new random output, adds the new relation input-output to the corresponding table, and responds to $\mathcal{A}_{\mathsf{UNF}}$ with this output value. Hash queries (m, j) to H_1 are answered in the following way. Let q_1 be the maximum number of such H_1 queries. $\mathcal{A}^{CDH}$ chooses at random an index $k^\star \in \{1, \ldots, q_1\}$ for a special query $(\tilde{m}, \tilde{j})$. For this special query, $\mathcal{A}^{CDH}$ chooses a random value $\tilde{\beta} \in \mathbb{Z}_p$ and defines the relation $H_1(\tilde{m}, \tilde{j}) = (g^a)^{\tilde{\beta}}$. For the rest of H_1 queries (m, j), $\mathcal{A}^{CDH}$ chooses a random value $\beta \in \mathbb{Z}_p$ and defines the relation $H_1(m, j) = g^\beta$. These relations are stored in TAB_1.

Key distribution. $\mathcal{A}_{\mathsf{UNF}}$ chooses the target collective $\mathcal{P}^\star = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, the decryption policies $\Gamma_j = T(t_j, n) \subset 2^B$ for $j = 1, \ldots, \ell$ where $t_1 < t_2 < \cdots < t_\ell$, and also the subset of corrupted members $\widetilde{B} \subset \mathcal{P}^\star$. For simplicity, we assume $\widetilde{B} = \{P_1, \ldots, P_{t^\star}\}$, where $1 \leq t^\star \leq n$. Let us define the set of indices $\mathbb{J}^\star = \{j \in \{1, \ldots, \ell\} \text{ s.t. } t_j \leq t^\star\}$, so that the corrupted players can trivially sign messages for signing policies Γ_j, if $j \in \mathbb{J}^\star$.

For the corrupted members of $\mathcal{P}^\star$, the algorithm $\mathcal{A}^{CDH}$ chooses randomly and independently the shares $\mathsf{sh}_i \in \mathbb{Z}_p$ producing the set $\{\mathsf{sh}_i\}_{\mathcal{P}_i \in \widetilde{B}}$.

For every index $j \in \mathbb{J}^\star$, the algorithm $\mathcal{A}^{CDH}$ chooses at random a secret $s_j \in \mathbb{Z}_p^*$ and a polynomial $f_j(x) \in \mathbb{Z}_p[x]$ of degree $t_j - 1$ such that $f_j(0) = s_j$. It computes (via the hash-table procedure) the values $h_{ij} = H_0(j, \mathsf{sh}_i)$, $r_{ij} = f_j(i) - h_{ij} \bmod p$, for all $P_i \in \widetilde{B}$. For the non-corrupted players, $P_i \notin \widetilde{B}$, the algorithm $\mathcal{A}^{CDH}$ chooses random and independent values $r_{ij} \in \mathbb{Z}_p$, then computes the values $f_j(i)$ by using the chosen polynomial. Finally, $\mathcal{A}^{CDH}$ computes the values $PK_j = g^{s_j}$ and $D_{ij} = g^{f_j(i)}$, for all $P_i \in \mathcal{P}^\star$.

For the rest of indices $j \in \{1, \ldots, \ell\}$, $j \notin \mathbb{J}^\star$, the algorithm $\mathcal{A}^{CDH}$ chooses at random $\alpha_j \in \mathbb{Z}_p$ and defines $PK_j = (g^b)^{\alpha_j}$ (which implicitly defines $s_j = b \cdot \alpha_j$). For each $j \notin \mathbb{J}^\star$, $\mathcal{A}^{CDH}$ chooses at random the values r_{ij}, for all $P_i \in \mathcal{P}$. In particular, this means that, for the corrupted members $\mathcal{P}_i \in \widetilde{B}$, we have that the values $r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$ are already determined. Let $f_j(x) \in \mathbb{Z}_p[x]$ be an implicit polynomial of degree $t_j - 1$ such that $f_j(0) = b \cdot \alpha_j \bmod p$ and $f_j(i) = r_{ij} + H_0(j, \mathsf{sh}_i) \bmod p$, for every corrupted player $\mathcal{P}_i \in \widetilde{B}$. Since $|\widetilde{B}| = t^\star < t_j$, the algorithm $\mathcal{A}^{CDH}$ can compute the values $D_{ij} = g^{r_{ij} + H_0(j, \mathsf{sh}_i)}$, for $\mathcal{P}_i \in \widetilde{B}$, and then combine these values with $PK_j = (g^b)^{\alpha_j}$ in order to obtain, by interpolation in the exponent, the rest of values $D_{ij} = g^{f_j(i)}$, for non-corrupted players $P_i \notin \widetilde{B}$.

Finally $\mathcal{A}^{CDH}$ sends to the adversary $\mathcal{A}_{\mathsf{UNF}}$ the secret keys $\{\mathsf{sh}_i\}_{\mathcal{P}_i \in \widetilde{B}}$ of the corrupted players, along with the public information $PK = (PK_1, \ldots, PK_\ell, \mathsf{out}_{\mathsf{pub}}, \{D_{ij}\}_{P_i \in \mathcal{P}^\star, 1 \le j \le \ell})$, where $\mathsf{out}_{\mathsf{pub}} = \{r_{ij}\}_{P_i \in \mathcal{P}^\star, j \in \{1, \ldots, \ell\}}$.

Simulating H_0. Once again, the simulation of the hash function H_0 is consistent as long as the H_0 hash queries from $\mathcal{A}_{\mathsf{UNF}}$ do not cause a collision with the implicitly determined values $\{\mathsf{sh}_i\}_{P_i \notin \widetilde{B}}$. If the number of hash queries for H_0 is q_0, such a collision happens with probability at most $\frac{q_0^2}{2p} + o\left(\left(\frac{q_0^2}{2p}\right)^2\right)$, which is negligible if $p > 2^{200}$.

Signing queries. Let (m, j) be a signing query asked by $\mathcal{A}_{\mathsf{UNF}}$. If $(m, j) = (\tilde{m}, \tilde{j})$, then $\mathcal{A}^{CDH}$ aborts and outputs $\perp$. Otherwise, $\mathcal{A}^{CDH}$ knows a value β such that $H_1(m, j) = g^\beta$. Then, $\mathcal{A}^{CDH}$ can easily compute correct signature shares $\sigma_{ij} = H_1(m, j)^{h_{ij} + r_{ij}} = \left(g^{h_{ij} + r_{ij}}\right)^\beta = D_{ij}^\beta$, for every player $P_i \in \mathcal{P}^\star$.

Forgery. At some point, and with non-negligible probability, $\mathcal{A}_{\mathsf{UNF}}$ outputs a valid signature $(\sigma^\star, j^\star)$ for some index $j \notin \mathbb{J}^\star$ and some message $m^\star$, different from the valid signatures obtained through signing queries. Since the signature is valid and H_1 behaves as a random function, A_{UNF} must have queried the pair $(m^\star, j^\star)$ to the hash oracle for H_1. With probability at least $1/q_1$, we have $(m^\star, j^\star) = (\tilde{m}, \tilde{j})$, and

in this case we have $H_1(m^\star, j^\star) = (g^a)^{\tilde{\beta}}$, for some value β known by $\mathcal{A}^{CDH}$, whereas the associated part of the public key is $PK_j = (g^b)^{\alpha_j}$, for some value α_j also known by $\mathcal{A}^{CDH}$. Since $\sigma^\star$ is a valid signature, we have that $\left(g, (g^b)^{\alpha_j}, (g^a)^{\tilde{\beta}}, \sigma^\star\right)$ is a valid Diffie-Hellman tuple, which means that $\sigma^\star = g^{ab\alpha_j\tilde{\beta}}$. In this case, $\mathcal{A}^{CDH}$ can easily obtain the desired solution g^{ab} of the given instance of the CDH problem. □

For simplicity, we have described a security reduction with a loss factor of q_1. It is possible to improve this reduction by using the techniques of Coron [26], and then the loss factor becomes linear in q_S, the number of signing queries, which is usually considered to be smaller than the number q_1 of hash queries.

4.3 Relations with Attribute-Based Cryptography

In an *attribute-based cryptosystem*, each user has a subset of attributes $A \subset \mathcal{P}$ from a universe $\mathcal{P} = \{\mathsf{at}_1, \ldots, \mathsf{at}_n\}$ of attributes, and receives from a trusted master entity a secret key according to those attributes. Later, the specific access policy $\Gamma \subset 2^{\mathcal{P}}$ defining the subsets of attributes that must be held by someone so that he is able to perform the secret task (either decrypting or signing) is chosen "on the fly", among all the possible (monotone increasing) access policies in $\mathcal{P}$.

Attribute-based cryptosystems have received a lot of attention from the cryptographic community in the last years, and different schemes have been proposed both for encryption (see for instance [38, 58]) and for signatures (in [65, 40]).

It is easy to see that attribute-based cryptosystems are a more general primitive than the primitives of multi-policy distributed cryptosystems that we have introduced in this chapter. Let us consider the case of encryption/decryption (the case of signatures work in an analogous way). Let us take an attribute-based encryption scheme, and let us associate each attribute $\mathsf{at}_i \in \mathcal{P}$ with one player $P_i \in \mathcal{P}$. Each player receives from the master entity the secret key (or secret share) sh_i corresponding to the fact of holding only attribute at_i (all these secret keys $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ are computed by the master entity in a single execution of the key generation protocol, with a common randomness). Later, the sender of a message m addressed to $\mathcal{P}$ chooses the desired decryption policy $\Gamma \subset 2^{\mathcal{P}}$ and encrypts m by using the attribute-based encryption protocol. Only if an authorized subset of players $A \subset \mathcal{P}$, $A \in \Gamma$ put together their secret shares, they will be able to run the attribute-based decryption protocol and recover the message m.

Therefore, it seems that the primitives of multi-policy distributed decryption and signature can already be implemented by using existing attribute-based cryptosystems, and actually the resulting schemes are more general, because the allowed decryption / signing policies must not necessarily be inside a pre-defined list

$\{\Gamma_1, \ldots, \Gamma_\ell\}$, as it happens in our multi-policy distributed cryptosystems; they can be whatever access policy defined on the set $\mathcal{P}$. However, we will explain below a list of drawbacks suffered by this attribute-based approach, as opposed to the direct approach that we have followed to design multi-policy distributed cryptosystems.

Restricted access policies. Even if, in theory, an attribute-based cryptosystem could allow encryptions or signatures for any access policy $\Gamma \subset 2^{\mathcal{P}}$, in specific proposals this is not always the case. For instance, the most efficient attribute-based cryptosystems proposed up to date (in terms of the length of ciphertexts or signatures, the computational cost of the protocols, etc.) admit only threshold policies [2, 40]. The resulting functionality can be achieved by our multi-threshold decryption and signature schemes, by taking $\ell = n$ and $t_j = j$, for all $j = 1, \ldots, n$.

Necessity of a trusted master entity. In any attribute-based cryptosystem, a master entity must generate and distribute the secret keys between users. This means that this entity has to be trusted, because otherwise it could impersonate any user in the system. Although we have described our multi-threshold schemes with a trusted dealer who generates and distributes the secret shares, this is not an intrinsic property of this kind of schemes, and actually we show in Section 3.5 how the own players in $\mathcal{P}$ can generate the public parameters and the secret shares by themselves, without the participation of any external (and trusted) dealer.

Length of ciphertexts, public parameters, signatures and secret shares. In most of the attribute-based cryptosystems proposed so far, the length of the ciphertexts or signatures is at least linear in the number of attributes involved in the access policy (which, for simplicity, we assume to be n). In our multi-threhold cryptosystems, the length of ciphertexts and signatures is constant. The only attribute-based cryptosystems with constant-length ciphertexts or signatures [2, 40], on the other hand, have secret keys whose length is at least linear in n. In our multi-threshold schemes, each secret key (or share) contains a single element in $\mathbb{Z}_p$. The length of the public parameters in our schemes, which is linear in $n \cdot \ell$, is comparable to the length of the public parameters in all existing attribute-based cryptosystems, which is at least linear in n, and sometimes linear in n^2.

Computational assumptions. Up to now, all the existing (and moderately efficient) attribute-based cryptosystems with a formal proof of security make use of bilinear pairings, and base their security on (sometimes, quite artificial) computational assumptions related to bilinear groups. For multi-policy distributed cryptosystems, we have seen that they can be constructed by combining, essentially, a multi-secret

sharing scheme with a standard distributed cryptosystem. The particular instantiation of a multi-policy distributed decryption scheme that we have described and analyzed in Section 4.1, for instance, is provably secure under the well-established assumption that the Computational Diffie-Hellman problem is hard.

4.4 Other Multi-Policy Cryptosystems

Up to now we have applied the multi-secret sharing scheme defined in Section 3.6 to both multi-policy distributed decryption schemes and multi-policy distributed signatures, where the distributed part is in the decryption and signature protocols respectively.

The multi-secret sharing schemes proposed in Chapter 3 can also be applied in the construction of multi-policy signcryption schemes with threshold unsigncryption and multi-policy digital signatures with distributed verification using the schemes described in Chapter 2. To achieve this goal, it is necessary to replace the underlying secret sharing scheme of the original schemes by one of the computational secure multi-secret sharing schemes proposed in Chapter 3. Moreover, multi-policy distributed signcryption schemes can result, when the distributed part relies on the signcryption protocol [64] instead of the unsigncryption protocol.

#	Distributed Cryptosystems	Studied in this Thesis
1	Distributed decryption	Section 4.1
2	Distributed signatures	Section 4.2
3	Distributed signcryption	-
4	Distributed unsigncryption	Sections 2.3 and 2.4
5	Signatures with distributed verification	Section 2.6
6	Distributed key distribution	-
7	Metering schemes	-
8	Fuzzy identity-based encryption	-

Table 4.1: Distributed cryptographic protocols.

Another distributed cryptographic protocols which are working with only one access structure, as summarized in Table 4.1, can be generalized to multi-policy with the same techniques used in this chapter: in distributed key distribution [70] and predistribution schemes a set of servers send to several users some information that allow the latest to compute common secret keys. In metering schemes [32] audit agencies are able to measure the interaction between clients and servers in the network during a certain number of time frames. Naor and Pinkas proposed in [69] a metering scheme in which any server is able to compute a proof to be sent to the audit agency if and

only if it has been visited by at least t clients, where t is a fixed threshold. In fuzzy identity-based encryption schemes, introduced by Sahai and Waters in [79], a user with the secret key for the identity ID can decrypt a ciphertext encrypted with the public key of identity ID' if and only if both identities differ in at most t positions, for a fixed threshold t. For example, in a social network each user dictates their own policy in order to restrict access to their personal information.

Conclusions

In this section we recapitulate the main results of this thesis and give an overview of open tasks which can lead to possible future work. Opposite to the traditional approach in the conclusion section, where the central points of the thesis are simply summed up, we explain here the results as a story. In the following we show how we came to the results from a time point of view (not exactly the same order as explained in this thesis) and how some of them motivated the research in other fields of cryptography.

In the beginning, motivated by distributed signatures we started to research how to produce signature schemes with the distributed part in the verification protocol instead of the signing protocol. A first proposal, based on Schnorr signature [80], was proved unforgeable but its indistinguishability property became true only assuming that an adversary is not allowed to ask signature queries to a verification oracle. As the security of this first proposal was not strong enough, it forced us to start studying other cryptographic primitives to achieve this goal. At this moment the question was: how can we find such schemes and proof their security in an acceptable level, similarly to IND-CCA for encryption schemes?

Moving the Research to Signcryption Schemes

The above primitive is very similar to signcryption schemes [93] but with the difference that the message is public in digital signatures. This key point motivated us to start studying signcryption schemes with distributed unsigncryption, an area not sufficiently investigated up to then.

The research in Chapter 2 on signcryption schemes is one of the main contributions in this thesis. We have considered in Section 2.1 the strong security properties that one could (or should) require for a signcryption scheme with threshold unsigncryption: existential unforgeability under insider chosen message attacks and indistinguishability under insider chosen ciphertext attacks, in a multi-user setting. Most of the (few) threshold unsigncryption schemes proposed in the literature, either in the traditional PKI or in the identity-based scenario, do not achieve this level of security. This includes generic constructions obtained by composing a secure signature scheme and a

secure threshold decryption scheme.

Moreover, we have constructed in Sections 2.3 and 2.4 two threshold unsigncryption schemes which achieve those strong security properties. We prove the security of the first one in the random oracle model [6]. On the other hand, we are able to prove the security of the second proposed scheme in the standard model, which employs two signature schemes and the ideas by Canetti-Halevi-Katz to achieve CCA security from identity-based selectively secure encryption [21]. The two schemes enjoy a "splitting" property which can be very useful for applications requiring some level of privacy for the sender of the digital information.

In parallel to this research, we were thinking that the dealer role could be replaced in the distributed parts of the schemes. As usual, we could modify some protocols in a way that the users take over this role or on the other hand use some verifiable secret sharing schemes to verify the validity of the shares distributed by the dealer. We presented in Section 3.1 a simple publicly verifiable secret sharing scheme based on the homomorphic Paillier's encryption scheme [73], which could be used to achieve this goal. The verification process in this proposal is made non-interactive without using the Fiat-Shamir technique or any additional Zero Knowledge proof. Moreover, this scheme also have homomorphic properties which could be used together with other cryptographic tools as e.g. multi-party computation.

With the original motivation in mind, we applied the proposed signcryption schemes to build digital signatures with distributed verification in Section 2.6, where the signer is able to restrict the verification capability. To achieve this goal, both the set of verifiers and also the access structures, which define the authorized subsets of verifiers, are fixed from the begining. For a given signature, its validity can be verified only if the verifiers of an authorized subset cooperate. Otherwise, no information can be obtained about the validity of a signature for a certain message. Two different examples have been proposed for this primitive in a threshold framework. The security of both signature schemes follow the security of those signcryption schemes, where they are based on. That is, both achieve IND-CCA security, one in the random oracle model and the other one in the standard model.

In our threshold unsigncryption approach, we considered a single user A, which creates the signcryption, and a collective B of receivers, where the cooperation of some authorized subset of users is necessary to perform the unsigncryption process. When working on that, we discussed the possibility that users could play both roles, sender and receiver at the same time. In this case, we can run the protocols for a set of participants $\mathcal{P} = \{P_1, \ldots, P_n\}$, where to each sender $P_i \in \mathcal{P}$ a different access structure $\Gamma_i \subset 2^{\mathcal{P}-\{P_i\}}$ would correspond, for $i \in \{1, \ldots, n\}$, for the role of authorized receivers.

Several solutions were discussed to achieve this goal, where the same secret information (or share) could be used to signcrypt and cooperatively unsigncrypt. The idea

to minimize the total number of secret information to be kept by the users, when only one secret is shared, does not work because security can not be ensured in general for some families of authorized subsets.

So, it seems that ℓ secrets have to be shared to avoid this problem and not only one secret, as thought at the beginning. The most simple (inefficient) solution is to run ℓ parallel instances of Shamir. The drawback with this solution is that every user keeps secret a vector of ℓ shares, one for every secret. That is, the secret information of every participant grows linearly with the number of secrets, which is very inefficient. The question at this point is how to share multiple secrets in an efficient way. That is, reducing (more than with the trivial solution) the quantity of secret information to be kept secretly by every user.

The Key Point are Computational Multi-Secret Sharing Schemes

Sharing multiple secrets led us to think about multi-policy versions of traditional distributed protocols; not only unsigncryption, but also more popular protocols like decryption or signature independently. We immediately realized that a key ingredient in the design of efficient multi-policy distributed cryptosystems would be efficient multi-secret sharing schemes, presented in Chapter 3, where ℓ different secrets could be distributed among the participants according to ℓ (possibly different) access strutures or policies.

We started to work on multi-secret sharing schemes (MSSSs) from an information-theoretical point of view, which led the results in Section 3.2. We proved then that, even for the more relaxed notion of unconditional security, and for some kinds of access structures (in particular, threshold ones), we have the same efficiency problem as with the strongest level of unconditional security: the length of each secret share must be linear in ℓ. Since we are looking for more efficient solutions, we moved to the scenario of MSSSs with computational security.

When working on this, we constructed a MSSS, denoted in Section 3.6 by Ω_3, where each vector of secret shares contains just only one element, reducing the amount of secret information considerably for every user. Moreover, we prove its computational security in the random oracle model, where a hash value is used to compute some values in the different protocols. As far as we know, this is the first formal security analysis for a computational multi-secret sharing scheme in the literature.

Coming back to multi-policy distributed cryptosystems, as presented in Chapter 4 we showed the utility of this sharing scheme Ω_3 by using it as a key ingredient in the design of two schemes which have two different functionalities: multi-policy decryption presented in Section 4.1 and multi-policy signatures presented in Section 4.2. The schemes proposed here are based on Shoup-Gennaro [88] and Boldyreva [10] for decryption and signature respectively. We prove the security of these two

new multi-policy cryptosystems in a formal security model. Obviously, the length of secret information kept by the users in the proposed distributed cryptosystems depends directly on the length of the shares produced by the underlying MSSS. The two new primitives provide similar functionalities as attribute-based cryptosystems, with some advantages and some drawbacks that are discussed in Section 4.3.

As a result of the multi-policy research, which outputs the sharing scheme Ω_3, we started working more in depth in MSSS. All that produces two new MSSSs that allowed us (as before) to distribute ℓ different secrets among a set of n players, each one according to an access structure. The first one, denoted in Section 3.3 by Ω_1, is inspired on the secret sharing scheme of Krawczyk [55] and the second scheme, denoted in Section 3.4 by Ω_2, is a generalization of some previous schemes [39, 62]. Both schemes are proved to enjoy computational security, by reduction to the semantic security of the underlying symmetric encryption scheme. Both security proofs are in the standard model and provide the exact relation between the security of the involved primitives. In fact, the sharing scheme Ω_2 is like the previous scheme Ω_3 defined in Section 3.6 but replacing the secure one-way hash function H of Ω_3 by an underlying symmetric encryption scheme Π. Consequently, Ω_3 is slightly more efficient than Ω_2 because hash functions are more efficient than encrypt and decrypt, but the price to pay is the security in the random oracle model.

Finally, we presented in Section 3.5 some results related to the efficiency of the schemes Ω_1 and Ω_2. In this part of the thesis we compare both sharing schemes, taking into account their efficiency properties, their security analysis and possible extensions. One of the schemes has very short secret shares (independently of the number ℓ of secrets) and can be easily extended to work without any trusted dealer. The other scheme has longer secret shares and the extension to work without a trusted dealer is much more complicated, but on the other hand it produces shorter public outputs and the security relation with the underlying symmetric encryption scheme is better, which may have consequences on the final efficiency of the scheme.

Future Work

As future work, one could investigate if other more efficient threshold unsigncryption schemes than the ones proposed in Chapter 2 can be designed. One possibility could be to look for a signcryption scheme with IND-CCA security in the standard model, where the bilinear maps are not used.

The multi-policy distributed cryptosystems presented in this thesis were possible using the multi-secret sharing scheme Ω_3. As both inherent distributed cryptosystems (single policy of Shoup-Gennaro [88] for decryption and Boldyreva [10] for signatures) are secure in the random oracle model, it was decided to use the sharing scheme Ω_3, which is the most efficient MSSS among all the proposed sharing schemes and with

security in the random oracle model. An open problem is to look for other distributed cryptosystems secure in the standard model to create multi-policy distributed cryptosystems secure in this model. In this case, the sharing schemes Ω_1 and Ω_2 would come into play due to their security in the standard model.

Appendix A

Published Papers

In this section we briefly present several papers which this thesis is based on. These papers have been published in different journals, conferences and workshops and cover the main results of this thesis.

Journals

- J. Herranz, A. Ruiz and G. Sáez. New Results and Applications for Multi-Secret Sharing Schemes. *Designs, Codes and Cryptography*, to appear, available at http://link.springer.com/article/10.1007/s10623-013-9831-6 (2013).

- J. Herranz, A. Ruiz and G. Sáez. Sharing Many Secrets with Computational Provable Security. *Information Processing Letters*, vol. **113**, pp. 572–579 (2013).

- J. Herranz, A. Ruiz and G. Sáez. Signcryption Schemes with Threshold Unsigncryption, and Applications. *Designs, Codes and Cryptography*, to appear, available at http://link.springer.com/article/10.1007/s10623-012-9688-0 (2013).

International Conferences

- J. Herranz, A. Ruiz and G. Sáez. Fully Secure Threshold Unsigncryption. *Proceedings of ProvSec'10*, LNCS **6402**, Springer-Verlag, pp. 261–278 (2010).

- A. Ruiz and J.L. Villar. Publicly Verifiable Secret Sharing from Paillier's Cryptosystem. *Proceedings of WEWoRC'05*, LNI **74**, Editorial GI, pp. 98–108 (2005).

National Workshops

- J. Herranz, A. Ruiz, G. Sáez. Firmas Digitales con Verificación Distribuida en el Modelo de Seguridad Estándar. *Actas de la XII Reunión Española de Criptología y Seguridad de la Información*, available at http://recsi2012.mondragon.edu/es/programa/recsi2012_submission_07.pdf (2012).

- J. Herranz, A. Ruiz, G. Sáez. Máxima Seguridad para Firmas Digitales con Verificación Distribuida. *Actas de la XI Reunión Española de Criptología y Seguridad de la Información*, pp. 97–103 (2010).

Bibliography

[1] J.H. An, Y. Dodis and T. Rabin. On the security of joint signature and encryption. *Proceedings of Eurocrypt'02*, LNCS **2332**, Springer-Verlag, pp. 83–107 (2002).

[2] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. De Panafieu and C. Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, vol. **422**, pp. 15–38 (2012).

[3] M. Bellare, A. Boldyreva and S. Micali. Public-key encryption in a multi-user setting: security proofs and improvements. *Proceedings of Eurocrypt'00*, LNCS **1807**, Springer-Verlag, pp. 259–274 (2000).

[4] M. Bellare, A. Boldyreva and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 171–188 (2004).

[5] M. Bellare, A. Desai, E. Jokipii and P. Rogaway. A concrete security treatment of symmetric encryption. *FOCS'97*, IEEE Society Press, pp. 394–403 (1997).

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pp. 62–73 (1993).

[7] M. Bellare and P. Rogaway. Introduction to Modern Cryptography. *Notes for the course CSE207 of the University of California, San Diego*. Available at http://cseweb.ucsd.edu/classes/sp99/cse207/index.html (2012).

[8] G.R. Blakley. Safeguarding Cryptographic Keys. *Proceedings of the National Computer Conference, American Federation of Information*, Processing Societies Proceedings **48**, pp. 313–317 (1979).

[9] C. Blundo, A. De Santis, G. Di Crescenzo, A.G. Gaggia and U. Vaccaro. Multi-secret sharing schemes. *Proceedings of Crypto'94*, LNCS **839**, Springer-Verlag, pp. 150–163 (1994).

[10] A. Boldyreva. Threshold Signatures, Multisignatures and blind signatures based on the Gap-Diffie-Hellman-Group signature scheme. *Proceedings of PKC'03*, LNCS **2567**, Springer-Verlag, pp. 31–46 (2003).

[11] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 223–238 (2004).

[12] D. Boneh, X. Boyen and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. *Proceedings of CT-RSA'06*, LNCS **3860**, Springer-Verlag, pp. 226–243 (2006).

[13] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, vol. **32** (3), pp. 586–615 (2003).

[14] D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, vol. **17** (4), pp. 297–319 (2004).

[15] D. Boneh, E. Shen and B. Waters. Strongly unforgeable signatures based on Computational Diffie-Hellman. *Proceedings of PKC'06*, LNCS **3958**, Springer-Verlag, pp. 229–240 (2006).

[16] C. Boyd. Digital Multisignatures. *Cryptography and Coding, Oxford University Press*, pp. 241–246 (1989).

[17] E. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **9**, pp. 105–113 (1989).

[18] C. Cachin. On-line secret sharing. *Proceedings of IMA Conference'95*, LNCS **1025**, Springer-Verlag, pp. 190–198 (1995).

[19] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Adaptive security for threshold cryptosystems. *Proceedings of Crypto'99*, LNCS **1666**, Springer-Verlag, pp. 98–115 (1999).

[20] R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. *Proceedings of STOC'98*, Vol. **30**, pp. 209–218 (1998).

[21] R. Canetti, S. Halevi and J. Katz. Chosen-ciphertext security from identity-based encryption. *Proceedings of Eurocrypt'04*, LNCS **3027**, Springer-Verlag, pp. 207–222 (2004).

[22] R. Canetti, H. Krawczyk and J.B. Nielsen. Relaxing chosen-ciphertext security. *Proceedings of Crypto'03*, LNCS **2729**, Springer-Verlag, pp. 565–582 (2003).

[23] D. Chaum. Designated Confirmer Signatures. *Proceedings of EUROCRYPT'94*, LNCS **435**, Springer-Verlag, pp. 86–91 (1994).

[24] D. Chaum and H. van Antwerpen. Undeniable Signatures. *Proceedings of Crypto'89*, LNCS **435**, Springer-Verlag, pp. 212–216 (1989).

[25] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. *Proceedings of Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Vol. **26**, pp. 383–395 (1985).

[26] J.S. Coron. On the exact security of Full Domain Hash. *Proceedings of Crypto'00*, LNCS **1880**, Springer-Verlag, pp. 229–235 (2000).

[27] L. Csirmaz. How to share secrets simultaneously. *Cryptology ePrint Archive*, available at http://eprint.iacr.org/2011/386 (2011).

[28] A. W. Dent and Y. Zheng. Practical Signcryption. *A volume in Information Security and Cryptography*, ISBN **978-3-540-89409-4**, Springer-Verlag, pages 274 (2010).

[29] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. *Proceedings of Crypto'89*, LNCS **435**, Springer-Verlag, pp. 307–315 (1989).

[30] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions of Information Theory*, Vol. **22 (6)**, pp. 644–654 (1976).

[31] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. *Proceedings of FOCS'87*, vol. **28**, pp. 427–437 (1987).

[32] M. Franklin and D. Malkhi. Auditable Metering with Ligthweigth Security. *Proceedings of FC'97*, LNCS **1318**, Springer-Verlag, pp. 151–160 (1997).

[33] E. Fujisaki and T. Okamoto. A Practical and Provably Secure Scheme for Publicy Verifiable Secret Sharing and Its Applications, *Proceedings of EUROCRYPT'98*, LNCS **1403**, Springer-Verlag, pp. 32–46 (1998).

[34] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *Proceedings of PKC'99*, LNCS **1560**, Springer-Verlag, pp. 53–68 (1999).

[35] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust threshold DSS signatures. *Proceedings of Eurocrypt'96*, LNCS **1070**, Springer-Verlag, pp. 354–371 (1996).

[36] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for Discrete-Log based cryptosystems. *Journal of Cryptology*, vol. **20** (1), pp. 51–83 (2007).

[37] S. Goldwasser, S. Micali, R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. of Computing* **17 (2)**, pp. 281–308 (1988).

[38] V. Goyal, O. Pandey, A. Sahai and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. *Proceedings of ACM CCS'06*, ACM Press, pp. 89–98 (2006).

[39] J. He and E. Dawson. Multisecret-sharing scheme based on one-way function. *Electronic Letters*, vol. **31** (2), pp. 93–95 (1995).

[40] J. Herranz, F. Laguillaumie, B. Libert and C. Ràfols. Short attribute-based signatures for threshold predicates. *Proceedings of CT-RSA'12*, LNCS **7178**, Springer-Verlag, pp. 51–67 (2012).

[41] J. Herranz, A. Ruiz and G. Sáez. Fully Secure Threshold Unsigncryption. *Proceedings of ProvSec'10*, LNCS **6402**, Springer-Verlag, pp. 261–278 (2010).

[42] J. Herranz, A. Ruiz, G. Sáez. Máxima Seguridad para Firmas Digitales con Verificación Distribuida. *Actas de la XI Reunión Española de Criptología y Seguridad de la Información*, pp. 97–103 (2010).

[43] J. Herranz, A. Ruiz, G. Sáez. Firmas Digitales con Verificación Distribuida en el Modelo de Seguridad Estándar. *Actas de la XII Reunión Española de Criptología y Seguridad de la Información*, available at http:// recsi2012.mondragon.edu/es/programa/recsi2012_submission_07.pdf (2012).

[44] J. Herranz, A. Ruiz and G. Sáez. Signcryption Schemes with Threshold Unsigncryption, and Applications. *Designs, Codes and Cryptography*, to appear, available at http://link.springer.com/article/10.1007/s10623-012-9688-0 (2013).

[45] J. Herranz, A. Ruiz and G. Sáez. New Results and Applications for Multi-Secret Sharing Schemes. *Designs, Codes and Cryptography*, to appear, available at http://link.springer.com/article/10.1007/s10623-013-9831-6 (2013).

[46] J. Herranz, A. Ruiz and G. Sáez. Sharing Many Secrets with Computational Provable Security. *Information Processing Letters*, vol. **113**, pp. 572–579 (2013).

[47] J. Herranz and G. Sáez. Verifiable secret sharing for general access structures, with application to fully distributed proxy signatures. *Proceedings of Financial Cryptography'03*, LNCS **2742**, Springer-Verlag, pp. 286–302 (2003).

[48] W.A. Jackson, K.M. Martin and C.M. O'Keefe. A Construction for Multisecret Threshold Schemes. *Design, Codes and Cryptography*, vol. **9** (3), pp. 287–303 (1996).

[49] M. Jakobsson, K. Sako and R. Impagliazzo. Designated Verifier Proofs and their Applications. *Proceedings of EUROCRYPT'96*, LNCS **1070**, Springer-Verlag, pp. 143–154 (1996).

[50] S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: introducing concurrency, removing erasures. *Proceedings of Eurocrypt'00*, LNCS **1807**, Springer-Verlag, pp. 221–242 (2000).

[51] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, vol. **16** (4), pp. 239–247 (2003).

[52] J. Katz and M. Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology*, vol. **19** (1), pp. 67–95 (2006).

[53] S.J. Kim, S.J. Park and D.H. Won. Zero-knowledge nominative signatures. *Proceedings of PragoCrypt'96*, International Conference on the Theory and Applications of Cryptology, pp. 380–392 (1996).

[54] J.H. Koo, H.J. Kim, I.R Jeong, D.H. Lee and J.I Lim. Jointly unsigncryptable signcryption schemes. *Proceedings of WISA'01*, vol. 2, pp. 397–407 (2001).

[55] H. Krawczyk. Secret sharing made short. *Proceedings of Crypto'93*, LNCS **773**, Springer-Verlag, pp. 136–146 (1993).

[56] H. Krawczyk and T. Rabin. Chameleon Signatures. *Proceedings of NDSS'00*, The Internet Society, pp. 143–154 (2000).

[57] F. Laguillaumie, D. Vergnaud. Multi-Designated Verifiers Signatures. *Proceedings of ICICS'04*, LNCS **3269**, Springer-Verlag, pp. 495–507 (2004).

[58] A. Lewko, T. Okamoto, A. Sahai, K. Takashima and B. Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. *Proceedings of EUROCRYPT'10*, LNCS **6110**, Springer-Verlag, pp. 62–91 (2010).

[59] F. Li, J. Gao and Y. Hu. ID-based threshold unsigncryption scheme from pairings. *Proceedings of CISC'05*, LNCS **3822**, Springer-Verlag, pp. 242–253 (2005).

[60] F. Li, X. Xin and Y. Hu. ID-based signcryption scheme with (t, n) shared unsigncryption. *International Journal of Network Security*, ovl. **3** (2), pp. 155–159 (2006).

[61] C.H. Lim, P.J. Lee. Modified Maurer-Yacobi's Scheme and its Applications. *Proceedings of AUSCRYPT'92*, LNCS **718**, Springer-Verlag, pp. 308–323 (1992).

[62] H.Y. Lin and Y.S. Yeh. Dynamic multi-secret sharing scheme. *Int. J. Contemp. Math. Sciences*, vol. **3** (1), pp. 37–42 (2008).

[63] A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: from cryptosystems to signature schemes. *Proceedings of Asiacrypt'01*, LNCS **2248**, Springer-Verlag, pp. 331–350 (2001).

[64] C. Ma, K. Chen, D. Zheng and S. Liu. Efficient and proactive threshold signcryption. *Proceedings of ISC'05*, LNCS **3650**, Springer-Verlag, pp. 233–243 (2005).

[65] H.K. Maji, M. Prabhakaran and M. Rosulek. Attribute-based signatures. *Proceedings of CT-RSA'11*, LNCS **6558**, Springer-Verlag, pp. 376–392 (2011).

[66] K.M. Martin, R. Safavi-Naini, H. Wang and P.R. Wild. Distributing the encryption and decryption of a block cipher. *Designs, Codes and Cryptography*, vol. **36**, pp. 263-287 (2005).

[67] B. Masucci. Sharing multiple secrets: models, schemes and analysis. *Designs, Codes and Cryptography*, vol. **39**, pp. 89–111 (2006).

[68] P. Mohassel. One-time signatures and chameleon hash functions. *Proceedings of SAC'10*, LNCS **6544**, Springer-Verlag, pp. 302–319 (2011).

[69] M. Naor and B. Pinkas. Secure and efficient metering. *Proceedings of EUROCRYPT'98*, LNCS **1403**, Springer-Verlag, pp. 576–590 (1998).

[70] M. Naor, B. Pinkas and O. Reingold. Distributed pseudo-random functions and KDCs. *Proceedings of EUROCRYPT'99*, LNCS **1592**, Springer-Verlag, pp. 327–346 (1999).

[71] J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: the non-comitting encryption case. *Proceedings of CRYPTO'02*, LNCS **2442**, Springer-Verlag, pp. 111–126 (2002).

[72] T. Okamoto and D. Pointcheval. The Gap-problems: a new class of problems for the security of cryptographic schemes. *Proceedings of PKC'01*, LNCS **1992**, Springer-Verlag, pp. 104–118 (2001).

[73] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Proceedings of EUROCRYPT'99*, LNCS **1952**, Springer-Verlag, pp. 223–238 (1999).

[74] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Proceedings of CRYPTO'91*, LNCS **576**, Springer-Verlag, pp. 129-140 (1991).

[75] A. Perrig. The BiBa one-time signature and broadcast authentication protocol. *Proceedings of CCS '01*, Vol. **8**, pp. 28–37 (2001).

[76] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, vol. **13** (3), pp. 361–396 (2000).

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

[78] A. Ruiz and J.L. Villar. Publicly Verifiable Secret Sharing from Paillier's Cryptosystem. *Proceedings of WEWoRC'05*, LNI **74**, Editorial GI, pp. 98–108 (2005).

[79] A. Sahai and B. Waters. Fuzzy identity-based encryption. *Proceedings of EUROCRYPT'05*, LNCS **3494**, Springer-Verlag, pp. 457–473 (2005).

[80] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, vol. **4**, Springer-Verlag, pp. 161–174 (1991).

[81] B. Schoenmakers. A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. *Proceedings of CRYPTO'99*, LNCS **1666**, Springer-Verlag, pp. 148–164 (1999).

[82] A. Shamir. How to share a secret. *Communications of the ACM*, vol. **22**, pp. 612–613 (1979).

[83] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, vol. **27** (2), pp. 379–423 and 623–656 (1948).

[84] C.E. Shannon. Communication Theory of Secrecy Systems. *The Bell System Technical Journal*, vol. **28** (4), pp. 656–715 (1949).

[85] S. Sharmila Deva Selvi, S. Sree Vivek, S. Priti and C. Pandu Rangan. On the security of identity based threshold unsigncryption schemes. *Proceedings of APWCS'2010*, available at http://eprint.iacr.org/2010/360 (2010).

[86] V. Shoup. Lower bounds for discrete logarithms and related problems. *Proceedings of Eurocrypt'97*, LNCS **1233**, Springer-Verlag, pp. 256–266 (1997).

[87] V. Shoup. Practical threshold signatures. *Proceedings of Eurocrypt'00*, LNCS **1807**, Springer-Verlag, pp. 207–220 (2000).

[88] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, vol. **15** (2), pp. 75–96 (2002).

[89] M. Stadler. Publicly Verifiable Secret Sharing. *Proceedings of Eurocrypt'96*, LNCS **1070**, Springer-Verlag, pp. 190–199 (1996).

[90] B. Yang, Y. Yu, F. Li and Y. Sun. Provably secure identity-based threshold unsigncryption scheme. *Proceedings of ATC'07*, LNCS **4610**, Springer-Verlag, pp. 114–122 (2007).

[91] G.M. Zaverucha and D.R. Stinson. Anonymity in shared symmetric key primitives. *Designs, Codes and Cryptography*, vol. **57** (2), pp. 139–160 (2010).

[92] Z. Zhang, C. Mian and Q. Jin. Signcryption scheme with threshold shared unsigncryption preventing malicious receivers. *Proceedings of TENCON'02*, IEEE Computer Society, vol. **2**, pp. 196–199 (2002).

[93] Y. Zheng. Digital signcryption or How to achieve cost(signature & encryption) << cost(signature) + cost(encryption). *Proceedings of Crypto'97*, LNCS **1294**, Springer-Verlag, pp. 165–179 (1997).

Index