UNIVERSITÀ
DI TRENTO
Department of Mathematics
Laboratory of Industrial Mathematics and Cryptography

Doctoral programme — XXXVI cycle

Algebraic Construction for Multi-Party Protocols with Focus on Threshold Signatures

Michele Battagliola

PhD thesis in Mathematics

Supervised by Nadir Murru

Board:

Ranise Silvio, Prof	Università di Trento
Bazzanella Danilo, Prof	Politecnico di Torino
Andrea Visconti, Prof	Università degli Studi di Milano Statale

Abstract

Secure multi-party computation (MPC) is a field of cryptography that aims to provide methods for parties to jointly compute a function over their inputs while keeping those inputs private. Unlike of traditional cryptography where adversary is outside the system of participants, the main task (and challenge) of MPC is to protect participants from internal adversaries, who participate in protocol and can therefore send corrupted.

The results presented in this thesis are three-fold. First, we study MPC from a theoretical standpoint, designing a new heuristic and a new proof system useful for proving the security of threshold signatures, a particular kind of MPC protocol. Next, we present new MPC primitives: a novel secret sharing scheme, a threshold version of Schnorr signature, a post quantum secure group action based threshold signature and finally a post quantum oblivious transfer. Lastly, we designed a coercion resistant e-voting protocol, that allows voters to freely votes without being afraid of external adversaries trying to pressure them to vote in a particular way.

Contents

Chapter 1

Introduction

1.1 Multi-Party Computation

Secure multi-party computation (MPC) is a branch of modern cryptography that focuses on the design of interactive protocols that allow a set of parties, each with its own input, to jointly compute a function over their inputs without revealing any information about them.

MPC was first informally theorised in 1979 by A. Shamir, R. Rivest and L. Adleman in their work on mental poker [SRA81]. A few years later, A. Yao formally introduced the theory of secure two-party computation for the Millionaires' Problem and then presented a generalisation to any feasible computation in [Yao82; Yao86]. The case where the involved parties are more than two was firstly considered by O. Goldreich, S. Micali and A. Wigderson in 1987 [GMW87].

The recent development of new technologies, such as cloud computing, mobile computing and the Internet of Things, coupled with the advent of more efficient infrastructures (in terms of bandwidth and latency) and more powerful devices, has sparked considerable interest in MPC cryptosystems. As a result, increasingly efficient MPC protocols have been proposed since the late 2000s, and MPC can now be considered as a viable solution for several real-world scenarios, such as privacy-preserving data analysis [Bog+ j; PBS12; Bog+16; Bog13; MKO16; KKB18; Tso+17], cloud computing [Ala+18], genomic sequence comparison [ZH17; BB17], distributed voting [JCJ10; ABR23], private bidding and auctions [Cho+12], distributed signatures [CKM23] or decryption functions and private information retrieval [Cho+95].

Given the growing interest in the subject, at the time of writing the literature on

it is quite extensive and an accurate taxonomy of existing protocols seems difficult. Nevertheless, it is possible to distinguish two main classes of MPC protocols: Boolean and Arithmetic MPC. In the cryptographic schemes of the former class, the function to be evaluated is represented by a Boolean circuit, while the protocols of the latter class are based on a variety of algebraic tools, such as homomorphic encryption or secret sharing techniques. In this work, we focus on arithmetic MPC, with particular attention to threshold signature schemes.

1.2 Threshold Cryptography

In many real-life situations, we cannot be sure whether a single entity (e.g. a single server) is trustworthy, but we can be reasonably sure that the majority of them are honest. Therefore, it may be a good idea to distribute computations among many parties, so that an adversary would have to corrupt many of them in order to successfully perform an attack.

The idea of threshold cryptography is to distribute sensitive cryptographic operations, such as decryption or signatures. In this setting, the secret key is split among a group of cooperating parties in such a way that a minimum number of them, the *threshold*, is required (and sufficient) to perform the operation successfully.

1.2.1 Threshold Signatures

A (t, n)-threshold digital signature scheme is a protocol designed to distribute the right to sign messages to any subset of at least t out of n key owners[1]. In order to streamline the introduction of the new protocol into the cryptographic landscape, threshold schemes usually translate a well-established signature scheme in a multi-party setting, producing signatures that are compatible with the centralized version. For this reason, much effort is devoted to finding efficient and secure algorithms for threshold versions of the most well-known digital signature schemes, namely ECDSA, EdDSA and Schnorr.

In 1996, a first $(T + 1, 2T + 1)$-threshold digital signature scheme was proposed [Gen+96]. A few years later, the same authors discuss the security of distributed key generation for the case of schemes based on the Discrete Logarithm Problem [Gen+07a; Gen+07b]. Since 2001, several authors started working first

[1]During this thesis we use "threshold t" to denote the minimum number of parties required to perform the operation. Notice that in literature this term sometimes refers to the maximum number of parties that are not allowed to perform the task.

on two-party variants of digital signatures [MR04] and then on ECDSA [Doe+18; Lin17]. The first general (T, N)-threshold scheme was proposed in 2016 [GGN16], improved first in 2017 [BGG17], and then again in 2018 [GG18]. In 2019, the work of [Doe+18] has been generalized by the same authors to the multi-party case [Doe+19]. Recently, the first adaptively secure[2] Schnorr signature protocol was proposed by Crites, Komlo, and Maller in [CKM23]. In Chapter 4 we show a threshold version of ECDSA and EdDSA with a particular key distribution algorithm that allows for offline participants, i.e. participant who do not participate in the protocol actively and instead delegate the generation of their secret share to the other parties.

Recently, driven by both the NIST call for Post-Quantum Standardization [NIS17] and the call for Multi-Party Threshold Schemes [BP], many researchers have started to wonder whether it could be possible to design post-quantum versions of threshold digital signature schemes. Since most of the existing literature for threshold schemes focuses on trapdoors that rely on the difficulty of the Discrete Logarithm Problem, new methods have to be investigated, likely starting with tools already utilized to design plain signatures, such as lattices, codes, multivariate equations etc. In [CS19], the (round 2) proposals of the standardization process were analyzed in order to determine ways to define threshold variants, eventually identifying multivariate schemes as the most suitable starting point, with UOV-based schemes being the most promising. Even though, from a theoretical point of view, it appears to be indeed possible to obtain a threshold version of UOV by exploiting LSSS-based MPC protocols, this approach remains, at the present time, only theoretical.

Notably, threshold signature schemes for cryptographic cyclic group actions have been already discussed in 2020 and applied to isogeny-based schemes [DFM20], where they proposed a way to apply a group actions in a threshold like way by using the classical Shamir Secret sharing on a group action induced by a cyclic group. They showed how to apply this for an El Gamal like encryption schemes and a signature based on Σ-protocols proving their simulatability, however this schemes are only secure in the honest-but-curious model and miss a distributed key generation mechanisms. In [CS20b] they showed a way to combine the use of zero-knowledge proofs and replicated secret sharing to obtain a secure threshold signature scheme from isogeny assumptions. The work is an important step for the research and can be extended to more general group actions, but the main drawbacks are the number of shares necessary to implement replicated secret sharing and the important slow down caused by the additional ZKPs required. In

[2] In this context, adaptively secure means that the adversary can decide which parties corrupt during the protocol and it is not asked to corrupt them beforehand.

[Beu+21] they showed how to define a distributed key generation algorithm by using a new primitive called *piecewise verifiable proofs*; proving their security in the quantum random oracle model. All previous techniques are then incorporated in [CM22] to have actively secure attributed based encryption and signature schemes, in which threshold signature are a particular case. In Chapter 5 we show a group action based threshold signature scheme which make no additional assumption over the group action, using it in black box way. This will allow our frameworks to be instantiated with a wider variety of candidates, such as code-based signature schemes.

1.3 Organization

In Chapter 2 we provide the mathematical background and the notation used in this work. Only the more general definitions are provided here, while the more technical ones are provided in the introductory section at the beginning of the relevant chapter.

Chapters 3 through 6 contain the core of the thesis. In Chapter 3 we describe a generalisation of the Fiat-Shamir transform in the multi-party setting. We introduce the concept of distributed identification protocol and we analyze its relations with threshold signatures.

In Chapter 4 we describe a new decentralised secret sharing scheme based on the classical Shamir scheme. We then show a threshold Schnorr signature, that use it for the key generation protocol, which is proved secure using the aforementioned generalised Fiat-Shamir transform.

In Chapter 5 we study group action cryptography, one of the most promising research areas for post-quantum cryptography. In particular, we show a threshold digital signature that makes black-box use of the group action and an oblivious transfer for commutative group actions.

In Chapter 6 we focus our attention on MPC protocols for e-voting. We describe one of the most secure and fast e-voting protocols currently proposed.

Finally, in Chapter 7 we draw conclusions from this work, summarizing the most relevant results and suggesting further research area arising from it.

We also included Appendix A, where we described a side project carried out during the PhD about mutidimensional continued fraction.

Chapter 2

Preliminaries

The purpose of this chapter is to provide the reader with the basic definitions and notation necessary to understand the rest of the thesis. Later on, each chapter have its own preliminaries, containing the specific information relevant to that chapter. In this way, with the exception of Chapter 4, each chapter is self-contained and can be read and understood without needing reference to the previous ones.

2.1 Notation and convention

In the following we present the notation used in the thesis. Individual chapters may have additional notation convention that are specified when needed.

2.1.1 Font

We use bold notation (`\mathbb`) to denote sets and algebraic structures.
We use calligraphic notation (`\mathcal`) to denote *named entities* in definitions and proofs, such as the forger $\mathcal{F}$ or the simulator $\mathcal{S}$.
We use serif font (`\mathsf`) to denote algorithms. Notice that the aforementioned simulator and forger are algorithms as well, but they have a different notation for the sake of clarity.
We use typewriter typestyle (`\mathtt`) to denote named cryptographic variables output of cryptographic protocols, such as the secret key `sk` and a commitment `cmt`.

2.1.2 Assignment, declaration and equality

If $\mathbb{S}$ is a set, $s \xleftarrow{\$} \mathbb{S}$ means that s is sampled uniformly at random from $\mathbb{S}$.

$y \leftarrow \mathsf{A}(x_1, x_2, \dots)$ is a deterministic algorithm taking in input the values $x_1, x_2, \dots$ and returning the value y. If $\mathsf{A}(x_1, x_2, \dots)$ is a probabilistic algorithm then we can use two notations for assigning to a variable y the output of $\mathsf{A}(x_1, x_2, \dots)$:

- $y \xleftarrow{\$} \mathsf{A}(x_1, x_2, \dots)$ where the symbol $\xleftarrow{\$}$ emphasizes the probabilistic nature of the algorithm $\mathsf{A}(x_1, x_2, \dots)$;
- $y \leftarrow \mathsf{A}(x_1, x_2, \dots; R)$ when the randomness R is set and thus the output y is uniquely determined.

When writing pseudocode we reserve the equality symbol = for the boolean operation, while we use := to denote the declaration of a variable (i.e. $y := \mathsf{Enc}(x, \mathtt{pk})$).

2.1.3 Miscellanea

We omit to explicitly write the randomness used or the input of a protocol when not strictly required. We indicate the concatenation of strings $x_1, x_2 \dots, x_n$ as $x_1||x_2|| \dots ||x_n$. We also assume that, given a context, any string x can be uniquely parsed as a the concatenation of substrings.

For the sake of readability, when having an index set $J \subseteq \{1, ..., n\}$ we write $\{a_i\}_J$ in place of $\{a_i\}_{i \in J}$.

In the following when we say that an algorithm is *efficient* we mean that it runs in (expected) polynomial time in the size of the input, possibly using a random source.

We say that a publicly-accessible algorithm is an oracle if parties are only given "black-box" access to it. For this reason, while still being an algorithm, it can be see as an "additional entity" in the proof, thus we always use $\mathcal{O}$ to denote oracles.

2.2 Mathematical Background

Definition 2.1. Let $f : \mathbb{N} \to \mathbb{R}$ be a function. We say that f is *negligible in* λ if for every $c \in \mathbb{N}$ there exists $m \in \mathbb{N}$ such that

$$|f(\lambda)| < \frac{1}{\lambda^c} \quad \forall \lambda > m.$$

Informally, we often state that an event happens with *overwhelming probability in* λ. That means that it happens with probability $1 - f(\lambda)$ where $f(\lambda)$ is a negligible function in λ.

Definition 2.2. Let $f : \mathbb{N} \to \mathbb{R}$. We say that f is *super-logarithmic* if and only if $f = \omega(\log n)$.

In the same way of Definition 2.2, we can define also super-polynomial functions.

A useful way to analyze the output distribution of a probabilistic algorithm A is to consider the min-entropy, which provides an upper bound to the probability that $\mathsf{A}(\dot{})$ generates a specific output in the output space. More formally we have:

Definition 2.3 (Min-entropy)**.** Let $\mathsf{A} : \mathbb{X} \to \mathbb{S}$ be a probabilistic algorithm. Let

$$\alpha(x) = \max_{s \in \mathbb{S}} \{\Pr[\mathsf{A}(x; R) = s | R \xleftarrow{\$} \mathsf{Coins}(\lambda)]\}$$

be the probability that A, executed on input x, outputs the most likely output s. The *min-entropy* function associated to A is defined as

$$\beta(\lambda) = \min_x \left\{ \log_2 \frac{1}{\alpha(x)} \right\}.$$

Usually we require algorithms having a uniformly distributed output in the output space $\mathbb{S}$. In this case $\alpha(x) = \frac{1}{|\mathbb{S}|}$ and the min-entropy $\beta(\lambda) = \log_2 |\mathbb{S}|$.

Finally, we often say that data sampled from one distribution D_1 are *indistinguishable* from data sampled from a second one D_2. Informally speaking that means that it is impossible to decide whether the original distribution is D_1 or D_2. Formally we have the following definition:

Definition 2.4 (Indistinguishability)**.** Let D_1 and D_2 be two probability distributions. We say that D_1 and D_2 are indistinguishable if, for any algorithm A, there exists a negligible function ν such that for all $n \in \mathbb{N}$

$$\Pr[\mathsf{A}(D_1(n)) = 1] - \Pr[\mathsf{A}(D_2(n)) = 1] \leq \nu(n).$$

The algorithm A in Definition 2.4 is often called *distinguisher*.

An immediate consequence of this definition, is that no algorithm behaves more than negligibly differently when given inputs sampled according to D_1 or D_2.

2.2.1 Basic Cryptography

In this section we provide some basic cryptography definitions, that are used during the whole paper.

Key Generation Algorithms Informally speaking, the goal of a key generation is to provide a secret-public key pair in a secure way, such that it is hard to recover the secret key key knowing only the public one. Formally:

Definition 2.5. Let KeyGen a key generation algorithm for a relation $\mathbb{L} \subseteq \mathbb{W} \times \mathbb{Y}$. Define the following experiment

$$\begin{array}{l} \mathsf{Exp}^{\mathsf{ow}}_{\mathcal{A}}(\lambda): \\ \hline 1: \quad (\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\lambda) \\ 2: \quad \mathtt{sk}' \xleftarrow{\$} \mathcal{A}(\mathtt{pk}) \\ 3: \quad \textbf{return } (\mathtt{sk}', \mathtt{pk}) \in \mathbb{L} \end{array}$$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{ow}}_{\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{ow}}_{\mathcal{A}}(\lambda) = 1)$$

We say that KeyGen is one-way if and only if $\mathbf{Adv}^{\mathsf{ow}}_{\mathcal{A}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

From now on, when speaking about key generation algorithms, we always implicitly consider algorithms for which the one-way property holds.

Encryption Schemes There are two families of encryption schemes: *symmetric cryptography*, where the sender and the receiver share a common secret key, and *asymmetric* or *public-key cryptography*, where the receiver holds a secret-public key pair while the sender only knows the public key. In this thesis we only use public-key cryptography. Formally we have:

Definition 2.6 (Public-Key Encryption)**.** A *public-key encryption scheme* is defined by the tuple

$$(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$$

where:

- $\mathsf{Setup}(\lambda)$, on input a security parameters λ, it outputs public parameters $\mathtt{pp}$.
- $\mathsf{KeyGen}(\mathtt{pp}, \lambda)$ is a probabilistic *key generation algorithm* that takes as input the public parameters $\mathtt{pp}$, the security parameter λ and outputs a public key $\mathtt{pk}$ and the corresponding secret key $\mathtt{sk}$
- $\mathsf{Enc}(\mathtt{pk}, \mathtt{m}; R)$ is a probabilistic algorithm called *encryption algorithm* that takes as input a public key $\mathtt{pk}$ and a message $\mathtt{m}$, called plaintext, and outputs an encryption $\mathtt{ct}$, called ciphertext;
- $\mathsf{Dec}(\mathtt{sk}, \mathtt{ct})$ is a deterministic algorithm, called *decription algorithm*, which takes as input a secret key and a ciphertext $\mathtt{ct}$ and outputs a message $\mathtt{m}$ or failure $\perp$.

We ask that $\Pr(\mathsf{Dec}(\mathtt{sk}, \mathtt{ct}) = \mathtt{m} | \mathsf{Enc}(\mathtt{pk}, \mathtt{m}; R) = \mathtt{ct}) = 1$.

With abuse of notation, in this thesis we refer to an encryption scheme as simply the couple $(\mathsf{Enc}, \mathsf{Dec})$.

The main security definition we ask for encryption schemes is indistinguishability under chosen plaintext attacks (IND-CPA). Informally, an encryption scheme is IND-CPA secure if it is impossible to distinguish encryption of different messages having access only to the public key. Formally:

Definition 2.7 (indistinguishability under chosen plaintext attacks)**.** Consider an encryption scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Consider the following experiment:

$\mathsf{Exp}^{\mathsf{ind-cpa}}_{\mathsf{Enc},\mathcal{A}}(\lambda)$

1 : $\mathtt{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\lambda)$
3 : $b \xleftarrow{\$} \{0, 1\}$
4 : $(\mathtt{m}_0, \mathtt{m}_1) \leftarrow \mathcal{A}(\lambda, \mathtt{pk})$
5 : $\mathtt{ct} \xleftarrow{\$} \mathsf{Enc}(\mathtt{pk}, \mathtt{m}_b)$
6 : $b' \leftarrow \mathbf{Adv}(\mathtt{pk}, \mathtt{ct}, \mathtt{m}_0, \mathtt{m}_1)$
7 : **return** $b = b$'

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{Enc},\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{ind-cpa}}_{\mathsf{Enc},\mathcal{A}}(\lambda) = 1)$$

We say that $(\mathsf{Enc}, \mathsf{Dec})$ is indistinguishable under chosen plaintext attacks if and only if $\mathbf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{Enc},\mathcal{A}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

Hash Functions Roughly speaking, a (cryptographic) hash function is a function that takes as input an arbitrary long string and outputs string with a fixed length that is hard to invert. More formally we have

Definition 2.8 (Hash Functions)**.** A *cryptographic hash function* with output length l is a function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^l$ satisfying the following properties:

- **Preimage Resistant** Given a digest `digest` it is hard to find `m` such that $\mathsf{H}(\mathtt{m}) = \mathtt{digest}$.
- **Second Preimage Resistant** Given a message `m` it is hard to find $\mathtt{m}' \neq \mathtt{m}$ such that $\mathsf{H}(\mathtt{m}') = \mathsf{H}(\mathtt{m})$.
- **Collision Resistant** It is hard to find $\mathtt{m}_0, \mathtt{m}_1$ with $\mathtt{m}_0 \neq \mathtt{m}_1$ such that $\mathsf{H}(\mathtt{m}_0) = \mathsf{H}(\mathtt{m}_1)$.

Commitment Schemes *Commitment schemes* are a cryptographic primitive that allows a sender to commit to a chosen value while keeping it hidden to others (sending the so called commitment). Later, by sending the decommitment, the sender is able to reveal (open) the commited value to the receiver. Commitment schemes are designed so that a party cannot change the value inside the commitment: for this reason they are usually used to achieve synchronous communications over an asynchronous channel.

Definition 2.9 (Commitment Schemes)**.** A *commitment scheme* is defined by the tuple $(\mathsf{Com}, \mathsf{Decom})$ where:

- $\mathsf{Com}(\mathtt{m}; R)$ is a probabilistic algorithm called *commitment algorithm* that takes as input a public key a message `m` and outputs a commitment, decommitment pair $\mathtt{cmt}, \mathtt{dcmt}$ to it.
- $\mathsf{Decom}(\mathtt{cmt}, \mathtt{dcmt})$ is a deterministic algorithm, called *decommitment algorithm*, which takes as input a commitment `cmt` and a decommitment `dcmt` and outputs a message `m` or failure $\perp$.

We ask that $\Pr(\mathsf{Decom}(\mathtt{cmt}, \mathtt{dcmt}) = \mathtt{m} | \mathsf{Com}(\mathtt{m}; R) = (\mathtt{cmt}, \mathtt{dcmt})) = 1$.

With abuse of notation we often refer to a commitment scheme $(\mathsf{Com}, \mathsf{Decom})$ as simply Com.

We ask that a commitment scheme is binding and hiding, namely that it is difficult to change a commited value once $\mathtt{cmt}$ is sent and that $\mathtt{cmt}$ do not reveal anything about the original message $\mathtt{m}$. Formally:

Definition 2.10 (Hiding and Binding)**.** Let $(\mathsf{Com}, \mathsf{Decom})$ be an commitment scheme. Consider the following experiment:

$\mathsf{Exp}^{\mathsf{hide}}_{\mathsf{Com},\mathcal{A}}(\lambda)$	$\mathsf{Exp}^{\mathsf{bind}}_{\mathsf{Com},\mathcal{A}}(\lambda)$
1 : $b \xleftarrow{\$} \{0,1\}$	1 : $\mathtt{m} \xleftarrow{\$} \mathcal{A}(\lambda)$
2 : $(\mathtt{m}_0, \mathtt{m}_1) \leftarrow \mathcal{A}(\lambda)$	2 : $\mathtt{cmt} \xleftarrow{\$} \mathsf{Com}(\mathtt{m}; R)$
3 : $\mathtt{cmt} \xleftarrow{\$} \mathsf{Com}(\mathtt{m}_b; R)$	3 : $\mathtt{dcmt} \leftarrow \mathcal{A}(\mathtt{m}, \mathtt{cmt})$
4 : $b' \leftarrow \mathbf{Adv}(\mathtt{cmt}, \mathtt{m}_0, \mathtt{m}_1)$	4 : $\mathtt{m}' \leftarrow \mathsf{Decom}(\mathtt{cmt}, \mathtt{dcmt})$
5 : **return** $b = b'$	5 : **return** $\mathtt{m}' \neq \mathtt{m}$ **and** $\mathtt{m}' \neq \perp$

Define the advantage of $\mathcal{A}$ in the hiding (binding) game as

$$\mathbf{Adv}^{\mathsf{hide}}_{\mathsf{Com},\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{hide}}_{\mathsf{Com},\mathcal{A}}(\lambda) = 1)$$
$$\mathbf{Adv}^{\mathsf{bind}}_{\mathsf{Com},\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{bind}}_{\mathsf{Com},\mathcal{A}}(\lambda) = 1)$$

We say that $(\mathsf{Com}, \mathsf{Decom})$ is hiding (binding) if and only if $\mathbf{Adv}^{\mathsf{hide}}_{\mathsf{Com},\mathcal{A}}(\lambda)$ ($\mathbf{Adv}^{\mathsf{bind}}_{\mathsf{Com},\mathcal{A}}(\lambda)$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

Observation 2.1. When referring to the security of commitment scheme we often say "statistically " hiding (binding) or "perfectly " hiding (binding). The difference is about the computational power of $\mathcal{A}$. When $\mathcal{A}$ is computationally unbounded then the scheme is perfectly secure, otherwise is only statistically secure. Notice that for a commitment scheme is impossible to be both perfectly hiding and perfectly binding.

2.3 Random Oracle Model

Introduced by Bellare and Rogaway in [BR93], the Random Oracle Model (ROM) is an idealised security model useful to analyze and prove the security of many cryptographic protocols.

Roughly speaking, let $\mathbb{S}, \mathbb{X}$ be two sets. A random oracle is a publicly-accessible oracle $\mathcal{O}^{\mathsf{H}} : \mathbb{S} \to \mathbb{X}$ that behaves as a perfectly random function. From a theoretical point of view no efficient algorithm can possibly be used as random oracle, since a truly random function would have a description that is exponentially large. For this reason random oracles are often substituted by hash functions in real world applications.

In particular, a protocol is secure in the random oracle model if it is secure when all the pseudo-random functions (usually hash functions) are replaced by random oracles, which returns truly random values upon invocation.

This model is in opposition with the Standard Model in which the adversary is only limited by the amount of time and computation available. A scheme is secure in the standard model if it can be proven secure only using complexity assumptions.

In this paper we always assume to work in the ROM.

2.4 Forking Lemma

First introduced by Pointcheval and Stern in [PS96], the Forking Lemma is one of the most used lemma in cryptography. In particular, it is the basis for all the proofs that require rewinding an adversary (i.e. halting the computation and restarting it from a previous state, usually changing some input data) in order to prove the security of a protocol.

Let $\mathcal{A}$ be an adversary initialized with a random tape and having access to a random oracle (modeled by an hash function, as explained in Section 2.3). While the behavior of the adversary is generally not defined, the adversary outputs some value that either satisfy some pre-defined conditions (thus winning the security game), or not. If $\mathcal{A}$ completes its attack successfully, the Forking Lemma gives a lower bound for the probability that $\mathcal{A}$ again win the security game in a second execution with the same random tape but with different outputs from the random oracle [Kom]. More formally we have the following lemma, by Bellare and Neven in [BN06]:

Lemma 2.1 (General Forking Lemma)**.** *Let $q \in \mathbb{Z}$ with $q \geq 1$, $\mathbb{H}$ be a set with $|\mathbb{H}| \geq 2$. Let IG be a randomized algorithm called input generator and let A be a randomized algorithm that, on input $x \xleftarrow{\$} \mathsf{IG}, h_1, ..., h_q \in \mathbb{H}$, returns a pair (J, σ) with J being an integer $0 \leq J \leq q$ and σ a side output. The accepting probability p of A, is defined as the probability that $J \geq 1$ in the experiment*

$$x \xleftarrow{\$} \mathsf{IG}; h_1, ..., h_q \xleftarrow{\$} \mathbb{H}; (J, \sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, ..., h_q)$$

The forking algorithm F_A *associated to* A *is the randomized algorithm that takes as input* x *and proceeds as follows:*

$\mathsf{F}_\mathsf{A}(x)$:
  $R \xleftarrow{\$} \{0,1\}^*$
  $h_1, ..., h_q \xleftarrow{\$} \mathbb{H}$
  $(J, \sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, ..., h_q; R)$
  if $J = 0$ **then**
    return $(0, \epsilon, \epsilon)$
  $h'_J, ..., h'_q \xleftarrow{\$} \mathbb{H}$
  $(J', \sigma') \xleftarrow{\$} \mathsf{A}(x, h_1, ..., h_{J-1}, h'_J, ..., h'_q; R)$
  if $(J = J' \wedge h_J \neq h'_J)$ **then**
    return $(1, \sigma, \sigma')$
  else
    return $(0, \epsilon, \epsilon)$

Then we have

$$\Pr[b = 1 | x \xleftarrow{\$} \mathsf{IG}; (b, \sigma, \sigma') \xleftarrow{\$} \mathsf{F}_\mathsf{A}(x)] \geq p\left(\frac{p}{q} - \frac{1}{|\mathbb{H}|}\right).$$

2.5 Sigma Protocols and Identification Schemes

A sigma protocol for a relation $\mathbb{L} \subseteq \mathbb{W} \times \mathbb{Y}$ is a three moves interactive protocol between a prover, holding a witness-statement pair $(w, y) \in \mathbb{L}$, and a verifier, knowing only the statement y. Roughly speaking, sigma protocols work as follows (see Figure 2.1):

(i) In the first step, the prover sends *commitment* $\mathtt{cmt} \in \mathbb{X}$ to the verifier.

(ii) Then verifier returns a *challenge* $\mathtt{ch}$ consisting of a random string of fixed length $c(\lambda)$ which depends on the security parameter λ.

(iii) Lastly, the prover answer with a *response* $\mathtt{rsp}$ that the verifier verifies according to $y, \mathtt{cmt}, \mathtt{ch}$.

At the end of the interaction we want that an honest prove is able to convince the verifier about it, without leaking any extra information. On the other hand, we want that a dishonest prover (not knowing w) is not able to convince the verifier. Formally we have:

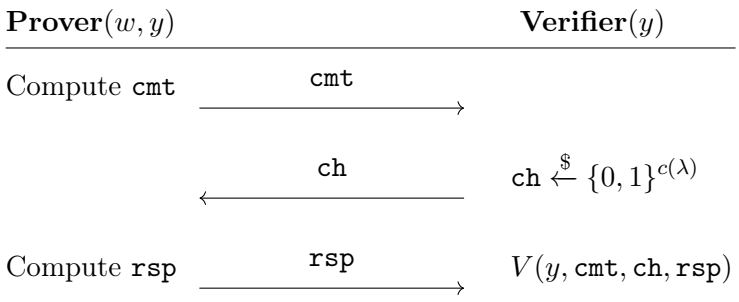


Figure 2.1: Structure of a sigma protocol

Definition 2.11 (Sigma protocol)**.** A three-moves protocol Π as Figure 2.1 between a Prover P and a Verifier V is said to be a sigma protocol for a relation $\mathbb{L}$ if:

- **Completeness:** if P follows the protocol on input (w, y), with $(w, y) \in \mathbb{L}$, the verifier accepts with overwhelming probability.
- **Special Soundness:** there exists an efficient deterministic algorithm $\mathcal{E}$, called extractor, with the following property: whenever $\mathcal{E}$ is given as input $y \in \mathbb{Y}$, two accepting conversations $(\mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$ and $(\mathtt{cmt}, \mathtt{ch}', \mathtt{rsp}')$, with $\mathtt{ch} \neq \mathtt{ch}'$, $\mathcal{E}$ outputs $w \in \mathbb{W}$ such that $(w, y) \in \mathbb{L}$.
- **Honest-Verifier Zero Knowledge:** There exists a polynomial-time algorithm $\mathcal{S}$, called simulator, which on input $y \in \mathbb{Y}$ and a random challenge $\mathtt{ch}$, outputs an accepting conversation $(\mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$, with the same probability distribution as conversations between an honest P and V on input y.

When the relation $\mathbb{L}$ is hard (i.e. given only $y \in \mathbb{Y}$ is hard to compute $w \in \mathbb{W}$ such that $(w, y) \in \mathbb{L}$) we can use sigma protocols to build identification schemes. Informally speaking, we can imagine an identification scheme as a sigma protocol equipped with a secure key generation algorithm for the relation $\mathbb{L}$ that provides in a secure way the couple $(w, y) \in \mathbb{L}$ to the prover. In the context of identification protocols we say that w is the secret key, denoted by $\mathtt{sk}$, while y is the public key, denoted by $\mathtt{pk}$.

Definition 2.12. A *canonical identification protocol* is an interactive protocol between a prover P and a verifier V and is defined by the tuple

$$\mathsf{Id} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{P}_{\mathtt{cmt}}, \mathsf{P}_{\mathtt{rsp}}, \mathsf{V})$$

where:

- $\mathsf{Setup}(\lambda)$, on input a security parameters λ, it outputs public parameters $\mathtt{pp}$.
- $\mathsf{KeyGen}(\mathtt{pp}, \lambda)$ is a probabilistic *key generation algorithm* that takes as input the public parameters $\mathtt{pp}$, the security parameter λ and outputs a public key $\mathtt{pk}$ and the corresponding secret key $\mathtt{sk}$
- $\mathsf{P}_{\mathtt{cmt}}(\mathtt{sk}; R)$ is a probabilistic algorithm called *prover commitment* that takes as input a secret key $\mathtt{sk}$ and outputs a commitment $\mathtt{cmt} \in \mathbb{X}$;
- $\mathsf{P}_{\mathtt{rsp}}(\mathtt{sk}, \mathtt{cmt}, \mathtt{ch}; R)$ is a probabilistic algorithm called *prover response* that takes as input a private key $\mathtt{sk}$, a commitment $\mathtt{cmt}$ and a challenge $\mathtt{ch}$ and outputs a response $\mathtt{rsp}$;
- $\mathsf{V}(\mathtt{pk}, \mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$ is a deterministic algorithm, called Verifier, which takes as input a public key, a commitment $\mathtt{cmt}$, a challenge $\mathtt{ch}$ and a response $\mathtt{rsp}$, and outputs accept or reject.

Moreover we also need an additional property on the commitment:

Definition 2.13 (Non-triviality)**.** A canonical identification scheme is called *non-trivial* if the min-entropy of the commitments is super-logarithmic in the security parameter λ.

Observation 2.2. If the commitment $\mathtt{cmt}$ is picked uniformly at random from $\mathbb{X}$, requiring a super-logarithmic min-entropy is equivalent to requiring a super-polynomial size of $\mathbb{X}$

To an identification scheme Id and a pair $(\mathtt{pk}, \mathtt{sk})$ it is associated a randomized transcript generation oracle $\mathcal{O}_{\mathsf{TrGen}}$ which takes no inputs and returns a random transcript of an honest execution.

An important notion of security for canonical identification schemes is the *security against impersonation under passive attack or (eavesdropping attack)*. In this notion we assume that the impersonator can see a polynomial number of transcripts of the real prover interacting with an honest verifier (in the security game, this is modeled by giving the access to the transcript generation oralce $\mathcal{O}_{\mathsf{TrGen}}$), then it must produce its impersonation attempt. More formally:

Definition 2.14 (Security against impersonation under passive attack)**.** Let Id be a canonical identification scheme and let $\mathcal{I}$ be an impersonator, $\mathtt{st}$ be its state and λ be the security parameter. Define the advantage of $\mathcal{I}$ as:

$$\mathbf{Adv}^{\mathsf{imp-pa}}_{\mathsf{Id},\mathcal{I}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{imp-pa}}_{\mathsf{Id},\mathcal{I}}(\lambda) = 1)$$

where the experiment is:

$\mathsf{Exp}^{\mathsf{imp-pa}}_{\mathsf{Id},\mathcal{I}}(\lambda)$:

1 : $\mathtt{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$

2 : $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{pp})$

3 : $\mathtt{st}||\mathtt{cmt} \xleftarrow{\$} \mathcal{I}^{\mathcal{O}_{\mathsf{TrGen}}}(\mathtt{pk})$

4 : $\mathtt{ch} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$

5 : $\mathtt{rsp} \xleftarrow{\$} \mathcal{I}(\mathtt{st}, \mathtt{ch})$

6 : **return** $\mathsf{V}(\mathtt{pk}, \mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$

We say that Id is polynomially secure against impersonations under passive attack if $\mathbf{Adv}^{\mathsf{imp-pa}}_{\mathsf{Id},\mathcal{I}}(\lambda)$ is negligible for every probabilistic polynomial time impersonator $\mathcal{I}$.

A standard way to prove a canonical identification scheme secure against impersonation under passive attacks is to show that $(\mathsf{P}_{\mathtt{cmt}}, \mathsf{P}_{\mathtt{rsp}}, \mathsf{V})$ forms a sigma protocol (i.e. proving the completeness, the special soundness and the honest-verifier zero knowledge).

2.6 Digital Signature Schemes

Digital Signature Schemes are cryptographic protocols used to provide *integrity*, *authenticity* and *non-repudiation* to electronic documents.

Definition 2.15. [Digital Signature Scheme] A *digital signature scheme* is defined by the tuple

$$\mathsf{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{V})$$

where:

- $\mathsf{Setup}(\lambda)$, on input a security parameters λ, it outputs public parameters $\mathtt{pp}$.

- $\mathsf{KeyGen}(\mathtt{pp}, \lambda)$ is a probabilistic *key generation algorithm* that takes as input the public parameters $\mathtt{pp}$, the security parameter λ and outputs a public key $\mathtt{pk}$ and the corresponding secret key $\mathtt{sk}$

- $\mathsf{Sign}(\mathtt{sk}, \mathtt{m}; R)$ is a probabilistic algorithm called *signature algorithm* that takes as input a private key $\mathtt{sk}$ and a message $\mathtt{m}$ and outputs a signature σ;

- $\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma)$ is a deterministic algorithm, called verification, which takes as input a public key, the commitment $\mathtt{m}$ and signature σ, and outputs accept or reject.

We ask that $\Pr(\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma) = \mathsf{accept} | \mathsf{Sign}(\mathtt{sk}, \mathtt{m}; R) = \sigma) = 1$.

A relevant notion of security for digital signature schemes is the notion of *unforgeability under chosen message attacks*, where an adversary has as many couple message-signature as it wish and is asked to produce a forgery (i.e. a valid signature without having direct access to the private key). Formally

Definition 2.16 (Existential unforgability under chosen message attack)**.** Let $\mathsf{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{V})$ be a digital signature scheme. Let $\mathcal{F}$ be a forger having access to a signing oracle $\mathcal{O}^{\mathsf{H}}_{\mathsf{DS}}(\cdot)$ and to the random oracle $\mathcal{O}_{\mathsf{H}}(\cdot)$. Define the following experiment where $\mathbb{M}$ represents the set of messages queried by $\mathcal{F}$, to $\mathcal{O}^{\mathsf{H}}_{\mathsf{DS}}(\cdot)$.

$\mathsf{Exp}^{\mathsf{euf-cma}}_{\mathsf{DS},\mathcal{F}}(\lambda)$:

1 : $\mathtt{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{pp})$
3 : $\mathtt{m}||\sigma \xleftarrow{\$} \mathcal{F}^{\mathcal{O}^{\mathsf{H}}_{\mathsf{DS}}(\cdot), \mathcal{O}_{\mathsf{H}}(\cdot)}(\mathtt{pk})$
4 : **if** $\mathtt{m} \in \mathbb{M}$
5 : **return** 0
6 : **else**
7 : **return** $V(\mathtt{pk}, \mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$

Define the advantage of $\mathcal{F}$ as $\mathbf{Adv}^{\mathsf{euf-cma}}_{\mathsf{DS},\mathcal{F}}(\lambda) = \Pr(\exp^{\mathsf{euf-cma}}_{\mathsf{DS},\mathcal{F}}(\lambda) = 1)$. DS is existential unforgeable against chosen message attacks if $\mathbf{Adv}^{\mathsf{euf-cma}}_{\mathsf{DS},\mathcal{F}}(\lambda)$ is negligible for every probabilistic polynomial time forger $\mathcal{F}$.

$\mathsf{Sign}(\mathtt{sk}, \mathtt{m})$

1 : $R \xleftarrow{\$} \mathsf{Coins}(\lambda)$
2 : $\mathtt{cmt} \leftarrow \mathsf{P}_{\mathtt{cmt}}(\mathtt{sk}; R)$
3 : $\mathtt{ch} \leftarrow \mathsf{H}(\mathtt{cmt}||\mathtt{m})$
4 : $\mathtt{rsp} \leftarrow \mathsf{P}_{\mathtt{rsp}}(\mathtt{sk}, \mathtt{cmt}||\mathtt{ch}; R)$
5 : **return** $\mathtt{cmt}||\mathtt{rsp}$

$\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma)$

1 : Parse σ as $\mathtt{cmt}|\mathtt{rsp}$
2 : $\mathtt{ch} \leftarrow \mathsf{H}(\mathtt{cmt}||\mathtt{m})$
3 : **return** $\mathsf{V}(\mathtt{pk}, \mathtt{cmt}||\mathtt{ch}||\mathtt{rsp})$

Figure 2.2: Signature agorithm from Fiat Shamir

The inclusion of the random oracle $\mathcal{O}_{\mathsf{H}}(\cdot)$ is not strictly required and its inclusion depends on whether the digital signature requires the ROM or not. Since this is the case for most signatures with provable security and for reasons that will more be clear in Chapter 3 we included it in Definition 2.16.

2.6.1 Fiat-Shamir Transform

Firstly introduced in [FS87], the Fiat-Shamir Transform is a widespread heuristic, used to design digital signature schemes starting from canonical identification schemes. Intuitively, the idea is to replace the challenge communication step with an hash function, taking as input the message and the commitment. Formally we have the following definition:

Definition 2.17 (**Fiat-Shamir transform**)**.** Let Id be a canonical identification scheme, let c be the challenge length's function and let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{c(k)}$ be a public hash function. The signature scheme DS uses the same setup and key generation algorithm as the identification scheme, while the signing and verification algorithms are the one of Figure 2.2, where, with abuse of notation, we called V both the signature verification and the identification scheme verification.

In [Abd+02] it is proved that, if a non-trivial canonical identification scheme is secure against eavesdropping attack, then the digital signature scheme obtained by applying the Fiat-Shamir Transform is unforgeable under chosen message attacks[1].

[1]In case the min-entropy of the commitments is less than super-logarithmic, it is always possible to consider a modified version of the protocol where $\mathtt{ch} = \mathsf{H}(\mathtt{cmt}||R_{\mathtt{cmt}}||\mathtt{m})$, where $R_{\mathtt{cmt}}$ is a random string of appropriate length, such that $\mathtt{cmt}||R_{\mathtt{cmt}}$ have the desired min-entropy. For more details about this, see [Abd+02].

2.7 Threshold Cryptography

2.7.1 Secret Sharing

Threshold Protocols are methods for distributing a "privilege" among a group of participants, such that a minimum number of them (the threshold) is required to exert it.

This can be done by sharing some values among the multiple parties using a secret sharing schemes.

Informally speaking, a (t, n)- secret sharing scheme allows one *dealer* to share a secret among n participants in a such a way that every set of at least t participants is able to recover the secret, while any other smaller set do not get any information about the secret.

We formalize the security of a secret sharing scheme with the following definition:

Definition 2.18. A (t, n)-secret sharing scheme between a dealer $\mathcal{D}$, holding a secret s and parties $P_1, ..., P_n$, each of them holding a share s_i of s, is (perfectly) secure if and only if

$$\Pr(\mathsf{secret} = s | \{s_i\}_{\mathbb{J}}) = \Pr(\mathsf{secret} = s' | \{s_i\}_{\mathbb{J}})$$

for all $\mathbb{J} \subseteq \{1, ..., n\}$ such that $|J| < t$.

In the following, we implicitly suppose that any secret sharing scheme is secure according to this definition.

A detailed discussion about secret sharing algorithm, their application and the possibility of removing the dealer, obtaining a fully decentralized algorithm, is done later in Chapter 4.

2.7.2 Thresold Signature

In a nutshell, a (t, n)-threshold signature is a multi-party protocol that allows any t parties out of a total of n to compute a signature that may be verified against a common public key.

Definition 2.19 (Threshold Signature Scheme)**.** A *threshold signature scheme* is defined by the tuple

$$\mathsf{TDS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TSign}, \mathsf{V}).$$

where:

- $\mathsf{Setup}(\lambda)$, on input a security parameters λ, it outputs public parameters $\mathtt{pp}$.
- $\mathsf{KeyGen}(\mathtt{pp}, n, t, \lambda)$, on input the number of participants n, the threshold t and the security parameter λ, it outputs a public key $\mathtt{pk}$ and a secret sharing $\mathtt{sk}_i$ of the corresponding secret key $\mathtt{sk}$, having each participant P_i holding $\mathtt{sk}_i$.
- $\mathsf{TSign}(\mathtt{m}, \{\mathtt{sk}_i\}_{\mathbb{J}})$ is a multi party protocol run by parties in $\mathbb{J}$. On input an agreed upon message $\mathtt{m}$ and shards $\mathtt{sk}_i$ from various players, it outputs a valid signature σ if $|\mathbb{J}| \geq t$,
- $\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma)$, on input a public key $\mathtt{pk}$, a message $\mathtt{m}$ and a signature σ, it outputs accept if the signature is valid, reject if not.

We ask that $\Pr(\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma) = \mathsf{accept} | \mathsf{Sign}(\{\mathtt{sk}_i\}_{\mathbb{J}}, \mathtt{m}; R) = \sigma) = 1$ when $|\mathbb{J}| \geq t$.

Informally, after an initial setup, any set of t parties who agree on a common message $\mathtt{m}$ is able to jointly perform TSign to sign it. The resulting signature is verifiable against the public key $\mathtt{pk}$ via the centralized verification algorithm V.

Depending on the nature of the KeyGen algorithm, one can distinguish:

- **Threshold signatures with dealer:** where a trusted dealer is in charge of creating $(\mathtt{sk}, \mathtt{pk})$ and distributing the share $\mathtt{sk}_i$ to each player.
- **Fully decentralized threshold signatures:** where the KeyGen is a decentralized protocol,

This existential unforgeability defined in Definition 2.16 can be extended also for threshold digital signature schemes, making a distinction between *active chosen message attacks* and *passive chosen message attacks*. This distinction is relevant because the creation of the signatures requires the parties executing the algorithm TSign to interact, and the two security notions capture distinct powers of the adversary.

(i) *passive chosen message attacks*: the adversary follows the protocol honestly and do not deviate from it. It can learn all the information stored, sent and received by the corrupted parties. This adversary is often referred as "Honest-but-Curious " or "Semi-Honest " .

(ii) *active chosen message attacks*: this attack model captures an adversary who can take full control over the device of the corrupted parties who might take part in the signing process. This adversary can deviate from an honest protocol execution.

Formally we have the following definitions:

Definition 2.20. Let $\mathsf{TDS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TSign}, \mathsf{V})$ be a (t, n)-threshold digital signature scheme with challenge length c and security parameter λ. Let $\mathcal{F}$ be a forger having access to a signing oracle $\mathcal{O}^{\mathsf{H}}_{\mathtt{view-TSign}}(\cdot)$ and to the random oracle $\mathcal{O}_{\mathsf{H}}(\cdot)$. Define the experiment $\mathsf{Exp}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda)$ as per Figure 2.3. Define the advantage of $\mathcal{F}$ as:

$$\mathrm{Adv}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda) = 1)$$

We say that TDS is *existentially unforgeable under passive chosen message attacks* if $\mathrm{Adv}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda)(\cdot)$ is negligible for every probabilistic polynomial time forger $\mathcal{F}$.

Definition 2.21. Let $\mathsf{TDS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TSign}, \mathsf{V})$ be a (t, n)-threshold digital signature scheme with challenge length c and security parameter λ. Let $\mathcal{F}$ be a forger having access to a signing oracle $\mathcal{O}^{\mathsf{H}}_{\mathsf{TSign}}(\cdot)$ and to the random oracle $\mathcal{O}_{\mathsf{H}}(\cdot)$. Define the experiment $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},F}(\lambda)$ as per Figure 2.3. Define the advantage of $\mathcal{F}$ as:

$$\mathrm{Adv}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda) = 1)$$

We say that TDS is *existentially unforgeable under active chosen message attacks* if $\mathrm{Adv}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda)(\cdot)$ is negligible for every probabilistic polynomial time forger $\mathcal{F}$.

Roughly speaking, $\mathcal{F}$ is able to corrupt at most $t-1$, receiving their private keys. Then it can interact with an hash oracle $\mathcal{O}_{\mathsf{H}}(\cdot)$ and with a signing oracle playing the role of the honest players in $\mathbb{J}_h$, not controlled by $\mathcal{F}$. After a polynomially large number of query, $\mathcal{F}$ outputs a signature $(\mathtt{m}, \sigma)$ and wins the game if and only if it is a new valid signature.

Informally, the difference between the active and the passive security is that in the passive case $\mathcal{F}$ queries a view oracle $\mathcal{O}_{\mathtt{View-TSign}}(\mathbb{J}, \mathbb{J}_h, \cdot)^{\mathsf{H}}$ that provides $\mathcal{F}$ of

$\mathsf{Exp}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda)$:

1 : $(\mathtt{pp}) \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{\mathtt{sk}_i\}, \mathtt{pk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{pp}, n, t)$
3 : $(\mathbb{J}, \{\mathtt{sk}_i\}_{i\in\mathbb{J}}) \xleftarrow{\$} \mathcal{F}(\mathtt{pp}, \mathtt{pk}, n, t)$
4 : // $|\mathbb{J}| < t$
5 : $(\mathtt{m}, \sigma) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathtt{view-TDS}}(\mathtt{pk},\mathbb{J},\mathbb{J}_h)^{\mathcal{O}_{\mathsf{H}}(\cdot)},\mathcal{O}_{\mathsf{H}}(\cdot)}$
6 : **if** $\mathtt{m} \in \mathbb{M}$
7 : **return** 0
8 : **else**
9 : **return** $\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma)$

$\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda)$:

1 : $\mathtt{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{sk_i\}, \mathtt{pk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{pp}, n, t)$
3 : $(\mathbb{J}, \{\mathtt{sk}_i\}_{i\in\mathbb{J}}) \xleftarrow{\$} \mathcal{F}(\mathtt{pp}, \mathtt{pk}, n, t)$
4 : // $|\mathbb{J}| < t$
5 : $(\mathtt{m}, \sigma) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathsf{TDS}}(\mathtt{pk},\mathbb{J},\mathbb{J}_h)^{\mathcal{O}_H(\cdot)},\mathcal{O}_H(\cdot)}$
6 : **if** $\mathtt{m} \in \mathbb{M}$
7 : **return** 0
8 : **else**
9 : **return** $\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma)$

$\mathcal{O}_{\mathtt{view-TDS}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$

Provides $\mathcal{F}$ with the view of parties in $\mathbb{J}$ who interact with the parties in $\mathbb{J}_h$ in a honest execution.

$\mathcal{O}_{\mathsf{TDS}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$

Controls the parties in $\mathbb{J}_h$, and interacts with $\mathcal{F}$ controlling the parties in $\mathbb{J}$.

Figure 2.3: Experiments for the unforgeability of a threshold digital signature against active and passive attacks. $\mathbb{J}_h \subset \{1, ..., n\} \setminus \mathbb{J}$ denotes the set of honest parties that the oracle controls and that the adversary can choose adaptively before each query. $\mathbb{M}$ is the set of messages queried by $\mathcal{F}$ to the sign oracle.

the view of executions of TSign on input a polynomially large number of messages which the adversary chooses adaptively. The views comprise every state of each party i with $i \in \mathbb{J}$ and every public message. In the active case, the adversary is allowed to interact with the oracle $\mathcal{O}_{\mathsf{TDS}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$ and participate in the signature computation: in particular at every step prescribed by the signing algorithm the adversary sends messages of its choice on behalf of the corrupted party to the oracle, acting as the honest parties.

Unless otherwise specified, we always mean active security when we talk about unforgeability in the context of threshold signatures.

2.8 Security Model for MPC

Informally, the goal of MPC protocols is to enable a set of parties, each having its own input, to jointly compute a function f over their inputs without leaking information about them. As such, adversarial models should assume the corruption of some parties, with the purpose, for example, of discovering the inputs of some of the remaining parties or to cause errors in the output computation.

Numerous security definitions for MPC protocols have been proposed so far. Arguably, the main security guarantees required by the distinct definitions are the following.

- Privacy: no party should be able to obtain any information other than the output of the function f.
- Correctness: the output received by each party should be correct, under the assumption that all of them behave honestly.
- Guarantee of output: corrupted parties should not be able to prevent honest parties from receiving a correct output.
- Fairness: malicious parties should receive the correct output of f if and only if the honest parties receive the correct output of f.
- Independence of input: the input chosen by a corrupted party should be independent from those of honest parties.

Whether the list above is complete or not is difficult to assess. For this reason, the Ideal/Real World paradigm has been introduced.

Suppose that parties $P_1, ..., P_n$ want to jointly compute the output of a function f on their inputs $x_1, \ldots, x_n$. Let $y_1, \ldots, y_n$ be the outputs viewed by parties $P_1, \ldots, P_n$, respectively. In the *Ideal World*, they securely compute it by privately sending their inputs to a fully-trusted third party F, called functionality. Then F simply computes $f(x_1, ..., x_n)$ and returns the result to all parties. In particular, $y_1 = \cdots = y_n$. In this scenario, an adversary that controls only a subset $J \subset \{P_1, .., P_n\}$ of the parties can learn no more than their inputs and $f(x_1, ..., x_n)$.

In the *Real World*, however, all parties communicate with each other using a protocol and there is not any fully-trusted third party. In this scenario an adversary can corrupt parties and act in many different ways (according to the threat models, which we will cover below). In this scenario, the y_i's may be distinct (so there might be an incorrect output). The Ideal World is then used as a "benchmark" against

which evaluating security of MPC protocols. More precisely, an MPC protocol is considered secure if any effect that an adversary can achieve in the Real World can also be achieved by a corresponding adversary in the Ideal World. In other words, the security of an MPC protocol in the real world (under a set of assumptions) should be equivalent to that in the ideal world.

Several security models have been introduced so far, depending on the adversary's behaviour and power. In this document, we focus on the most common scenarios: the semi-honest-adversary security and the malicious-adversary security. The former is usually the minimum security requirement for MPC protocols, even though the latter represents a compromise between enhanced security and efficiency.

Semi-Honest-Adversary Security A semi-honest adversary $\mathcal{A}$ is supposed to corrupt parties, but must follow the protocol. For this reason, semi-honest adversaries are also said to be *passive* (or *honest but curious*), in the sense that they cannot deviate from the prescribed execution of the protocol and may only try to learn as much as possible from the messages they receive from other parties. Note that this may imply that several colluding corrupt parties pool their views together in order to learn information.

Definition 2.22. An MPC protocol π securely computes a function f in the presence of a semi-honest adversary if there exists a simulator $\mathcal{S}$ such that, for every proper subset of corrupted parties $\mathbb{J}$ and all inputs $x_1, ..., x_n$, the two distributions

$$\mathsf{Real}_\pi(\lambda, \mathbb{J}, x_1, ..., x_n)$$

and

$$\mathsf{Ideal}_{\mathsf{F},\mathcal{S}}(\lambda, \mathbb{J}, x_1, ..., x_n)$$

are indistinguishable, where

- λ is the security parameter,
- F is the functionality that computes f,
- $\mathsf{Ideal}_{\mathsf{F},\mathcal{S}}(\lambda, J, x_1, ..., x_n)$ outputs both $\mathcal{S}(\mathbb{J}, (x_i, y_i)|i \in \mathbb{J})$, with y_i the P_i's output produced by F on input $(x_1, \ldots, x_n)$,
- $\mathsf{Real}_\pi(\lambda, J, x_1, ..., x_n)$ outputs $\{\mathtt{view}_i|i \in J\}, (\bar{y}_1, ..., \bar{y}_n)$, with $\mathtt{view}_i$ the final view of the party P_i, and $\bar{y}_i$ its final output (obtained by running the protocol honestly).

Malicious Adversary A malicious adversary may instead corrupt parties and deviate arbitrarily from the protocol. It is important to notice that when corrupted parties deviate from the protocol there is the possibility that honest parties' outputs are affected. No guarantees can be made on the final outputs of corrupted parties, since they can output whatever they like.

With the same notation of definition Definition 2.22 we have:

Definition 2.23. A protocol π securely computes a function f in the presence of a malicious adversary if for every Real World adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ with $\texttt{corrupt}(\mathcal{S}) = \texttt{corrupt}(\mathcal{A}) = \mathbb{J}$ (where $\texttt{corrupt}(\mathcal{A})$ denotes the set of parties that are corrupted, while $\texttt{corrupt}(\mathcal{S})$ denotes the set of parties that are corrupted by the ideal adversary, i.e. $\mathcal{S}$) such that, for all honest parties possible inputs $\{x_i | i \notin \mathbb{J}\}$, the distributions:

$$\mathsf{Real}_{\pi}(\lambda, \mathbb{J}, \{x_i | i \notin \mathbb{J}\})$$

and

$$\mathsf{Ideal}_{\mathsf{F},\mathcal{S}}(\lambda, \mathbb{J}, \{x_i | i \notin \mathbb{J}\})$$

are indistinguishable.

Chapter 3

Generalized Fiat-Shamir Transform

3.1 Introduction

The concept of distributed identification protocol has very few example in cryptography: it was firstly introduced in [BO+88], by Ben-Or, Goldwasser, Kilian, and Wigderson who introduced the concept of *multi prover zero knowledge proof.*They are quite limited in scope and deal with protocols with only two provers who do not communicate after interacting with the verifier. The concept was later revised by Desmedt, Di Crescenzo, and Burmester in [DDCB95], who maintained the setting of no communication between the provers, and by Pedersen in [Ped91], which introduced the concept of multiple provers in the context of undeniable signatures. Pedersen focuses on robustness of the obtained signature and does not make any consideration about the security of threshold identification schemes and their relation with threshold signatures. Lastly, Keller, Mikkelsen, and Rupp in [KMR12] introduced the concept of *multiple prover with combiner*: each of these provers communicates with a combiner, a trusted party that combines the received messages and handle the communication with the verifier, effectively playing the role of the prover in a standard ZKP. In a certain sense, the idea of using a combiner can be seen more as an ideal functionality in the Ideal World than a feasible protocol in the Real World Section 2.8.

Finally, a completely different approach is presented in [Bau+22] by Baum, Jadoul, Orsini, Scholl, and Smart. Authors introduce the concept of multiple verifiers that cooperate to verify a proof made by a single prover.

In this chapter we flip the last approach [Bau+22] generalizing the work in [KMR12], introducing the notion of *distributed identification protocol.* In this

context, the knowledge of the witness is shared among multiple provers cooperating to produce a proof, that will be later verified by a single verifier. Contrary to the previous works such as [BO+88; DDCB95], we allow for communication between the provers even after the challenge step and we do not rely on the presence of combiner. Indeed we formalize the concept of identification protocol in the MPC setting, where each prover engage in an MPC protocol with the other provers and the verifier. In this setting we define the concept of security against impersonation under passive (and active) attacks Definition 2.14 and we show that our definitions can lead to secure threshold signature schemes, miming the approach used in the centralized case in [Abd+02].

Finally, we propose a convenient way to prove the security of a threshold identification protocol: in the same way that centralised identification protocols are usually proved secure by proving the honest verifier zero knowledge and the special soundness of the underlying sigma protocol, we provide analogous definitions for the distributed case and obtain similar results.

The results of this chapter are contained in [BF24], that is currently under review for CRYPTO2024.

3.2 Distributed Identification Schemes and Fiat-Shamir Transform

In this section, we extend the definition of identification scheme to *threshold identification scheme* and propose a generalisation of the Fiat-Shamir Transform for threshold signature schemes. The generalisation is not straightforward and requires additional hypotheses about the structure of the threshold identification scheme, this is formalised in Section 3.2.2.

3.2.1 Threshold identification schemes

We generalise Definition 2.12 and define protocols that allow multiple provers $P_1, ..., P_n$, holding a secret sharing of a secret $\mathtt{sk}$, to prove their joint knowledge of $\mathtt{sk}$. The idea is to replace both $\mathsf{P}_{\mathtt{cmt}}$ and $\mathsf{P}_{\mathtt{rsp}}$ with multi-party protocols that fulfill the same role (called $\mathsf{TP}_{\mathtt{cmt}}$ and $\mathsf{TP}_{\mathtt{rsp}}$). In particular $\mathsf{TP}_{\mathtt{cmt}}$ is run by a set $\mathbb{J}$ of provers to jointly produce a common commitment $\mathtt{cmt}$, then, after receiving a challenge $\mathtt{ch}$, the parties in $\mathbb{J}$ jointly run $\mathsf{TP}_{\mathtt{rsp}}$ to produce a response $\mathtt{rsp}$. We emphasise that, as with threshold signatures, the verifier remains centralised in this setting.

Definition 3.1 (Canonical $(t, n)-$ identification protocol)**.** Let $P_1, \ldots, P_n$ be a set of players. A *threshold identification protocol* is defined by the tuple

$$\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$$

- $\mathsf{Setup}(\lambda)$: on input a security parameter λ, it outputs public parameters $\mathtt{pp}$.
- $\mathsf{KeyGen}(n, t, \mathtt{pp}; R)$: is a probabilistic *key generation algorithm* that takes as input the public parameters $\mathtt{pp}$, the number of participants n and the threshold t, and outputs a public key $\mathtt{pk}$ and a secret sharing $\{\mathtt{sk}_i\}_{[n]}$ of the secret key $\mathtt{sk}$, with each participant P_i holding $\mathtt{sk}_i$;
- $\mathsf{TP}_{\mathtt{cmt}}(\{\mathtt{sk}_i\}_{\mathbb{J}}; R)$[1]: is a probabilistic multi-party protocol run by parties in $\mathbb{J}$ called *threshold prover commitment*. On input the private keys $\mathtt{sk}_i$ it outputs a common commitment $\mathtt{cmt}$;
- $\mathsf{TP}_{\mathtt{rsp}}(\{\mathtt{sk}_i\}_{\mathbb{J}}, \mathtt{cmt}, \mathtt{ch}; R)$: is a probabilistic multi-party protocol run by parties in $\mathbb{J}$ called *threshold prover response*. It takes as input shards $\mathtt{sk}_i$ from the various player, a commitment $\mathtt{cmt}$ and a challenge $\mathtt{ch}$, and outputs a valid response $\mathtt{rsp}$ if $|\mathbb{J}| \geq t$;
- $\mathsf{V}(\mathtt{pk}, \mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$: is a centralised protocol called Verifier which takes a public key, a commitment $\mathtt{cmt}$, a challenge $\mathtt{ch}$ and a response $\mathtt{rsp}$ as inputs, and outputs accept or reject.

We ask that

$$\Pr(\mathsf{V}(\mathtt{pk}, \mathtt{cmt}, \mathtt{ch}, \mathtt{rsp}) = \mathsf{accept} | \mathsf{TP}_{\mathtt{rsp}}(\{\mathtt{sk}_i\}_{\mathbb{J}}, \mathtt{cmt}, \mathtt{ch}) = \mathtt{rsp}) = 1$$

when $|\mathbb{J}| \geq t$ and where we omitted the randomness R for the sake of readability.

From now on, we suppose the presence of a trusted dealer, thus both the Setup and KeyGen are not considered in our discussion. We show later, in Chapter 4 a possible way to achieve a secure decentralised secret sharing. We do this since we want to obtain general results about the security of threshold identification scheme, making as little hypotheses as possible about its structure. For this reason, we put ourselves in a situation where parties have a secure sharing of the secret $\mathtt{sk}$ and we analyze how to keep the security during the proof systems.

[1] to be precise, R should be a vector, since each party in $\mathbb{J}$ must have its own randomness. For sake of readability we simply write R. For the sake of readability we always write a single randomness, the correct meaning will be clear from the context.

3.2.2 Requirements on $\mathsf{TP}_{\mathtt{cmt}}$

Just as we need commitment with high min entropy in the centralised case, we need an analogous idea for the distributed case. However, it is not enough to ask for a super-polynomial challenge space, since the adversary can now participate in the $\mathsf{TP}_{\mathtt{cmt}}$ protocol and thus can bias the distribution. This is especially important for active security, as we will see later.

First of all, we want to characterise protocols that produce "good" commitments when at least one party involved is honest. This captures the idea of the non-triviality of the identification protocols in the centralised case. We refer to this class of $\mathsf{TP}_{\mathtt{cmt}}$ as *unpredictable*.

Definition 3.2 (Unpredictable $\mathsf{TP}_{\mathtt{cmt}}$)**.** Let TId be a (t, n)-threshold identification scheme with $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$. We say that $\mathsf{TP}_{\mathtt{cmt}}$ is *unpredictable*, if and only if the output $\mathtt{cmt}$ has super-logarithmic min-entropy when at least one party is honest.

Observation 3.1. Notice that, at best, the distribution of $\mathtt{cmt}$ is uniform in the commitment space $\mathbb{X}$, so also for the distributed case we need $|\mathbb{X}|$ to be super polynomial. However this is not enough, indeed suppose that $\mathbb{X} = \mathbb{F}_p$, the finite field having p elements, for a large enough p. Suppose that $\mathsf{TP}_{\mathtt{cmt}}$ works as follows: first each party P_i chooses $x_i \in \mathbb{F}_p$ and then sends it to the other parties. Define $\mathtt{cmt} = \prod_i x_i$. This is clearly not unpredictable, since an adversary sending 0 can force $\mathtt{cmt} = 0$ whatever the other x_i are. While this is a problem in the active case, when the adversary is able to decide its own input, this would not be a problem in the passive one, where each party choose x_i randomly. Still, asking for simply super polynomial size of $\mathbb{X}$ is not enough even in the passive case. Indeed, suppose that $\mathtt{cmt} = \min_i(\{x_i\}_{\mathbb{J}})$, where we fixed an order over $\mathbb{F}_p$ (it is enough to consider $\mathbb{F}_p$ as $\mathbb{Z}$ and consider $\mathbb{Z} = [p]$. This protocols clearly does not outputs data with uniform distribution and thus, depending on p, could have min-entropy lower than super-logarithmic.

One way to design an unpredictable $\mathsf{TP}_{\mathtt{cmt}}$ algorithm is to require each party P_i to produce and simultaneously publish a partial commitment $\mathtt{cmt}_i$. Then $\mathtt{cmt}$ is computed using with an agreed upon function such that if at least one party P_j in the group is honest, then $\mathtt{cmt}$ has high min-entropy.

Formally we have the following definition:

Definition 3.3 (Commit-Release $\mathsf{TP}_{\mathtt{cmt}}$)**.** Let TId be a threshold identification scheme with $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$. We say that $\mathsf{TP}_{\mathtt{cmt}}$ *is commit-*

release if and only if the $\mathsf{TP}_{\mathtt{cmt}}$ protocol is unpredictable and has the following structure:

- $\mathsf{TP}^{\mathsf{Com}}_{\mathtt{cmt}}(\mathtt{sk}_i; R)$: a non interactive protocol run locally by each party that outputs a one-way binding commitment (Definition 3.4) $\mathsf{Com}(\mathtt{ssid}||\mathtt{cmt}_i)$ where $\mathtt{cmt}_i$ is picked uniformly at random in $\mathbb{X}$, and $\mathtt{ssid}$ is a session identifier shared among all the parties involved in the execution.
- $\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}(\{\mathsf{Com}(\mathtt{ssid}||\mathtt{cmt}_i)\}_{\mathbb{J}})$: an interactive deterministic protocol run by all the parties involved in the threshold identification execution that release $\mathtt{cmt}_i$, opening $\mathsf{Com}(\mathtt{ssid}||\mathtt{cmt}_i)$, and output a common $\mathtt{cmt}$ obtained deterministically by combining the partial commitments $\mathtt{cmt}_i$.

Note that it is required for the commitment scheme to be binding, so that once created the cryptographic commitment to $\mathtt{cmt}_i$, P_i can only reveal its partial commitment, but the hiding property is not required [KL20]. Indeed, we ask for a weaker privacy property, namely the one-way property. A commitment with randomness length r is one-way when if, when considering the relation $\mathbb{L} \subseteq (\mathbb{M} \times \{0,1\}^r) \times \mathbb{X}$ such that $((\mathtt{m}, R), x) \in \mathbb{L}$ if and only if $\mathsf{Com}(\mathtt{m}, R) = x$, it can be modeled as a (one-way) key generation algorithm for $\mathbb{L}$, as per Definition 2.5. In particular we have the following definition:

Definition 3.4 (One-way commitment scheme)**.** Let $(\mathsf{Com}, \mathsf{Decom})$ be a commitment scheme. We say that $(\mathsf{PGen}, \mathsf{Com}, \mathsf{Open})$. Consider the following experiment:

$\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}}(\lambda)$:

1 : $\mathtt{m} \xleftarrow{\$} \mathbb{M}$
2 : $R \xleftarrow{\$} \{0,1\}^r$
3 : $x \xleftarrow{\$} \mathsf{Com}(\mathtt{m}, R)$
4 : $\mathtt{m}' \xleftarrow{\$} \mathcal{A}(c)$
5 : **return** $\mathtt{m}' = \mathtt{m}$

Define the advantage of an adversary $\mathcal{A}$ playing the game above as

$$\mathbf{Adv}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}}(\lambda) = 1].$$

We say that Com is one way if $\mathbf{Adv}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

It is pretty easy to see that an hiding commitment is also one-way. Formally:

Proposition 3.1. *If* $(\mathsf{Com}, \mathsf{Decom})$ *is an hiding commitment scheme as per Definition 2.10, then it is one-way.*

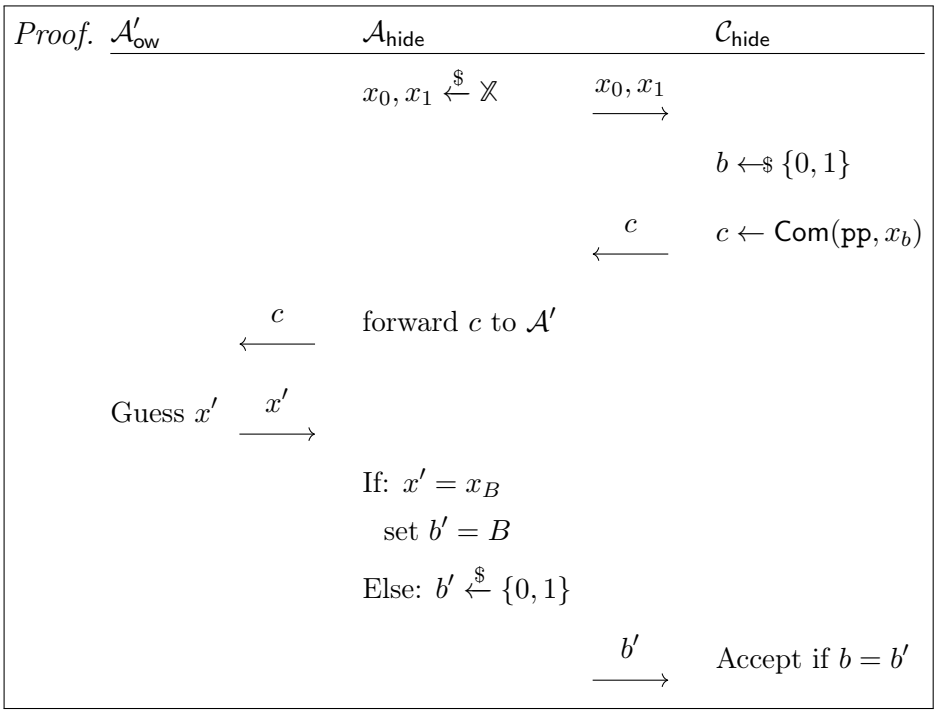


Figure 3.1: Description of the adversary $\mathcal{A}$ of the hiding game which uses the adversary $\mathcal{A}'$ of the one-way experiment as a subroutine.

We show that an adversary $\mathcal{A}'$ who wins the experiment $\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{Com},\mathcal{A}}(\lambda) : (\lambda)$ with non-negligible advantage $\nu(\lambda)$ can be used as a subroutine of an adversary $\mathcal{A}$ capable to win the hiding experiment with non-negligible advantage.

Figure 3.1 shows the reduction schematically. First $\mathcal{A}$ generates two messages $\mathtt{m}_0, \mathtt{m}_1$ and sends them to the challenger of the hiding game, receiving $\mathtt{cmt} = \mathsf{Com}(x_b, R)$ for a random unknown bit b. $\mathcal{A}$ can simulate the challenger of the one-way experiment by sending the commitment to $\mathcal{A}'$ which returns $\mathtt{m}'$ to $\mathcal{A}$. If $\mathtt{m}' = \mathtt{m}_i$ for some i, $\mathcal{A}$ sets $b' = i$. Otherwise $\mathcal{A}$ picks a random bit b'.

Clearly the simulation is correct, the only difference is that $\mathcal{A}$ does not know the commited value inside `cmt`, however this is indistinguishable from the real game since `cmt` is generated honestly by the challenger for the hiding game.

Now we show that $\mathcal{A}$ wins the hiding experiment with non-negligible advantage. When $\mathtt{m}' \neq \mathtt{m}_i$, $\mathcal{A}'$ has lost the one-way game, and this happens with probability at most $p_1 < 1 - \nu$ for a non-negligible function ν. With probability greater $\nu(\lambda)$ instead $\mathcal{A}'$ returns $\mathtt{m}' = \mathtt{m}_b$. In this case either $\mathcal{A}'$ wins the one-way experiment, and thus $\mathcal{A}$ wins the hiding experiment, or it looses its experiment but guesses x_{1-b}, the other message that $\mathcal{A}$ randomly picked during the hiding experiment. The second case, where $\mathcal{A}'$ does not win and outputs the other message $\mathtt{m}_{1-b}$ happens with probability $\frac{1}{|\mathbb{X}|}$, that is negligible when $\mathbb{X}$ has super-polynomial size.

To summarize,

$$\begin{aligned}
\Pr(\mathcal{A} \text{ wins}) &= \Pr((b = b' \wedge ((x' \neq x_0) \wedge (x' \neq x_1))) \vee x' = x_b) = \\
&= \Pr(b = b' \wedge ((x' \neq x_0) \wedge (x' \neq x_1))) + \Pr(x' = x_b) = \\
&= \Pr(b = b' | (x' \neq x_0) \wedge (x' \neq x_1)) \Pr(((x' \neq x_0) \wedge (x' \neq x_1))) + \nu = \\
&= \frac{1}{2}\left(1 - \left(\nu + \frac{1}{|\mathbb{X}|}\right)\right) + \nu = \frac{1}{2} + \frac{1}{2}\nu - \frac{1}{2|\mathbb{X}|}
\end{aligned}$$

therefore the advantage of $\mathcal{A}$ in winning the hiding experiment is $\frac{1}{2}\nu - \frac{1}{N}$ which is non-negligible.

□

Security notions for threshold identification schemes.

Let TId be a threshold identification protocol, with public key $\mathtt{pk}$ and secret keys $\{\mathtt{sk}_i\}_{i\in[n]}$, we associate to TId two oracles that will be used in the experiments that define the security notions below:

- a transcript generation oracle $\mathcal{O}_{\mathtt{view-TId}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$ that takes as input the public key, two non-empty sets of parties $\mathbb{J}$ and $\mathbb{J}_h$ such that $|\mathbb{J} \cup \mathbb{J}_h| = t$ and returns random transcript of an honest execution of TId, including all the public messages and the internal state of parties in $\mathbb{J}$.
- a threshold identification oracle $\mathcal{O}_{\mathsf{TId}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$ that takes as input the public key, two non-empty sets of parties $\mathbb{J}$ and $\mathbb{J}_h$ such that $|\mathbb{J} \cup \mathbb{J}_h| = t$. When an adversary $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{TId}}$, the oracle interacts with $\mathcal{A}$ in an execution of TId. In particular, $\mathcal{A}$ controls the parties in $\mathbb{J}$, while $\mathcal{O}_{\mathsf{TId}}$ controls both the parties in $\mathbb{J}_h$ and the verifier in real execution of the protocol.

These two oracles aim to capture the differences between a passive adversary and an active one. In particular, when $\mathcal{A}$ queries $\mathcal{O}_{\mathtt{view-TId}}$, it is not able to

decide the messages sent by the corrupted parties ($\mathbb{J}$), it simply learns all their the internal states and the messages they received. Instead $\mathcal{O}_{\mathsf{TId}}$ allows the adversary to participate in the execution of TId, controlling all the parties in $\mathbb{J}$, while the oracle controls all the honest parties ($\mathbb{J}_h$).

Definition 3.5 (Security against impersonation under passive attacks)**.** Let TId be a (t, n)-threshold identification scheme with $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$, having challenge length c and security parameter λ. Let $\mathcal{A}$ be an impersonator having access to a threshold transcript generation oracle $\mathcal{O}_{\mathtt{view-TId}}(\cdot)$.

We define the advantage of $\mathcal{A}$ in winning the experiment $\mathsf{Exp}^{\mathsf{p-imp}}_{\mathsf{TId},\mathcal{A}}(\lambda)$ described in Figure 3.2 as

$$\mathbf{Adv}^{\mathsf{p-imp}}_{\mathsf{TId},\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{p-imp}}_{\mathsf{TId},\mathcal{A}}(\lambda) = 1)$$

We say that TId is *secure against impersonation under passive attack* if $\mathbf{Adv}^{\mathsf{p-imp}}_{\mathsf{TId},\mathcal{A}}$ is negligible for every probabilistic polynomial time impersonator $\mathcal{A}$.

Definition 3.6 (Security against impersonation under active attack)**.** Let TId be a (t, n)-threshold identification scheme with $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$, having challenge length c and security parameter λ. Let $\mathcal{A}$ be an impersonator having access to a threshold identification oracle $\mathcal{O}_{\mathsf{TId}}(\cdot)$.

Define the advantage of $\mathcal{A}$ in winning the experiment $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{A}}(\lambda)$ described in Figure 3.2 as

$$\mathbf{Adv}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TDS},\mathcal{A}}(\lambda) = 1)$$

We say that TId is *secure against impersonation under active attack* if $\mathbf{Adv}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{A}}$ is negligible for every probabilistic polynomial time impersonator $\mathcal{A}$.

There are two main differences between the two definitions: one is about the oracle $\mathcal{A}$ can interact with and the second one is about when they can interact with it.

Indeed, in both cases $\mathcal{A}$ can corrupt at most $t - 1$ parties and obtain their private keys. However, in the active case, $\mathcal{A}$ interacts with the identification oracle $\mathcal{O}_{\mathsf{TId}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$ that plays the role of the honest parties (that the adversary can adaptively choose), while in the passive case it can only query the transcript generation oracle $\mathcal{O}_{\mathtt{view-TId}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$, that provides $\mathcal{A}$ with transcripts of honest executions of the identification protocol.

The second difference is that, whereas in the passive case $\mathcal{A}$ can be assumed to receive all the transcripts from the $\mathcal{O}_{\mathtt{view-TId}}(\cdot)$ before it creates the commitment

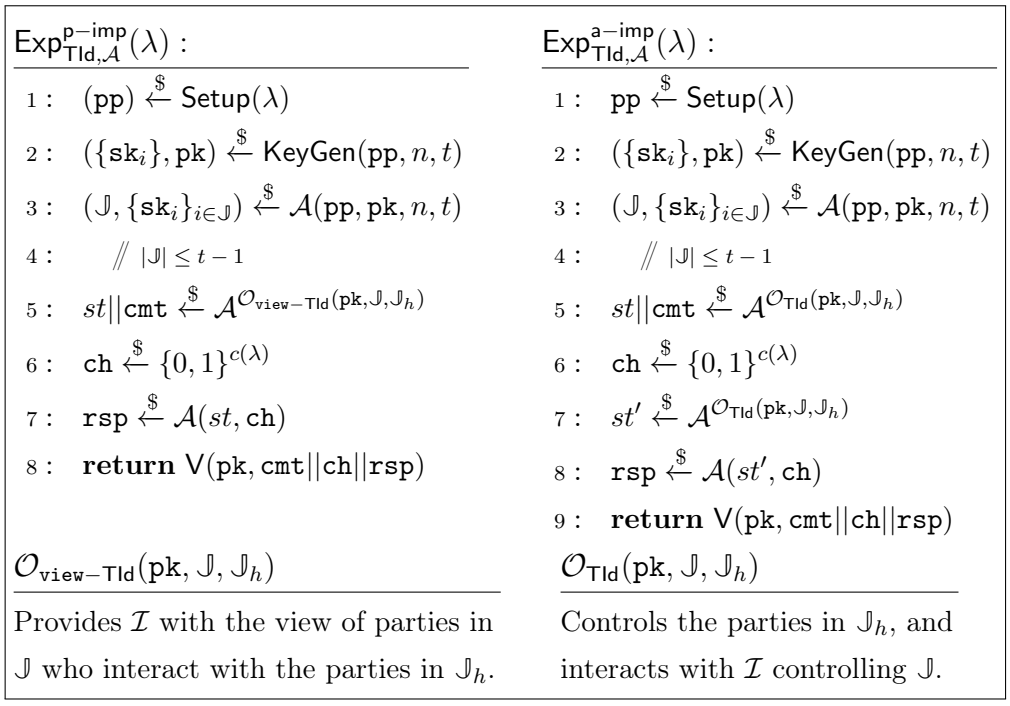


Figure 3.2: Experiments of active and passive impersonation attacks. $\mathbb{J}_h \subset \{1, ..., n\} \setminus \mathbb{J}$ denotes the set of honest parties that the oracle controls and that the adversary can choose adaptively before each query.

$\mathsf{TSign}(\mathtt{m}, \{\mathtt{sk}_i\}_{i\in\mathbb{J}})$:	$\mathsf{V}(\mathtt{pk}, \mathtt{m}, \sigma)$:
1 : $R \xleftarrow{\$} \mathsf{Coins}^t(\lambda)$	1 : Parse σ as $\mathtt{cmt}\|\mathtt{rsp}$
2 : $\mathtt{cmt} \leftarrow \mathsf{TP}_{\mathtt{cmt}}(\{sk_i\}_{i\in\mathbb{J}}; R)$	2 : $\mathtt{ch} \leftarrow \mathsf{H}(\mathtt{cmt}\|\|\mathtt{m})$
3 : $\mathtt{ch} \leftarrow \mathsf{H}(\mathtt{cmt}\|\|\mathtt{m})$	3 : **return** $\mathsf{V}(\mathtt{pk}, \mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$
4 : $\mathtt{rsp} \leftarrow \mathsf{TP}_{\mathtt{rsp}}(\{sk_i\}_{i\in\mathbb{J}}, \mathtt{cmt}, \mathtt{ch}; R)$	
5 : **return** $\mathtt{cmt}\|\|\mathtt{rsp}$	

Figure 3.3: Signing and verification algorithm for a threshold signature obtained by using the distributed Fiat-Shamir transform

$\mathtt{cmt}$ of the impersonation attempt, in the active case $\mathcal{A}$ is allowed to interact with the identification oracle $\mathcal{O}_{\mathsf{TId}}(\cdot)$ even after it has sent the commitment of the impersonation attempt and has received the challenge $\mathtt{ch}$ from the verifier (Figure 3.2, $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{A}}(\lambda)$, line 7). This choice is intended to broaden the options of an attacker running the experiment as much as possible.

3.2.3 Distributed Fiat-Shamir Transform

We are now ready to generalise the definition of *Fiat-Shamir transform* presented in [Abd+02] to the distributed case. The definitions above allows for a very natural and intuitive construction: to sign a message, the provers run the prover commitment protocol $\mathsf{TP}_{\mathtt{cmt}}$, compute the challenge as $\mathsf{H}(\mathtt{cmt}||\mathtt{m})$ as in the centralised case and finally they jointly compute the signature performing $\mathsf{TP}_{\mathtt{rsp}}$. Notice that, as expected, the threshold signature obtained maintains the same threshold, since $\mathsf{TP}_{\mathtt{rsp}}$ require t participants. The verification step is adapted in the same way as the standard Fiat-Shamir Transform.

Definition 3.7 (Distributed Fiat-Shamir transform)**.** Let TId be a canonical threshold identification scheme with $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$.

We define the threshold digital signature TDS derived from the canonical $(t, n)-$identification scheme TId using the Fiat-Shamir transform as the tuple $\mathsf{TDS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TSign}, \mathsf{V})$ where the Setup and KeyGen algorithms are the same as the identification scheme, the output length of the hash function is equal to the challenge length of the identification scheme and the signing and the verification algorithms are defined as per Figure 3.3 ($\mathbb{J} \geq t$).

3.3 Security of the Distributed Fiat-Shamir Transform

In this section, we state and prove the main result of this chapter: the relation between the security of the threshold identification protocol and the security of the threshold signature obtained by applying the distributed Fiat-Shamir Transform. First, we analyse the security against active adversaries in Section 3.3.1 , then we deal with the passive case in Section 3.3.2, for which we only provide a proof sketch, since it is almost identical to the active one.

3.3.1 Active security

The need to deal also with active adversaries is the main difference between the multi-party setting and the centralised one from [Abd+02]. In the centralised case, we do not need to consider adversaries who are able to influence the distribution of messages during the sign requests (in fact, the adversary only receives the signatures of the requested messages), while in the distributed case we must also consider this possibility. For this reason we are not able to treat the identification protocol in a "black-box" way, but we need to make some assumptions about the how the parties interact, namely about the $\mathsf{TP}_{\mathtt{cmt}}$ protocol, as anticipated in Section 3.2.2. This leads to a security theorem that is not an equivalence result, as is possible in the centralised case instead. While we are not able prove that our hypothesis about the structure of the protocol is the minimal one, the proposed structure is perfectly reasonable and shared by the vast majority of threshold signatures [CKM23; Lin22; Bat+22a].

Theorem 3.1 (Active security)**.** *Let* $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$ *be a canonical threshold identification scheme. Consider the associated signature scheme* $\mathsf{TDS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TSign}, \mathsf{Ver})$ *as per Definition 3.7. Then the following implications hold:*

(i) $(\mathsf{TId} \implies \mathsf{TDS})$*: if* $\mathsf{TP}_{\mathtt{cmt}}$ *satisfies the commit-release property as per Definition 3.3 and* TId *is secure against impersonation under active attacks, then* TDS *is secure against active chosen-message attacks.*

(ii) $(\mathsf{TDS} \implies \mathsf{TId})$*: If* TDS *is secure against active chosen-message attacks, then* TId *is secure against impersonation under active attacks.*

We prove the two implications separately, providing a game-based proof.

Lemma 3.1 ($\mathsf{TId} \implies \mathsf{TDS}$)**.** *Under the assumptions of Theorem Theorem 3.1, if* $\mathsf{TP}_{\mathtt{cmt}}$ *satisfies the commit-release property and* TId *is secure against impersonation under active attacks, then* TDS *is unforgeable against active chosen-message attacks.*

First, we give an intuitive idea of the proof, with the help of Figure 3.4. The impersonator $\mathcal{I}$ who participates in the experiment $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{I}}$ interacts with a challenger $\mathcal{C}_{\mathsf{TId}}$, who initialises the experiment sends the challenge $\mathtt{ch}^*$ during the impersonation attempt, and the oracle $\mathcal{O}_{\mathsf{TId}}$, which plays the role of the honest parties when $\mathcal{I}$ asks to take part in the threshold identification protocol.

The goal is to use the forger $\mathcal{F}$ for the distributed signature to win $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{I}}$. To do so, $\mathcal{I}$ must simulate the experiment $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{F}}$, therefore it should simulate the challenger $\mathcal{C}_{\mathsf{TDS}}$, the random oracle $\mathcal{O}_{\mathsf{H}}$ and the signature generation oracle $\mathcal{O}_{\mathsf{TDS}}$. In the picture they are denoted with a bar, to better visualize that $\mathcal{I}$ simulates them and are not real interaction with the oracles.

The simulation comprises four parts, each of them denoted by a different enumeration system. Namely

Numbers (1)-(6): the initialisation of the security game of TId. $\mathcal{I}$ uses the same data in the initialisation of TDS for $\mathcal{F}$. This allows $\mathcal{I}$ to correctly simulate $\mathcal{C}_{\mathsf{TDS}}$. Notice that $\mathcal{I}$ corrupts the same parties chosen by $\mathcal{F}$.

Lower case letters (a)-(o): the simulation of the sign queries made to $\mathcal{O}_{\mathsf{TDS}}$ by $\mathcal{F}$. In particular. $\mathcal{F}$ sends to $\mathcal{I}$ a sign query for $\mathtt{m}$, asking for the cooperation of the parties in $\mathbb{J}_h$. $\mathcal{I}$, to simulate the sign oracle, starts an interaction with $\mathcal{O}_{\mathsf{TId}}$ asking for the same $\mathbb{J}_h$. $\mathcal{F}$ forwards the messages received by $\mathcal{O}_{\mathsf{TId}}$ to $\mathcal{F}$ (steps (c-d) and (g-h)) and vice versa (steps (e-f) and (i-j)). In step (k), when $\mathcal{I}$ receives the challenge $\mathtt{ch}$ from $\mathcal{O}_{\mathsf{TId}}$, it updates the hash table setting $\mathsf{HT}[\mathtt{m}||\mathtt{cmt}] = \mathtt{ch}$. Finally $\mathcal{I}$ carries out the whole signing protocol with the support of $\mathcal{O}_{\mathsf{TId}}$.

Greek letters $(\alpha) - (\beta)$: the simulation of the hash queries. $\mathcal{I}$ can safely answer randomly, except when asked multiple times on the same message. For this reason $\mathcal{I}$ populates an hash table HT to keep track of all the previous queries and responds accordingly..

The only exception is during the $\mathtt{fp}$-th hash query. In this case $\mathcal{I}$ parses x as $\mathtt{m}^*||\mathtt{cmt}^*$ and starts the impersonation attempt (Capital letters).

Capital letters (A)-(D): $\mathcal{I}$ starts the impersonation attempt during the $\mathtt{fp}$-th hash query of $\mathcal{F}$ (step (A) and (B)). After a polynomial number of hash queries and sign queries the forger $\mathcal{F}$ outputs a forgery $(\widehat{\mathtt{cmt}}, \widehat{\mathtt{ch}}, \widehat{\mathtt{rsp}})$ (step (C)). At

this point $\mathcal{I}$ uses it in its impersonation attempt. In particular $\mathcal{I}$ sends $\widehat{\texttt{rsp}}$ to $\mathcal{C}_{\mathsf{TId}}$ (step (D)) as response.

Proof. Let $\mathcal{F}$ be a forger that wins the $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},F}(\lambda)$ with non-negligible advantage $\epsilon(\lambda)$. First, we require that $\mathcal{F}$ some additional properties, as in [Abd+02]:

- all of its hash queries have the form $\texttt{cmt}||\texttt{m}$ with $\texttt{cmt} \in \mathbb{X}, \texttt{m} \in \{0,1\}^*$;
- before outputting a forgery $(\texttt{m}, \texttt{cmt}||\texttt{rsp})$, $\mathcal{F}$ has performed an hash query for $(\texttt{cmt}||\texttt{m})$;
- if $\mathcal{F}$ outputs $(\texttt{m}, \texttt{cmt}||\texttt{rsp})$, $\texttt{m}$ was never a sign query.

It is easy to see that if there exists a forger $\mathcal{F}'$ which does not satisfy these requirements, it is possible to build a forger $\mathcal{F}$ which does satisfy the requirements using $\mathcal{F}'$ as a subroutine, as discussed in [Abd+02], Proof of Lemma 3.5.

The proof is structured as follows: first we describe how the simulation works, in particular we show how $\mathcal{I}$ can simulate the challenger of the experiment $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{F}}(\lambda)$ both in the initialisation and in the training phase. Then, we show that the simulation is successful with overwhelming probability, and we show how $\mathcal{I}$ can exploit $\mathcal{F}$'s forgery to carry out its impersonation attempt. Finally, we show the advantage that $\mathcal{I}$ has in winning $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{I}}(\lambda)$.

Initialisation. $\mathcal{I}$ receives from $\mathcal{C}_{\mathsf{TId}}$ the public parameters $\texttt{pp}$ of the identification protocol and the public key $\texttt{pk}$. $\mathcal{I}$ forwards this information to $\mathcal{F}$ and initializes the hash query counter $\texttt{hc} = 0$, the sign query counter $\texttt{sc} = 0$, an empty hash table $\mathsf{HT} = \emptyset$ and an empty query table $\mathsf{QT} = \emptyset$. Lastly, $\mathcal{I}$ generates a random forge pointer $\texttt{fp} \in [q_h(\lambda)]$.

The hash table and the query table are used by $\mathcal{I}$ to maintain the coherence during the simulation, avoiding contradictions in the simulation (e.g. to avoid answering different hash queries with the same input with two different digests). The forge pointer is guess made by $\mathcal{I}$. Since $\mathcal{F}$ performs an hash query with input the message it decides to forge, $\mathcal{I}$ guesses ahead of time the position of this query. We show later that the guess is correct with non negligible probability and that a correct guess leads to the successful execution of the impersonation attack.

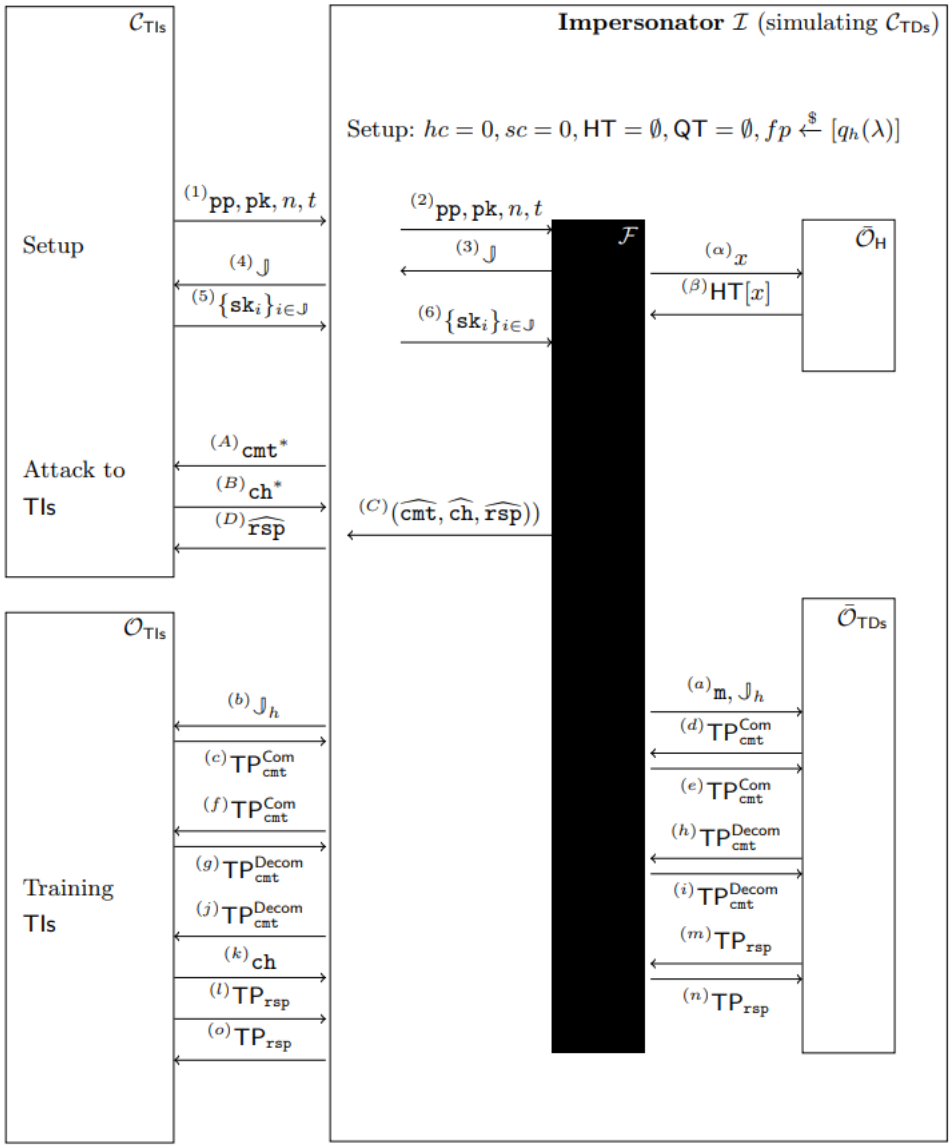


Figure 3.4: High level description of the impersonator $\mathcal{I}$ using a forger $\mathcal{F}$ as a subroutine.

Training phase. $\mathcal{F}$ chooses the the set $\mathbb{J}$ (with $|\mathbb{J}| \leq t-1$) of actors it wants to control. $\mathcal{I}$ chooses the same set $\mathbb{J}$ and sends it to $\mathcal{C}_{\mathsf{Tld}}$, receiving the secret keys of the players in $\mathbb{J}$, finally $\mathcal{I}$ forwards this information to $\mathcal{F}$. This simulate perfectly the first three lines of the $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{Tld},\mathcal{F}}$ game of Figure 2.3 of Definition 2.21.

Now we need to show how can $\mathcal{I}$ can simulate the random oracle $\mathcal{O}_{\mathsf{H}}$ and the sign oracle $\mathcal{O}^{\mathsf{H}}_{\mathsf{TDS}}$ in the EUF-CMA security game. In the first case $\mathcal{I}$ uses the hash table HT to answer, while in the second $\mathcal{I}$ performs an identification query to its oracle $\mathcal{O}_{\mathsf{Tld}}$ using the same input as part of the $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{Tld},\mathcal{I}}(\lambda)$ game. Specifically, the simulation works as follows:

- **Hash Query**: when $\mathcal{F}$ performs an hash query to $\mathcal{O}_{\mathsf{H}}$ on input x, $\mathcal{I}$ returns $\mathsf{HT}[x]$ if it is already defined. Otherwise, $\mathcal{I}$ increases the counter $\mathtt{hc}$ by 1 and sets $\mathsf{QT}[\mathtt{hc}] = x$, then, if $\mathtt{hc} \neq \mathtt{fp}$, $\mathcal{I}$ picks uniformly at random $d \in \{0,1\}^{c(\lambda)}$, sends it to $\mathcal{F}$ and sets $\mathsf{HT}[x] = d$. If $\mathtt{hc} = \mathtt{fp}$, instead of answering randomly, $\mathcal{I}$ starts the impersonation attempt. In particular, it parses x as $\mathtt{cmt}^*||\mathtt{m}^*$, sends $\mathtt{cmt}*$ to the challenger $\mathcal{C}_{\mathsf{Tld}}$ as the first move of the impersonation attempt of the $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},F}(\lambda)$ game and receives back a challenge $\mathtt{ch}^*$. In this case, $\mathcal{I}$ sets $\mathsf{HT}[x] = \mathtt{ch}^*$ and sends $\mathtt{ch}^*$ to $\mathcal{F}$. This procedure allows $\mathcal{I}$ to perfectly simulate the random oracle $\mathcal{O}_{\mathsf{H}}$. Indeed, the only difference between a real execution and the simulation is during the $\mathtt{fp}$-th query. In this case, instead of choosing a random value, as a real random oracle, $\mathcal{I}$ forwards the value $\mathtt{ch}^*$ received from $\mathcal{C}_{\mathsf{Tld}}$. However, since $\mathtt{ch}^*$ is chosen uniformly at random in the challenge space, this is indistinguishable from a real execution.

- **Sign Query**: to start a sign query, $\mathcal{F}$ chooses $\mathbb{J}_h$, the set of the honest players who participates in the computation of a signature of $\mathtt{m}$ alongside parties in $\mathbb{J}$. In theory, $\mathcal{F}$ could include only a proper subset of $\mathbb{J}$ in the query, instead of the whole $\mathbb{J}$. Since the simulation is identical in both cases, we suppose that $\mathcal{F}$ decides to include the whole set $\mathbb{J}$, for the sake of readability.

  After receiving $\mathbb{J}, \mathbb{J}_h$ and $\mathtt{m}$, $\mathcal{I}$ increases the signature counter $\mathtt{sc}$ and sends to $\mathcal{O}_{\mathsf{Tld}}(\mathtt{pk}, \mathbb{J}, \mathbb{J}_h)$ a request to perform the threshold identification protocol. The impersonator $\mathcal{I}$ acts as a "man-in-the-middle" between $\mathcal{F}$ and $\mathcal{O}_{\mathsf{Tld}}$, and repeats the following operations for each step prescribed by the algorithm $\mathsf{TP}_{\mathtt{cmt}}$:

  (i) $\mathcal{O}_{\mathsf{Tld}}$ produces the messages for the participants in $\mathbb{J}_h$ and sends them to $\mathcal{I}$ in the execution of the steps prescribed by $\mathsf{TP}^{\mathsf{Com}}_{\mathtt{cmt}}$;

(ii) $\mathcal{I}$ forwards to $\mathcal{F}$ the messages received from $\mathcal{O}_{\mathsf{TId}}$;

(iii) $\mathcal{F}$ produces the messages executing $\mathsf{TP}^{\mathsf{Com}}_{\mathtt{cmt}}$ on behalf of the corrupted participants in $\mathbb{J}$.

(iv) $\mathcal{I}$ forwards the messages received from $\mathcal{F}$ to $\mathcal{O}_{\mathsf{TId}}(\mathtt{pk}, J, \mathbb{J}_h)$.

These steps are repeated for the protocol $\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}$, leading to the computation of the shared $\mathtt{cmt}$. Notice that, in general, we should assume that the adversary $\mathcal{F}$ always speaks last and thus obtains the output before $\mathcal{I}$. This could lead to collision, however this happen with negligible probability thanks to the structure of $\mathsf{TP}_{\mathtt{cmt}}$, as we show later. After obtaining $\mathtt{cmt}$, $\mathcal{I}$ forwards it to $\mathcal{O}_{\mathsf{TId}}$, outputs a random challenge $\mathtt{ch}$. $\mathcal{I}$ sets $\mathsf{HT}[\mathtt{cmt}||\mathtt{m}] = \mathtt{ch}$. The simulation is then repeated during the computation of $\mathsf{TP}_{\mathtt{rsp}}$.

This concludes the description of the simulation EUF-CMA game. Since TId and TDS are identical, the messages sent by $\mathcal{I}$ to $\mathcal{F}$ forms a perfect simulation and are indistinguishable from messages in a real execution. The only exception is when $\mathcal{I}$ update the hash table, setting $\mathsf{HT}[\mathtt{cmt}||\mathtt{m}] = \mathtt{ch}$. We now show that this step leads to failure with negligible probability, making the simulation indistinguishable from a real execution.

Simulation failure. We have shown that the simulation of $\mathcal{I}$ fails only if $\mathcal{I}$ is forced to overwrite the hash table HT during a sign query performed by $\mathcal{F}$. This can be due to a previous sign query or a previous hash query, we treat the two cases separately.

(i) **During a sign query:** this means that before producing the commitment $\mathtt{cmt}$ during the current sign query for $\mathtt{m}$, $\mathcal{F}$ has performed another sign query for $\mathtt{m}$, having the same $\mathtt{cmt}$. For $n \in [q_s(\lambda)]$, let $\mathbb{X}_n \subset \mathbb{X}$ the set of commitments $\mathtt{cmt}$ generated during the previous $n-1$ sign queries, then the failure probability of the simulation during sign query n for a collision of the commitment with the commitment of a previous query is:

$$\Pr[\mathtt{cmt} \in \mathbb{X}_n] = \frac{n-1}{2^{\beta(\lambda)}}.$$

(ii) **During an hash query:** this means that $\mathcal{F}$ has already performed an hash query for $\mathtt{cmt}||\mathtt{m}$. This can happen in three circumstances, either before the sign query, during the sign query but before the execution of $\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}$ or after the execution of $\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}$.

– The first case happens with negligible probability. Since $\mathsf{TP}_{\mathtt{cmt}}$ has high min-entropy, the probability of guessing $\mathtt{cmt}$ is negligible.

– The third case also happens with negligible probability. Indeed, since Com is one-way, the probability of guessing $\mathtt{cmt}$ is negligible even after knowing $\mathsf{Com}(\mathtt{cmt}_i)$ at the end of $\mathsf{TP}^{\mathsf{Com}}_{\mathtt{cmt}}$.

– The second case is the trickier one. Indeed, at the end of $\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}$, $\mathcal{F}$ knows the value of $\mathtt{cmt}$ before $\mathcal{I}$ and thus can ask for a sign query on $\mathtt{m}||\mathtt{cmt}$ without $\mathcal{I}$ realizing it and causing a collision with high probability. To avoid that, $\mathcal{I}$ rewinds the adversary. In particular, $\mathcal{I}$ follows the simulation normally,and stores the challenge $\mathtt{ch}$ received from the oracle $\mathcal{O}_{\mathsf{Tld}}$. If $\mathcal{F}$ ask for $\mathtt{m}||\mathtt{cmt}$, when $\mathcal{I}$ learns $\mathtt{cmt}$ it rewinds the forger to the moment in which it performs the random oracle query for $\mathtt{m}||\mathtt{cmt}$. This time $\mathcal{I}$ sets the digest to $\mathtt{ch}$, and since the algorithm $\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}$ is deterministic, when $\mathcal{F}$ will complete the algorithm $\mathsf{TP}_{\mathtt{cmt}}$ releasing its partial commitments, the final commitment will result to be $\mathtt{cmt}$ again. This means that after the rewinding of $\mathcal{F}$ the simulation does not fail and is correct since the value $\mathtt{ch}$ was picked uniformly at random by $\mathcal{O}_{\mathsf{Tld}}$.
  If the commitment scheme used in $\mathsf{TP}_{\mathtt{cmt}}$ is done via a random oracle (e.g. $\mathsf{Com}(\mathsf{pp}, x) = \mathsf{H}_{\mathsf{Com}}(x)$) then the simulation does not require any rewind. Indeed, in order to create its cryptographic commitment to $\mathtt{cmt}_i$, $\mathcal{F}$ must have previously sent to the random oracle a query for $\mathtt{cmt}_i$. This allows $\mathcal{I}$ to extract all the partial commitments of $\mathcal{F}$ and thus it can compute the value $\mathtt{cmt}$ ahead of time. Notice that if the value sent by the adversary after $\mathsf{TP}^{\mathsf{Com}}_{\mathtt{cmt}}$ is not a previously outputted value the adversary would not be able to produce the decommitment with overwhelming probability, causing a failure.
  While this requirement is not necessary for a game based proof, it is required for proving the protocol secure in the UC model, as noted later in Section 3.3.3.

Therefore the probability that $\mathcal{I}$ fails its simulation and overwrites the hash table is:

$$\Pr[\mathcal{I} \text{ fails}] \leq \sum_{n=1}^{q_s(\lambda)} \frac{(n-1)+q_h(\lambda)}{2^{\beta(\lambda)}} =$$
$$= \frac{q_h(\lambda)q_s(\lambda)}{2^{\beta(\lambda)}} + \sum_{n=1}^{q_s(\lambda)} \frac{(n-1)}{2^{\beta(\lambda)}} =$$
$$= \frac{q_h(\lambda)q_s(\lambda) + q_s(\lambda)(q_s(\lambda)-1)/2}{2^{\beta(\lambda)}}.$$

Therefore it holds that

$$\Pr[\mathcal{I} \text{ fails}] \leq \frac{q_s(\lambda)(q_h(\lambda)+q_s(\lambda)-1)}{2^{\beta(\lambda)}}. \tag{3.3.1}$$

which is negligible in λ.

Exploit of $\mathcal{F}$'s forgery. Once $\mathcal{F}$ has concluded the training phase, $\mathcal{F}$ outputs a forgery $(\widehat{\texttt{cmt}}, \widehat{\texttt{rsp}})$ of a message $\widehat{m}$ not previously queried. Then $\mathcal{I}$ concludes its impersonation attempt by sending the message $\widehat{\texttt{rsp}}$ as a response to the challenge $\texttt{ch}^*$ received after the $\texttt{fp}$-th hash query, associated to the commitment $\texttt{cmt}^*$.

Note that if $\widehat{\texttt{cmt}} = \texttt{cmt}^*$, $\widehat{\texttt{m}} = \texttt{m}^*$ and $(\widehat{\texttt{cmt}}, \widehat{\texttt{rsp}})$ is a valid forgery of $\widehat{\texttt{m}} = \texttt{m}^*$, which happens if $\texttt{fp}$ was guessed by $\mathcal{I}$, then the impersonator will be successful in its impersonation attempt.

Evauation of $\mathcal{I}$'s advantage. In order to win the game $\mathcal{I}$ needs

- to successfully simulate;
- to guess correctly the forge pointer $\texttt{fp}$;
- that $\mathcal{F}$ outputs a valid forgery.

Finally we can find a lower bound to the probability of success of the impersonator $\mathcal{I}$ in playing the experiment:

$$\begin{aligned}
&\Pr[\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{Tld},\mathcal{I}} = 1] \geq \Pr[\mathcal{F}^{\mathcal{I}} \text{ wins} \wedge \mathcal{I} \text{ guesses } \texttt{fp} \wedge \mathcal{I} \text{ simulates}] = \\
&= \Pr[\mathcal{F}^{\mathcal{I}} \text{ wins} \wedge \mathcal{I} \text{ guesses } \texttt{fp} \mid \mathcal{I} \text{ simulates}] \cdot \Pr[\mathcal{I} \text{ simulates}] = \\
&= \Pr[\mathcal{F}^{\mathcal{I}} \text{ wins} \mid \mathcal{I} \text{ simulates}] \cdot \Pr[\texttt{fp} \text{ is guessed} \mid \mathcal{I} \text{ simulates}] \cdot \Pr[\mathcal{I} \text{ simulates}] \geq \\
&\geq \epsilon(\lambda) \frac{1}{q_h(\lambda)} \left(1 - \frac{q_s(\lambda)(q_h(\lambda)+q_s(\lambda)-1)}{2^{\beta(\lambda)}}\right)
\end{aligned}$$

which is non-negligible in the security parameter λ.

Notice that in the second equality we used the fact that $\mathtt{fp}$ is sampled uniformly at random by $\mathcal{I}$ before it starts interacting with $\mathcal{F}$ and the value of $\mathtt{fp}$ does not affect the simulation of the experiment with $\mathcal{F}$, and in the third equality we used the lower bound to the probability that $\mathcal{I}$ fails the simulation described in Equation Equation (3.3.1).

This is a contradiction, since TId was assumed secure against impersonation under active attacks. Therefore the digital signature TDS is unforgeable under active attacks, and this concludes the proof. □

Observation 3.2. The hypothesis about the structure of $\mathsf{TP}_{\mathtt{cmt}}$ is crucial to reproduce the simultaneity of the exchange of messages between the parties involved in the creation of $\mathtt{cmt}$. Without this structure the simulator would not be able to extract the adversary $\mathtt{cmt}_i$, either by rewind or by simulating the random oracle.

This is not only needed from a "formal " standpoint. Indeed, let us consider the following a $\mathsf{TP}_{\mathtt{cmt}}$ in which each party choose randomly $\mathtt{cmt}_i$ and publishes it, then all the parties set $\mathtt{cmt} = \sum_{i \in J} \mathtt{cmt}_i$. This protocol is clearly not secure, indeed $\mathcal{F}$ might force two consecutive signing sessions on different messages to have the same $\mathtt{cmt}$, while the challenge will be different with high probability. This can cause attacks, in particular it leads to key recovery attacks if the protocol has the analogous property of the standard special soundness defined in Definition 2.11.

We now prove the opposite implication, showing that secure digital signatures lead to secure identification schemes.

Lemma 3.2 ($\mathsf{TDS} \implies \mathsf{TId}$)**.** *Under the assumptions of Theorem 3.1, if* TDS *is unforgeable against active chosen-message attacks then* TId *is secure against impersonation under active attacks in the random oracle model.*

The proof is very similar to the previous one and follows the same blueprint.

Proof. Let $\mathcal{I}$ be an impersonator which wins the experiment $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{I}}(\lambda)$ with non-negligible probability, then we build a forger $\mathcal{F}$ which uses $\mathcal{I}$ as a subroutine who wins the experiment $\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ with non-negligible probability.

$\mathcal{F}$ needs to simulate the identification oracle, by interacting with $\mathcal{O}^{\mathsf{H}}_{\mathsf{TDS}}(\cdot)$ and $\mathcal{O}_{\mathsf{H}}(\cdot)$. Notice that, since $\mathcal{I}$ do not have access to a random oracle, $\mathcal{F}$ do not need to simulate it. This makes the proof easier, since it removes the main issue in the proof of Lemma 3.1.

Initialisation $\mathcal{F}$ receives from $\mathcal{C}_{\mathsf{TDS}}$ the public parameters $\mathtt{pp}$ and the public key of the n $\mathcal{F}$ simulates $\mathcal{O}_{\mathsf{Tld}}(\cdot)$ and forwards this information to $\mathcal{I}$.

Training phase $\mathcal{I}$ chooses the set $\mathbb{J}$ of parties it wants to corrupt. $\mathcal{F}$ chooses the same set $\mathbb{J}$ and sends it to $\mathcal{C}_{\mathsf{TDS}}$, receiving the secret keys of the players in $\mathbb{J}$. Finally $\mathcal{F}$ forwards this information to $\mathcal{I}$. This simulate perfectly the first three line of $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{Tld},\mathcal{I}}(\lambda)$ in Figure 3.2.

Now we need to show how $\mathcal{F}$ can simulate the identification queries.

To start an identification query $\mathcal{I}$ chooses $\mathbb{J}_h$, the sets of honest players who participates in the threshold identification protocol alongside players in $\mathbb{J}$. As before we can suppose that all the parties in $\mathbb{J}$ takes part in the protocol. $\mathcal{F}$ generates a random message $\mathtt{m}$ and performs a sign query to $\mathcal{O}^{\mathsf{H}}_{\mathsf{TDS}}(\mathtt{m}, \mathbb{J}, \mathbb{J}_h)$.

In the same way of the previous proof, $\mathcal{F}$ acts as "man-in-the-middle " during the execution of $\mathsf{TP}_{\mathtt{cmt}}$, forwarding every message received by $\mathcal{O}^{\mathsf{H}}_{\mathsf{TDS}}$ to $\mathcal{I}$ and vice versa. When it comes the time for $\mathcal{F}$ to send the challenge $\mathtt{ch}$ to $\mathcal{I}$, $\mathcal{F}$ queries $\mathcal{O}_{\mathsf{H}}(\cdot)$ on $(\mathtt{m}||\mathtt{cmt})$ and obtains $\mathtt{ch}$ which forwards to $\mathcal{I}$. Since it is the first time that $\mathcal{F}$ queries the random oracle on $(\mathtt{m}||\mathtt{cmt})$, $\mathtt{ch}$ is uniformly random and thus $\mathcal{F}$ correctly simulates the oracle $\mathcal{O}_{\mathsf{Tld}}$. Finally as with the protocol $\mathsf{TP}_{\mathtt{cmt}}$, $\mathcal{F}$ acts as a man in the middle in the execution of $\mathsf{TP}_{\mathtt{rsp}}$ between $\mathcal{I}$ and $\mathcal{O}^{\mathsf{H}}_{\mathsf{TDS}}(\cdot)$.

Simulation failure The simulation never fails because $\mathcal{F}$ always receives new random challenges from $\mathcal{O}_{\mathsf{H}}$ since it provides $\mathcal{O}_{\mathsf{H}}$ always with different inputs obtained by increasing $\mathtt{m}$ every time it performs a new sign query.

Exploit of $\mathcal{I}$'s impersonation When $\mathcal{I}$ produces its impersonation attempt, it sends to $\mathcal{F}$ a commitment $\mathtt{cmt}^*$ as if it were produced by executing $\mathsf{TP}_{\mathtt{cmt}}$. Then $\mathcal{F}$ starts preparing its forgery by sending to $\mathcal{O}_{\mathsf{H}}$ an hash query with input $(\mathtt{m}^*||\mathtt{cmt}^*)$ fresh new $\mathtt{m}^*$ that has never been used before and that will be the message that will be signed in the forgery. The oracle $\mathcal{O}_{\mathsf{H}}$ returns to $\mathcal{F}$ the challenge $\mathtt{ch}^*$ that $\mathcal{F}$ sends to $\mathcal{I}$ correctly simulating the transcript oracle $\mathcal{O}_{\mathsf{Tld}}$ in the generation of a random challenge. When $\mathcal{I}$ concludes its impersonation by sending the response $\mathtt{rsp}^*$, $\mathcal{F}$ use the same $\mathtt{rsp}^*$ to produce a forgery. It is easy to see that $\mathcal{F}$ wins with the same probability of $\mathcal{I}$, leading to a contraddiction, since TDS is unforgeable. □

The proofs of Lemma 3.1 and Lemma 3.2 prove Theorem 3.1.

3.3.2 Passive security

In this section we present the security result which considers passive adversaries. The ideas behind the proofs in this case are similar to the ones proposed in the active case, therefore a sketch

Theorem 3.2 (Passive security result)**.** *Let* TId *be a canonical threshold identification scheme, with* $\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$ *and let* $\mathsf{TDS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TSign}, \mathsf{Ver})$ *be the associated signature scheme as per Definition 3.7. Then the following implications hold:*

(i) $(\mathsf{TId} \implies \mathsf{TDS})$*: if* TId *is secure against impersonation under passive attacks and* $\mathsf{TP}_{\mathtt{cmt}}$ *is unpredictable as per Definition 3.2, then* TDS *is secure against active chosen-message attacks.*

(ii) $(\mathsf{TDS} \implies \mathsf{TId})$*: if* TDS *is secure against impersonation under passive attacks, then* TId *is secure against active chosen-message attacks.*

As before, we show the two implications separately:

Lemma 3.3 $((\mathsf{TId} \implies \mathsf{TDS}))$**.** *Under the assumptions of Theorem 3.2, if* TId *is secure against impersonation under passive attacks and* $\mathsf{TP}_{\mathtt{cmt}}$ *is unpredictable as for Definition 3.2, then* TDS *is secure against passive chosen-message attacks.*

Proof Sketch. We assume that there exists a forger $\mathcal{F}$ with non-negligible advantage in winning the $\mathsf{Exp}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},\mathcal{F}}$. Without loss of generality we require that $\mathcal{F}$ satisfies the following properties, as it was required in Lemma 3.1:

- all of its hash queries have the form $\mathtt{cmt}||\mathtt{m}$ with $\mathtt{cmt}, \mathtt{m} \in \{0,1\}^*$;
- before outputting a forgery $(\mathtt{m}, \mathtt{cmt}||\mathtt{rsp})$, $\mathcal{F}$ has performed an hash query for $(\mathtt{cmt}||\mathtt{m})$;
- if $\mathcal{F}$ outputs $(\mathtt{m}, \mathtt{cmt}||\mathtt{rsp})$, $\mathtt{m}$ was never a sign query.

Now we show how to define the impersonator $\mathcal{I}$ starting from the forger $\mathcal{F}$. $\mathcal{I}$ will act as a "man-in-the-middle" between $\mathcal{F}$ and the challenger $\mathcal{C}_{\mathsf{TId}}$. In particular it forwards the initial message containing the public parameters and the public keys of the parties received by $\mathcal{C}_{\mathsf{TId}}$ to $\mathcal{F}$. Then, when $\mathcal{F}$ decides the set $\mathbb{J}$ to corrupt during the experiment, and during each sign query the set $\mathbb{J}_h$ of honest parties

who contribute, $\mathcal{I}$ makes the same choices. Now $\mathcal{I}$ needs to simulate the sign query and the hash query. To do so, $\mathcal{F}$ initialise an empty hash table HT and:

- when $\mathcal{F}$ performs an hash query with input $x \in \{0,1\}^*$, $\mathcal{I}$ returns $\mathsf{HT}[x]$ if it is defined, otherwise it returns a random value and saves it in $\mathsf{HT}[x]$ for all but one query. In that specific query for $x = \mathtt{cmt}^*||m^*$, the $\mathtt{fp}$-th query, where $\mathtt{fp}$ is randomly selected in $[q_h(\lambda)]$ at the beginning of the experiment, $\mathcal{F}$ forwards $\mathtt{cmt}^*$ to $\mathcal{C}_{\mathsf{TId}}$ as part of the impersonation attempt of $\mathsf{Exp}^{\mathsf{p-imp}}_{\mathsf{TId},\mathcal{I}}(\lambda)$ and get a challenge $\mathtt{ch}^*$. It then returns $\mathtt{ch}^*$ to $\mathcal{F}$ and updates HT setting $\mathsf{HT}[\mathtt{cmt}^*||m^*] = \mathtt{ch}^*$.
- When $\mathcal{F}$ performs a sign query for $\mathtt{m}$ and $\mathbb{J}_h$, $\mathcal{I}$ queries the oracle $\mathcal{O}_{\mathtt{view-TId}}$ who provides it with a transcript of an identification scheme execution performed by the parties in $\mathbb{J} \cup \mathbb{J}_h$. Being $\mathtt{cmt}$ and $\mathtt{ch}$ the commitment and the challenge included in the transcript, $\mathcal{I}$ updates the hash table HT setting $\mathsf{HT}[\mathtt{cmt}||\mathtt{m}] = \mathtt{ch}$ and forwards it to $\mathcal{F}$.

The simulation may fail if $\mathcal{I}$ must overwrite the hash table HT but this happens with negligible probability being $\mathsf{TP}_{\mathtt{cmt}}$ unpredictable, therefore the commitments are generated with super-logarithmic min-entropy. After at most q_h hash queries and q_s sign queries, $\mathcal{F}$ will eventually output a forgery $(\widehat{\mathtt{cmt}}, \widehat{\mathtt{rsp}})$ of $\widehat{\mathtt{m}}$. If $\mathcal{F}$ successfully produce a forgery and $\mathcal{I}$ correctly guessed the hash query corresponding to it, i.e. $\mathtt{cmt}^* = \widehat{\mathtt{cmt}}$ and $\mathtt{m}^* = \widehat{\mathtt{m}}$, $\mathcal{I}$ also wins the impersonation game.

□

Observation 3.3. Note that, since the transcript oracle generates honest transcripts, where all the parties involved behave honestly, we would not necessarily need an unpredictable $\mathsf{TP}_{\mathtt{cmt}}$, which guarantees a sufficiently random output if at least one party is honest, but it would be enough a $\mathsf{TP}_{\mathtt{cmt}}$ that returns a random output when all the parties involved are honest (e.g. the toy $\mathsf{TP}_{\mathtt{cmt}}$ described in Observation 3.2 could be suitable for a passive secure signature).

Lemma 3.4 ($(\mathsf{TDS} \implies \mathsf{TId})$)**.** *Under the assumptions of Theorem 3.2, if* TDS *is secure against active chosen-message attacks, then* TId *is secure against impersonation under active attacks in the random oracle model.*

Proof Sketch. Let $\mathcal{I}$ be an impersonator which wins the experiment $\mathsf{Exp}^{\mathsf{p-imp}}_{\mathsf{TId},\mathcal{I}}(\lambda)$ with non-negligible probability, then we build a forger $\mathcal{F}$ which uses $\mathcal{I}$ as a subroutine who wins the experiment $\mathsf{Exp}^{\mathsf{p-euf-cma}}_{\mathsf{TDS},F}(\lambda)$ with non-neligible probability.

Initialisation. $\mathcal{F}$ interacts with $\mathcal{O}_{\texttt{view-TDS}}(\cdot)$ and $\mathcal{O}_{\mathsf{H}}(\cdot)$ who provides it with the public parameters $\texttt{pp}$ and the public key of the n parties among which $t-1$ can be corrupted by $\mathcal{F}$. $\mathcal{F}$ simulates $\mathcal{O}_{\texttt{view-TId}}(\cdot)$ and forwards these information to $\mathcal{I}$.

Training phase. The impersonator $\mathcal{I}$ selects the set $\mathbb{J}$ of parties it wants to corrupt and sends it to $\mathcal{F}$, who makes the same choice and sends it to $\mathcal{O}_{\texttt{view-TDS}}(\cdot)$. The oracle sends to $\mathcal{F}$ the secret keys of the parties in $\mathbb{J}$, and $\mathcal{F}$ forwards it to $\mathcal{I}$ who can start the training phase in which it asks $\mathcal{F}$ for transcripts of the identification scheme executed with the parties in $\mathbb{J}_h \subset [n] \setminus \mathbb{J}$ such that $|\mathbb{J} \cup \mathbb{J}_h| = t$. $\mathcal{F}$ simulates the oracle $\mathcal{O}_{\texttt{view-TId}}(\texttt{pk}, \texttt{pp})$ by querying, for each identification transcript query, a digital signature query to $\mathcal{O}_{\texttt{view-TDS}}(\cdot)$ for message $\texttt{m} \in \{0,1\}^{\lambda}$. The oracle $\mathcal{O}_{\texttt{view-TDS}}(\texttt{pk}, \texttt{pp})$ answers with a signature of $\texttt{m}$ together with the public messages exchanged between the parties in $\mathbb{J}$ and $\mathbb{J}_h$ and the state of the parties in $\mathbb{J}$, the ones corrupted by $\mathcal{F}$. $\mathcal{F}$ forwards the messages received from $\mathcal{O}_{\texttt{view-TDS}}(\texttt{pk}, \texttt{pp})$ which are indistinguishable from a real execution of the threshold identification scheme since, according to Definition 3.7 the creation of $\texttt{cmt}$ is exactly the same as in the associated canonical identification scheme, the challenge is the output of a random oracle on input $(\texttt{cmt}||\texttt{m})$ which is a random element, and the response is again computed as in the canonical identification scheme. $\mathcal{F}$ every time it must provide $\mathcal{I}$ with a new identification transcript must query the sign oracle with a new sign query, every time for a different message. One way to do this is the following: $\mathcal{F}$ can treat the message $\texttt{m}$ used in the sign query by $\mathcal{F}$ as an element in $\mathbb{Z}_{2^{\lambda}}$ and for new identification transcript queries performed by $\mathcal{I}$, $\mathcal{F}$ always updates $\texttt{m}$ setting $\texttt{m} \leftarrow \texttt{m} + 1$. Therefore, for each transcript query from $\mathcal{I}$, $\mathcal{F}$ will provide it with a transcript with challenge which is the output of the random oracle $\mathcal{O}_{\mathsf{H}}(\cdot)$ always on distinct inputs.

Exploit of $\mathcal{I}$'s impersonation. When $\mathcal{I}$ starts its impersonation attempt, it sends a commitment $\texttt{cmt}^*$. $\mathcal{F}$ computes a fresh new $\texttt{m}^*$ and sends a random oracle query to $\mathcal{O}_{\mathsf{H}}(\cdot)$ with input $\texttt{cmt}^*||\texttt{m}^*$ receiving $\texttt{ch}^*$. $\mathcal{F}$ sends $\texttt{ch}^*$ to $\mathcal{I}$, correctly simulating the oracle $\mathcal{O}_{\texttt{view-TId}}(\cdot)$ being $\texttt{ch}^*$ the output of a random oracle of an input that has never been queried before. $\mathcal{I}$ generates a valid response $\texttt{rsp}^*$ and $\mathcal{F}$ use it to generate its forgery $(\texttt{cmt}^*||\texttt{rsp}^*)$ to the message $\texttt{m}$ which has never been queried in a sign query. Whenever the impersonator succeeds in its impersonation attempt, also $\mathcal{F}$ succeeds and creates a valid forgery.

□

The proofs of Lemma 3.3 and Lemma 3.4 prove Theorem Theorem 3.2.

3.3.3 UC Security

Introduced by R. Canetti in [Can01], Universal Composability (UC) is a widely used framework for the design and analysis of protocols due to the very strong security guarantees it provides. In particular, a protocol that is UC secure maintains its security properties when run together with other protocols and allows for both parallel and sequential composition. With regards of threshold digital signature, different UC security definitions are used, in particular we can distinguish a stronger definition, that essentially states that a threshold signature is a UC secure MPC protocol that outputs a signature [Lin22]. This means that the distribution of output signatures must be the same as the distribution output by the centralised (non-threshold) signing algorithm, except for whatever bias the adversary can introduce by aborting. On the other hand, a weaker definition is often used, designing a threshold signature functionality that models both signing and verification. There is no requirement that the threshold signature algorithm should produce the same distribution as the centralised one. Instead, it is only required not to allow forgeries [Can+20], in the same way as in the centralised definition [Can04].

In this work we tackle for the first time the problem of adapting the Fiat-Shamir Transform to a distributed setting, which already requires defining several new cryptographic protocols. For this reason, we favour a more straightforward approach both in terms of security definitions and security proofs. In particular our security analysis is game based, as the ones in [Abd+02; BR06], and provides security guarantees about a specific property, namely the unforgeability.

It is worth noticing that, under particular hypothesis, the proof of Theorem 3.1 does not require any rewinding, which suggests that it should be easy to adapt our proof in the UC setting, proving the weaker version of UC secure signature [Can+20]. Proving our approach secure in the stronger version would require more work and a completely different approach, since it would require to consider the centralised version of the distributed protocol and compare its output distribution and the one of the distributed protocol, while we focused on the distributed protocol on its own.

3.4 Threshold Sigma Protocols

In the same way we defined canonical identification schemes by adding a key generation algorithm to sigma protocols, we can adapt Definition 3.1 to define threshold sigma protocol by considering only $\mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}$ and V. Formally we have

Definition 3.8 (Sigma protocol)**.** Let $\mathbb{L} \subseteq \mathbb{W} \times \mathbb{Y}$ be a relation. A (t, n)-*passive (active) threshold sigma protocol* Σ for $\mathbb{L}$ is defined by the tuple

$$\Sigma_{\mathbb{L}} = (\mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V}$$

where the algorithms $\mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V}$ are defined as for threshold identification schemes (Definition 3.1) satisfying the following properties:

- **Completeness:** if a set $\mathbb{J}$ of players, with $|\mathbb{J}| \geq t$ follows the protocol on input $(\{w_i\}_{\mathbb{J}}, y)$, with $\{w_i\}$ being valid shares of a witness w such that $(w, y) \in \mathbb{L}$, then verifier accepts with overwhelming probability.
- **Special Soundness:** there exists an efficient deterministic algorithm $\mathcal{E}$, called extractor, with the following property: whenever $\mathcal{E}$ is given as input a statement $y \in \mathbb{Y}$, two accepting conversation $(\mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$ and $(\mathtt{cmt}, \mathtt{ch}', \mathtt{rsp}')$, with $\mathtt{ch} \neq \mathtt{ch}'$, $\mathcal{E}$ outputs $w \in \mathbb{W}$ such that $(w, y) \in \mathbb{L}$.
- **Passive (Active) Zero Knowledge**, see below Definition 3.9 and Definition 3.10.

Definition 3.9 (Passive Zero Knowledge)**.** Let $\mathcal{S}$ be an efficient probabilistic algorithm, called simulator that takes as input $y \in \mathbb{Y}$, a random challenge $\mathtt{ch}$ and a set $\mathbb{J}$ of parties with $|\mathbb{J}| < t$ as well as their shares $\{w_i\}_{\mathbb{J}}$.

A threshold sigma protocol is *passive zero knowledge* if, for any set of parties $\mathbb{J}_{\mathcal{S}}$ such that $|\mathbb{J}| + |\mathbb{J}_{\mathcal{S}}| \geq t$ and $\mathbb{J}_{\mathcal{S}} \cap \mathbb{J} = \emptyset$, $\mathcal{S}$ can generate $(\mathtt{cmt}, \mathtt{rsp})$ and a transcript Π for all messages exchanged in the execution of $\mathsf{TP}_{\mathtt{cmt}}$ and $\mathsf{TP}_{\mathtt{rsp}}$ by parties in $\mathbb{J} \cup \mathbb{J}_{\mathcal{S}}$, as well as the internal state of the parties in $\mathbb{J}$ such that:

- $(\mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$ form an accepting conversation for y;
- for all $(w, y) \in \mathbb{L}$, $(\mathtt{cmt}, \mathtt{rsp}, \Pi) \xleftarrow{\$} \mathcal{S}(y, \mathtt{ch}, \{w_i\}_{i \in \mathbb{J}})$ has the same distribution as that of transcript of a conversation between the parties in $\mathbb{J} \cup \mathbb{J}_{\mathcal{S}}$ acting honestly.

Definition 3.10 (Active Zero Knowledge)**.** Let $\mathcal{S}$ be an efficient probabilistic algorithm, called simulator that takes as input $y \in \mathbb{Y}$, a random challenge $\mathtt{ch}$ and a set $\mathbb{J}$ of parties with $|\mathbb{J}| < t$ as well as their shares $\{w_i\}_{\mathbb{J}}$.

A threshold sigma protocol is *active zero knowledge* if $\mathcal{S}$, controlling any set of parties $\mathbb{J}_{\mathcal{S}}$ such that $|\mathbb{J}| + |\mathbb{J}_{\mathcal{S}}| \geq t$ and $\mathbb{J}_{\mathcal{S}} \cap \mathbb{J} = \emptyset$, can interact with an adversary $\mathcal{A}$ controlling the parties in $\mathbb{J}$ executing $\mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}$ producing $(\mathtt{cmt}, \mathtt{rsp})$ and transcript Π for all the messages sent by party in $\mathbb{J}_{\mathcal{S}}$ to party in $\mathbb{J}$ such that

- if $\mathcal{A}$ acts honestly, $(\mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$ is an accepting conversation for y.
- for all $(w, y) \in \mathbb{L}$, the following two distributions

$$(\mathtt{cmt}, \mathtt{rsp}, \Pi) \xleftarrow{\$} \mathcal{S}^{\mathcal{A}}(y, \mathtt{ch}, \{w_i\}_{i \in \mathbb{J}})$$

$$(\mathtt{cmt}_h, \mathtt{rsp}_h, \Pi_h) \xleftarrow{\$} \mathcal{A}^{\{P_i\}_{\mathbb{J}_{\mathcal{S}}}}(\mathtt{ch}, \{w_i\}_{i \in \mathbb{J}})$$

  are indistinguishable, where $\mathcal{A}^{\{P_i\}_{\mathbb{J}_{\mathcal{S}}}}$ denotes a real execution between the adversary $\mathcal{A}$ and honest parties in $\mathbb{J}_{\mathcal{S}}$, with challenge $\mathtt{ch}$, while $\mathcal{S}^{\mathcal{A}}$ denotes the simulation performed by $\mathcal{S}$ and $\mathcal{A}$.

Informally speaking, passive zero knowledge requires the existence of a simulator that, having received the challenge $\mathtt{ch}$ in advance, is able to produce accepting conversations between all the parties in $\mathbb{J} \cup \mathbb{J}_{\mathcal{S}}$, knowing only the secret shares of those in $\mathbb{J}$, thus simulating the execution for those in $\mathbb{J}_{\mathcal{S}}$. The idea is that having access to accepting transcripts does not reveal any information about the parties involved, either from the point of view of the verifier or from the point of view of the other parties involved.

For the case of active zero knowledge, the key difference is that $\mathcal{S}$ is not allowed to compute the transcript by itself but instead it needs to to simulate a real execution of the protocol, controlling parties in $\mathbb{J}_{\mathcal{S}}$, interacting with an adversary, controlling parties in $\mathbb{J}$. The idea is that participating in executions of the protocol does not leak any information about the private input to the other participants. Notice that in this case we do not require $(\mathtt{cmt}, \mathtt{ch}, \mathtt{rsp})$ to be always an accepting conversation. Indeed, if this would be the case, the simulator should produce accepting conversation even when the adversary is malicious and sends wrong data. This would badly formalise the concept of "the simulator simulates honest participants" , since it would have more power (it is clearly impossible for an honest participant to force correct execution when interacting with malicious adversary). Moreover this would clash with the second conditions, indeed when the adversary sends wrong data the two distributions are different: in the first one $\mathtt{rsp}$ is an accepting response, in the second one it is not.

Theorem 3.3. *Let* $\Sigma = (\mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$ *be a* $(t, n)-$ *active (passive) threshold Sigma protocol for a relation* $\mathbb{L} \subseteq \mathbb{W} \times \mathbb{Y}$ *with super-polynomial challenge space* $\mathbb{C}$*. Let*

$$\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$$

be the threshold identification scheme obtained by equipping Σ *with a setup protocol* Setup *and a key generation algorithm* KeyGen*. If* KeyGen *is one-way, then the* TId *is secure against active (passive) impersonator attack.*

We show only the active case, the passive case can be done in the same way.

Proof. We want to show that if there exists an impersonator $\mathcal{I}$ able to win the $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{I}}$ game, then it is possible to build an attacker $\mathcal{S}$ that is able to win the $\mathsf{Exp}^{\mathsf{ow}}_{\mathsf{KeyGen},\mathcal{S}}$ game.

Firstly, $\mathcal{S}$ receives a challenge $y \in \mathbb{Y}$, having the goal of finding $w \in \mathbb{W}$ such that $(w, y) \in \mathbb{L}$. $\mathcal{S}$ sets y as the public key $\mathtt{pk}$ for the $\mathsf{Exp}^{\mathsf{a-imp}}_{\mathsf{TId},\mathcal{I}}$ and sends it to $\mathcal{I}$, who answers with the set $\mathbb{J}$ of participants it desires to corrupt.

Then $\mathcal{S}$ sends to $\mathcal{I}$ random shares w_i to simulate the secret sharing of $w' \in \mathbb{W}$ such that $(w', y) \in \mathbb{L}$. By the security property of Definition 2.18, this is indistinguishable from an execution of a real secret sharing, since $\mathcal{I}$ controls less than t parties. Then $\mathcal{S}$ needs to simulate the oracle $\mathcal{O}_{\mathsf{TId}}(\cdot)$ for $\mathcal{I}$. This is possible thanks to the active zero knowledge property of Σ.

$\mathcal{I}$ will eventually perform a successful impersonation. At this point $\mathcal{S}$ rewinds $\mathcal{I}$ and changes the challenge sent. Since the challenge space $\mathbb{C}$ is super-polynomial, with non-negligible probability this yields the two required accepting conversation (by the Forking Lemma, see Section 2.4), thus $\mathcal{S}$ can use the extractor $\mathcal{E}$ from the special soundness to extract a witness w, breaking the one-way assumption on the key generation. □

Theorem 3.4. *Let* $\Sigma = (\mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$ *be a* $(t, n)-$ *threshold sigma protocol for a relation* $\mathbb{L} \subseteq \mathbb{W} \times \mathbb{Y}$*, with super-polynomial challenge space* $\mathbb{C}$*. Let*

$$\mathsf{TId} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TP}_{\mathtt{cmt}}, \mathsf{TP}_{\mathtt{rsp}}, \mathsf{V})$$

be the threshold identification scheme obtained by equipping Σ *with a setup protocol* Setup *and a key generation algorithm* KeyGen*. If* KeyGen *is one-way, then* $\mathsf{TP}_{\mathtt{cmt}}$ *is unpredictable as per Definition 3.2.*

We show the proof only for the passive case, the active one follows immediately as well.

Proof. Suppose that the output $\mathtt{cmt}$ of $\mathsf{TP}_{\mathtt{cmt}}$ has low min-entropy even if at least one of the parties involved in the execution behaves honestly. Since TId has Passive Zero Knowledge, there exists a simulator $\mathcal{S}$ that outputs accepting conversations having the same distribution as real conversations. By querying a polynomial number $q(\lambda)$ of time $\mathcal{S}(\cdot, \mathtt{ch}_i, \cdot)$, $i \in [q(\lambda)]$ with $\mathtt{ch}_i \neq \mathtt{ch}_j \ \forall i \neq j$, with non-negligible probability we obtain two transcripts with the same $\mathtt{cmt}^*$. By the special soundness property it is possible to retrieve the secret key w. □

3.5 Conclusions and future works

Although threshold signature schemes have been known for some time and are more popular than ever, the concept of threshold identification schemes has received very little attention. In particular, previous works focused their attention on protocols that do not allow communication between provers, relying either on some pre-computation or on the presence of a trusted third party (the combiner).

In this chapter we propose a new definition for threshold identification schemes, with the aim of capturing their multi-party nature. We model our definition to mimic the traditional structure of threshold signature schemes in order to establish a link between the two worlds, thanks to a generalised version of the Fiat-Shamir transform.

Next, following the footprint of M. Abdalla et al. in [Abd+02], we show that threshold signatures obtained by applying the Fiat-Shamir transform are secure, under similar conditions of the centralised case.

Finally, we turn our attention to threshold sigma protocols and their connection to threshold identification schemes. Similar to the centralised case, we define properties of the sigma protocols which, if satisfied, guarantee that the associated identification schemes are secure.

In the next chapter we show the power of our definitions and results, showing an easy proof for a threshold Schnorr Signature.

Future Works Our approach could streamline the security analysis of many threshold signatures, however it covers only static corruptions, where the adversary decide which party to corrupt at the beginning of the protocol. While this is a

relevant security notion, often used as in [Bat+23b; Lin22; Chu+23], many protocols are also proved secure in the adaptive case, where the adversary can, at any time, corrupt parties and learn their state [CKM23]. It would be interesting to extend our analysis to the adaptive case. The structure of the proof of Theorem 3.1 suggests that if a threshold identification scheme is secure against adaptive adversaries (this can be done by adding an additional oracle $\mathcal{O}_{\texttt{Corrupt}}$ that can be adaptively called to learn honest parties input) also the derived threshold signature scheme is secure against adaptive attacks. In this case, the real challenge would be to define properties on the threshold sigma protocol, in the same vein of the zero knowledge properties, to achieve the adaptive security of the threshold identification scheme.

Chapter 4

Decentralized Secret Sharing and Threshold Signatures

4.1 Introduction

In Chapter 3 we propose a framework for designing threshold signature schemes, focusing on the signature generation algorithm. To complete the work, the key generation algorithm should also be considered. In particular, it is desirable to have distributed key generation instead of relying on a trusted third party.

A common way to achieve a fully decentralised system is to use a variant of the Shamir secret sharing scheme, where the dealer is not a single authority, such as the scheme described in [Ped92, Section 5.2], but is a subset of the participants.

In this chapter we present a completely decentralised *extensible and verifiable* secret sharing scheme based on Shamir's, and we enhance it with the possibility of having *offline participants*, firstly introduced in [Bat+22b]. In particular, the proposed protocol allows for the addition of new parties after the initial secret sharing, a property that can be useful to increase the resilience of the secret reconstruction, providing additional protection against the loss of shares.

Extensible secret sharing schemes can be seen as a generalization of repairable secret sharing schemes, that instead allow a participant that lost its share to retrieve it with the help of other parties. In particular our protocol is a generalization of the Enrolment Protocol presented in [SW18], that we enhanced with the verifiability property, i.e. the added participant is able to verify that the received share is correct.

Lastly, we use show a potential application for our scheme, presenting a $(t, n)-$threshold Schnorr Signature, that use it during the Key Generation. The resulting key generation is similar to the one presented in [Gen+07a], however our signature is secure for an arbitrary t instead of requiring $t \leq \frac{n}{2}$, thanks also to the usage of additional ZKPs. Moreover, we also show that the resulting signature can support the addition of an arbitrary number of new participants, thanks to the extensible property of the Secret Sharing Scheme used. We prove the signature unforgeable as per Definition 2.16, using the heuristic proposed in the previous chapter.

Notice that the proposed Secret Sharing Scheme is suitable algorithm for performing the Key Generation in many Discrete Logarithm based threshold signature, such as ECDSA or EdDSA [Bat+22b; Bat+23b], however we decided to focus our attention on the Schnorr's one, due to the increasing interest in this field. Our approach is similar to [BDN18], [Max+19] and [Nic+03]. However, these three signatures work only in the (n, n) case, while ours works works for an arbitrary threshold. More recently a general (t, n) Schnorr Signature was proposed, FROST, however their assumptions are not classical, while we only rely on classical ones. Lastly, concurrently with this work, Sparkle [CKM23] was proposed, that require only standard assumptions in the static case and it is very similar to our work. The two works were made independently, we discuss the small difference between them in Section 4.5.1.

The results of this chapter are contained in [BLM22], that is currently under review for WCC2024 and was presented at CANS2022. The signature presented here is a generalisation of [Bat+22a] presented during at DLT2022. It is immediate to apply the heuristic here to the ECDSA and the EdDSA algorithm presented in [Bat+23b] and [Bat+22c], published at the beginning of the PhD.

4.2 Preliminaries

4.2.1 From MDS Codes to Secret Sharing

Let $\mathbb{F}_q$ be the finite field with q elements and let α be an agreed-upon primitive element of $\mathbb{F}_q$. Let $\{p^{(i)}\}_{i=1,\dots,\tau} \subseteq \mathbb{F}_q[x]$ be a set of τ polynomials of degree $t-1$, so $p^{(i)} = \sum_{k=0}^{t-1} p_k^{(i)} x^k$, where $p_k^{(i)} \in \mathbb{F}_q$ is the k-th coefficient of the polynomial $p^{(i)}$.

Let $p = \sum_{i=1}^{\tau} p^{(i)}$, with coefficients $p_k = \sum_{i=1}^{\tau} p_k^{(i)}$ for $k = 0, \dots, t-1$, and define $\beta_j = p(\alpha^j)$. Note that, if we define $\beta_{i,j} = p^{(i)}(\alpha^j)$ for $i \in \{1, \dots, \tau\}$ and $j \in \{1, \dots, q-1\}$, then we have that $\beta_j = \sum_{i=1}^{\tau} \beta_{i,j}$.

Definition 4.1. Let $\mathbb{J} = [j_1, \ldots, j_n]$ be a list of $1 \leq n \leq q-1$ distinct integers in $\{1, \ldots, q-1\}$. We define $G_{\mathbb{J}}$ as the $(t \times n)$ matrix:

$$G_{\mathbb{J}} = \left[\alpha^{j \cdot k}\right]_{k \in \{0,\ldots,t-1\},\, j \in \mathbb{J}}$$

If $n = 1$ then $\mathbb{J} = [j]$ and we sometimes simply use G_j instead of $G_{[j]}$.

Lemma 4.1. *For any $t \leq n \leq q-1$ and for any $\mathbb{J} = [j_1, \ldots, j_n]$, the matrix $G_{\mathbb{J}}$ is the generator matrix of a punctured $[n, t]_q$ Reed-Solomon code. In particular:*

- *$G_{\mathbb{J}}$ has maximum rank for any $\mathbb{J} = [j_1, \ldots, j_n]$;*
- *if $n = t$ then $G_{\mathbb{J}}$ is invertible;*
- *if $n = q-1$ then $G_{\mathbb{J}}$ is a standard generator matrix (given as a Vandermonde matrix) of a $[q-1, t]_q$ Reed-Solomon code.*

Lemma 4.1 summarises the properties of the matrix defined in Definition 4.1 and the link with Reed-Solomon codes [RS60]. An interested reader can refer to [Rot06] for a comprehensive introduction to Coding Theory with a focus on Reed-Solomon codes and algebraic codes. We remark that the link with Reed-Solomon codes derives from the matrix in Definition 4.1.

An alternative and more general approach would be to use any $t \times n$ matrix with coefficients in $\mathbb{F}_q$. In this case Lemma 4.1 would become a summary of the required properties that the matrix should satisfy in order to achieve similar results. In particular, we remark that it is possible to substitute our definition with the one of Extended Generalised Reed-Solomon codes, a choice that would allow a broader set of acceptable parameters (e.g. in Definition 4.1 n can be at most $q+1$ instead of $q-1$). We focus however on Vandermonde matrices to exploit the link between Reed-Solomon codes and the classical version of Shamir's Secret Sharing Scheme [MS81].

Now we show that, since p has degree at most $t-1$, given a list $\mathbb{J} \subseteq \{1, \ldots, q-1\}$ of cardinality at least t, with the list of evaluations $[\beta_j]_{j \in \mathbb{J}}$ it is possible to interpolate the polynomial p. That is, the coefficients p_k can be reconstructed and therefore the evaluation $p(\gamma)$ in any element $\gamma \in \mathbb{F}_q$ can be computed.

Proposition 4.1. *Let $\mathbb{J} = [j_1, \ldots, j_t]$ be a list of t distinct integers in $\{1, \ldots, n\}$, and let $G_{\mathbb{J}}$ be the square matrix constructed as in Definition 4.1. Then:*

$$(p_0, \ldots, p_{t-1}) = (\beta_{j_1}, \ldots, \beta_{j_t}) \cdot G_{\mathbb{J}}^{-1}.$$

Proof. For any $j \in \{1, \ldots, q\}$ we have that $\beta_j = p(\alpha^j) = \sum_{k=0}^{t-1} p_k \cdot (\alpha^j)^k = (p_0, \ldots, p_{t-1})G_j$, thus:

$$(p_0, \ldots, p_{t-1}) \cdot G_{\mathbb{J}} = (\beta_{j_1}, \ldots, \beta_{j_t}). \tag{4.2.1}$$

By Lemma 4.1, since $\mathbb{J}$ has cardinality t, then $G_{\mathbb{J}}$ is invertible, so we can multiply both sides of Equation (4.2.1) by $G_{\mathbb{J}}^{-1}$ and conclude our proof. □

Proposition 4.2. *Let h be any integer in $\{1, \ldots, n\}$, let $\mathbb{J} = [j_1, \ldots, j_t]$ be a list of t distinct integers in $\{1, \ldots, n\}$, and let e_ℓ be the ℓ-th element of the standard basis of $\mathbb{F}_q^t$. Then:*

$$\beta_h = \sum_{\ell=1}^{t} f(\beta_{j_\ell}, h, \mathbb{J}, \ell),$$

where for any $\ell \in \{1, \ldots, t\}$ we define the function f as:

$$f(x, h, \mathbb{J}, \ell) = x \cdot e_\ell G_{\mathbb{J}}^{-1} G_h. \tag{4.2.2}$$

Proof. Observe that $e_\ell \cdot G_{\mathbb{J}}^{-1}$ is the ℓ-th row of $G_{\mathbb{J}}^{-1}$. By linearity, from Proposition 4.1 we have:

$$(p_0, \ldots, p_{t-1}) = \sum_{\ell=1}^{t} \beta_{j_\ell} e_\ell \cdot G_{\mathbb{J}}^{-1}.$$

So:

$$\sum_{\ell=1}^{t} f(\beta_{j_\ell}, h, \mathbb{J}, \ell) = \sum_{\ell=1}^{t} \beta_{j_\ell} e_\ell G_{\mathbb{J}}^{-1} G_h = (p_0, \ldots, p_{t-1})G_h = \beta_h,$$

as shown in the proof of Proposition 4.1. □

Observation 4.1. An interesting consequence of Proposition 4.2 is that t distinct shares are sufficient to compute any other share. However, observe that it is possible to obtain β_{j_ℓ} from $f(\beta_{j_\ell}, h, \mathbb{J}, \ell)$, since both $G_{\mathbb{J}}$ and G_h can be easily computed even without knowing anything about the polynomials. This means that Proposition 4.2 should not be used directly to distribute new shares of a secret, in order to preserve the privacy of the old shares.

A simple workaround is to split these secret values. Let $b_{h,\mathbb{J},\ell,k}$ be chosen at random in $\mathbb{F}_q$ for $k \in \{1, \ldots, t\} \setminus \{\ell\}$, and set $b_{h,\mathbb{J},\ell,\ell} = f(\beta_{j_\ell}, h, \mathbb{J}, \ell) - \sum_{k=1, k\neq\ell}^{t} b_{h,\mathbb{J},\ell,k}$. If we define $b_{h,\mathbb{J},k} = \sum_{\ell=1}^{t} b_{h,\mathbb{J},\ell,k}$, then we have that:

$$\sum_{k=1}^{t} b_{h,\mathbb{J},k} = \sum_{k=1}^{t} \left(\sum_{\ell=1}^{t} b_{h,\mathbb{J},\ell,k} \right) = \sum_{\ell=1}^{t} \left(\sum_{k=1}^{t} b_{h,\mathbb{J},\ell,k} \right) = \sum_{\ell=1}^{t} f(\beta_{j_\ell}, h, \mathbb{J}, \ell) = \beta_h \tag{4.2.3}$$

Note that the random values are completely canceled out only when summing all the $b_{h,\mathbb{J},k}$, this means that the values $\beta_{j\ell}$ remain hidden, so this is a safe way to generate new shares.

4.2.2 Homomorphic Commitment

For our Extensible Decentralised Verifiable Secret Sharing Scheme, described in Section 4.3, we need a homomorphic commitment, that is a commitment HCom for which the following properties hold for all $m_0, m_1, z_0, z_1, \gamma \in \mathbb{F}_q$:

$$\mathsf{HCom}(m_0; z_0) \cdot \mathsf{HCom}(m_1; z_1) = \mathsf{HCom}(m_0 + m_1; z_0 + z_1),$$
$$\mathsf{HCom}(m_0; z_0)^\gamma = \mathsf{HCom}(\gamma \cdot m_0; \gamma \cdot z_0).$$

The Pedersen commitment [Ped92], based on the difficulty of the discrete logarithm, is a perfectly hiding homomorphic commitment scheme which works as follows:

Setup let $\mathbb{G}$ be a group of prime order q where the DLOG problem is hard (for the binding property to hold), and g, h be random generators of $\mathbb{G}$, then the message space of the commitment scheme is $\mathbb{Z}_q$, the randomiser space is $\mathbb{Z}_q$ and the commitment space is $\mathbb{G}$;

Commitment to commit to $m \in \mathbb{Z}_q$ using the randomiser $z \in \mathbb{Z}_q$, the committer computes $C = \mathsf{HCom}(m, z) = g^m \cdot h^z$;

Verification the decommitment is the pair (m, z), and $\mathsf{Ver}(C, m, z)$ simply outputs m if $C = g^m \cdot h^z$, $\perp$ otherwise.

4.3 Extensible Decentralised Verifiable Secret Sharing Protocol

In this section we give a description of our decentralised variant of the verifiable secret sharing scheme (VSSS) by Pedersen [Ped92], which includes the feature of adding new users.

Let $P_1, \ldots, P_n$ be n parties participating in the secret sharing scheme, and let $t \leq n$ be the chosen threshold.

We assume that q is big enough that, given n polynomials of degree d sampled uniformly at random from $\mathbb{F}_q[x]$, the probability of their sum to be of degree $d' < d$

$\mathsf{SecGen}(\mathsf{pp})$

1 : $p^{(i)} \xleftarrow{\$} \mathbb{F}_q[x], \deg(p^{(i)}) = t-1$

2 : $z^{(i)} \xleftarrow{\$} \mathbb{F}_q[x], \deg(z^{(i)}) = t-1$

3 : Publish $\mathsf{C}_{0,i,k} := \mathsf{HCom}(p_k^{(i)}; z_k^{(i)})$

4 : Sends to P_j $\beta_{i,j} := p^{(i)}(\alpha^j),\ \gamma_{i,j} := z^{(i)}(\alpha^j)$

5 : // To all $j \in [n]$

6 : **if** $\mathsf{HCom}(\beta_{j,i}; \gamma_{j,i}) \neq \prod_{k=0}^{t-1} (\mathsf{C}_{0,j,k})^{(\alpha^i)^k}$ **then**

7 : **return** $\perp$

8 : $\beta_i := \sum_{j=1}^{\tau} \beta_{j,i}; \quad \gamma_i := \sum_{j=1}^{\tau} \gamma_{j,i}$

9 : **return** β_i, γ_i

Figure 4.1: Secret Generation algorithm

is negligible. Finally, we use a homomorphic commitment HCom as defined in Section 4.2.2.

4.3.1 Secret Generation

First, all the parties agree on some public parameters pp, namely a prime q defining a finite fields $\mathbb{F}_q$, a primitive element α and the parameters of HCom (e.g. $\mathbb{G}$ and the generators g, h for Pedersen commitment of Section 4.2.2)

The distributed secret generation algorithm is carried out by $\tau \leq n$ parties, WLOG we can assume $\{P_1, \ldots, P_\tau\}$.

The detailed algorithm is presented in Figure 4.1. First each party chooses two random polynomials $p^{(i)}$ and $z^{(i)}$, implicitly defining its own secret as $p^{(i)}(0)$. Then, each party publishes homorphic commitments to all the coefficient, using the polynomial $z^{(i)}$ as randomness. Finally, each party distributes its own secret, sending $\beta_{i,j} = p^{(i)}(\alpha^j)$ and $\gamma_{i,j} := z^{(i)}(\alpha^j)$. If the received value are consistent with the commitment published each party can obtain its private share β_i for the common secret $\sum_{i=0}^{\tau} p^{(i)}(0)$.

We remark that the τ parties involved in the secret generation algorithm are always capable of determining the secret p_0, regardless of the value t. We have two possible cases:

- $\tau \geq t$: in this case we have a standard (t, n)-VSSS, and no group of less than

t parties can work together to reconstruct the secret;

- $\tau < t$: in this case, $P_1, \ldots, P_\tau$ can reconstruct the secret, no matter what t is.

In particular, observe that if $\tau = 1$ then our protocol behaves as a VSSS in which the shares are created and distributed by a centralised authority.

4.3.2 Secret Reconstruction

If $\mathbb{J} \subseteq \{1, \ldots, q\}$ is a set of t distinct indexes, then with the vector of shares $(\beta_j)_{j \in \mathbb{J}}$ it is possible to reconstruct the secret p_0 as follows:

$$p_0 = (\beta_j)_{j \in \mathbb{J}} \cdot G_{\mathbb{J}}^{-1} \cdot e_1^T,$$

which is a direct consequence of Proposition 4.1. Let $\ell \in \{1, \ldots, t\}$ be the position of j inside the list $\mathbb{J}$, note that the Shamir share β_j can be converted into an additive share ω_j:

$$\begin{aligned} \omega_j &= \beta_j e_\ell \cdot G_{\mathbb{J}}^{-1} \cdot e_1^T, \\ p_0 &= \sum_{j \in \mathbb{J}} \omega_j. \end{aligned} \tag{4.3.1}$$

4.3.3 Addition of New Parties

Let $\mathbb{J}$ be a set of t parties. Figure 4.2 how can they add a new party P_{n+1} (i.e. generate its share β_{n+1}). Initially each party P_i generates a (t, t)-additive secret sharing of its own secret value $f(\beta_i, n+1, \mathbb{J}, i)$, as described in Proposition 4.2 and in the Observation 4.1 below. This values allows for the computation of the new share β_{n+1} of P_{n+1}. The correctness of the computation of the additive secret sharing is then checked in Lines 6,8 and 11. If all the checks are correct, the values are then sent to P_{n+1}.

P_{n+1} retrieves its share as: $\beta_{n+1} = \sum_{j=1}^{t} b_{n+1,\mathbb{J},j}$, and the checking value as: $\gamma_{n+1} = \sum_{j=1}^{t} z_{n+1,\mathbb{J},j}$. Then it checks their consistency with the commitments by verifying:

$$\mathsf{HCom}\left(b_{n+1,\mathbb{J},j}; z_{n+1,\mathbb{J},j}\right) \stackrel{?}{=} \prod_{k=1}^{t} \mathsf{C}_{n+1,\mathbb{J},k,j}, \tag{4.3.2}$$

for $j \in \{1, \ldots, t\}$.

$\mathsf{PAdd}(\mathbb{J}, \{\beta_i\}_{\mathbb{J}}, \{\gamma_i\}_{\mathbb{J}})$

1 : $b_{n+1,\mathbb{J},i,j}, z_{n+1,\mathbb{J},i,j} \xleftarrow{\$} \mathbb{F}_q,\ k \in [t], j \neq i$

2 : $b_{n+1,\mathbb{J},i,i} := f(\beta_i, n+1, \mathbb{J}, i) - \sum_{j=1, j\neq i}^{t} b_{n+1,\mathbb{J},i,j}$

3 : $z_{n+1,\mathbb{J},i,i} := f(\gamma_i, n+1, \mathbb{J}, i) - \sum_{j=1, j\neq i}^{t} z_{n+1,\mathbb{J},i,j}$

4 : // f as per Equation (4.2.2)

5 : Publish $\mathsf{C}_{n+1,\mathbb{J},i,j} = \mathsf{HCom}\left(b_{n+1,\mathbb{J},i,j}; z_{n+1,\mathbb{J},i,j}\right)$

6 : **if** $\prod_{j=1}^{t} \mathsf{C}_{n+1,\mathbb{J},i,j} \neq \left(\prod_{j=0}^{t-1} \left(\prod_{k=1}^{\tau} \mathsf{C}_{0,k,j}\right)^{(\alpha^i)^j}\right)^{e_i G_{\mathbb{J}}^{-1} G_{n+1}}$ **then**

7 : **return** $\perp$

8 : **if** $\prod_{j=1}^{t} \prod_{i=1}^{t} \mathsf{C}_{n+1,\mathbb{J},i,j} \neq \prod_{j=0}^{t-1} \left(\prod_{k=1}^{\tau} \mathsf{C}_{0,k,j}\right)^{(\alpha^{n+1})^j}$ **then**

9 : **return** $\perp$

10 : Sends to P_j $b_{n+1,\mathbb{J},i,j}$ and $z_{n+1,\mathbb{J},i,j}$

11 : **if** $\mathsf{HCom}\left(b_{n+1,\mathbb{J},j,i}; z_{n+1,\mathbb{J},j,i}\right) \neq \mathsf{C}_{n+1,\mathbb{J},k,i}$ **then**

12 : **return** $\perp$

13 : $b_{n+1,\mathbb{J},i} := \sum_{j=1}^{t} b_{n+1,\mathbb{J},j,i} \quad z_{n+1,\mathbb{J},i} := \sum_{j=1}^{t} z_{n+1,\mathbb{J},j,i}$

14 : Sends to P_{n+1} $b_{n+1,\mathbb{J},i}$ and $z_{n+1,\mathbb{J},i}$

15 : **return** 0

Figure 4.2: Algorithm for the addition of new parties

Note that at the end of the procedure, P_{n+1} has its own secret values just like the other parties, so it can participate in the secret reconstruction or the addition of further parties.

4.3.4 Security of the Secret Sharing

In this section we prove the correctness and security of the secret sharing scheme described in Section 4.3.1, reducing it to the correctness and security of a centralised version, which are a direct consequence of the binding and hiding properties of the commitment scheme. For the correctness we refer to Definition 4.1 of [Ped92] which includes the verifiability, for the security we refer to Definition 2.18.

We start by defining a centralised version of the protocol:

Definition 4.2 (centralised Secret Sharing)**.** The centralised version of the scheme described in Section 4.3.1 between a dealer $\mathcal{D}$ and players $P_1, \dots, P_n$ with threshold t of a secret $s \in \mathbb{F}_q$ proceeds as follows:

(i) $\mathcal{D}$ chooses two random polynomials $p, z \in \mathbb{F}_q[x]$ of degree $t-1$ such that $p_0 = s$;

(ii) $\mathcal{D}$ computes and publishes $\mathsf{C}_k = \mathsf{HCom}(p_k, z_k)$ for $k = 0, \dots, t-1$;

(iii) $\mathcal{D}$ sends $\beta_j = p(\alpha^j)$ and $\gamma_j = z(\alpha^j)$ to P_j;

(iv) each P_j checks that their share is correct by verifying:

$$\mathsf{HCom}(\beta_j, \gamma_j) \stackrel{?}{=} \prod_{k=0}^{t-1} \mathsf{C}_k^{(\alpha^j)^k} \tag{4.3.3}$$

The secret s can be reconstructed as usual by interpolating $\{\beta_j\}_{j \in \mathbb{J}}$ where $\mathbb{J}$ is a set of at least t indexes.

Lemma 4.2 (Correctness)**.** *If* HCom *is binding then the secret sharing scheme of Definition 4.2 is correct. If* HCom *is perfectly binding then the secret sharing scheme of Definition 4.2 is correct even if* $\mathcal{D}$ *has unbounded computational power.*

Proof. For the homomorphic properties of HCom when the dealer is honest then Equation (4.3.3) holds so an honest P_j accepts.

In order to make the players reconstruct a different secret s', there should be a player P_j such that $\mathcal{D}$ has sent to P_j a wrong share $\beta_j \neq p(\alpha^j)$. P_j accepts β_j only if Equation (4.3.3) holds. this means $\mathcal{D}$ was able to find γ_j such that $\mathsf{HCom}(\beta_j, \gamma_j) = \mathsf{HCom}(p(\alpha^j), z(\alpha^j))$, contradicting the binding property of HCom. □

The correctness of the decentralised protocol follows trivially, since it comprise τ parallel and independent execution of the centralised protocol, each of them having a different dealer P_i.

Lemma 4.3 (Security)**.** *If* HCom *is hiding then the secret sharing scheme of Definition 4.2 is secure. If* HCom *is perfectly hiding then the secret sharing scheme of Definition 4.2 is secure even if the adversary has unbounded computational power.*

Proof. To prove that an adversary with the views of up to $t-1$ players does not gain any information about the secret s, we prove that, for any $s' \neq s$ this adversary cannot distinguish a view of the sharing of s from a view of the sharing of s'. To achieve this, we prove that the existence of an adversary that has more than negligible advantage in winning the game defined below breaks the hiding property of HCom.

$\mathsf{Exp}^{\mathsf{ss-dist}}_{\mathsf{SecGen},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}()$
2 : $s_0, s_1 \xleftarrow{\$} \mathbb{F}_q$
3 : $b \xleftarrow{\$} \{0,1\}$
4 : $\mathbb{J} \leftarrow \mathcal{A}((s_b, s_{1-b}))$ // $|\mathbb{J}| \leq t-1$
5 : $p^{(0)}, p^{(1)}, z^{(0)}, z^{(1)} \xleftarrow{\$} \mathbb{F}_q[x]$
6 : // $p^{(i)}(0) = s_i$ and $p^{(0)}(\alpha^j) = p^{(1)}(\alpha^j)$ and $z^{(0)}(\alpha^j) = z^{(1)}(\alpha^j)$ for all $j \in \mathbb{J}$
7 : $\mathsf{C}_k^{(0)} := \mathsf{HCom}(p_k^{(0)}, z_k^{(0)})$ $\mathsf{C}_k^{(1)} := \mathsf{HCom}(p_k^{(1)}, z_k^{(1)})$
8 : $b' \leftarrow \mathcal{A}(\{\mathsf{C}_k^{(0)}\}, \{\mathsf{C}_k^{(1)}\}, \{p^{(0)}(\alpha^j)\}_{\mathbb{J}}, \{z^{(0)}(\alpha^j)\}_{\mathbb{J}})$
9 : **return** $b = b'$

The advantage of $\mathcal{A}$ is

$$\mathbf{Adv}^{\mathsf{ss-dist}}_{\mathsf{SecGen},\mathcal{A}}(\lambda) = \Pr(b' = 1 | b = 1) - \Pr(b' = 1 | b = 0).$$

Let $\mathcal{A}$ be an adversary that has a non-negligible advantage ε in winning the game $\mathsf{Exp}^{\mathsf{ss-dist}}_{\mathsf{SecGen},\mathcal{A}}(\lambda)$, then we can violate the hiding property of HCom as defined

in Definition 2.10 by simulating a game for $\mathcal{A}$. Notice that we use a slightly modified game, where instead of receiving only $\mathtt{cmt} = \mathsf{Com}(\mathtt{m}_b; R)$, the adversary receive both $\mathsf{Com}(\mathtt{m}_b; R)$ and $\mathsf{Com}(\mathtt{m}_{1-b}; R)$ and need to guess the order. The simulator $\mathcal{S}$ sets $s_0 = \mathtt{m}_0$ and $s_1 = \mathtt{m}_1$, $\mathtt{C}_0 = \mathtt{cmt}$, $\mathtt{C}_1 = \mathtt{cmt}_1$, and chooses randomly $\beta_j, \gamma_j \in \mathbb{F}_q$ for $j \in \mathbb{J}$ (WLOG we can suppose that $|\mathbb{J}| = t - 1$). During the simulation, $\mathcal{S}$ force β_j, γ_j to be the evaluation respectively of $p^{(0)}$ (and thus $p^{(1)}$) and $z^{(0)}$ (and thus $z^{(1)}$) on α^j, for all j Note that if we implicitly define $p^{(0)}, p^{(1)} \in \mathbb{F}_q[x]$ as the polynomials such that $p^{(0)}(0) = \mathtt{m}_0$, $p^{(1)}(0) = \mathtt{m}_1$, $p^{(0)}(\alpha^j) = p^{(1)}(\alpha^j) = \beta_j$, then $\exists \lambda_{j,k}$ for $k = 1, \ldots, t-1$ such that $p_k = \lambda_{0,k} \cdot \mathtt{m}_0 + \sum_{j \in \mathbb{J}'} \lambda_{j,k} \cdot \beta_j$ and $p'_k = \lambda_{0,k} \cdot \mathtt{m}_1 + \sum_{j \in \mathbb{J}'} \lambda_{j,k} \cdot \beta_j$. So, for all $k = 1, \ldots, t-1$ if we define:

$$\mathtt{C}_k = \mathtt{C}_0^{\lambda_{0,k}} \cdot \prod_{j \in \mathbb{J}'} \mathsf{HCom}(\beta_j, \gamma_j)^{\lambda_{j,k}}, \quad \mathtt{C}_k^1 = \mathtt{C}_0^{1\,\lambda_{0,k}} \cdot \prod_{j \in \mathbb{J}'} \mathsf{HCom}(\beta_j, \gamma_j)^{\lambda_{j,k}}$$

then we implicitly set $\gamma_j = z^{(0)}(\alpha^j) = z^{(1)}(\alpha^j)$ where the coefficients of the polynomials $z^{(0)}, z^{(1)} \in \mathbb{F}_q[x]$ are defined as the coefficients of p, p' using γ_j instead of β_j, r_1 instead of m_1 and r_2 instead of m_2. This means that the Equation (4.3.3) holds for both sets of commitments $\{\mathtt{C}_0, \ldots, \mathtt{C}_{t-1}\}$, $\{\mathtt{C}_0^1, \ldots, \mathtt{C}_{t-1}^1\}$. So $\mathcal{S}$ can simulate the game by sending to $\mathcal{A}$ the ordered pair (s_0, s_1), then if $\mathcal{A}$ answers with 0 the simulator guesses that $s_0 = \mathtt{m}_0$ and $s_1 = \mathtt{m}_1$, otherwise the simulator guesses the reverse order. Note that the simulation is perfect so we have the same non-negligible advantage ε in breaking the hiding property. □

Theorem 4.1. *If* HCom *is hiding, then the secret sharing scheme described in Section 4.3.1 is secure.*

Proof. For the sake of simplicity we suppose that $\tau = t$ but the same proof can be adapted for an arbitrary τ.

Since HCom is hiding, then the secret sharing scheme of Definition 4.2 is secure.

Let us suppose that the adversary controls $P_2, ..., P_t$. We show that after the Secret Generation (Section 4.3.1) it has no information about the secret p_0.

First of all, notice that $p_0^{(1)}$ is uniformly distributed, thus p_0 is uniformly distributed as well.

Then notice also that steps 1 to 6 are t independent executions of the Verifiable Secret Sharing scheme described in Definition 4.2 with n participants and threshold t, each having as dealer a different P_i, $i = 1, ..., t$, thus the adversary does not gain

any information about $p_0^{(1)}$, the secret of the honest player. Moreover the last step does not involve any new message exchange, thus does not reveal anything.

Hence, the adversary has no information about p_0.

□

Now we need to prove the security of the Addition of New Parties. Informally, we need to show that an adversary controlling at most $t-1$ participants is not able to learn anything about the secret of the other parties or the secret itself. More formally we have the following definition (WLOG we suppose that the parties involved are $P_1, ..., P_t$ and P_{n+1}):

Definition 4.3. Let $S \subseteq \{1, ..., t, n+1\}$ be a set such that $|S| = t-1$ and $\texttt{view}_S$ be the set of all the messages that parties in S see. Then

$$\Pr(P_i \text{ has secret } \omega_i | \texttt{view}_S) = \Pr(P_i \text{ has secret } \omega_i)$$

for $i \notin S$. Moreover

$$\Pr(\text{The shared secret is } p_0 | \texttt{view}_S) = \Pr(\text{The shared secret is } p_0).$$

Theorem 4.2. *If* HCom *is hiding, then the Addition of New Parties described in Section 4.3.3 is secure.*

Proof. Initially we suppose that the adversary does not control P_{n+1}, but only $t-1$ out of the t parties which perform the protocol to add P_{n+1}. WLOG we can suppose that these parties are $P_1, \ldots, P_t$ and that the adversary controls $P_2, \ldots, P_t$.

Since HCom is hiding, then the secret sharing scheme of Definition 4.2 is secure.

We can notice that Lines 1, 2 and 3 are a (t,t) additive secret sharing of $f(\beta_1, n+1, \mathbb{J}, 1)$, with dealer P_1, verified with a homomorphic commitment. This is secure and does not leak any information about β_1 or β_{n+1}.

The following steps do not require any additional computation or communication involving the secret $b_{n+1,\mathbb{J},1,1}$, so the security is trivial.

Now we need to deal with the case of the adversary controlling P_{n+1} and $t-2$ among $P_1, \ldots, P_t$. WLOG we can suppose that the adversary controls $P_3, \ldots, P_t$.

The same considerations as before hold for lines 1,2 and 3. However, now the adversary is also able to learn $b_{n+1,\mathbb{J},1}$ and $b_{n+1,\mathbb{J},2}$ in the last line. In the computation of each $b_{n+1,\mathbb{J},1}$ and $b_{n+1,\mathbb{J},2}$ there are two unknown and uniformly distributed

addends, that is the adversary learns $b_{n+1,\mathcal{J},1,1} + b_{n+1,\mathcal{J},2,1}$ and $b_{n+1,\mathcal{J},1,2} + b_{n+1,\mathcal{J},2,2}$, but these sums give no information on the addends, so the adversary is not able to learn anything more. □

Theorem 4.3. *If* HCom *is binding, then the Addition of New Parties described in Section 4.3.3 is robust, i.e. an adversary controlling at most* $t-1$ *parties is not able to corrupt the protocol without being noticed.*

Proof. Suppose that the adversary controls $P_2, ..., P_t$. To prevent the correct execution of the protocol the adversary could send wrong data either during Lines 10 or 14.

In the first case a cheating behaviour is caught thanks to the check in Lines 6 and 8 unless the adversary is able to produce $\tilde{b}_{n+1,\mathcal{J},j,1} \neq b_{n+1,\mathcal{J},j,1}$ and $\tilde{z}_{n+1,\mathcal{J},j,1} \neq z_{n+1,\mathcal{J},j,1}$ such that $\mathsf{HCom}(\tilde{b}_{n+1,\mathcal{J},j,1}, \tilde{z}_{n+1,\mathcal{J},j,1}) = \mathsf{HCom}(b_{n+1,\mathcal{J},j,1}, z_{n+1,\mathcal{J},j,1})$. This is impossible due to the binding property of HCom.

In the second case a cheating behaviour is caught in the same way thanks to Equation (4.3.2).

□

4.4 Threshold Schnorr Signature

In this section we describe a possible use case of our extensible secret sharing scheme and the framework presented in Chapter 3: a (t, n)-threshold variant of Schnorr's digital signature algorithm with offline participants.

In particular, we need to design a secure key generation and a secure threshold identification protocol, such that, after applying the Fiat-Shamir transform, we obtain a threshold version of Schnorr signature.

We require that at least $\tau \geq t$ users are online for the setup, in the following we suppose there are exactly $\tau = t$ online parties in the key generation phase, namely $P_1, \ldots, P_t$.

4.4.1 Setup and Key Generation

All the parties need to agree on a group $\mathbb{G}$ of prime order q with generator g where the DLOG problem is assumed to be hard. Note that this means that the

KeyGen(pp):

1 : $y_i \xleftarrow{\$} \mathbb{Z}_q$

2 : $Y_i := g^{y_i}$

3 : $(\mathtt{C}_i, \mathtt{D}_i) \leftarrow \mathsf{Com}(Y_i)$

4 : Publishes $\mathtt{C}_i$

5 : Publishes $\mathtt{D}_i$

6 : $Y := \prod_{i=1}^{t} Y_i$

7 : Use Schnorr protocol to prove the knowledge of y_i

8 : Perform the distributed secret sharing of Section 4.3 for the secret y_i

9 : $w_i := \sum_{j=1}^{t} \beta_{i,j}$

10 : // $\beta_{i,j}$ is the share received by P_i from P_j during the secret sharing

11 : **return** Y, w_i

Figure 4.3: Key-generation algorithm for the threshold Schnorr Signature

field $\mathbb{F}_q$ is isomorphic to the ring $\mathbb{Z}_q$. Moreover the hardness of DLOG implies that the size of q is exponential in the security parameter, thus any practical application necessarily has a number of users $n \ll q$. They also need to agree on a commitment scheme Com and an hash function $\mathsf{H}_{\mathsf{com}}$.

Figure 4.3 shows the key-generation algorithm.

We now show that the protocol is secure in the presence of a malicious adversary, as per Section 2.8. We suppose that the adversary corrupt $P_2, ..., P_t$. In particular, we need to show that there exist a simulator $\mathcal{S}$ that, on input a public key Y_c received from the trusted third party, is able to interact with the adversary $\mathcal{A}$ of the real protocol of Figure 4.3, forcing the execution to end with output Y_c.

In the simulation we need an equivocal commitment scheme, i.e. a commitment scheme having a secret trapdoor, known only by $\mathcal{S}$, that allows to easy violate the binding properties (equivocate). In case of the Pedersen commitment scheme described in Section 4.2.2, knowing the discrete logarithm k of $h = g^k$ allows to equivocate the commitment.

(i) $\mathcal{S}$ randomly chooses $y_1 \in \mathbb{Z}_q$ and follows the protocol normally until line 7;

(ii) during the Schnorr proofs, $\mathcal{S}$ rewind the adversary to extract $y_2, ..., y_n$.

(iii) $\mathcal{S}$ rewinds the adversary to step 5.

(iv) $\mathcal{S}$ computes $\hat{Y} = \frac{Y_c}{\prod_2^t g^{y_i}}$, computes the commitment $(\hat{\mathsf{C}}_1, \hat{\mathsf{D}}_1) = \mathsf{Com}(\hat{Y})$. Implicitly define the discrete logarithm of $\hat{Y}$ as $\hat{y}$

(v) P_i follows the protocol normally. Due to the value chosen, in step 5, $Y = Y_c$.

(vi) $\mathcal{S}$ simulates the Schnorr ZK proof of knowledge of $\hat{a}$, since it does not know this value.

(vii) $\mathcal{S}$ generates uniformly at random $\hat{\beta}_{1,j}$ for $j = 2, \ldots, t$, then simulates a fake Decentralised Secret Sharing Protocol since it cannot compute a polynomial $\hat{p}^{(1)}$ such that $\hat{p}^{(1)}(j) = \hat{\beta}_{1,j}$ and $\hat{p}^{(1)}(0) = \hat{y}$:

  (i) similarly as in the proof of Lemma 4.3, $\exists \lambda_{j,k}$ for $k = 1, \ldots, t-1$ such that $\hat{p}_k^{(1)} = \lambda_{0,k} \cdot \hat{a} + \sum_{j=2}^{t} \lambda_{j,k} \cdot \hat{\beta}_{1,j}$;

  (ii) $\mathcal{S}$ sets the commitment $\hat{\mathsf{C}}_{0,1,0}$ of Line 3 of Figure 4.1 as the commitment to a random value (for the hiding property of HCom this is indistinguishable to a real commitment since it does not need to be opened)[1]

  (iii) $\mathcal{S}$ computes the commitments to the other coefficients of its unknown secret polynomial $\hat{p}^{(1)}$ as:

$$\hat{\mathsf{C}}_{0,1,k} = (\hat{\mathsf{C}}_{0,1,0})^{\lambda_{0,k}} \cdot \prod_{j=2}^{t} \mathsf{HCom}(\hat{\beta}_{1,j}, \hat{\gamma}_{1,j})^{\lambda_{j,k}}$$

  where $\hat{\gamma}_{1,j} \in \mathbb{Z}_q$ for $j = 2, \ldots, t$ are randomly chosen.

  (iv) note that by interpolating at the exponent $\hat{Y}$ and $g^{\hat{\beta}_{1,2}}, \ldots, g^{\hat{\beta}_{1,t}}$, $\mathcal{S}$ is able to compute $g^{\hat{\beta}_{1,1}}$ where $\hat{\beta}_{1,1}$ is implicitly defined as $\hat{p}^{(1)}(1)$;

(viii) $\mathcal{S}$ sends $\hat{\beta}_{1,j}, \hat{\gamma}_{1,j}$ to P_j.

The proof of the correctness of the simulation is stated in the following lemmas. The proofs are trivial and use the same argument of the one presented in [Bat+22c].

Lemma 4.4. *If the Decisional Diffie-Hellman assumption holds, then the simulation terminates in expected polynomial time and is indistinguishable from the real protocol.*

[1] Note that if HCom is Pedersen's commitment [Ped92], then $\mathcal{S}$ can compute a real commitment as $\mathsf{HCom}(\hat{y}, z_0^{(1)}) = \hat{Y} \cdot h^{z_0^{(1)}}$, where $z_0^{(1)} \in \mathbb{Z}_q$ is chosen randomly.

Proof. Let ϵ be the (non negliglible) probability that the adversary correctly decommits in step 5. Due to the Forking Lemma the rewinding is performed at most a polynomial number of times. The only difference from the real protocol is that $\mathcal{S}$ does not know the discrete logarithm of $\hat{Y}$ and so it performs a fake verification protocol. However, this is indistinguishable due to the honest verifier zero knowledge property. □

Lemma 4.5. *For a polynomial number of inputs the simulation terminates with output Y_c except with negligible probability.*

Proof. This is because of the binding property of the commitment scheme: if $\mathcal{A}$ correctly decommits twice it must do so to the same string, no matter what P_1 decommits (except with negligible probability). Because of the construction of $\hat{Y}$, the output is Y_c. □

Thus the key generation is secure.

4.4.2 Threshold identification protocol

We now proceed with the identification protocol. The idea is to design a threshold version of the standard Schnorr protocol.

Theorem 4.4. *If the discrete logarithm is hard in $\mathbb{G}$, then the threshold identification protocol of Figure 4.4 is secure against impersonation under active attacks.*

Proof. Our goal is to use Theorem 3.3.

We start by proving that the threshold sigma protocol is special sound and then we prove the active and passive zero knowledge property.

- **Special soundness.** The special soundness property is trivial and follows immediately from the special soundness of the standard Schnorr protocol [Sch91].

  Indeed, suppose to have two accepting transcripts $(R, \mathtt{ch}, z)$ and $(R, \mathtt{ch}', z')$ with $\mathtt{ch} \neq \mathtt{ch}'$. Then it would be possible to compute the discrete logarithm of $\mathtt{pk} = y$ by simply computing $w = (z - z')(\mathtt{ch} - \mathtt{ch}')^{-1}$.

- **Active zero knowledge.** To prove that the protocol is active zero knowledge, we must show that it can be simulated by a simulator $\mathcal{S}$ taking in input

$\mathsf{TP}^{\mathsf{Com}}_{\mathtt{cmt}}(\mathtt{ssid}, \{w_i\}_{i\in\mathbb{S}}; \mathbf{R}) \to \{\mathtt{cmt}_i\}_{i\in\mathbb{S}}$

1 : $r_i \xleftarrow{\$} \mathbb{Z}_p$
2 : $R_i \leftarrow g^r$
3 : $\mathtt{cmt}_i \leftarrow \mathsf{H}_{\mathsf{com}}(\mathtt{ssid}, \mathbb{S}, R_i)$
4 : **return** $\mathtt{cmt}_i$

$\mathsf{TP}_{\mathtt{rsp}}(\{w_i\}_{i\in\mathbb{S}}, \mathtt{cmt}, \mathtt{ch}) \to (\mathtt{rsp})$

1 : $z_i \leftarrow r_i + \mathtt{ch}(\lambda_i x_i)$
2 : // λ_i is the Lagrange
3 : // coefficient of i w.r.t. $\mathbb{S}$
4 : Party P_i sends z_i
5 : $z \leftarrow \sum_{i\in\mathbb{S}} z_i$
6 : **return** $(R, z) \leftarrow \sigma$

$\mathsf{TP}^{\mathsf{Decom}}_{\mathtt{cmt}}(\mathtt{ssid}, \{w_i\}_{i\in\mathbb{S}}, \{\mathtt{cmt}_i\}_{i\in\mathbb{S}}) \to \mathtt{cmt}$

1 : Party P_i sends R_i
2 : **if** $\mathtt{cmt}_j \neq \mathsf{H}_{\mathsf{com}}(\mathtt{ssid}, \mathbb{S}, R_j), j \in \mathbb{S}$
3 : **return** $\perp$
4 : $R = \prod_{i\in\mathbb{S}} R_i$
5 : **return** $\mathtt{cmt} \leftarrow R$
6 :

$V(y, \sigma) \to 0/1$

1 : Parse $(R, z) \leftarrow \sigma$
2 : **if** $RY^{\mathtt{ch}} = g^z$ **then**
3 : **return** accept
4 : **return** reject

Figure 4.4: Threshold sigma protocol for Sparkle.

$(y = g^w, \mathtt{ch}^*)$ and the $t-1$ shares of the private key controlled by the adversary. Without loss of generality we suppose that the adversary controls $P_1, \ldots, P_{t-1}$ and $w_1, \ldots, w_{t-1}$ are their shares of the witness which are given also to the simulator $\mathcal{S}$ who must impersonate P_t without knowing w_t.

The simulation resembles the simulation of the classical Schnorr protocol.

The simulator $\mathcal{S}$ samples uniformly at random $z_t \in \mathbb{Z}_q$ and defines

$$R_t = g^{z_t} y^{-\mathtt{ch}^*} \prod_{j=1}^{t-1} g^{\lambda_j w_j \mathtt{ch}^*},$$

where $\mathtt{ch}^*$ is the challenge it received in input and λ_j is the Lagrange coefficient of j.

Note that, even if $\mathcal{S}$ does not know w_t, by definition of Shamir secret sharing $w = \sum_{i\in[t]} \lambda_i w_i$ and $y^{-\mathtt{ch}^*} = g^{w(-\mathtt{ch}^*)}$, therefore $y^{-\mathtt{ch}^*} \prod_{j=1}^{t-1} g^{\lambda_j w_j \mathtt{ch}^*} = g^{\lambda_t w_t(-\mathtt{ch}^*)}$, then it holds that $g^{z_t} = R_t g^{\lambda_t w_t \mathtt{ch}^*}$.

This means that the transcript $(R_t, \mathtt{ch}^*, z_t)$ is valid and, being z_t sampled uniformly at random, and R_t being uniquely determined from $(z_t, \mathtt{ch}^*)$, $(R_t, \mathtt{ch}^*, z_t)$ is indistinguishable from an honest transcript (generated starting from R_t).

Finally $\mathcal{S}$ executes $\mathsf{TP}_{\mathtt{cmt}}^{\mathsf{Com}}$ computing $\mathsf{com}_t = \mathsf{H}_{\mathsf{com}}(\mathtt{m}, \mathcal{S}, R_t)$, then it executes $\mathsf{TP}_{\mathtt{cmt}}^{\mathsf{Decom}}$ by revealing R_t. The commitments are aggregated computing R, then the challenge $\mathtt{ch}^*$ will be used as the challenge of the protocol and $\mathcal{S}$ simulates the algorithm $\mathsf{TP}_{\mathtt{rsp}}$ by broadcasting the responses $\mathtt{rsp}_t = z_t$ it sampled randomly at the beginning of the simulation.

Note that the transcripts $(R, \mathtt{ch}^*, z)$, together with the transcript generated by the messages sent by $\mathcal{S}$, form an accepting transcript as long as the other parties act correctly. Also, the transcripts of $\mathcal{S}$ are indistinguishable from a real execution since the messages that $\mathcal{S}$ must send are independent of the messages sent by the adversary who could be potentially malicious. Therefore the sigma protocol is active zero-knowledge according to Definition 3.10.

□

To derive a secure signature we also need to prove that $\mathsf{TP}_{\mathtt{cmt}}$ is secure.

Lemma 4.6. *The threshold identification protocol of Figure 4.4 has commit release* $\mathsf{TP}_{\mathtt{cmt}}$ *as per Definition 3.3.*

Proof. It is immediate to see that $\mathsf{TP}_{\mathtt{cmt}}^{\mathsf{Com}}$ does not require any interaction between the parties and outputs $\mathsf{H}_{\mathsf{com}}(\mathtt{ssid}, \mathbb{S}, R_i)$ that is a one-way commitment as long as $\mathsf{H}_{\mathsf{com}}$ is a secure cryptographic hash function. The function used to reconstruct the commitment $\mathtt{cmt} = R$ is $R = \prod_{i \in \mathbb{S}} R_i$ where the computations are executed in $\mathbb{G}$, therefore if at least one party in $\mathbb{S}$ is honest, the value R will be uniformly distributed in $\mathbb{G}$. Lastly, due to the binding property of $\mathsf{H}_{\mathsf{com}}$, $\mathsf{TP}_{\mathtt{cmt}}^{\mathsf{Decom}}$ is a deterministic protocol. □

By equipping the threshold sigma protocol with the Setup and KeyGen explained before we obtain a threshold identification scheme TId which has a one-way KeyGen, a super-polynomial challenge space and is special sound, active zero-knowledge and passive zero knowledge. Therefore by Theorem 3.3, TId is secure against active and passive impersonation attacks.

4.4.3 Finishing the Proof

Combining Lemma 4.4 and Lemma 4.5, as well as Lemma 4.6 and Theorem 4.4 we obtain a secure signature with distributed key generation, as per Theorem 3.1.

The security of the addition of new user, that follows directly from the security of the Participant Addition Protocol of Section 4.3.3, stated in Theorem 4.2.

4.5 Conclusions and future works

The main result presented in this chapter in the context of decentralised secure protocols is twofold: on one hand we describe and prove the security of a variant of Shamir's Linear Secret Sharing Scheme, on the other hand we develop a (t, n)-threshold variant of the Schnorr Signature Scheme which allows to add participants after the Key Generation.

Our variant of Linear Secret Sharing is based on the link between Shamir's scheme and linear MDS codes and it allows two interesting properties: there is no need for an authority that manages and learns the shares of a common secret, moreover any group of at least t authorised parties can extend the Secret Sharing Scheme by adding new participants who have the same powers as those belonging to the initial set of n parties.

Due to the properties of the decentralised VSSS, the threshold Schnorr signature protocol described in this work is completely decentralised and is proven secure under standard hypotheses (i.e. the centralised Schnorr signature protocol is unforgeable, the commitment schemes are secure, and the Decisional Diffie-Hellman Assumption holds).

Notice that, by adopting an extensible decentralised VSSS, it is possible to modify other existing schemes in order to obtain secure (t, n)-threshold variants in which new participants can be added at any time by a legitimate group of users. In particular, since our VSSS scheme can be seen as a generalization of the techniques adopted in [Bat+22b], it is possible to obtain (t, n) threshold variants of ECDSA.

Finally, we remark that, by further decentralising the role of the parties $P_1, \ldots, P_\tau$, it is possible to design a VSSS with complex access-policies to the secret p_0. However, the attempt of creating a decentralised VSSS generic enough to define any possible access control structure seems to present several difficulties, and the feasibility of this line of research is still under investigation.

4.5.1 Comparison with Concurrent Works

As mentioned in Section 1.1, the core of this chapter is contained in [BLM22], a follow up of [Bat+22a], presented as a poster at CANS2022 and currently under review for WCC2024. During Crypto2023, Crites, Komlo, and Maller proposed Sparkle [CKM23], a new $(t, n)-$threshold Shnorr Signature. The two protocols are very similar and have almost the exact structure.

In [CKM23] there is a deep security analysis focused on adaptive corruption of parties after the key generation. However, a key difference between our and their proofs is that Sparkle's security proof does not allow the adversary to participate in the key generation phase, and thus the adversary is not able to choose its secret key freely. One may see our analysis as covering adversaries that participate in the key generation and Sparkle's analysis as covering adversaries that corrupt parties afterwards, thus the two somewhat complete each other.

We remark that the two works were made independently, roughly at the same time (the original preprint of this work and the first presentation at CANS 2022 both predate the first appearance of Sparkle online) with no communication or plagiarism between the teams.

Chapter 5

Group Action Cryptography

5.1 Introduction

With the threat of quantum computers looming ever closer, the community has moved to develop alternative cryptographic solutions that will be resistant to quantum algorithms, particularly with the NIST call for standardization [NIS17]. While the first standards covering key encapsulation and signatures are about to be drafted, the situation with the latter is not considered fully satisfactory, so NIST has launched an "on-ramp" process to standardise new signature designs [NIS23]. there is a lack of threshold friendly schemes among the current solutions, which is prompting more research in this area and will lead to its own standardisation process [BP].

Group actions are one of the most promising areas in post-quantum cryptography. Long used, often without the "explicit label" , with the discrete logarithm problem, group actions gained traction in post quantum cryptography with isogeny-based schemes, due to the flexibility of group based constructions. Recently, Code Equivalence and other Isomorphism Problems have also been (re)discovered as (non-abelian) group actions suitable for cryptographic use. Non-commutative actions have advantages from a security point of view, since they prevent quantum attacks on commutative group actions, such as Kuperberg's algorithm for the dihedral hidden subgroup problem [Kup13]. However, this significantly limits the possible cryptographic primitives based on them, since, for example, they do not allow for an immediate Diffie-Hellman-like key exchange. This sparked the interest of the community in studying cryptographic group actions in a more general framework. Not only to improve current schemes and understand their limitations, but also to design new functionalities such as threshold signatures.

In [CS19], the (round 2) proposals of the standardization process were analyzed in order to determine ways to define threshold variants, eventually identifying multivariate schemes as the most suitable starting point, with schemes based on the Unbalanced Oil and Vinegar (UOV) framework being the most promising. Even though, from a theoretical point of view, it appears to be indeed possible to obtain a threshold version of UOV by exploiting MPC protocols using Linear Secret Sharing Schemes (LSSS), this approach remains, at the present time, only theoretical.

Notably, threshold signature schemes for cryptographic cyclic group actions have been already discussed in 2020 and applied to isogeny-based schemes [DFM20], where they proposed a way to apply a group actions in a threshold like way by using the classical Shamir Secret sharing on a group action induced by a cyclic group. They showed how to apply this for an El Gamal like encryption schemes and a signature based on Σ-protocols proving their simulatability, however this schemes are only secure in the honest-but-curious model and miss a distributed key generation mechanisms. In [CS20b] they showed a way to combine the use of zero-knowledge proofs and replicated secret sharing to obtain a secure threshold signature scheme from isogeny assumptions. The work is an important step for the research and can be extended to more general group actions, but the main drawbacks are the number of shares necessary to implement replicated secret sharing and the important slow down caused by the additional ZKPs required. In [Beu+21] they showed how to define a distributed key generation algorithm by using a new primitive called *piecewise verifiable proofs*; proving their security in the quantum random oracle model. All previous techniques are then incorporated in [CM22] to have actively secure attributed based encryption and signature schemes, in which threshold signature are a particular case.

In this chapter, we investigate constructions for post-quantum threshold signature schemes using cryptographic group actions as the main building block. Our work takes a black-box approach, making no additional assumptions, such as commutativity, on the group action beyond its security. This allows our frameworks to be instantiated with a wider variety of candidates, such as the code-based signature schemes mentioned above.

The first contribution, is construction for a "full" (n, n)-threshold signature scheme with a distributed key generation mechanism. We then prove its security via a reduction to the original centralised signature, since the approach of Chapter 3 is not suitable due to the round robin structure of the protocol, which is tipical (and in some sense, necessary) when dealing with group action in a black box way [CG23]

Next, we show a (t, n) version of scheme. Since we cannot assume any properties

on the groups (except the security of the group actions), our construction is quite inefficient in terms of memory required, since we have to rely on replicated secret sharing instead of the linear one. Nevertheless, our construction remains practical for certain use cases, especially for low values of t and n, or whenever t and n are close, i.e. with very high or low threshold.

The results of this chapter are contained in [Bat+23a], that is currently accepted at CT-RSA 2024.

5.2 Preliminaries

For all the chapter we use multiplicative notation for groups, that are usually denoted as $\mathbb{G}$.

Definition 5.1 (Group Action)**.** Let $\mathbb{G}$ be a group and $\mathbb{X}$ be a set. A group action is a function $\star$:

$$\begin{aligned} \star : \mathbb{G} \times \mathbb{X} &\to \mathbb{X} \\ (g, x) &\to g \star x \end{aligned} \tag{5.2.1}$$

with the following properties:

- for all $x \in \mathbb{X}$ we have $e \star x = x$;
- for all $g, h \in \mathbb{G}$, it holds that $h \star (g \star x) = (h \cdot g) \star x$.

We indicate it by the tuple $(\mathbb{G}, \mathbb{X}, \star)$ and we say that $\mathbb{G}$ acts on $\mathbb{X}$.

Often, we ask for additional properties, namely:

Definition 5.2. We say that a group action is:

- *Transitive*, if for every $x, y \in \mathbb{X}$, there exists $g \in \mathbb{G}$ such that $y = g \star x$;
- *Faithful*, if $g \in \mathbb{G}$ is such that $x = g \star x$ for all $x \in \mathbb{X}$, then $g = e$;
- *Free*, if $g \in \mathbb{G}$ and there exist $x \in \mathbb{X}$ such that $x = g \star x$, then $g = e$;
- *Regular*, if it is free and transitive.

Definition 5.3. Given a group action $(\mathbb{G}, \mathbb{X}, \star)$ for a set element $x \in \mathbb{X}$ we can define its orbit as

$$\mathbb{O}_x := \{g \star x \mid g \in \mathbb{G}\}$$

and its stabilizer as

$$G_x := \{g \in \mathbb{G} \mid g \star x = x\} \ .$$

For finite groups we have a classical result called Orbit-Stabilizer Theorem that states

$$|G| = |tele\mathbb{O}_x| \cdot |G_x| \quad \forall x \in \mathbb{X} \ .$$

Notably, we can always have a transitive group action by restricting $\mathbb{X}$ to an orbit $\mathbb{O}_x$. If the action is also free we get by the theorem the equality $|G| = |\mathbb{X}|$, that is trivially implied by the bijection $g \mapsto g \star x$. This bijection implies that for any pair x, y of elements in $\mathbb{X}$ there exists one and only one group element g with $g \star x = y$, we label this group element as $\delta_\star(x, y)$.

5.2.1 Group Action in Cryptography

We need additional properties for a group action to be useful for cryptographic tasks. First of all we need to be able to efficiently perform operations on the group $\mathbb{G}$, the set $\mathbb{X}$ and their interactions.

Definition 5.4. A group action $(\mathbb{G}, \mathbb{X}, \star)$ is said effective if:

(i) It is possible to work efficiently on the group $\mathbb{G}$, in particular we can perform efficiently:

  (i) given $g, h \in \mathbb{G}$; compute their product gh;

  (ii) given $g \in \mathbb{G}$, compute the inverse g^{-1};

  (iii) sample an element g from $\mathbb{G}$ from a uniform distribution (or a distribution statistically close to the uniform one;

  (iv) given $g, h \in \mathbb{G}$, compute $g = h$;

  (v) Testing if one string represent a valid group element in $\mathbb{G}$.

(ii) It is possible to verify efficiently that a string corresponds to an element in $\mathbb{X}$ and compute efficiently a unique representation for it;

(iii) There exists at least on element $x_0 \in \mathbb{X}$ we can represent using a finite length string;

(iv) Given any $g \in \mathbb{G}$ and $x \in \mathbb{X}$ we can efficiently compute $g \star x$.

Furthermore, for a group action to be useful for cryptographic purposes, there must be some hard problem associated with it.

The natural problem that arises from group action is the *Group Action Inverse Problem (GAIP)*, sometimes called *vectorization problem*, that basically states that a group action is a one-way function.

Definition 5.5 (GAIP)**.** Let $(\mathbb{G}, \mathbb{X}, \star)$ be an effective group action. Define the following experiment

$\mathsf{Exp}^{\mathsf{GAIP}}_{\mathcal{A}}(\lambda)$:

1 : $x, y \xleftarrow{\$} \mathbb{X}$

2 : $g \xleftarrow{\$} \mathcal{A}((x, y))$

3 : **return** $g \star x = y$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{GAIP}}_{\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{GAIP}}_{\mathcal{A}}(\lambda) = 1)$$

We say that GAIP is difficult in $(\mathbb{G}, \mathbb{X}, \star)$ if and only if $\mathbf{Adv}^{\mathsf{GAIP}}_{\mathcal{A}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

It can also be useful to define the decisional version of Definition 5.6:

Definition 5.6 (d-GAIP)**.** Let $(\mathbb{G}, \mathbb{X}, \star)$ be an effective group action. Define the following experiment

$\mathsf{Exp}^{\mathsf{d-GAIP}}_{\mathcal{A}}(\lambda)$:

1 : $x \xleftarrow{\$} \mathbb{X}$

2 : $b \xleftarrow{\$} \{0, 1\}$

3 : **if** $b = 0$ **then**

4 : $\quad y \xleftarrow{\$} \mathbb{O}_x$

5 : **else**

6 : $\quad y \xleftarrow{\$} \mathbb{X} \setminus \mathbb{O}_x$

7 : $b' \xleftarrow{\$} \mathcal{A}((x, y))$

8 : **return** $b = b'$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{d-GAIP}}(\lambda) = \Pr(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{d-GAIP}}(\lambda) = 1) - \frac{1}{2}$$

We say that d-GAIP is difficult in $(\mathbb{G}, \mathbb{X}, \star)$ if and only if $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{d-GAIP}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

Another related problem is the *parallelization* problem, that is the generalization of the classic computational Diffie-Hellman problem.

Definition 5.7 (c-GAIP)**.** Let $(\mathbb{G}, \mathbb{X}, \star)$ be an effective group action. Define the following experiment

$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{c-GAIP}}(\lambda)$:

1 : $x \xleftarrow{\$} \mathbb{X}$

2 : $g, h \xleftarrow{\$} \mathbb{G}$

3 : $y \xleftarrow{\$} \mathcal{A}((g \star x, h \star x))$

4 : **return** $gh \star x = y$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{c-GAIP}}(\lambda) = \Pr(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{c-GAIP}}(\lambda) = 1)$$

We say that c-GAIP is difficult in $(\mathbb{G}, \mathbb{X}, \star)$ if and only if $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{c-GAIP}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

This problem is particularly interesting if $\mathbb{G}$ is an abelian group, since it trivially imply a key exchange à la Diffie-Hellman. Note that if c-GAIP is hard, then so is GAIP.

It is important to notice that in all the three definitions above, the adversary $\mathcal{A}$ only sees a single pair x, y. In many algorithms, the participants often compute the same action several times over different elements $x_i \in \mathbb{X}$, so the above definitions are not suitable for modelling the security of these algorithms.

In this case, we need to provide to the adversary an oracle $\mathcal{O}_g$ that, on input a set element x, returns $g \star x$. We have the following two definitions:

Definition 5.8 (Weak Unpredictability)**.** Let $(\mathbb{G}, \mathbb{X}, \star)$ be an effective group action. Define the following experiment

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{w-unp}}(\lambda):}$

$x \xleftarrow{\$} \mathbb{X}$

$g \xleftarrow{\$} \mathbb{G}$

$y \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_g}(x)$

return $g \star x = y$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{w-unp}}(\lambda) = \Pr(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{w-unp}}(\lambda) = 1)$$

We say that $(\mathbb{G}, \mathbb{X}, \star)$ is weakly unpredictable if and only if $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{w-unp}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

Definition 5.9 (Weak Pseudorandomness)**.** Let $(\mathbb{G}, \mathbb{X}, \star)$ be an effective group action. Define the following experiment

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{w-prand}}(\lambda):}$

$g \xleftarrow{\$} \mathbb{G}$

$b \xleftarrow{\$} \{0,1\}$

if $b = 0$ **then**

  $\mathcal{O}_g \leftarrow \mathcal{O}_{\mathbb{X}}$

$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_g}()$

return $b' = b$

where $\mathcal{O}_{\mathbb{X}}$ is a Random Oracle in $\mathbb{X}$. Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{w-prand}}(\lambda) = \Pr(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{w-prand}}(\lambda) = 1)$$

We say that $(\mathbb{G}, \mathbb{X}, \star)$ is weakly pseudorandom if and only if $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{w-prand}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

Hard Homogeneous Spaces Especially in isogeny-based cryptography, it is often used the term *Hard Homogeneous Spaces* (HHS), that is an effective group action $(\mathbb{G}, \mathbb{X}, \star)$ where c-GAIP is difficult, as per Definition 5.7.

5.2.2 Examples of Group Actions

The most classic example of a group action is the discrete logarithm, which is unfortunately not secure in a post-quantum setting, since Schorr's algorithm solves it in polynomial time (and thus solves the GAIP).

Isogenies of supersingular elliptic curves Isogenies are one of the most famous and used group actions, since they are the only commutative one. Thanks to that, they allow for an easy construction of a post-quantum Diffie-Hellman like key exchanges. We now briefly explain the CSIDH group action, that was introduced in [Cas+18] and has several example, such as [BKP20; BKV19; CS20b].

Let $\mathbb{F}_p$ be a prime field, with $p \geq 5$. In the following $\mathbb{E}$ and $\mathbb{E}_0$ denote elliptic curves defined over $\mathbb{F}_p$. An isogeny $\phi : \mathbb{E} \to \mathbb{E}_0$ is a non-constant morphism mapping $0_{\mathbb{E}}$ to $0_{\mathbb{E}_0}$. Each coordinate of $\phi(x, y)$ is then the fraction of two polynomials in $\bar{\mathbb{F}}_p[x, y]$, where $\bar{\mathbb{F}}$ denotes the algebraic closure of $\mathbb{F}_p$.

We restrict our attention to separable isogenies between supersingular elliptic curves, i.e. curves $\mathbb{E}$ defined over $\mathbb{F}_p$ whose set of rational points has cardinality $p+1$. The set $\mathrm{End}(\mathbb{E})$ of all endomorphisms of $\mathbb{E}$ that are defined over $\mathbb{F}_p$ together with the zero map form a commutative ring under pointwise addition and composition, which is isomorphic to an order $\mathcal{O}$ of the quadratic field $\mathbb{K} = \mathbb{Q}(\sqrt{-p})$. A fractional ideal a of $\mathcal{O}$ is a finitely generated $\mathcal{O}$-submodule of $\mathbb{K}$. We say that a fractional ideal a is invertible if there exists another fractional ideal b such that $ab = \mathcal{O}$, and that it is principal if $a = \alpha\mathcal{O}$ for some $\alpha \in \mathbb{K}$. The invertible fractional ideals of $\mathcal{O}$ form an abelian group whose quotient by the subgroup of principal fractional ideals is finite. This quotient group is called the ideal class group of $\mathcal{O}$, and denoted by $C\ell(O)$. The ideal class group $C\ell(O)$ acts freely and transitively on the set $E\ell\ell_p(\mathcal{O}, \pi)$, which contains all supersingular elliptic curves $\mathbb{E}$ over $\mathbb{F}_p$ - modulo isomorphisms defined over $\mathbb{F}_p$ - such that there exists an isomorphism between $\mathcal{O}$ and $\mathrm{End}(\mathbb{E})$ mapping $\sqrt{-p} \in \mathcal{O}$ into the Frobenius endomorphism $(x, y) \to (x_p, y_p)$ [Cas+18].

Isomorphism Problems The task of finding an isomorphism, if any, between two category objects can be seen as a particular instance of GAIP. In general, these classes of problems do not rely on abelian isomorphism groups. For the scope of this work, the most important is the code equivalence problem.

Definition 5.10. A map $\phi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is said an *isometry* for the distance d if it leaves the metric invariant, i.e.

$$d(\phi(x), \phi(y)) = d(x, y) \text{ for all } x, y \in \mathbb{F}_q^n \ .$$

If ϕ it is also linear we call it *linear isometry*. Two $[n, k]$-codes $\mathbb{C}, \mathbb{C}'$ are said *equivalent* with respect to the metric d if it exists a linear isometry for the relative metric that maps $\mathbb{C}$ to $\mathbb{C}'$.

We often omit to explicit write the metric when not necessary. We can now define define a general version of the code equivalence problem:

Definition 5.11 (General Code Equivalence)**.** Consider two equivalent linear codes $\mathbb{C}$ and $\mathbb{C}'$ over $\mathbb{F}_q$. Compute the isometry between them.

We can see this as a particular instance of GAIP (Definition 5.5), where the group of isometries act on the set of codes. Indeed, to prove that isometries forms a group under the composition, it is sufficient to notice that clearly the identity is an isometry and the composition of two isometries is still an isometry. Also they are invertible since the set $\mathbb{F}_q^n$ is finite and the maps are injective, in fact by the definition of distance:

$$x \neq y \Leftrightarrow 0 \neq d(x, y) = d(\phi(x), \phi(y)) \Rightarrow \phi(x) \neq \phi(y) \ .$$

Define $\mathbb{X}$ as the set containing $[n, k]$-codes and $\mathbb{I}$ the group of isometries, we can consider the following group action associated to the linear code equivalence:

$$\begin{aligned} \star : \mathbb{I} \times \mathbb{X} &\to \mathbb{X} \\ (\phi, \mathbb{C}) &\to \phi \star \mathbb{C} := \phi(\mathbb{C}) \end{aligned} \tag{5.2.2}$$

In the following we work with Hamming metric, thus the code equivalence problem of definition 5.11 can be stated as follows:

Definition 5.12 (Linear Code Equivalence)**.** Let $\mathbb{C}$ be a $[n, k]$-linear code over $\mathbb{F}_q$, with generator matrix G. Consider the following problem

$\mathsf{Exp}^{\mathsf{LCE}}_{\mathcal{A}}(\lambda)$:

1 : $S, Q \xleftarrow{\$} GL_k(q) \times M_n(q)$
2 : $G' \leftarrow SGQ$
3 : $S', Q' \in GL_k(q) \times M_n(q) \xleftarrow{\$} \mathcal{A}((x, y))$
4 : **return** $G' = S'GQ'$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{LCE}}_{\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{LCE}}_{\mathcal{A}}(\lambda) = 1)$$

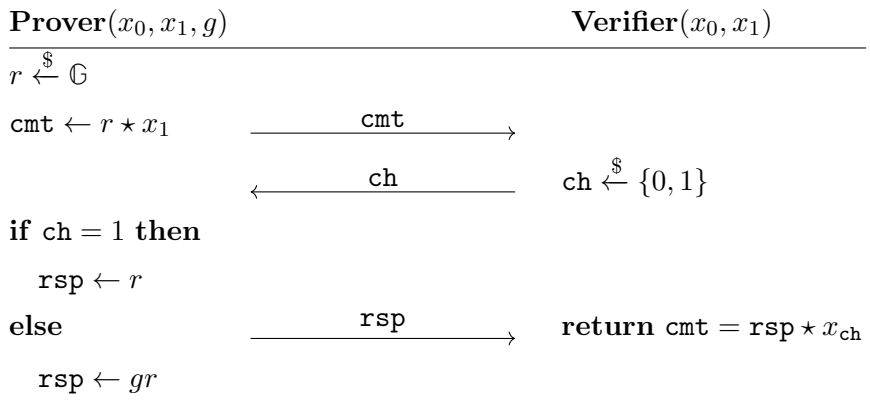


Figure 5.1: Structure of a sigma protocol

We say that the linear code equivalence problem is hard if and only if $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{GAIP}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

There is no formal proof about the hardness of the linear code equivalence problem. On the contrary, it was shown in [PR97] that the permutation equivalence problem is unlikely to be NP-complete, since this would imply a collapse of the polynomial hierarchy. However, while the problem can be solved efficiently for some families of codes, there are many cases that remain intractable after nearly 40 years of study [Bia+20].

Lastly, it is possible to prove that instead of S we can always choose G and G' to be in systematic form, i.e. $G = [I|R]$ where I is the $k \times k$ identity matrix.

5.3 Threshold Signature

In this section we show how it is possible to obtain a group action based threshold signature.

Consider the sigma protocol of Figure 5.1. It is easy to see that if GAIP is hard then the above sigma protocol is honest verifier zero knowledge, has special soundness and completeness. Thus, it is possible to obtain a signature by simply considering parallel repetition of it and applaying the Fiat-Shamir Transform.

In this section we design a threshold signature of it.

KeyGen(pp):

1 : $g_i \xleftarrow{\$} \mathbb{G}$
2 : Publishes $\mathtt{cmt}_i := \mathsf{Com}(g_i)$
3 : **for** i **in**$\{1, ..., n\}$ **do**
4 : P_i computes $x_i = g_i \star x_{i-1}$
5 : P_i publishes a ZKP $x_i = \mathsf{Decom}(\mathtt{cmt}_i) \star x_{i-1}$
6 : P_i sends x_i to P_{i+1} // when $i < N$
7 : **return** $y := x_N$ // The private key of P_i is g_i

Figure 5.2: Key-generation for the group action threshold signature. The public parameters pp comprise the origin x_0, the group $\mathbb{G}$ and an order among the players.

5.3.1 Full Threshold

We start our discussion by showing an algorithm for the *full threshold* cases, where all the users are required to produce a signature (i.e. $t = n$).

Decentralised Key Generation Algorithm The goal of this protocol is to produce a common public key $y = g \star x_0$ with $g = g_1 \cdot ... \cdot g_N$, where each party holds one g_i, in the same way of [Bas+23; CS20b]. To do so the users sequentially apply a previously committed random group element to the origin x and add a non-interactive Zero-Knowledge proof to show the correctness of the computation. The resulting protocol is shown in Figure 5.2. The main difference with [CS20b] is that our scheme is specialised for non-abelian group actions and we are able to prove the security with only one ZKP per user, compared to the two required by [CS20b].

At first glance, the fact that each user P_i needs to receive the set element x_{i-1} by P_{i-1} before starting its computation is a big limitation, since it does not allow for any parallelization. However, it was recently shown, that the round robin structure is optimal, when dealing with group actions in a black box way[CG23].

We now show two possible ZKPs for line 2 and 5. The most immediate idea is to define $x'_i = g_i \star x_0$ and set $\mathsf{Com}(g_i) = x'_i$. Then the relation of line 5 is the following

$$x'_i = g_i \star x \wedge x_i = g_i \star x_{i-1} \ . \tag{5.3.1}$$

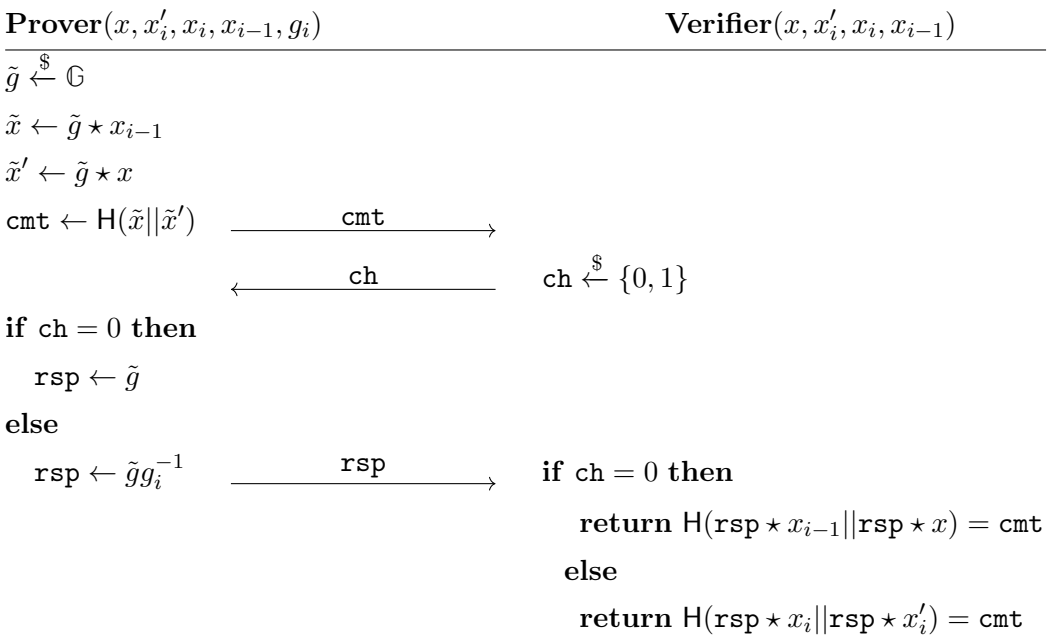


Figure 5.3: Sigma protocol for the relation of Equation (5.3.1)

The ZKP is shown in Figure 5.3.

Proposition 5.1. *If the action is weakly pseudorandom then the protocol in Figure 5.3 is complete, sound and computationally zero-knowledge. Moreover, it can be rendered to a non interactive computationally zero-knowledge quantum proof of knowledge in the QROM.*

Proof. First we prove that the underlying protocol is complete, sound and computationally zero-knowledge. The completeness is straightforward. We need to prove soundness and zero knowledge.

- **Soundness:** suppose that the prover is able to answer both challenges with u_0 and u_1, by the collision resistance of the hash function at this point we would retrieve g as $u_1^{-1}u_0$ against the one wayness of the group action and having that the public keys are generated by the same group elements.
- **Zero Knowledge:** to simulate the protocol without knowing the secret g and for any pairs of elements (x'_i, x), (x_i, x_{i-1}) the simulator $\mathcal{S}$ flips a coin c. If $c = 0$, then $\mathcal{S}$ follows the protocol normally and is able to answer if $b = 0$. If $c = 1$, it computes $\bar{x}' = \bar{g}x$ and $\bar{x} = \bar{g}x_{i-1}$ and sends them in place of $\tilde{x}'$ and

$\tilde{x}$. In this way it is able to answer when $b = 1$. Thus, if $c = b$ the prover can convince the verifier, otherwise it rewinds the verifier and try again. Since at every iteration the prover has probability $\frac{1}{2}$ of guessing the correct c the simulation ends in expected polynomial time. Note that this transcript is indistinguishable from the honestly-obtained one if the action is the weakly pseudorandomn.

For the quantum resistance we can observe that since the automorphisms are all trivial the sigma protocol has perfect unique responses (see [Blä+22, Lemma 1]) then by [Don+19, Theorem 25] the protocol is a quantum proof of knowledge. Then the protocol has completeness, high min entropy and HVZK and is zero-knowledge against quantum adversaries thanks to [Unr17]. □

The problem of the above protocol is that it requires a weakly pseudorandom group actions. It was recently shown that many commonly used group actions, such as the linear code equivalence, are not weakly pseudorandom and having access to $(x, g \star x, y, g \star y)$ allows to easily recover g [Bud+24]. In this case we need a different and less efficient zero knowledge proof. The strategy is basically to prepare the whole proof before the key generation, commiting to both the possible response. Then, finishing the proof during the key generation phase, opening only the relevant response. Formally, in line 2, P_i picks $\{\tilde{g}_k\}_{[\lambda]}$ randomly from $\mathbb{G}$, compute $\texttt{root}_i = \mathsf{Merkle}(\{(\tilde{g}_k, \tilde{g}_k g_i)\}_k)$, the root of a Merkle Tree having the set of all $(\tilde{g}_k, \tilde{g}_k g_i)$ as leafs and define $\mathsf{Com}(g_i) = \texttt{root}_i$. Later, in line 5, P_i performs the protocol of Figure 5.4.

Observation 5.1. The zero knowledge property is trivial to prove, the simulator can simply adopt the strategy used for the previous protocol. The special soundness is more tricky. The problem is that the protocol does not ensure that all the leaf of $\texttt{root}_i$ are correctly formed, indeed suppose that an adversary cheat on one leaf, then probability of being detected is only $\frac{1}{2}$, since half of the time that leaf is not open. What the protocol ensure, is that in the majority of the leaf the correct g_i is used. Moreover, if two accepting transcript are different on one bit, then both leaves must be correct, thus the above extractor works fine.

Signing Algorithm The signing protocol generalises the one presented in [CS20b; DFM20] for non-abelian group actions.

In the commitment phase, each user P_i receives x_{i-1}^j, computes $x_i^j = \tilde{g}_i^j \star x_{i-1}^j$ for random $\tilde{g}_i$ and outputs it. During the response phase P_i get u_{i-1}^j and outputs

Prover$(\texttt{root}_i, x_i, x_{i-1}, g_i)$		**Verifier**$(\texttt{root}_i, x_i, x_{i-1})$
$\{\tilde{x}_k\}_{[\lambda]} \leftarrow \{\tilde{g}_k x_i\}_{[\lambda]}$		
$\texttt{cmt} \leftarrow \{\tilde{x}_k\}_{[\lambda]}$	$\xrightarrow{\texttt{cmt}}$	
	$\xleftarrow{\texttt{ch}}$	$\texttt{ch} \xleftarrow{\$} \{0,1\}^\lambda$
if $\texttt{ch}_k = 0$ **then**		
$\quad \texttt{rsp}_k \leftarrow \tilde{g}_k \| \mathsf{H}(\tilde{g}_k g_i^{-1})$		
else		
$\quad \texttt{rsp}_k \leftarrow \tilde{g}_k g_i^{-1} \| \mathsf{H}(\tilde{g}_k)$		
$\texttt{rsp} \leftarrow \texttt{rsp}_1 \| ... \| \texttt{rsp}_k$	$\xrightarrow{\texttt{rsp}}$	Check all $\texttt{rsp}_k$ and the Merkle Tree.

Figure 5.4: Sigma protocol for a non pseudorandom group action.

$u_i^j = \tilde{g}_i^j u_{i-1}^j g_i^{-\texttt{ch}_j}$. For the challenge $\texttt{ch}_j = 0$ the parties verify that $\tilde{x}_i^j = u_i^j \star x$, while in the other case they check $\tilde{x}_i^j = u_i^j \star x_i$.

The idea of this multiparty protocol is illustrated in Figure 5.5.

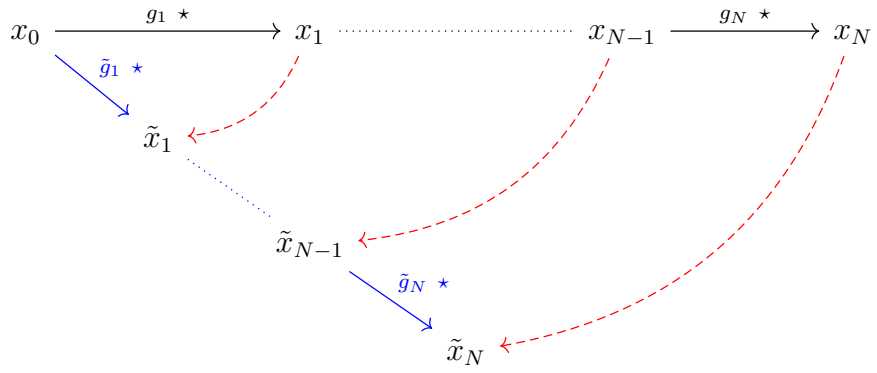


Figure 5.5: Scheme representing the idea behind the protocol in Figure 5.6. In blue are the ephemeral group elements revealed on $\texttt{ch} = 0$, while in red the map reconstructed for $\texttt{ch} = 1$.

A detailed description of the algorithm is given in Figure 5.6.

A key feature of Figure 5.6, with respect to the previous literature, is the use of secure salt during the challenge evaluation (line 12 and 13), a technique used also in [Cha22]. The salt is crucial to reduce the number of ZKPs in the signing protocol while maintaining security in the presence of malicious users.

$\mathsf{TSign}(\{g_i\}_{[n]}, \mathtt{m})$:

1 : $x_0^j \leftarrow x$ **for** $j \in \{1, ..., \lambda\}$
2 : **for** $i \in \{1, ..., n\}$ **do**
3 : P_i picks $\mathtt{salt}_i \xleftarrow{\$} \{0,1\}^{c(\lambda)}$
4 : $\mathsf{CMT_i} \leftarrow \mathsf{Com}(\mathtt{salt}_i)$
5 : **for** $i \in \{1, ..., n\}$ **do**
6 : **if** $i > 1$ **then** P_i receives all x_{i-1}^j from P_{i-1}
7 : **for** $j \in \{1, ..., \lambda\}$ **do**
8 : P_i picks $\tilde{g}_i^j \xleftarrow{\$} \mathbb{G}$, and sets $x_i^j \leftarrow \tilde{g}_i^j \star x_{i-1}^j$
9 : P_i sends x_i^j
10 : $x^j \leftarrow x_N^j$ **for** $j \in \{1, ..., \lambda\}$
11 : Each party publishes $\mathsf{Decom}(\mathsf{CMT_i})$
12 : $\mathtt{salt} \leftarrow \bigoplus_i \mathtt{salt}_i$
13 : $\mathtt{ch} \leftarrow \mathsf{H}(x^1||...||x^\lambda||\mathtt{salt}||\mathtt{m})$
14 : P_1 set $u_0^j \leftarrow e$**for** $j \in \{1, ..., \lambda\}$
15 : **for** $i \in \{1, ..., n\}$ **do**
16 : **if** $i > 1$ **then** P_i receives all u_{i-1}^j from P_{i-1}
17 : **for** $j \in \{1, ..., \lambda\}$ **do**
18 : P_i computes $u_i^j \leftarrow \tilde{g}_i^j u_{i-1}^j g_i^{-\mathtt{ch}_j}$
19 : P_i outputs u_i^j and all users verify it
20 : $\mathtt{rsp}_j \leftarrow u_N^j$ **for** $j \in \{1, ..., \lambda\}$
21 : $\sigma = \mathtt{ch}||\mathtt{salt}||\mathtt{rsp}_1||...||\mathtt{rsp}_\lambda$
22 : **return** σ

Figure 5.6: Signature algorithm

Indeed, without the salt verification, the scheme can be attacked by an adversary opening several concurrent sessions. Intuitively, suppose that the adversary is in control of the N-th user and wants to sign the message $\mathtt{m}$ for the public key $y = g \star x$, knowing only g_N. He can proceed in the following way:

(i) The adversary starts λ signing sessions for any messages $\mathtt{m}_1$,...,$\mathtt{m}_\lambda$.

(ii) For every session s, he receives by P_{N-1} $x^1_{N-1}, ..., x^\lambda_{N-1}$. At this point he evaluates $x^1_N = \tilde{g}^1_N \star x^1_{N-1}$ for each session s as described in the protocol. Let us call this element $\hat{x}^s$ for each session.

(iii) He evaluates the challenge $\mathtt{ch} = \mathsf{H}(\hat{x}^1||...||\hat{x}^\lambda||\mathtt{m})$.

(iv) For each session s, the adversary then evaluates $x^2_N, ..., x^{\lambda-1}_N$ legitimately, then chooses $\tilde{g}^\lambda_N$ so that the first bit of $\mathsf{H}(x^1_N||...||x^\lambda_N||\mathtt{m}_i)$ is equal to the s-th bit of $\mathtt{ch}$. This would not be possible if we had a secure salt.

(v) Finally, the adversary closes all the concurrent sessions obtaining, for the session s, the response u^1_{N-1} received from P_{N-1}, which is used to evaluate $\mathtt{rsp}_1$. This can be used to answer $\mathtt{ch}_s$ and obtain a valid signature $\mathtt{ch}||\hat{\mathtt{rsp}}_1||...||\hat{\mathtt{rsp}}_\lambda$.

We now prove the security of the scheme.

Theorem 5.1. *Let $(\mathbb{G}, \mathbb{X}, \star)$ be a group action such that GAIP is hard, if the centralised signature is unforgeable in the quantum random oracle model, then the full-threshold signature scheme composed by* KeyGen, TSign *is EUF-CMA secure in the quantum random oracle model.*

First of all, we need to prove the security of KeyGen.

Lemma 5.1. *Let $(\mathbb{G}, \mathbb{X}, \star)$ be a group action such that GAIP is hard, the protocol* KeyGen *can be simulated in the quantum random oracle model in polynomial time so that any probabilistic polynomial time adversary is convinced that the public key is any fixed pair $x, y \in X$.*

The main idea of the proof is to use the ZKPs to extract the secret from the adversary and use them to simulate a view of the protocol. Unlike [CS20b], here we only have one ZKP for any user, thus we rely in rewinding the tape to change the set element sent by the simulator in line 4. Notice that depending on whether the action is weakly pseudorandom or not we can use the protocol of Figure 5.3 or Figure 5.4.

Proof. Figure 5.7 shows the simulation strategy. We need to prove that the simulation terminates in expected polynomial time, it is indistinguishable from a real execution, and outputs y. The simulation terminates in polynomial time with non-negligible probability, indeed we have to carry over:

- one rewind of $\mathcal{A}$;

KeyGen simulation:	
1 :	$\mathcal{S}$ sends $\mathsf{Com}(g_{\mathcal{S}})$ to $\mathcal{A}$
2 :	$\mathcal{S}$ follows the protocol normally
3 :	$\mathcal{S}$ sends a random $x'_{i_{\mathcal{S}}}$ instead of the correct one
4 :	$\mathcal{S}$ simulate the ZKP
5 :	$\mathcal{S}$ follows the protocol normally and use the ZKP to extract $g_i, i > i_{\mathcal{S}}$
6 :	$\mathcal{S}$ rewinds the adversary before sending $x'_{i_{\mathcal{S}}}$
7 :	$x'_{i_{\mathcal{S}}} \leftarrow (g_{i_{\mathcal{S}}+1}^{-1} .... g_n^{-1}) \star y$
8 :	$\mathcal{S}$ follows the protocol normally with the new $x'_{i_{\mathcal{S}}}$ simulating the ZKP

Figure 5.7: KeyGen simulation

- at most $N-1$ extractions of secrets from the ZKPs, that can be carried over in polynomial time using the Forking Lemma on the single ZKP. The probability for the adversary to fake the ZKP where a share does not exists is negligible, assuming that GAIP is hard.

Note that the rewinding can be performed since the adversary has already committed to the values g_i before the rewinding phase. In addition, thanks to the ZKPs, these group elements must exist, and the adversary is forced to apply them on $x_{i_0} = (g_{i_0+1}^{-1} ... g_N^{-1}) \star y$, so that the output of the simulation is the public key x, y as desired. □

The proof of Theorem 5.1 follows the game-based argument proposed in [Gri+21, Theorem 3]. The key idea is to reduce the security of the full threshold signature to the security of the centralised one.

Proof Theorem 5.1. Consider an adversary $\mathcal{A}$ that make up to q_s sign queries and q_h quantum call to the random oracle $\mathcal{O}^{\mathsf{H}}$ that wins the $\mathsf{Exp}_{\mathsf{TDS}}^{\mathsf{a-euf-cma}}$ with non negligible advantage ϵ. By in Lemma 5.1 we can simulate the KeyGen on any public key x, y.

Since the protocol TSign and KeyGen are executed in multiparty, if by any reason the protocol is aborted because of $\mathcal{A}$ misbehaviour, the game ends and returns $\perp$. We now describe a sequence of game, game G_0 is the EUF-CMA game, while game G_2 is the real execution of the protocol.

Game G_0. This game is the same one played for the EUF-CMA security in Definition 2.21, thus $\Pr(G_0^{\mathcal{A}} \to 1) = \epsilon$ by definition.

Game G_1. In this game we set $\texttt{ch}$ at random and we reprogram the random oracle, instead of obtaining $\texttt{ch}$ from the random oracle. We can observe that any statistical difference between the games can be used to build a distinguisher for the reprogramming of the oracle; in particular we can adapt the distinguisher from the proof of [Gri+21, Theorem 3]. In total, we reprogram the oracle q_s times (one for every signature) and $\mathcal{A}$ performs q_h quantum calls. Moreover, note that $x^1, ..., x^\lambda, \texttt{m}$ are (at least partially) controlled by the adversary, while $\texttt{salt}$ is randomly sampled thanks to the initial commitments and the secure aggregation. This ensure us that the probability of having a collision is negligible and that $\mathcal{A}$ is not able to guess the input H, except with negligible probability (for a more detailed discussion, see the proof Lemma 3.1, in particular the paragraph about simulation failures during has queries. Notice that the commitment to $\texttt{salt}_i$ is done via a random oracle, so the simulation is simpler and does not require rewinding, as noted in the last part of the proof). Thus, by [Gri+21, Proposition 1] we have:

$$|\Pr[G_0^{\mathcal{A}} \to 1] - \Pr[G_1^{\mathcal{A}} \to 1]| \leq \frac{3g_s}{2^{1+\lambda}}\sqrt{q_h} \tag{5.3.2}$$

Game G_2. First of all, note that during the computation of the response, it is possible to check whether the received u_i^j is correct or not, if the user $i+1$ saved all the x_i during the key generation step. We exploit this property in our simulation. Indeed, to simulate a signature, the simulator first acts honestly and follows the protocol. Upon receiving all the responses u_i^j of $P_1, ..., P_{i_0-1}$, it checks the correctness of all of them. If they are all correct, it rewinds the adversary up until receiving $\tilde{x}_{i_0-1}$ and chooses $\tilde{x}_{i_0}$ according to challenge $\texttt{ch}_j$ (Figure 5.8 shows schematically of how the simulation strategy works). In particular:

- linking $\tilde{x}_{i_0-1}$ and $\tilde{x}_{i_0}$ on challenge $\texttt{ch}_j = 0$;
- linking x_{i_0} and $\tilde{x}_{i_0}$ on challenge $\texttt{ch}_j = 1$;

The idea is that every time the adversary acts honestly until P_{i_0}, the simulator produces an indistinguishable transcript that will not be rejected during the response computation. When, instead, the adversary sends something wrong before P_{i_0}, the simulation is perfect. Indeed, even if P_{i_0} is not able to answer to the challenge, the error spotted allows for an early abort and the simulation is indistinguishable.

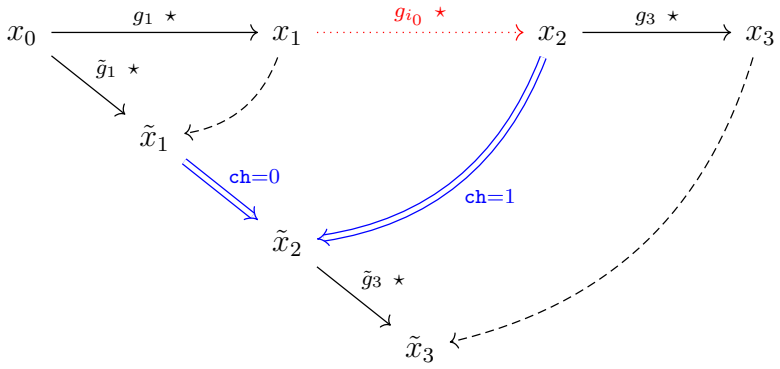


Figure 5.8: Example of simulation for $N = 3$ and $i_0 = 2$, in red the missing link, while in blue the elements used to generate x_{i_0} and to answer the challenge.

We have shown that G_2 simulates the multiparty signature protocol TSign, thus we need to bound the distance between the two last games. We are able to prove that the two views have the same distribution, implying null game distance.

If the simulator spots an error and aborts, the simulation is correct and indistinguishable from the real execution, since P_{i_0} followed the protocol normally. If the simulator rewind the adversary, then the view is given by $\mathtt{salt}_{i_0}$, $x^j_{i_0}$, $\tilde{g}^j_{i_0}$ for all $j = 1, ..., \lambda$. The salt and the group elements are uniformly distributed both in the signature and in the simulation, so they are indistinguishable even for an unbounded adversary. Also for j with $\mathtt{ch}_j = 0$ the set elements $x^j_{i_0}$ are indistinguishable since the simulator is just following the protocol TSign.

For j with $\mathtt{ch}_j = 1$ we consider the tuples $(\tilde{x}^j_{i_0-1}, \tilde{x}^j_{i_0})$ with $\tilde{x}^j_{i_0} = \tilde{g}^j_{i_0} \star \tilde{x}^j_{i_0-1}$ in the honest execution and $\tilde{x}^j_{i_0} = \tilde{g}^j_{i_0} \star x_{i_0}$ in the simulated ones.

After the rewinding, we know that $\tilde{x}^j_{i_0-1} = u^j_{i_0-1} \star x_{i_0-1} \in \mathbb{O}_{x_{i_0-1}} = \mathbb{O}_x$. Since the group action is free, there exists a unique $\tilde{h}$ with $\tilde{x}^j_{i_0} = \tilde{h} \star \tilde{x}^j_{i_0-1}$. The element $\tilde{h}$ has the same distribution as $\tilde{g}^j_{i_0}$ thanks to the uniqueness of the solution; it follows that these pairs are again indistinguishable.

Finally, we observe that game G_2 is executed entirely without the use of the secret share g_{i_0}, thanks to the simulation, and so succeeding in the game implies being able to forge a signature for the centralised scheme in the quantum random oracle. Since we assumed quantum unforgebility for the centralised signature, this probability is negligible. Combining all the game distances we prove the desired

reduction by the resulting equivalence:

$$\mathbf{Adv}^{\mathsf{a-euf-cma}}_{\mathsf{TDS},\mathcal{A}} \leq \frac{3g_s}{2^{1+\lambda}}\sqrt{q_h} + \mathsf{negl}(n)(\lambda) \ .$$

□

Observation 5.2. A thoughtful reader may be wondering why we have not used the results of Chapter 3 to prove the security of our scheme. While this is a suitable strategy for the passive case (that follows trivially from the active one), we are not able to use Theorem 3.1 in this context, since we are not able to design a secure $\mathsf{TP}_{\mathtt{cmt}}$ protocol, according to Definition 3.3. This is an immediate consequence of the round robin structure: we are not able to produce commitment and the combine them, since no operation is defined on the set, and we also are not able to simply commit to $\tilde{g}_i^j$ and then reveal it, otherwise the scheme would be trivially insecure. The presence of the salt solves the issue, allowing the simulator to act on it without the need of extracting all the ephimeral secrets $\tilde{g}_i^j$ from $\mathcal{A}$, nevertheless this is not enough to use Theorem 3.1.

This is a further testament to the fact that we were not able to prove the minimal conditions, but instead our conditions were only sufficient. However, being able to exactly capture the proof strategy adopted in this proof is non trivial and difficult to generalise.

5.3.2 General Scheme

We can now explain how to obtain a t, n scheme from the full threshold one, via replicated secret sharing[1] [ISN89]. Our approach was first proposed in [CS20b].

Definition 5.13. A *monotone access structure* $\mathbb{A}$ for the parties $\mathbb{P} := \{P_1, .., P_N\}$ is a family of subsets $\mathbb{S} \subset \mathbb{P}$ that are authorised (to sign a message) such that given any $\mathbb{S} \in \mathbb{A}$ and $\mathbb{S}' \supset \mathbb{S}$ then $\mathbb{S}' \in \mathbb{A}$. We sometimes call $\mathbb{A}$ the authorised set (or authorised parties). To each access structure we can associate a family of unqualified sets $\mathbb{U}$ that satisfies that for all $\mathbb{S} \in \mathbb{A}$, $\mathbb{S}_{\mathbb{U}} \in \mathbb{U}$ then $\mathbb{S} \cap \mathbb{S}_{\mathbb{U}} = \emptyset$. For all the section we will define the unqualified sets in the canonical way as $\mathbb{U} = 2^{\mathbb{P}} \setminus \mathbb{A}$.

If we want to share a secret s for a monotone access structure $\mathbb{A}$, we need to consider the family $\mathbb{U}^+$ of the maximal unqualified set with respect to inclusion

[1] Unfortunately, while standard linear secret sharing would be more efficient, it is difficult to use in a non-abelian setting.

and define $\mathbb{I}$ as the family of complements for $\mathbb{U}^+$, i.e.

$$\mathbb{I} := \{\mathbb{S}_\mathbb{I} \in \mathbb{A} \mid \forall \mathbb{S}_\mathbb{U} \in \mathbb{U} \,.\, \mathbb{S}_\mathbb{U} \supseteq \mathbb{P} \setminus \mathbb{S}_\mathbb{U} \implies \mathbb{S}_\mathbb{U} = \mathbb{P} \setminus \mathbb{S}_\mathbb{I}\} \ .$$

Having fixed $M = \#\mathbb{I}$, we sort the elements in $\mathbb{I}$ as $\mathbb{I}_1, \mathbb{I}_2, ...$ and for each $l \in \{1, ..., M\}$ we define the shares s_l so that $s = s_1 \cdots s_M$; each party P_i is then given access to s_l if and only if $\mathbb{I}_l \ni i$. This leads to the following (already known) result.

Proposition 5.2. *Any authorised subset of users can get the secret s, whilst any non-authorised set of users cannot retrieve at least one share.*

By using this proposition, the parties in the authorised set can recover the secret just by agreeing on which one of them should be the one sharing each share, i.e. by agreeing on a turn function τ.

For the t, n scheme, the authorised sets are the ones having cardinality at least t. In this way, $\mathbb{U}^+$ are all the subsets with at most $t-1$ element, $\mathbb{I}$ the ones of cardinality $N - T + 1$ and $M = \#\mathbb{I} = \binom{N}{T-1}$.

Distributed key generation. The distributed key generation protocol of Figure 5.2 can be used also in this threshold case. The main difference is that during the generation each share g_i should be known to multiple users, so to apply it on x_{i-1} they can either:

(i) jointly generate a shard of it and then combine the shard. Since the element g_i should be known to all the party involved, a simply commit-release protocol is suitable for the task;

(ii) delegate one of the users that should know a share to apply it; said user then share it with the others, who can check the correctness of the received shard easily by simply compute the action.

We prefer the second option since it has a lower latency for the non-abelian case, but still achieves the same security, assuming that all the users take part to at least one generation round, thanks to the zero-knowledge proofs.

Signature algorithm The signature algorithm is also performed in the same way as the full threshold scheme, using the turn function τ to determine which party sends which messages at each round. The proof of security for this scheme is practically equal to the full threshold one: in fact, one can imagine that, after an initial phase to see who has the required shares, the scheme is essentially an (M, M)-threshold scheme.

Theorem 5.2. *Let* $(\mathbb{G}, \mathbb{X}, \star)$ *a free group action where GAIP is hard. If the centralised signature is unforgeable in the quantum random oracle model, then the* (t, n)*-threshold signature scheme composed by* KeyGen*,* TSign *adjoined with replicated secret sharing is EUF-CMA secure in the quantum random oracle model.*

Sketch. The proof is very similar to that of the full threshold case (Theorem 5.1). First of all, note that, since the adversary controls at most $t-1$ players, there must be at least a set $\mathbb{I}_h \in \mathbb{I}$ composed only by honest players on which the adversary has no control. Thus we just use the strategies from the full threshold case using users in $\mathbb{I}_h$ as non corrupted users. □

Number of shares The main drawback of replicated secret sharing is that the number of shares grows proportionally to the cardinality of $\mathbb{U}^+$, which is usually exponential in the number of parties. In particular, in the threshold case, there are $\binom{n}{t-1}$ shares in total, and each party needs to save $\binom{n}{t}$ shares. Since the group is non-abelian, the number of rounds cannot be reduced and it is equal to the total number of shares.

For this reason, the protocol has several usability limitations and is feasible only for particular t and n, like $t = n$ (full threshold) or small n. For the case $t = n-1$ and $n > 3$, the number of the shares is already linear in n and the number of rounds is quadratic in n. Nevertheless, we would like to point out that for the most used combinations of (t, n) such as $(2, 3)$ or $(3, 5)$, the number of shares (and rounds) is manageable and the protocol maintains an acceptable level of efficiency.

5.4 Optimizations and Performance Evaluation

In this section, we show a couple of optimizations typically used with group action that can also be used for our threshold protocol. We later show the parameters of the protocol, when considering the LESS signature, based on the linear code equivalence problem [Bar+21]. We will denote by ξ the bit-weight of an element of $\mathbb{X}$, and γ to denote that of an element of $\mathbb{G}$.

5.4.1 Multi-bit Challenges

Multi-bit challenges are a way to reduce the computational time at the price of bigger keys and are widely used in signature design (e.g. [DFG19]). In a nutshell,

this method consists in replacing a binary challenge space with a larger one, where each challenge value corresponds to a different public key. In this way, it is possible to amplify soundness, at the cost of an increase in public key size. Security is then based on a new problem:

Pr[[]mGAIP: Multiple Group Action Inverse Problem] Given a collection $x_0, ..., x_{r-1}$ in $\mathbb{X}$, find, if any, an element $g \in \mathbb{G}$ and two different indices $j \neq j'$ such that $x_{j'} = g \star x_j$.

It is possible to prove that this problem is equivalent to GAIP [Bar+21]. We can then consider $r-1$ public keys $x_1, ..., x_{r-1}$ generated from the initial element x_0 by $r-1$ shared keys $g^{(1)}, ..., g^{(r-1)}$ (with the notation $g^{(0)} = e$). At this point the challenge is generated as an integer $\mathtt{ch} \in \{0, ..., r-1\}$, thus to evaluate the response P_i computes $u_i^j = \tilde{g}_i^j u_{i-1}^j (g_i^{(\mathtt{ch}_j)})^{-1}$. As mentioned above, the soundness error is reduced to r^{-1}, thus in the signing algorithm we only need to execute $\lceil \frac{\lambda}{\log_2(r)} \rceil$ rounds, reducing both signature size and computational cost, but increasing the public key size.

5.4.2 Fixed-weight challenges

When there is a meaningful difference between the size of $\mathtt{rsp}$ when $\mathtt{ch} = 0$ and $\mathtt{ch} = 1$, a common idea is to choose an hash function having a fixed number 0 and 1, minimizing the occurrence of the bigger response, at the cost of increasing the challenge length [BKP20; Bar+21]. In our case, while $\mathtt{ch} = 1$ requires to send a group element, in the case $\mathtt{ch} = 0$ the prover can simply send the PRNG seed used to generate the random group element $\tilde{g}$, that is usually much shorter than a a group element.

Let consider an hash function H that returns a vector of fixed weight ω and length t. To avoid a security loss we need to have a *preimage security* (the difficulty of guessing in the challenge space) of still λ bits, thus t, ω are such that: $\binom{t}{\omega} \geq 2^\lambda$. In this way, we can obtain shorter signature size at the price of an higher number of rounds.

To further reduce the signature size, it is possible to send multiple seeds at the same time by using a *seed tree*. This primitive uses a secret master seed to generate t seeds recursively exploiting a binary structure: each parent node is used to generate two child nodes via a PRNG. When a subset of $t - \omega$ seeds is requested for the signature, we only need to send the appropriate nodes, reducing the space required for the seeds from $\lambda(t-\omega)$ to a value bounded above by $\lambda N_{\mathsf{seeds}}$, where

$$N_{\mathsf{seeds}} = 2^{\lceil \log(\omega) \rceil} + \omega(\lceil \log(t) \rceil - \lceil \log(\omega) \rceil - 1) \ ,$$

as shown in [GPS22; Cho+23]. In [Cha22], the author noted that, to avoid collisions attacks, a fresh salt should be used in combination of the seed tree structure.

Applying this optimization to a threshold signature is not straightforward and requires particular parameters to be used. Indeed, the parties can not share a single seed used for the generation of the ephemeral map $\tilde{g}$, but have to share $M = \binom{N}{T-1}$ of them. Thus, if the challenge bit is 0, the parties need to send all the M bits, and the total communication cost becomes $M \cdot \lambda$. So, for this strategy to make sense, we need $M\lambda$ to be smaller than the weight of the group element.

5.4.3 Scheme Parameters

When the two approaches are combined, the final signature weight is $(n_{\mathsf{seeds}}M + 2)\lambda + \omega\gamma + t$ with t the number of rounds ($\#\mathsf{rounds}$) satisfying

$$\binom{t}{\omega}(r-1)^{\omega} \geq 2^{\lambda} \ .$$

In our signing algorithm, for each of the $\binom{n}{t-1}$ iteration of the for loop over $1, ..., M$, each user needs to send the following quantities to the next user:

- $\#\mathsf{rounds} \cdot \xi$ bits for the commitment phase,
- $\#\mathsf{rounds}\cdot\gamma + 2\lambda$ bits in general and $(N_{\mathsf{seeds}}M+2)\lambda+\omega\gamma$ when using fixed-weight challenges.

At this point, we can see specific choices for LESS. In our analysis, we choose the public parameters that satisfy the requirement of 128 bits of classical security and at least 64 bits of quantum security, and evaluate ξ and γ accordingly. We include here the data for the original signature schemes, as well as parameters that we found in order to optimize the sum $|\mathsf{pk}| + |\sigma|$ for the cases $(2, 3)$, $(3, 5)$ and the case without fixed-weight challenges.

Instantiations with LESS.

From [Bal+23] we have taken the secure balanced LESS parameters for the NIST Security Category 1 $N = 252, K = 126$ (length and dimension of the code), $\mathbb{F}_q = \mathbb{F}_1 27$. We obtain that the size of a single code in systematic form is given by $(N-K)K\lceil \log_2(q) \rceil$ bits, so $\xi = 13.7\mathsf{KiB}$. Instead, to send a monomial map, we can use the IS-LEP technique from [PS23]. This recent optimization requires the use of

Case	Variant	t	ω	$\lvert\mathsf{pk}\rvert$ (KiB)	$\lvert\mathsf{sig}\rvert$ (KiB)	Exc. (MiB)
centralised	Fixed	247	30	13.7	8.4	-
(2,3)	Fixed	333	26	13.7	10.59	13.30
(3,5)	Fixed	333	26	13.7	21.09	44.43
(N,T)	$[440, 50]_{127}$	-	-	16.68	12.55	$\binom{N}{T-1}2.19$

Table 5.1: Parameters for the threshold version of LESS

a new canonical representation of the generator matrices via information sets. In this way, the equality can be verified using only the monomial map, truncated on the preimage of the information set, thus nearly halving the communication cost to $K(\lceil \log_2(q-1) \rceil + \lceil \log_2(N) \rceil)$ bits for each group element. This optimization (and any other possible new optimization based leveraging modified canonical forms, such as [CPS23]) can be used also for the threshold protocol since:

- for the commitment phase, the last user can simply commit using the modified canonical form, then store the additional information received (the information set used);
- for the response phase, when the monomial map $g^{-1}\tilde{g}$ is recovered, it can be truncated again by the last user by using the additional information from the commitment.

For the cases in which fixed-weight cannot be used, we simply send all the truncated monomial maps. In this case, we can cut the signature size without enlarging too much the public key, by decreasing the code dimension to $K = 50$ at the price of a longer code with $N = 440$ for $q = 127$ leading to a public key size of 17.1KiB and truncated monomial map size of 100B. Numbers are reported in Table 5.1, where we report, in the last column, also the total size of the exchanged data.

5.5 Oblivious Transfer

An oblivious transfer (OT) protocol is a 2-party protocol in which a sender sends one among many pieces messages to a receiver, but remains oblivious as to what message has been sent. On the other hand, the receiver does not learn anything about the other messages.

In this section we present a group action based oblivious transfer (OT). While requiring communication during the key generation phase, our OT require a single group action evaluation per input message and the communication of a single set element per message, making it way faster and compact compared to the one presented in [Ala+], which instead require $\log(\lambda)$ group action evaluation per input message. Moreover, our construction allows for a straightforward generalization to n input messages and t received ones, while the one in [Ala+] is limited to the classical 1-out-of-2.

We can define an OT and its usual security requirements as follows.

Definition 5.14 (Statistically Sender-Private OT)**.** A two-message statistically sender-private OT is a triple of algorithms $(\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$, such that:

- $\mathsf{OTR}(\lambda, b)$: on input a security parameter λ and a bit b, outputs a message $\mathtt{ot}_1$ and a secret $\mathtt{sk}$.
- $\mathsf{OTS}(\lambda, \mathtt{m}_0, \mathtt{m}_1, \mathtt{ot}_1)$: on input a security parameter λ, a pair of messages $(\mathtt{m}_0, \mathtt{m}_1)$ and a message $\mathtt{ot}_1$, outputs a message $\mathtt{ot}_2$.
- $\mathsf{OTD}(\lambda, \mathtt{sk}, b, \mathtt{ot}_2)$: on input a security parameter λ, a secret $\mathtt{sk}$, a bit b and a message $\mathtt{ot}_2$ it outputs the message $\mathtt{m}'$.

that satisfies the following properties:

- **Correctness**: for any bit $b \in \{0, 1\}$, any pair of messages $(\mathtt{m}_0, \mathtt{m}_1)$ and

  $$(\mathtt{ot}_1, \mathtt{sk}) = \mathsf{OTR}(\lambda, b), \mathtt{ot}_2 = \mathsf{OTS}(\lambda, \mathtt{m}_0, \mathtt{m}_1, \mathtt{ot}_1), \mathtt{m}' = \mathsf{OTD}(\lambda, \mathtt{sk}, b, \mathtt{ot}_2)$$

  we have $\mathtt{m}' = \mathtt{m}_b$ except with negligible probability.
- **Receiver privacy**: the output distributions of $(\mathtt{ot}_1, \mathtt{sk}) = \mathsf{OTR}(\lambda, 0)$ and $(\mathtt{ot}'_1, \mathtt{sk}') = \mathsf{OTR}(\lambda, 1)$ are indistinguishable.
- **Statistical sender privacy**: for any bit b, any message $\mathtt{ot}_1$ and two pairs of messages $(\mathtt{m}_0, \mathtt{m}_1)$, $(\mathtt{m}'_0, \mathtt{m}'_1)$ such that $\mathtt{m}_b = \mathtt{m}'_b$, we have that the ouptut distributions of $\mathsf{OTS}(\lambda, \mathtt{m}_0, \mathtt{m}_1, \mathtt{ot}_1)$ and $\mathsf{OTS}(\lambda, \mathtt{m}'_0, \mathtt{m}'_1, \mathtt{ot}_1)$ are indistinguishable.

From now on, let $(\mathbb{G}, \mathbb{X}, \star)$ be a group action where cGAIP is hard. Suppose now that the receiver owns a public key (x, x_0, x_1) for which it knows a group element that maps x to one among x_0 and x_1 but it does not know a map from x to the

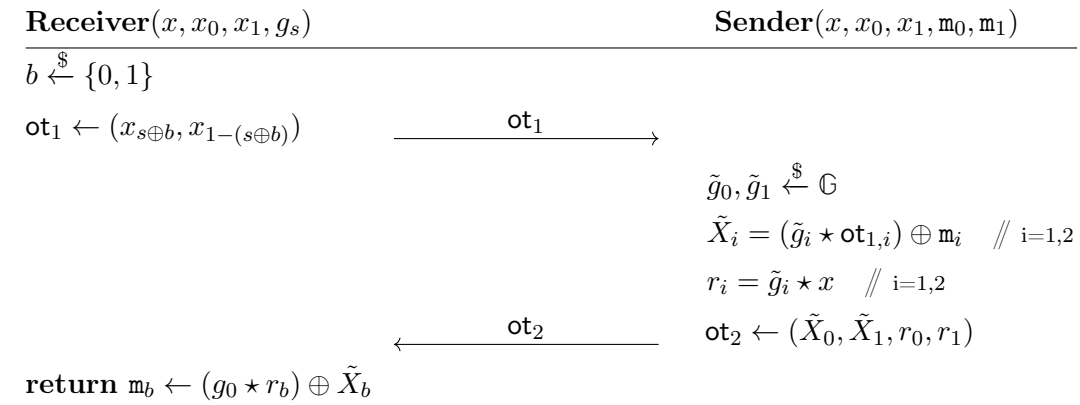


Figure 5.9: Structure of a sigma protocol

other element. We refer to this group element as g_s and the $x_s = g_s \star x$. Moreover, the sender does not know whether $s = 0$ or $s = 1$. We deal with how to obtain such a key and how to prove the "ignorance" of g_{1-s} later. Figure 5.9 shows the protocol.

We now prove the security of the described OT.

Theorem 5.3. *Let $(\mathbb{G}, \mathbb{X}, \star)$ be a group action for which cGAIP is hard. Then the protocol of Figure 5.9 is a statistically sender-private OT as per Definition 5.14.*

Proof. The correctness is straightforward and follows from basic arithmetic. We prove now receiver privacy and statistical sender privacy.

Receiver privacy. Since the sender does not knows whether the receiver knows g_0 or g_1, the two couple (x_0, x_1) and (x_1, x_0) have trivially the same distribution. Indeed $\Pr(\mathsf{ot}_1 = (x_1, x_0) | b = 0 \wedge s = 0) = \Pr(\mathsf{ot}_1 = (x_1, x_0) | b = 1 \wedge s = 1)$.

Statistical sender privacy. Let $\mathcal{A}$ be a malicious receiver, that can distinguish between $\mathsf{OTS}(\lambda, \mathtt{m}_0, \mathtt{m}_1, \mathsf{ot}_1)$ and $\mathsf{OTS}(\lambda, \mathtt{m}'_0, \mathtt{m}'_1, \mathsf{ot}_1)$, with $\mathtt{m}_b = \mathtt{m}'_b$. Without loss of generality we can suppose that $\mathcal{A}$ knows only g_0, we show later how to prevent a malicious adversary from knowing both g_0 and g_1. Suppose that $b = 0$, we show that $\tilde{X}_1$ is indistinguishable from a random string. Indeed, since the the cGAIP is hard, we have that, for a fixed g, $(g \star x, r \star x, gr \star x)$ and $(g \star x, r \star x, y)$ with y sampled uniformly at random from $\mathbb{X}$ have the same distribution. From the security of H, it follows immediately that also $\mathsf{H}(gr \star x)$ and $\mathsf{H}(y)$ are both uniformly

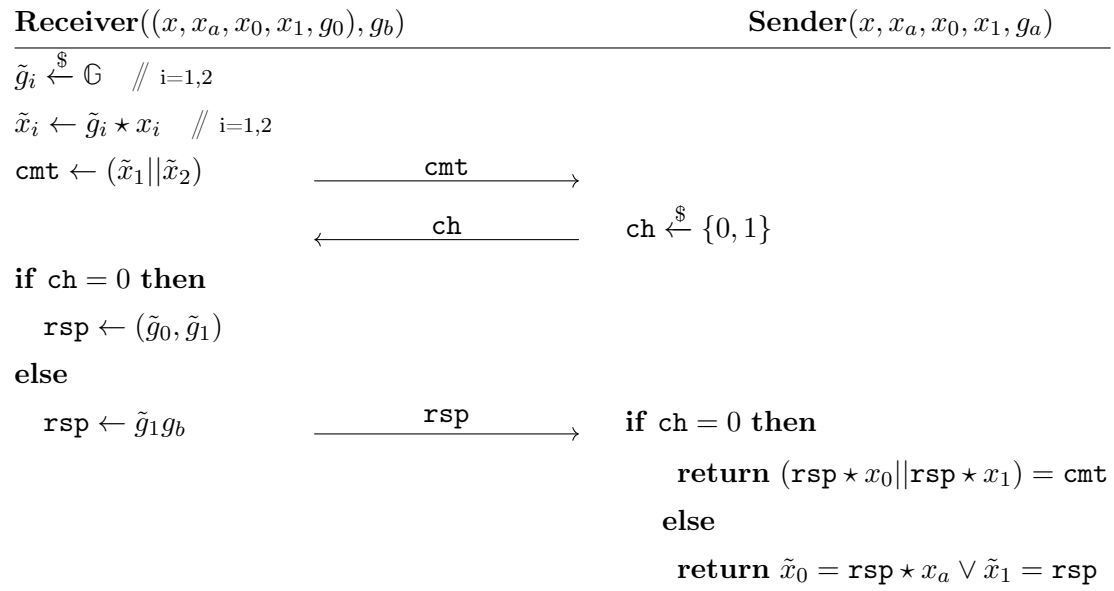


Figure 5.10: Sigma protocol for the knowledge of g_b, and thus the "ignorance" of g_1.

distributed in $\{0,1\}^{l(\mathsf{H})}$, the output domain of H. Lastly, notice that $\tilde{X}_1$ is simply a one-time pad encryption of $\mathtt{m}_1$ with key $\mathsf{H}(gr \star x)$, which is uniformly distributed. Thus, also $\tilde{X}_1$ is indistinguishable from a uniformly distributed random string. □

We now need to prove how is it possible to create a valid pubic key, such that the receiver knows only one among the two actions and the sender is not able to detect which one. WLOG we suppose that the receiver knows g_0. The idea is that the sender generates a random g_a and sends $x_a = g_a \star x$ to the receive. At this point the receiver computes $x_0 = g_0 \star x$ and $x_1 = g_b \star x_a$, then prove the knowledge of g_b by doing the equivalent of a ring signature with ring of keys formed by (x_a, x_0) and (x_a, x_1). Since it knows g_b but not g_a, this is enough to prove that it does not know g_1 such that $x_1 = g_1 \star x$.

It is easy to prove that the above protocol is a sigma protocol for the relation $((x, x_a, x_0, x_1, g_0), g_b) \in \mathbb{L}$ if and only if $(x_0 = g_b \star x_a) \vee (x_1 = g_b \star x_a)$.

Sketch. The completeness is trivial. Moreover, it is clear that having both $\tilde{g}_0$,$\tilde{g}_1$ and $\tilde{g}_1 g_b$ allows to immediately compute g_b. To prove the honest verifier zero knowledge is enough to notice that on challenge $\mathtt{ch} = 0$ the prover can act honestly, while on challenge $\mathtt{ch} = 1$ it can simply compute $\tilde{x}_1 = g \star x_a$ and then set $\mathtt{rsp} = g$. □

Lastly, we need to prove that the receiver does not learn anything about the

action g_1 that links x and x_1. This is an immediate consequence of the GAIP.

Sketch. Let $\mathcal{A}$ be an adversary that after the protocol can output g_1 such that $x_1 = g_1 \star x$. Let x, y be a GAIP challenge. The simulator $\mathcal{S}$ sets $x_a = y$ and forwards it to adversary to start the proof. When the adversary outputs g_1, $\mathcal{S}$ rewinds the adversary, changes the challenge and obtains the secret g_b. From g_1 and g_b it computes $g_a = g_b^{-1} g_1$ and wins the GAIP game. □

5.5.1 Remarks

It is easy to see that the above protocol can easily be adapted to a t-out-of$n$ OT, where the sender has n messages and the receiver obtains only t of them. To do so it is enough to modify the protocol of Figure 5.10 such that the receiver generates t keys starting from x and $n - t$ keys starting from x_a, the proof and the remaining part of the protocol can be generalized trivially.

Notice that the protocol requires the action to be commutative, otherwise the last step, when the receiver retrieve the message, fails, since $g_b r \star x \neq r g_b \star x$. Generalizing this protocol to non commutative group action is still a work in progress and the task of obtaining an OT protocol from black box group action is still an open problem.

Chapter 6

E-Voting

6.1 Introduction

Internet voting is a type of electronic voting (e-voting) that allows voters to cast their ballot remotely through the Internet, without the need of physically going to a polling station. Since the first attempts of introducing the Internet as a legally binding way of casting votes in Estonia and the United States in the early 2000's, Internet voting solutions increased in popularity and are currently used to varying degrees in several countries around the world [Ifesa; Idea]. Prominent examples include Switzerland [HPT22], Canada [CAE19] Australia [HT15] and Estonia, where recently becomes the favourite voting methods [Eest].

As with other electronic voting initiatives, the promises of Internet voting are higher voter turnout, lower cost and accessibility [Lic+21]; potentially at the expense of simplicity, transparency and privacy. Cryptographic protocols are particularly suited to the task, and in recent years many protocols were designed to achieve secure Internet-based elections that ensure voter privacy, vote verifiability and the correctness of the outcome [Adi08; CCM08; RRI16].

There is one additional threat, however, that is equally crucial to address in a fair and democratic election process: *coercion resistance*. Informally, a coercion-resistant protocol must defend voters from attackers that pressure them to vote in a specific way, either through threats or rewards. Because of its remote nature, Internet voting substantially increases the attacker option, since the safety offered by the polling station is not present anymore. These include forcing voters to reveal the voting material, or monitoring their behavior during the election day.

Juels, Catalano and Jakobsson [JCJ10] proposed one of the first formal defini-

tions of coercion resistance and designing a protocol to achieve it. To this date, JCJ still remains the reference point for research on the topic, however recently, the security definition presented in JCJ was disputed, in [HS19] and particularly in [CGY22b], who proposed a stronger security notion and a coercion resisntance e-voting protocol.

In this chapter we present an e-voting protocol that, at the state of the art, achieve the stronger security notion and the better efficiency among the peers.

The results presented in this chapter were presented at EVote-ID 2023 and are published in the proceedings of the conference [ABR23][1]. During CIFRIS23 a possible improvement of this protocol was discussed, regarding the usage of Class Group Encryption to achieve better performance from a memory standpoint, as hinted in Section 6.6. Research in this direction is not yet finished and is currently in progress but is bringing, for now, interesting preliminary results, not contained in this thesis, that we plan to finalize and send to EVote-ID 2024.

6.2 Preliminaries

6.2.1 Internet Voting

Internet voting protocol are generally divided in three phases:

(i) **Registration Phase:** voters authenticate themselves with the relevant authorities and receive voting materials, usually containing their voting credentials and a proof about their correctness.

(ii) **Voting Phase:** voters vote using the obtained credentials. Voters can vote more than once and thus change their previous vote, depending on the election policy (usually a revote invalidates previous votes). During this phase voters should be able to verify the correctness of the protocol, checking that the vote was casted-as-intended and recorded-as-casted. This is usually done via a combination of ZKP, the usage of a public board and device auditing techniques like the Benaloh Challenge [Ben06].

(iii) **Tally Phase:** the election result is calculated and published. Note that it is often required that no partial result is ever published or known, so the tallying protocol is usually a multi-party protocol that revolves around a

[1]The proceedings are not yet published, but they will be published soon.

threshold encryption scheme. The result is usually published along with a ZKP about the correctness of the calculation.

The participants in the protocol are:

- The *public board* $\mathbb{B}$, an append-only list of data, where all the other participants can write. The contents of the board can be read by anyone at any time, and the board is assumed to be honest.
- The election *trustees*, a set of n_T authorities that performs the cleansing and the tally. In our protocol, we assume that there are most t dishonest trustees, where $t < n_T$ is the threshold of the encryption protocol used.
- The *voters*. There are n_V voters and we assume that the adversary is able to control at most $n_V - 2$ of them.
- The *auditors*, a set of parties that check the consistency of the data published on the board. In particular auditors need to check the validity of all the ZKPs. We only need one auditor to be honest. Since every check involves only public data, any party could serve as auditor.
- The *registrars*, a second set of n_R authorities that provide credentials to voters. In our protocol, we assume that all the registrars are honest.

We define a voting protocol as follows:

Definition 6.1 (Voting Protocol)**.** A *voting protocol* is defined by the tuple

$$(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally})$$

where:

- $\mathsf{Setup}(\lambda, t, n_T)$, is a multi party protocol run by the trustees. On input a security parameters λ, the number of trustees n_T and the threshold t it outputs public parameters $\mathtt{pp}$, a public key $\mathtt{sk}$ and shares of the private key $\mathtt{pk}_i$, one for each trustees.
- $\mathsf{Register}()$ is a multi party protocol run by the registrars, which take no input and generates the private credentials $\{\mathtt{s}_i\}_{\mathbb{V}}$ of the voters and outputs their public part $\mathbb{R}$.
- $\mathsf{Vote}(\nu, \tilde{\mathtt{s}}, \mathtt{pk})$ which takes a voting choice ν, a credential $\tilde{\mathtt{s}}$ and a public key $\mathtt{pk}$ and returns a ballot.

- $\mathsf{Tally}(\mathbb{B}, \mathbb{R}, \{\mathtt{sk}_i\})$ a multi party protocol run by at least t trustees. On input the public board $\mathbb{B}$, the public credentials $\mathbb{R}$ and at least t shares of the private key, it outputs the election result.

Note that the above definition is deliberately vague and not exhaustive. In fact, there are many voting protocols that do not fit precisely this definition. The aim is to provide a relatively formal description of the algorithms that make up the protocol presented in this chapter, and that are necessary to understand the definition of security below.

6.2.2 Coercion Resistance

The most common definition of coercion resistance is by Juels, Catalano and Jakobsson [JCJ10]. Roughly speaking, a voting protocol is coercion resistant if and only if voters are able to generate some kind of *fake credential* that could be handed over to the coercer in case of attack, preserving the original legitimate ones and thus their ability to vote [JCJ10]. Votes with fake credentials are discarded later, in the *cleansing* phase of the election process.

Recently, the security definition presented in JCJ was disputed [CGY22b; HS19], due to its limitation in handling revotes and ballots cast under invalid credentials. Ideally, the only types of leakage that should be allowed are those that inevitably arise from the election result, namely the number of discarded votes (that is unavoidable, since it is the difference between the total processed ballots and the number of valid votes). Cortier, Gaudry, and Yang, in [CGY22b], showed that the JCJ protocol leaks significantly more than this simple difference. In particular, during the tallying phase of the election, votes with duplicate credentials (i.e. the revotes) and votes with invalid credentials are handled separately, thus leaking the size of both sets individually, instead of leaking only the size of their union. The most up to date security definition for coercion resistance is the following:

Definition 6.2 (Coercion Resistance [CGY22b])**.** We say that a voting protocol $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Vote}, \mathsf{Tally})$ is coercion resistance if there exist an algorithm $\mathsf{FakeCred}$ that, on input a credential $\mathtt{s}$ outputs a random credential $\tilde{\mathtt{s}}$ such that for every adversary $\mathcal{A}$, for all parameters $n_T, t, n_V, n_\mathcal{A}, n_C$ and for all distribution D, there exists a simulator $\mathcal{S}$ such that

$$\Pr(\mathsf{Ideal}^{\mathsf{CR}}_{\mathcal{S}}(\lambda, n_V, n_\mathcal{A}, n_C, D) = 1) - \Pr(\mathsf{Real}^{\mathsf{CR}}_{\mathcal{A}}(\lambda, n_T, t, n_V, n_\mathcal{A}, n_C, D) = 1)$$

is negligible, where $\mathsf{Ideal}^{\mathsf{CR}}$ and $\mathsf{Real}^{\mathsf{CR}}$ are as defined in Figure 6.1.

$\mathsf{Real}^{\mathsf{CR}}$		$\mathsf{Ideal}^{\mathsf{CR}}$	
1 :	$\mathbb{B} \leftarrow \emptyset$	1 :	
2 :	$\mathtt{pk}, \mathtt{sk}_i, h_i \leftarrow \mathsf{Setup}^{\mathcal{A}}(\lambda, n_T, t)$	2 :	
3 :	$\{\mathtt{s}^i\}_{i \in \mathbb{V}}, R \leftarrow \mathsf{Register}(\lambda, \mathtt{pk}, n_V)$	3 :	
4 :	$\mathbb{V}_{\mathcal{A}} \leftarrow \mathcal{A}()$	4 :	$\mathbb{V}_{\mathcal{A}} \leftarrow \mathcal{A}()$
5 :	$(j, \beta) \leftarrow \mathcal{A}(\{\mathtt{s}^i\}_{i \in V}, R)$	5 :	$(j, \beta) \leftarrow \mathcal{A}()$
6 :	**if** $\lvert\mathbb{V}_{\mathcal{A}}\rvert \neq n_A$ or $j \notin \mathbb{V} \setminus \mathbb{V}_{\mathcal{A}}$ or $\beta \notin [n_C]$	6 :	**if** $\lvert\mathbb{V}_{\mathcal{A}}\rvert \neq n_A$ or $j \notin \mathbb{V} \setminus \mathbb{V}_{\mathcal{A}}$ or $\beta \notin [n_C]$
7 :	**return** 0	7 :	**return** 0
8 :	$\mathbb{B} \leftarrow D(n_V - n_{\mathcal{A}}, n_C)$	8 :	$B \leftarrow D(n_V - n_{\mathcal{A}}, n_C)$
9 :	**for** $(i, *) \in B, i \notin [n_V]$ **do**	9 :	
10 :	$\mathtt{s}^i \leftarrow \mathsf{FakeCred}(\mathtt{s}^i)$	10 :	
11 :	$b \xleftarrow{\$} \{0, 1\}$	11 :	$b \xleftarrow{\$} \{0, 1\}$
12 :	$\tilde{\mathtt{s}} \leftarrow \mathtt{s}^j$	12 :	
13 :	**if** $b = 1$	13 :	**if** $b = 1$
14 :	remove all $(j, *)$ from B	14 :	Remove all $(j, *)$ from B
15 :	**else**	15 :	**else**
16 :	Remove all $(j, *)$ from B but the last	16 :	Remove all $(j, *)$ from B but the last
17 :	Replace it with (j, β)	17 :	Replace it with (j, β)
18 :	$\tilde{\mathtt{s}} \leftarrow \mathsf{FakeCred}(\mathtt{s}^j)$	18 :	
19 :	$\mathcal{A}(\tilde{\mathtt{s}})$	19 :	
20 :	**for** $(i, \alpha) \in B$ **do**	20 :	$(\nu_i)_{i \in \mathbb{V}_{\mathcal{A}}}, \beta' \leftarrow \mathcal{A}(\lvert B\rvert)$
21 :	$\mathbb{M} \leftarrow \mathcal{A}(\mathbb{B})$	21 :	**if** $b = 1$ and $\beta \neq \emptyset$
22 :	$\mathbb{B} \leftarrow \mathbb{B} \cup \{\mathtt{m} \in \mathbb{M} \mid \mathtt{m} \text{ valid}\}$	22 :	$B \leftarrow B \cup \{(j, \beta')\}$
23 :	$\mathbb{B} \leftarrow \{\mathsf{Vote}(c_i, \alpha, \mathtt{pk})\}$	23 :	$B \leftarrow B \cup \{(i, \nu_i) \mid i \in \mathbb{V}_{\mathcal{A}}, \nu_i \in [n_C]\}$
24 :	$\mathbb{M} \leftarrow \mathcal{A}(\mathbb{B})$	24 :	
25 :	$\mathbb{B} \leftarrow \mathbb{B} \cup \{\mathtt{m} \in \mathbb{M} \mid \mathtt{m} \text{ valid}\}$	25 :	
26 :	$X, \Pi \leftarrow \mathsf{Tally}^{\mathcal{A}}(\mathbb{B}, R, \mathtt{pk}, \{h_i, s_i\}, t)$	26 :	$X \leftarrow \mathtt{result}(\mathtt{cleanse}(B))$
27 :	$b' \leftarrow \mathcal{A}()$	27 :	$b' \leftarrow \mathcal{A}(X)$
28 :	**return** $b = b'$	28 :	**return** $b = b'$

Figure 6.1: Security game of coercion-resistance. λ is the security parameter, n_T the number of talliers, t the threshold, n_V the number of voters, $n_{\mathcal{A}}$ the number of corrupted voters, n_C the number of voting options and D the distribution.

Informally, in the real game, the adversary takes part of the setup process (line 2) and decides the set of voters $\mathbb{V}_{\mathcal{A}}$ it controls and the coercion target (lines 4-5), receiving its index j and its voting choice β. Afterwards, votes are drawn according to a distribution D and added to the list B, containing all the votes in order, this models the voting process, see Observation 6.1 for further details. Lines 13-19 model the coercion: if $b = 1$, the coerced voter obeys, hence any vote from j is removed from B and the real credential $\mathtt{s}^j$ is handled to the adversary. If $b = 0$ the

voter follows the evasion strategy, i.e. they cast a vote for their intended preference β and give to the adversary a fake credential $\tilde{\mathtt{s}} = \mathsf{FakeCred}(\mathtt{s}_j)$.

Votes are then added to $\mathbb{B}$, according to the sequence B (lines 20-25). After each vote the adversary is allowed to see the board and add votes. Lastly $\mathcal{A}$ participate in the tally, controlling at most $t-1$ trustees, and finally the adversary guesses whether the evasion strategy was followed or not.

In the ideal world, the adversary only selects the set of voters $\mathbb{V}_{\mathcal{A}}$ it controls and the coercion target (4- 5). Then votes from $\mathbb{V}_{\mathcal{A}}$ (line 20) and, possibly, the coerced votes (line 21-22) are directly added to B. Then B is handled to the tally functionality that publishes the result of the election X, without revealing anything else.

Observation 6.1. Notice that in Definition 6.2 and also in Definition 6.1 we avoid to discuss how each voter receive its own credential $\mathtt{s}$ from the registrars. Indeed, this is a separate issue we talk about later, since it is crucial when discussing about the evasion strategy and the protocol $\mathsf{FakeCred}$. However, it requires some extra notion and can be vastly different between voting protocols, since it often relies on external authentication methods.

Moreover, this allows for a more abstract definition, since the "real" voting process is replaced by a random distribution that outputs all the ballots.

6.2.3 Cryptographic Primitives

We now list some of the cryptographic primitives needed for our protocol.

ElGamal Encryption Scheme. Due to its homomorphic properties, the ElGamal encryption scheme [Gam85] is a popular choice for designing voting schemes.

Let $\mathbb{G}$ be a multiplicatively written group of order q, with generator g, for which solving the Decisional Diffie Hellman (DDH) problem is hard. The private key $\mathtt{sk}$ is sampled at random from $\mathbb{Z}_q$, while the public key $\mathtt{pk}$ is $g^{\mathtt{sk}}$. The encryption of a message m is defined as $\mathsf{Enc}(m, \mathtt{pk}; r) = (g^r, g^m \cdot \mathtt{pk}^r) \in \mathbb{G}^2$ where $r \in \mathbb{Z}_q$ is a random value. We omit the randomness when not explicitly necessary.

Let $E_0 = (1,1)$, $E_1 = (1,g)$, and $E_{-1} = (1,g^{-1})$ be the respective encryptions of $0, 1, -1$ with randomness 0. Re-encryption can be done by multiplying a ciphertext by an encryption of 0. In particular, let $X \in \mathbb{G}^2$ be an ElGamal ciphertext, then we define $\mathsf{ReEnc}(X, \mathtt{pk}; r) = X \cdot \mathsf{Enc}(0, \mathtt{pk}; r)$, where the multiplication operation is component-wise.

For a number n_T of election trustees, we use a (t, n_T) threshold version of ElGamal, so $\mathtt{pk}$ is produced via a distributed key generation, and a minimum of $t+1$ parties are required to jointly decrypt.

Designated-Verifier Zero-Knowledge Proof. Similarly to JCJ and CHide, our protocol uses Designated-Verifier Zero Knowledge Proofs (DVZKPs) [JSI96]. Roughly speaking, a DVZKP is a zero-knowledge proof (ZKP) in which only the verifier designated by the prover is able to be convinced about the correctness of the proof. In particular the verifier V holds a key pair. Using the public key, the prover produces a proof for a statement, such that only V is convinced that the statement is true. This is achieved by allowing V to produce fake but valid DVZKPs for any statement, using their private key. The basic construction for designing DVZKP for a relation $\mathbb{L} \subseteq \mathbb{W} \times \mathbb{Y}$ is to consider the a new relation $\mathbb{L}' \subseteq \mathbb{W}' \times \mathbb{Y}'$ such $((w, \mathtt{sk}_V), (y, \mathtt{pk}_V)) \in \mathbb{L}'$ if and only if "$(w, y) \in \mathbb{L}$ or $(\mathtt{sk}, \mathtt{pk})$ is a valid secret-public key pair" . In this way, the original prover can prove that $(w, y) \in \mathbb{L}$, since it does not know the secret key $\mathtt{sk}_V$ of the verifier. Later, the verifier can produce a proof for any $(x', w) \in \mathbb{W} \times \mathbb{Y}$ (thus even for $(x', w) \notin \mathbb{L}$) since it knows $\mathtt{sk}_V$ and thus can prove the second branch of the "or" .

In particular, the usage of a DVZKP instead of a traditional ZKP is crucial for the evasion strategy, since it allows voters to be sure about the credentials received and, at the same time, they are able to produce fake credentials alongside fake proofs to hand over in case of attacks.

Circuits over encrypted bits. The basic building block for our tallying algorithm is the CGate protocol, originally presented in [ST04], in the re-randomized version [CGY22a]. Informally, on input of two encryptions X, Y of x and y, respectively, with $y \in \{0, 1\}$ it outputs a ciphertext Z which is the encryption of xy. If both x and y are bits, this allows to compute the conjunction And. Since the Not operator can be computed as $\mathsf{Not}(X) = E_1 \cdot X^{-1}$, every other Boolean operator can be easily implemented by combining these two.

In particular we need equality $\mathsf{Eq}(X, Y) = \mathsf{Not}(XY/\mathsf{CGate}(X, Y)^2)$ and a less-than operator $\mathsf{Less}(X, Y) = Y/\mathsf{CGate}(X, Y)$. These allow to build a circuit able to compute the less-than operator over strings encrypted bit by bit. Indeed, let a, b be two $k-$bit values and $A_1, ..., A_k$ and $B_1, ..., B_k$ their bitwise encryptions. To check securely compute $a < b$ we use this recursive formula: $L_0 = 0$ and

$$L_i = \mathsf{Less}(A_i, B_i) \cdot \mathsf{CGate}(L_{i-1}, \mathsf{Eq}(A_i, B_i)) \tag{6.2.1}$$

for $i = 1, ..., k$.At the end L_k is the encryption of $a < b$.

In [ST04] the authors proved that the CGate algorithm is UC-secure.

Distributed Random Bit Generation. In the same way as CHide, credentials are generated by a particular set of authorities and are encrypted bit by bit. In order to do so, they need to use a distributed random bit generation protocol. In particular, they jointly produce an encrypted bit $\mathsf{Enc}(b, \mathtt{pk})$, for which each participant knows only a share b_i of b. Furthermore, the transcript of the protocol communication is used as a DVZKP for the correctness of the protocol. We use the $\mathsf{RandBit}$ protocol proposed in [CGY22b].

Mixnet. Mixnets are widely used in secure e-voting systems. Informally, a mixnet allows a set of participants to shuffle and re-encrypt a set of ciphertexts, without needing to know the secret key (or a secret sharing of it). On a high level, participants privately shuffle all inputs and eventually publish them re-encrypted in random order. Informally, we say that a mixnet is secure if, given at least one honest participant, the permutation from the input to the output remains secret for all the participants involved.
In the protocol we will need a verifiable mixnet, that ensures the correctness of the output (i.e. the output is indeed a permutation and re-encryption of the input). A suitable candidate for our protocol is the mixnet presented in [Wik09] or [Wik04], which is UC-secure.

6.3 Protocol Description

Our protocol is an improved version of the original CHide, presented in [CGY22b][2]. The participants, the Setup phase, the Registration phase and the Voting phase are essentially the same, while we changed both the Tallying phase, to substantially reduce the computational complexity.

The main difference is that for each ballot, CHide requires to compare the encrypted credential to every successive one and to every credentials in the register (thus having quadratic complexity), while in our protocol we first perform a sorting algorithm on the encrypted votes. At the end of the protocol, votes with the same credentials and authorized credentials are consecutive, allowing the election authorities to recognize valid votes faster.

[2]Authors of CHide independently updated their work, using a similar technique to achieve comparable performances. More about this in Section 6.5

6.3.1 Overview

Setup Phase A security parameter λ is chosen. The election trustees jointly run the distributed key generation protocol presented in [Gen+07a], obtaining a public key $\mathtt{pk}$ at the appropriate security level. Each trustee publishes a commitment h_i to its private share of $\mathtt{pk}$ on the public board, as well as $\mathtt{pk}$. The private shares are denoted $\mathtt{sk}_i$ for $i = 1, ..., n_T$.

Registration phase As in CHide, credentials are created by a designated set of registrars, encrypted bitwise, sent to the voters and published on the public board. Let $\mathtt{s} = (\mathtt{s}^1, \ldots, \mathtt{s}^k)$ be the k-bit credential of voter V and let $\mathtt{S} = (\mathtt{S}^1, ..., \mathtt{S}^k)$ the bitwise encrypted values published on the board. Each credential is sent privately to the voter, with designated zero-knowledge proofs to guarantee voters that their credential is valid.[3] Let $\mathbb{R}$ be the list of all the authorized credentials, we sometimes call $\mathbb{R}$ as the "public credentials" .

Voting Phase To cast a vote for candidate ν, voter V computes an encryption of their voting choice $\mathtt{ct}^1 = \mathsf{Enc}(\nu, \mathtt{pk})$ and a bitwise encryption of their credential $\mathtt{ct}^2 = (\mathsf{Enc}(\mathtt{s}^1, \mathtt{pk}), ..., \mathsf{Enc}(\mathtt{s}^k, \mathtt{pk}))$, as well as two ZKPs: one to prove that $\mathtt{ct}^2$ contains encryptions of bits, and a second one proving knowledge of the randomness used in $\mathtt{ct}^1$ and that ν is a valid voting option. These ZKPs are also used to link together $\mathtt{ct}^1$ and $\mathtt{ct}^2$, making the tuple $C = (\mathtt{ct}^1, \mathtt{ct}^2)$ non-malleable. The tuple and the corresponding ZKPs are published on the public board using an anonymous channel.

During the Voting Phase, each voter can vote multiple times, for simplicity the policy we implement is that only the last vote is counted. During this step the auditors verify the uniqueness of each ballot and that every ZKP is valid.

Tally Phase Once the Voting Phase is finished, the election trustees count the votes. Let $\mathbb{B} = \{C_i\}$ the list of all the votes, listed in chronological order, and $\mathbb{R} = \{\mathtt{S}_i\}$ the public credentials.

First of all, all the invalid votes marked by the auditors are discarded. Then the election trustees parse each element $\mathtt{e}_i$ of $\mathbb{B}||\mathbb{R}$ as $(\mathtt{Data}_i, \sigma_i, \mathtt{f}_i, \mathtt{c}_i)$ where:

- $\mathtt{Data}_i \leftarrow \mathtt{ct}_i^1$ if $\mathtt{e}_i \in \mathbb{B}$; otherwise $\mathtt{Data}_i$ is set to be a random encryption.

[3] Voter authentication is out of the scope of this paper but, for example, could be done via a digital signature by the user with a long-term key pair.

- $\sigma_i \leftarrow \mathtt{ct}_i^2$ if $\mathtt{e}_i \in \mathbb{B}$; $\sigma_i \leftarrow \mathtt{S}_i$ otherwise.
- $\mathtt{f}_i \leftarrow \mathsf{Enc}(0, \mathtt{pk})$ if $\mathtt{e}_i \in \mathbb{B}$; $\mathtt{f}_i \leftarrow \mathsf{Enc}(1, \mathtt{pk})$ otherwise.
- $\mathtt{c}_i$ is the bitwise encryption of an increasing counter and represents the chronological order of the votes.

Then the trustees apply a mixnet protocol (for example [Wik09] or [Cha81]) on $\mathbb{B}||\mathbb{R}$ and produce a verification transcript. For simplicity we will refer to each element after the mixnet using the same notation as before, i.e. each element is in the form $(\mathtt{Data}_i, \sigma_i, \mathtt{f}_i, \mathtt{c}_i)$.

The election trustees perform a sorting algorithm on the set, with the following relation:

$$\mathtt{e}_i <_{\mathtt{Tally}} \mathtt{e}_j \Leftrightarrow \mathsf{Dec}(\sigma_i)||\mathsf{Dec}(\mathtt{f}_i)||\mathsf{Dec}(\mathtt{c}_i) <_{\mathtt{Lex}} \mathsf{Dec}(\sigma_j)||\mathsf{Dec}(\mathtt{f}_j)||\mathsf{Dec}(\mathtt{c}_j) \quad (6.3.1)$$

where, with an abuse of notation, $\mathsf{Dec}(\sigma_i)$ and $\mathsf{Dec}(\mathtt{c}_i)$ denote the concatenation of the decryptions of every ciphertext in σ_i and $\mathtt{c}_i$ and $<_{\mathtt{Lex}}$ is the lexicographical order. It is important to note that:

- If two votes $\mathtt{e}_i,\ \mathtt{e}_j \in \mathbb{B}$ have the same credentials, then they are sorted chronologically thanks to the counters $\mathtt{c}_i, \mathtt{c}_j$. Moreover if $\mathtt{e}_h$ is such that $\mathtt{e}_i <_{\mathtt{Tally}} \mathtt{e}_h <_{\mathtt{Tally}} \mathtt{e}_j$ then $\mathtt{e}_h$ has the same credential of both $\mathtt{e}_i$ and $\mathtt{e}_j$.
- If $\mathtt{e}_i \in \mathbb{B}$ and $\mathtt{e}_j \in \mathbb{R}$ have the same credentials (i.e. $\mathtt{e}_i$ is a ballot cast with an authorized credential) then $\mathtt{e}_i <_{\mathtt{Tally}} \mathtt{e}_j$. Moreover if $\mathtt{e}_h$ is such that $\mathtt{e}_i <_{\mathtt{Tally}} \mathtt{e}_h <_{\mathtt{Tally}} \mathtt{e}_j$ then $\mathtt{e}_h$ has the same credential of both $\mathtt{e}_i$ and $\mathtt{e}_j$.
- No two distinct elements $\mathtt{e}_i, \mathtt{e}_j$ will compare equally in this ordering, thanks to the counter $\mathtt{c}_i$ in each ballot.

Informally, the ordered list is formed by blocks of consecutive ballots cast with the same credential, ending with the corresponding element in $\mathbb{R}$ if they were made with an authorized one.

In order to prove the correctness of the sorting algorithm, the trustees add the proofs of the correctness of every CGate computation as well the correctness of the decryption.

After the sorting, for every pair of consecutive elements $(\mathtt{e}_i, \mathtt{e}_{i+1})$ in the ordered list, the election trustees check whether $\mathsf{Dec}(\sigma_i) = \mathsf{Dec}(\sigma_{i+1})$. This produces an encrypted bit $\mathtt{I}_i^1$. Trustees the compute $\mathtt{I}_i = \mathsf{CGate}(\mathtt{I}_i^1, \mathtt{f}_{i+1})$, obtaining encryption of the conjunction between the bits encrypted in $\mathtt{I}_i^1$ and $\mathtt{f}_{i+1}$. In particular $\mathtt{I}_i$ is

$\mathsf{Tally}(\mathbb{B}, \mathbb{R}, \{\mathtt{sk}_i\})$:

1 : Parse each element in $\mathbb{B}||\mathbb{R}$ as described

2 : $\mathbb{B}_{\mathsf{Unsort}}, \Pi_1^{\mathsf{Mixnet}} \xleftarrow{\$} \mathsf{Mixnet}(\mathbb{B}||\mathbb{R})$

3 : $\mathbb{B}_{\mathsf{Sort}}, \Pi^{\mathsf{Sort}} \leftarrow \mathsf{Sort}(\mathbb{B}_{\mathsf{Unsort}})$

4 : **for** $\mathtt{e}_i \in \mathbb{B}_{\mathsf{Sort}}$ **do**

5 : $\quad \mathtt{I}_i \leftarrow \mathsf{CGate}(\mathsf{Eq}(\sigma_i, \sigma_{i+1}), \mathtt{f}_i)$

6 : $\quad \mathtt{Data}^i \leftarrow \mathsf{CGate}(\mathtt{Data}_i, \mathtt{I}_i)$

7 : $\mathbb{B}_{\mathsf{Mixnet}}, \Pi_2^{\mathsf{Mixnet}} \xleftarrow{\$} \mathsf{Mixnet}(\mathbb{B}_{\mathsf{Sort}})$

8 : **return** $X = \mathsf{Dec}(\mathtt{Data}), \mathtt{Data} \in \mathbb{B}_{\mathsf{Mixnet}}$ and $\Pi = \Pi_i^{\mathsf{Mixnet}}||\Pi^{\mathsf{Sort}}||\Pi^{\mathsf{CGate}}$

9 : // where Π^{CGate} is the verification transcript of all CGate computations in the cycle.

Figure 6.2: Tally protocol. The sorting algorithm can be any comparison sort, see Observation 6.2.

an encryption of 1 if and only if $\mathtt{e}_i$ is a vote with a valid credential and the last vote with that credential. At this point the trustees multiply the plaintext in $\mathtt{I}_i$ and $\mathtt{Data}_i$, computing $\mathsf{CGate}(\mathtt{Data}_i, \mathtt{I}_i)$, apply a second mixnet on the resulting list, and decrypt every vote. Figure 6.2 shows the tally protocol.

Observation 6.2. The `Sort` algorithm can be any suitable comparison sort, such as Quicksort or Mergesort, thanks to the mixnet (the stability property is guaranteed by the flag $\mathtt{f}_i$ and the counter $\mathtt{c}_i$, that also ensures that all elements are different). The crucial part is the evaluation of the comparison, done using the circuit presented in Section 6.2.3 in Equation (6.2.1).

The result of every comparison can then be decrypted and used according to the chosen sorting algorithm, without leaking anything because of the mixing. Sorting $\mathbb{B}||\mathbb{R}$ without mixing would leak the number of votes between two authorized credentials and could lead to potential attack (for example, if an attacker votes with a fake credential that is greater than any authorized one it would easily detect the lie). In fact, due to the mixnet, any adversary would have no information about the terms of each comparison, thus the result of the comparison is meaningless and can be simulated, as shown in the next section.

Evasion Strategy To evade coercion a voter can simply lie about their credential s, giving a fake credential $\bar{s}$ to the coercer, and manipulating the DVZKP accordingly.

In this way, voters are also able to vote with their correct credentials. So, the algorithm FakeCred of Definition 6.2, simply output a random string $\bar{s}$ with the correct length and produce a DIVZKP about its correctness.

6.4 Security Proof

The proof is very similar to the one presented in [CGY22b]. We consider a distribution $\mathbb{D}$ of sequence of pairs (j, ν) where j is a voter and ν is a voting option. Additionally, fake votes are modeled as pairs where $j \notin [1, n_V]$.

Theorem 6.1. *Under the DDH assumption and in the Random Oracle Model, the voting system presented in Section 6.3 is coercion resistant according to Definition 6.2.*

Proof. Let $\mathcal{A}$ be an adversary for the real game. We give to $\mathcal{A}$ the power to impersonate t among n_T election trustees and up to $n_{\mathcal{A}}$ voters. Our goal is to build an adversary $\mathcal{S}$ for the ideal game. $\mathcal{S}$ use $\mathcal{A}$ as a subroutine, simulating the real game, in order to win the ideal one. In particular, $\mathcal{S}$ control the remaining $n_T - t$ trustees and need to simulate the Setup as well as the Tally.

First of all, $\mathcal{S}$ and $\mathcal{A}$ run the `Setup` algorithm to generate a common public key `pk`, secret shares of the private key $\mathtt{sk}_1, ..., \mathtt{sk}_{n_T}$ and the public commitments $h_1, ..., h_{n_T}$. During this step $\mathcal{S}$ is also able to reconstruct the secret key `sk` by extracting $\mathcal{A}$'s secrets.

Then $\mathcal{S}$ follows the real game normally, getting $\mathbb{V}_{\mathcal{A}}, j$ and β from the adversary. In the ideal game $\mathcal{S}$ sends the same choices for $\mathbb{V}_{\mathcal{A}}, j, \beta$.

When asking for the credential of voter j, $\mathcal{S}$ provides to $\mathcal{A}$ the real credential $\mathtt{s}^j$. From the ideal game $\mathcal{S}$ learns the size $|B|$ of the ideal board and uses it to simulate the voting process. For $|B|$ times:

- $\mathcal{S}$ calls $\mathcal{A}$ with input $\mathbb{B}$ getting $\mathbb{M}$
- $\mathcal{S}$ decrypts all the valid votes and credentials in M. For every authorized credential $\mathtt{s}^i$, $\mathcal{S}$ saves the tuple $(\mathtt{s}^i, \nu)$ or updates a previously saved $(\mathtt{s}^i, \nu')$.
- $\mathcal{S}$ adds all valid ballots in $\mathbb{M}$ to $\mathbb{B}$

- $\mathcal{S}$ chooses a random voter and a valid voting option and casts a valid vote, adding it to $\mathbb{B}$.

At the end of the voting process, $\mathcal{S}$ add the same votes in the ideal game.

$\mathcal{S}$ learns X and its at the end of the ideal game and use it to simulate the tallying process in the real game:

- $\mathcal{S}$ runs the first mixnet for the honest authorities, while $\mathcal{A}$ uses the dishonest ones.
- To perform the sorting, $\mathcal{S}$ simulates all the CGate operations. This can be done since CGate is a UC-secure protocol, as shown in [CGY22a]. $\mathcal{S}$ also simulates the decryption step and thus randomly sorts the list.
- $\mathcal{S}$ runs the second mixnet for the honest authorities, while $\mathcal{A}$ uses the dishonest ones.
- $\mathcal{S}$ chooses $|X|$ entries at random and simulates its partial decryption: every entry not chosen is decrypted to 0, while such $|X|$ entries are decrypted such that the result is exactly X.

At this point $\mathcal{A}$ makes its guess b and $\mathcal{S}$ forward the same guess in the ideal game.

The differences between a real execution and the simulation are:

- In the real game $\mathcal{A}$ can get either the real credential $\mathtt{s}^j$ or a fake one. In the simulation $\mathcal{A}$ always receives $\mathtt{s}^j$. Since in both the real and ideal worlds fake credentials have uniformly random distribution and the DVZKP could be simulated, $\mathcal{A}$ can only distinguish a real execution from a simulated one if and only if it is able to distinguish whether the received credential is a plaintext of one of the encrypted credentials in $\mathbb{R}$ or not. Since the ElGamal encryption is IND-CPA secure under the DDH Assumption this is impossible, and thus the simulation is indistinguishable from a real execution.
- During the simulation of the voting loop $\mathcal{S}$ adds random ballot, while in the real game ballots are drawn according to $\mathbb{D}$. As before, since the ballots are encrypted, the simulation is indistinguishable from the real game under the DDH Assumption. Notice that decrypting the votes is not possible, since

after the eventual decryption $\mathcal{A}$ would learn the real distribution of the votes. For this reason the votes are not decrypted and instead $\mathcal{S}$ use the same result of the ideal game, as we discuss later.

- During the tally $\mathcal{S}$ simulates the execution of the CGate protocol. By the security of the CGate, the simulation is indistinguishable from the real game[CGY22a].

- In the real game, the ballots are sorted as per relation 6.3.1, while in the ideal game each comparison is simulated and thus the order is random. Being able to distinguish between the correct order and a fake one would mean either being able to distinguish the ballots, that is unfeasible due to the IND-CPA security of the encryption scheme, or being able to recognize the ballots after the mixnet, that is unfeasible thanks to the security of the mixnet.

- In the simulation the result always include all the last valid ballots cast by honest voters. In a real execution the adversary may change it by casting ballots on behalf of an honest voter. However, to do so, the adversary must be able to create a valid ZKP about the credential used, and this is unfeasible.

- $\mathcal{S}$ simulates the decryption protocol at the end. This simulation is indistinguishable from the real world under the DDH assumption in the ROM.

□

6.4.1 Note about Privacy and Verifiability

In this chapter we focus our attention to coercion resistance, however privacy and verifiability are crucially important as well. We provide here an informal discussion about them, since the proofs are trivial. Before discussing them however, we first need to define IND-PA0 security, that is indistinguishability under parallel chosen plaintext attack.

Definition 6.3 (indistinguishability under parallel chosen ciphertext attacks)**.** Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Consider the following experiment:

$\mathsf{Exp}^{\mathsf{ind-pa0}}_{\mathsf{Enc},\mathcal{A}}(\lambda)$

$\mathtt{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$

$(\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$

$(\mathtt{m}_0, \mathtt{m}_1) \leftarrow \mathcal{A}(\lambda, \mathtt{pk})$

$\mathtt{ct} \xleftarrow{\$} \mathsf{Enc}(\mathtt{pk}, \mathtt{m}_b)$

$\{\mathtt{ct}'_i\} \xleftarrow{\$} \mathcal{A}(\mathtt{ct})$

$\{\mathtt{m}'_i\} \leftarrow \{\mathsf{Dec}(\mathtt{ct}_i, \mathtt{sk})\}$

$b' \leftarrow \mathcal{A}\{\mathtt{pk}, \mathtt{ct}, \mathtt{m}_0, \mathtt{m}_1, \{\mathtt{ct}'_i\}, \{\mathtt{m}'_i\}\} b' \leftarrow \mathbf{Adv}(\mathtt{pk}, \mathtt{ct}, \mathtt{m}_0, \mathtt{m}_1)$

return $b = b'$

Define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathsf{ind-pa0}}_{\mathsf{Enc},\mathcal{A}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{ind-pa0}}_{\mathsf{Enc},\mathcal{A}}(\lambda) = 1)$$

We say that $(\mathsf{Enc}, \mathsf{Dec})$ is indistinguishable under chosen plaintext attacks if and only if $\mathbf{Adv}^{\mathsf{ind-pa0}}_{\mathsf{Enc},\mathcal{A}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

It is possible to show that the voting map of Section 6.3

$$(\mathtt{s}, \nu) \to (\mathsf{Enc}(\nu, \mathtt{pk}), \mathsf{Enc}(\mathtt{s}^1, \mathtt{pk}), ..., \mathsf{Enc}(\mathtt{s}^k, \mathtt{pk}), \Pi_1, \Pi_2)$$

where Π_1 is a proof that $\mathsf{Enc}(\mathtt{s}^1, \mathtt{pk}), ..., \mathsf{Enc}(\mathtt{s}^k, \mathtt{pk})$ are encryptions of bit, Π_2 is a proof that ν is a valid voting option and Π_1 and Π_2 are linked together to make the tuple non malleable is an IND-PA0 encryption scheme.

Privacy informally, privacy means that it is impossible to guess which option a voter chose. Formally, in the privacy game, the adversary $\mathcal{A}$ chooses two voting options ν_0, ν_1 and an "observed voter" v_o, who picks a random bit b and vote ν_b. The adversary, controlling $t-1$ trustees, wins if guesses b, see Appendix C of [CGY22b]. The proof is an immediate reduction to the IND-PA0 security of the encryption protocol used, indeed, suppose that $\mathcal{A}$ is an adversary that wins the privacy game non negligible advantage, we show how to build an adversary $\mathcal{S}$ for the IND-PA0 that wins with non negligible advantage. Indeed, $\mathcal{S}$ chooses ν_0, ν_1 as the plaintext for the IND-PA0 game. Before the tally, $\mathcal{S}$ use the whole board $\mathbb{B}$ as query in the IND-PA0 game, except for the vote casted by v_o, for which $\mathcal{S}$ choose randomly one of the two voting option. At this point, $\mathcal{S}$ can simulate the

whole tally knowing the end result and guesses whatever bit b $\mathcal{A}$ guesses. When $\mathcal{S}$ choose correctly the vote for v_o the simulation is perfect and $\mathcal{S}$ win every time $\mathcal{A}$ win, which happens with probability $\frac{1}{2} + \varepsilon$ for a non negligible ε. If it chooses wrongly, that happens half of the time, $\mathcal{S}$ wins with probability $\frac{1}{2}$. Overall the winning probability of $\mathcal{S}$ is $\frac{1}{2} + \frac{\varepsilon}{2}$ that is non negligible.

Verifiability the universal verifiability is granted by the ZKPs produced by the trustees and by the honesty of the bulletin board. Cast-as-intended instead is more tricky, since it should require to design algorithms for voters to inquire their own devices. Many protocols, such as the Benaloh challenge [Ben06], are suitable, however they usually relies more on " a responsible behaviour"from the user, instead of solid security proof. Moreover they often fall short when analyzed from a game theory standpoint, like in [Jam23] where Jamroga suggests that the optimal strategy is to (almost) always ignore the audit step and cast the vote immediatly. For this reason we heve left cast-as-d out of the discussion and therefore assumed the existence of a trusted device.

6.5 Performance

The main goal of the paper is to improve the performance of the tallying protocol in CHide and JCJ. This is achieved by performing a preliminary sorting step, that reduces the complexity of the tallying from quadratic to quasi-linear.

A performance comparison between our protocol and CHide can be performed by couting the number of CGate operations. We use as example the recent Estonian election, where for the first time more than half of the voters used a remote voting system, for a total of a little more than 3×10^5 valid votes. [Eest]. Since the Estonian voting system does not track the number of revotes and removed ballots, we suppose that a total of 6×10^5 votes where submitted (i.e. only half of the total votes are valid votes) and that every registered voter voted (i.e. the list of authorized credentials $|R|$ contains 3×10^5 registered credentials). In the following k is the bit-length of voters' credentials.

Each comparison during the sorting algorithm requires $3k$ CGate computations, as explained in Section 6.2. Thus for the sorting phase our algorithm requires $3k(9\times10^5\times\log_2(9\times10^5)) \approx 54k\times10^6$ CGate computations and 18×10^6 decryptions. Then, to compute the check bit $\mathtt{I}^i$ for every pair of votes the protocol requires $2k \times 9 \times 10^5$ CGate computations. In total, our protocols require around $56k \times 10^6$ CGate computations, 18×10^6 intermediate decryptions and two mixnet applications.

Table 6.1: Performance comparison between CHide and our protocol with respect of the security parameter k.

	CGate	Mixnet	Preliminary Decryptions
CHide	$720k \times 10^9$	1	-
Our Protocol	$56k \times 10^6$	2	18×10^6

The CHide protocol instead requires to check that the credentials of each casted votes unique, comparing it with each subsequent vote, and that it is an authorized one, comparing it with every registered credentials. Each equality operation requires only k CGate computations, thus for finding duplicates CHide requires $k(2 \times 6 \times 10^5 \times 3 \times 10^5) = 360k \times 10^9$ CGate computations and the same number of computations for checking authorized credentials. Then a mixnet is applied and the votes are decrypted. Thus, CHide requires a total of $720k \times 10^9$ CGate computations and one mixnet application.

Recent Updates. The CHide preprint was independently updated by the authors to address the quadratic complexity of the protocol. Their solution is quite similar to our solution, leveraging the CGate protocol to sort all the votes and achieve a quasi-linear complexity.

While sharing the same philosophy and the same asymptotic complexity, the two protocols have a meaningful difference that could lead to different running times. Updated CHide avoids the preliminary mixnet by using a swap operation between ciphertexts, instead of simply decrypting the output of each comparison. This restricts their choice of sorting algorithms to be *data-oblivious*, with complexity $O(n \log^2 n)$. Moreover, instead of using a single bit, they use a fixed "special" counter for registered credentials, thus performing more comparisons in the last part of the tally (the computation of $\mathtt{I}_i$, as per our notation).

6.5.1 Comparison with related works

During the last years many different coercion resistant protocols have been proposed, usually with the goal of reducing the quadratic complexity that is typical of protocols descending from JCJ. Notable examples of more efficient protocols are VoteAgain [LQT20], AFT [AFT10], Athena [Smy19] and protocols based on hash tables like [Røn+20] and [WAB07]. The linear-time version of the JCJ protocol proposed in [Røn+20] also uses fully homomorphic encryption. Table 6.1 summarizes the comparison between this and related work in terms of security and

Table 6.2: Comparison with other coercion resistant protocols.

Protocol	Complexity	Security
JCJ[JCJ10]	$O(n^2)$	JCJ
Civitas [CCM08]	$O(n^2)$	JCJ
AKLM [Ach+15]	$O(n^2)$	AKLM
Revote [LHK16]	$O(n^2)$	AKLM
CHide[CGY22b]	$O(n^2)$ or $O(n \log^2 n)$	CHide
VoteAgain [LQT20]	$O(n \log n)$	VoteAgain
AFT [AFT10]	$O(n)$	JCJ
Athena [Smy19]	$O(n)$	JCJ + Dups
Hash-based [Røn+20; WAB07]	$O(n)$	JCJ + Dups
This work	$O(n \log n)$	CHide

In the table, the security levels are defined as:
complexity.

- JCJ is the security level achieved by the original JCJ protocol.
- JCJ+Dups is at lower security level than JCJ, where the number of votes for each credential also leak.
- AKLM is at lower security than JCJ, in which it is assumed that voters revote at the end of the voting period to escape from adversarial control.
- CHide is the security level achieved by CHide, higher than JCJ.
- VoteAgain follows its own coercion resistance definition introduced in [LQT20] and it is not comparable with the others.

From the state of the art, achieving a better or equivalent complexity than our protocol requires to either change the security definition (as per [LQT20]) or to increase the leakage.

6.6 Conclusions and future works

In this work we presented an enhanced version of CHide, that drastically reduces the computational complexity of the tallying from $O(n^2)$ to $O(n \log n)$, which is currently the best efficiency among voting protocols satisfying a stronger notion of coercion resistance.

A possible way to speed up the tally even further is amortizing the process through the whole voting phase, instead of waiting until the end of the election. A possible approach would consist of using a bucket sorting algorithm, like the one presented in [Ash+20]. As votes come in, they are assigned to buckets. When the first two buckets are full, the first step of bucket sorting is performed. When the next two buckets are full, the authorities perform the first step of the sorting process on them and the second step on the whole for the bucket, and so on. While maintaining the same asymptotic complexity, this approach could lead to a vastly reduced delay between the end of the voting phase and the publication of the result. However, bucket sorting is usually susceptible to "overflow" attacks. Indeed, typical bucket sorting algorithms like [Ash+20] allow for a fixed maximum number of elements in each bucket, thus an attacker could vote multiple time with the same credential, causing the corresponding bucket to overflow and making the sorting fail. In the end we not find any solution to this problem but it is a topic worthy of further examination.

Unfortunately, our protocol still has the same issue of CHide regarding the dimension of the credentials, that are encryptions of individual bit instead of a single encrypted string. The bitwise encryption is required to realize a secure tally, since it allows to multiply plaintexts using the CGate algorithm. As a way to reduce the credentials length we are currently studying class group encryption (CGE). Introduced in 2015 by G. Castagnos and F. Laguillaumie [CL15], CGE is the first discrete logarithm based scheme that allows for an unlimited number of linear operations over plaintexts. Recently, the work by L. Braun et al. [BDO22] introduced the notion of threshold CGE, which allows efficient plaintext multiplication using a multiparty protocol. This feature is particularly helpful to design e-voting protocols since it allows for efficient equality testing, a crucial step for schemes that adhere to the security definition given by JCJ [JCJ10]. More in detail, we are currently adapting the mixnet protocol from [BG12] to be used with CGE and we are designing Zero Knowledge Proofs useful for the protocol. However, the main drawback of CGE is the lack of an "easy and natural" ordering algorithm. A work in progress version of this work was presented at CIFRIS23, as a short abstract.

Chapter 7

Conclusions

The recent surgence of new technologies like cloud computing and the Internet of Things paved the way for many new applications, such as privacy-preserving data analysis, online voting and distributed key management. The increasing request for more complex applications caused the need of of new MPC protocol.

In this thesis we presented result from three area of MPC: first, in Chapter 3 we study "abstract" MPC, studying a new heuristic for threshold signature design. Unlike in the centralised case, where the security of digital signatures is typically demonstrated according to well-established frameworks (mainly the Fiat-Shamir heuristic), in the MPC world each threshold signature security proof "follows its own scheme" , often adhering to different security definitions and flavour, from classical game based ones to the most recent UC ones. Our is aim is two-fold, on one hand we want to streamline the security analysis of many systems and on the other hand we want to provide a common framework for proving the security of the signature, that resembles as much as possible the classic Fiat-Shamir transform .

Next, in Chapter 4 and Chapter 5 we showed two threhsold signature, the first using classical assumption and the second one with post quantum security. Besides the importance of the two protocols on their own, these two chapter constitute a good example about bot the power and the current limitations of the heuristic proposed in Chapter 3. Indeed, while the in Chapter 4 we could prove the security of the signature very easily thanks, in Chapter 5 we could not use the results of Chapter 3, showing the unbalance between the necessary and sufficient conditions of our heuristic. Albeit very small, such unbalance is particularly meaningful for round robin threshold signature, that are very common in post quantum cryptography.

As an additional result, in Chapter 5 we also showed a group action based OT, which has competitive parameters with other state-of-the-art protocols. Unfortu-

nately, our construction does not fill the gap of having an OT based on black box usage usage group actions (in particular not commutative), which therefore still remains an open problem.

Lastly, in Chapter 6 we presented a new evoting protocol. Our protocol is an improved version of CHide, that drastically reduces the computational complexity from $O(n^2)$ to $O(n \log n)$, while keeping the highest possible security. Our work is, as the current literature, the best protocol from an asymptotic computational complexity standpoint, however it suffers from having an heavy memory complexity. For this reason we hinted some current work in progress that aim to reduce the burden of such large memory requirement using a new encryption techniques.

Bibliography

[Abd+02] M. Abdalla, J. H. An, M. Bellare, and C. Namprempre. "From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security". In: *Advances in Cryptology — EUROCRYPT 2002*. Ed. by L. R. Knudsen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 418–433.

[ABR23] D. F. Aranha, M. Battagliola, and L. Roy. *Faster coercion-resistant e-voting by encrypted sorting*. Cryptology ePrint Archive, Paper 2023/837. `https://eprint.iacr.org/2023/837`. 2023.

[Ach+15] D. Achenbach, C. Kempka, B. Löwe, and J. Muüller-Quade. "Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting". In: *USENIX Journal of Election Technology and Systems (JETS)* 3.2 (Aug. 2015), pp. 26–45.

[Adi08] B. Adida. "Helios: Web-based Open-Audit Voting". In: *USENIX Security Symposium*. USENIX Association, 2008, pp. 335–348.

[AFT10] R. Araújo, S. Foulle, and J. Traoré. "A Practical and Secure Coercion-Resistant Scheme for Internet Voting". In: *Towards Trustworthy Elections*. Vol. 6000. LNCS. Springer, 2010, pp. 330–342.

[Ala+] N. Alamati, L. De Feo, H. Montgomery, and S. Patranabis. "Cryptographic group actions and applications". In: *ASIACRYPT 2020*. Springer.

[Ala+18] M. Alam, N. Emmanuel, T. Khan, A. Khan, N. Javaid, K.-K. R. Choo, and R. Buyya. "Secure policy execution using reusable garbled circuit in the cloud". In: *Future Gener. Comput. Syst.* 87 (2018), pp. 488–501.

[Arn+01] P. Arnoux, V. Berthé, H. Ei, and S. Ito. "Tilings, Quasicrystals, Discrete Planes, Generalized Substitutions, and Multidimensional Continued Fractions". In: *Discrete Models: Combinatorics, Computation, and Geometry*. 2001. URL: https://api.semanticscholar.org/CorpusID:669324.

[Ash+20] G. Asharov, T. H. Chan, K. Nayak, R. Pass, L. Ren, and E. Shi. "Bucket Oblivious Sort: An Extremely Simple Oblivious Sort". In: *SOSA*. SIAM, 2020, pp. 8–14.

[ATBQ00] F. E. S. Arthur T. Benjamin and J. J. Quinn. "Counting on Continued Fractions". In: *Mathematics Magazine* 73.2 (2000), pp. 98–104. DOI: 10.1080/0025570X.2000.11996816.

[Bal+23] M. Baldi et al. *Matrix Equivalence Digital Signature*. Accessed: 2023-09-15. 2023. URL: https://www.less-project.com/LESS-2023-08-18.pdf.

[Bar+21] A. Barenghi, J.-F. Biasse, E. Persichetti, and P. Santini. "LESS-FM: fine-tuning signatures from the code equivalence problem". In: *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*. Springer. 2021, pp. 23–43.

[Bas+23] A. Basso, G. Codogni, D. Connolly, L. De Feo, T. B. Fouotsa, G. M. Lido, T. Morrison, L. Panny, S. Patranabis, and B. Wesolowski. "Supersingular curves you can trust". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 405–437.

[Bat+22a] M. Battagliola, A. Galli, R. Longo, and A. Meneghetti. "A Provably-Unforgeable Threshold Schnorr Signature With an Offline Recovery Party". In: *DLT@ITAcSEC*. 2022. URL: https://ceur-ws.org/Vol-3166/paper05.pdf.

[Bat+22b] M. Battagliola, R. Longo, A. Meneghetti, and M. Sala. "Threshold ECDSA with an Offline Recovery Party". In: *Mediterranean Journal of Mathematics* 19.1 (2022), pp. 1–29.

[Bat+22c] M. Battagliola, R. Longo, A. Meneghetti, and M. Sala. "Threshold ECDSA with an Offline Recovery Party". In: *Mediterranean Journal of Mathematics* 19.4 (2022).

[Bat+23a] M. Battagliola, G. Borin, A. Meneghetti, and E. Persichetti. *Cutting the GRASS: Threshold GRoup Action Signature Schemes*. Cryptology ePrint Archive, Paper 2023/859. https://eprint.iacr.org/2023/859. 2023. URL: https://eprint.iacr.org/2023/859.

[Bat+23b] M. Battagliola, R. Longo, A. Meneghetti, and M. Sala. "Provably Unforgeable Threshold EdDSA with an Offline Participant and Trustless Setup". In: *Mediterranean Journal of Mathematics* 20.5 (2023), p. 253.

[Bau+22] C. Baum, R. Jadoul, E. Orsini, P. Scholl, and N. P. Smart. "Feta: Efficient Threshold Designated-Verifier Zero-Knowledge Proofs". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. Association for Computing Machinery, 2022, 293–306. DOI: 10.1145/3548606.3559354. URL: https://doi.org/10.1145/3548606.3559354.

[BB17] M. Blanton and F. Bayatbabolghani. "Improving the Security and Efficiency of Private Genomic Computation Using Server Aid". In: *IEEE Security Privacy* 15.5 (2017), pp. 20–28. DOI: 10.1109/MSP.2017.3681056.

[BDN18] D. Boneh, M. Drijvers, and G. Neven. "Compact Multi-signatures for Smaller Blockchains". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by T. Peyrin and S. Galbraith. Cham: Springer International Publishing, 2018, pp. 435–464.

[BDO22] L. Braun, I. Damgård, and C. Orlandi. "Secure Multiparty Computation from Threshold Encryption based on Class Groups". In: *IACR ePrint Arch.* (2022), p. 1437. URL: https://eprint.iacr.org/2022/1437.

[Ben06] J. Benaloh. "Simple verifiable elections". In: EVT'06. USENIX Association, 2006, p. 5.

[Beu+21] W. Beullens, L. Disson, R. Pedersen, and F. Vercauteren. "CSI-RAShi: distributed key generation for CSIDH". In: *International Conference on Post-Quantum Cryptography*. Springer. 2021, pp. 257–276.

[BF24] M. Battagliola and A. Flamini. *Distributed Fiat-Shamir Transform*. Cryptology ePrint Archive, Paper 2024/214. https://eprint.iacr.org/2024/214. 2024. URL: https://eprint.iacr.org/2024/214.

[BG12] S. Bayer and J. Groth. "Efficient Zero-Knowledge Argument for Correctness of a Shuffle". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 263–280.

[BGG17] D. Boneh, R. Gennaro, and S. Goldfeder. "Using level-1 homomorphic encryption to improve threshold dsa signatures for bitcoin wallet security". In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2017, pp. 352–377.

[Bia+20] J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. "LESS is More: Code-Based Signatures Without Syndromes". In: *AFRICACRYPT 2020*. Springer International Publishing, 2020.

[BJ14] B. Balof and H. Jenne. "Tilings, Continued fractions, Derangements, Scramblings, And e". In: *Journal of Integer Sequences [electronic only]* 17 (Jan. 2014).

[BKP20] W. Beullens, S. Katsumata, and F. Pintore. "Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices". In: *Advances in Cryptology – ASIACRYPT 2020*. Ed. by S. Moriai and H. Wang. Cham: Springer International Publishing, 2020, pp. 464–492.

[BKV19] W. Beullens, T. Kleinjung, and F. Vercauteren. "CSI-FiSh: efficient isogeny based signatures through class group computations". In: *ASIACRYPT 2019*. Springer. 2019.

[BL11] V. Berthé and S. Labbé. "An Arithmetic and Combinatorial Approach to Three-Dimensional Discrete Lines". In: *Discrete Geometry for Computer Imagery*. 2011. URL: `https://api.semanticscholar.org/CorpusID:357989`.

[Blä+22] M. Bläser, Z. Chen, D. H. Duong, A. Joux, N. T. Nguyen, T. Plantard, Y. Qiao, W. Susilo, and G. Tang. "On digital signatures based on isomorphism problems: QROM security, ring signatures, and applications". In: *Cryptology ePrint Archive* (2022).

[BLM22] M. Battagliola, R. Longo, and A. Meneghetti. "Extensible Decentralized Secret Sharing and Application to Schnorr Signatures". preprint: `https://eprint.iacr.org/2022/1551`. 2022.

[BMS22] M. Battagliola, N. Murru, and G. Santilli. "Combinatorial properties of multidimensional continued fractions". In: *Discret. Math.* 346 (2022), p. 113649. URL: https://api.semanticscholar.org/CorpusID:252367238.

[BN06] M. Bellare and G. Neven. "Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. Association for Computing Machinery, 2006, 390–399. DOI: 10.1145/1180405.1180453. URL: https://doi.org/10.1145/1180405.1180453.

[BO+88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. "Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions". In: STOC '88. Association for Computing Machinery, 1988, 113–131. DOI: 10.1145/62212.62223. URL: https://doi.org/10.1145/62212.62223.

[Bog+ j] D. Bogdanov, L. Kamm, B. Kubo, R. Rebane, V. Sokk, and R. Talviste. "Students and Taxes: a Privacy-Preserving Study Using Secure Computation". In: *Proceedings on Privacy Enhancing Technologies* 2016.3 (1Jul. 2016), pp. 117 –135. DOI: https://doi.org/10.1515/popets-2016-0019. URL: https://content.sciendo.com/view/journals/popets/2016/3/article-p117.xml.

[Bog13] D. Bogdanov. "Sharemind: programmable secure computations with practical applications". PhD thesis. University of Tartu, 2013. URL: http://hdl.handle.net/10062/29041.

[Bog+16] D. Bogdanov, M. Joemets, S. Siim, and M. Vaht. *Privacy-preserving tax fraud detection in the cloud with realistic data volumes*. Tech. rep. T-4-24. https://cyber.ee/research/reports/: Cybernetica AS, 2016.

[BP] L. Brandão and R. Peralta. "NIST First Call for Multi-Party Threshold Schemes". In: DOI: 10.6028/NIST.IR.8214C.ipd. URL: https://doi.org/10.6028/NIST.IR.8214C.ipd.

[BQ03] A. Benjamin and J. Quinn. *Proofs That Really Count (The Art of Combinatorial Proof)*. Vol. 58. Jan. 2003. DOI: 10.5948/9781614442080.

[BR06] M. Bellare and P. Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *Advances in Cryptology - EUROCRYPT 2006*. Ed. by S. Vaudenay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 409–426.

[BR93] M. Bellare and P. Rogaway. "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. CCS '93. New York, NY, USA: Association for Computing Machinery, 1993, 62–73. DOI: 10.1145/168588.168596.

[BS22] D. Bowman and H. D. Schaumburg. "Combinatorics of continuants of continued fractions with 3 limits". In: *Journal of Combinatorial Theory, Series A* 186 (2022), p. 105556. DOI: https://doi.org/10.1016/j.jcta.2021.105556. URL: https://www.sciencedirect.com/science/article/pii/S0097316521001552.

[Bud+24] A. Budroni, J.-J. Chi-Domínguez, G. D'Alconzo, A. J. D. Scala, and M. Kulkarni. *Don't Use It Twice! Solving Relaxed Linear Code Equivalence Problems*. Cryptology ePrint Archive, Paper 2024/244. https://eprint.iacr.org/2024/244. 2024. URL: https://eprint.iacr.org/2024/244.

[CAE19] A. Cardillo, N. Akinyokun, and A. Essex. "Online Voting in Ontario Municipal Elections: A Conflict of Legal Principles and Technology?" In: *E-VOTE-ID*. Vol. 11759. LNCS. Springer, 2019, pp. 67–82.

[Can01] R. Canetti. "Universally composable security: a new paradigm for cryptographic protocols". In: *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888.

[Can04] R. Canetti. "Universally composable signature, certification, and authentication". In: *Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004.* 2004, pp. 219–233. DOI: 10.1109/CSFW.2004.1310743.

[Can+20] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. "UC Non-Interactive, Proactive, Threshold ECDSA with Identifiable Aborts". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. CCS '20. Association for

Computing Machinery, 2020, 1769–1787. DOI: 10.1145/3372297.3423367. URL: https://doi.org/10.1145/3372297.3423367.

[Cas+18] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. "CSIDH: An Efficient Post-Quantum Commutative Group Action". In: *Advances in Cryptology – ASIACRYPT 2018*. Ed. by T. Peyrin and S. Galbraith. Cham: Springer International Publishing, 2018, pp. 395–427.

[CCM08] M. R. Clarkson, S. Chong, and A. C. Myers. "Civitas: Toward a Secure Voting System". In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2008, pp. 354–368.

[CG23] D. Cozzo and E. Giunta. "Round-Robin is Optimal: Lower Bounds for Group Action Based Protocols". In: *Theory of Cryptography*. Ed. by G. Rothblum and H. Wee. Cham: Springer Nature Switzerland, 2023, pp. 310–335.

[CGY22a] V. Cortier, P. Gaudry, and Q. Yang. "A Toolbox for Verifiable Tally-Hiding E-Voting Systems". In: *ESORICS (2)*. Vol. 13555. LNCS. Springer, 2022, pp. 631–652.

[CGY22b] V. Cortier, P. Gaudry, and Q. Yang. "Is the JCJ voting system really coercion-resistant?" working paper or preprint. 2022. URL: https://hal.inria.fr/hal-03629587.

[Cha22] A. Chailloux. "On the (In) security of optimized Stern-like signature schemes". In: WCC. 2022.

[Cha81] D. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms". In: *Commun. ACM* 24.2 (1981), pp. 84–88. URL: https://doi.org/10.1145/358549.358563.

[Cho+12] S. G. Choi, K.-W. Hwang, J. Katz, T. Malkin, and D. Rubenstein. "Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces". In: *Topics in Cryptology – CT-RSA 2012*. Ed. by O. Dunkelman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 416–432.

[Cho+23] T. Chou, R. Niederhagen, E. Persichetti, T. H. Randrianarisoa, K. Reijnders, S. Samardjiska, and M. Trimoska. "Take your meds: Digital signatures from matrix code equivalence". In: *International Conference on Cryptology in Africa*. Springer. 2023.

[Cho+95] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. "Private information retrieval". In: *Proceedings of IEEE 36th Annual Foundations of Computer Science* (1995), pp. 41–50. URL: https://api.semanticscholar.org/CorpusID:544823.

[Chu+23] H. Chu, P. Gerhart, T. Ruffing, and D. Schröder. "Practical Schnorr Threshold Signatures Without the Algebraic Group Model". In: *Advances in Cryptology – CRYPTO 2023*. Ed. by H. Handschuh and A. Lysyanskaya. Cham: Springer Nature Switzerland, 2023, pp. 743–773.

[CKM23] E. Crites, C. Komlo, and M. Maller. "Fully Adaptive Schnorr Threshold Signatures". In: *Cryptology ePrint Archive* (2023).

[CL15] G. Castagnos and F. Laguillaumie. "Linearly Homomorphic Encryption from DDH". In: *CT-RSA*. Vol. 9048. LNCS. Springer, 2015, pp. 487–505.

[CM22] F. Campos and P. Muth. "On actively secure fine-grained access structures from isogeny assumptions". In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 375–398.

[CPS23] T. Chou, E. Persichetti, and P. Santini. *On Linear Equivalence, Canonical Forms, and Digital Signatures*. https://tungchou.github.io/papers/leq.pdf. Accessed: 2023-09-20. 2023.

[CS19] D. Cozzo and N. P. Smart. "Sharing the LUOV: threshold post-quantum signatures". In: *IMA International Conference on Cryptography and Coding*. Springer. 2019, pp. 128–153.

[CS20a] I. Canacki and R. Schiffler. "Snake graphs and continued fractions". In: *European Journal of Combinatorics* 86 (2020), p. 103081. DOI: https://doi.org/10.1016/j.ejc.2020.103081. URL: https://www.sciencedirect.com/science/article/pii/S0195669820300020.

[CS20b] D. Cozzo and N. P. Smart. "Sashimi: Cutting up CSI-FiSh Secret Keys to Produce an Actively Secure Distributed Signing Protocol". In: *Post-Quantum Cryptography*. Ed. by J. Ding and J.-P. Tillich. Cham: Springer International Publishing, 2020, pp. 169–186.

[DDCB95] Y. Desmedt, G. Di Crescenzo, and M. Burmester. "Multiplicative non-abelian sharing schemes and their application to threshold cryptography". In: *Advances in Cryptology — ASIACRYPT'94*. Ed. by J. Pieprzyk and R. Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 19–32.

[DFG19] L. De Feo and S. D. Galbraith. "SeaSign: compact isogeny signatures from class group actions". In: *EUROCRYPT 2019*. Springer. 2019.

[DFM20] L. De Feo and M. Meyer. "Threshold schemes from isogeny assumptions". In: *PKC 2020*. Springer. 2020.

[Doe+18] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. "Secure two-party threshold ECDSA from ECDSA assumptions". In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 980–997.

[Doe+19] J. Doerner, Y. Kondi, E. Lee, and A. Shelat. "Threshold ECDSA from ECDSA assumptions: The multiparty case". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 1051–1066.

[Don+19] J. Don, S. Fehr, C. Majenz, and C. Schaffner. "Security of the Fiat-Shamir transformation in the quantum random-oracle model". In: *CRYPTO 20199*. Springer. 2019.

[Eest] *How did Estonia carry out the world's first mostly online national elections*. `https://e-estonia.com/how-did-estonia-carry-out-the-worlds-first-mostly-online-national-elections/`. Accessed: 2023-05-06.

[EK13] C. Elsner and A. Klauke. "Transcendence results and continued fraction expansions obtained from a combinatorial series". In: *Journal of Combinatorial Number Theory* 5 (2013), pp. 53–79.

[F82] "On congruences and continued fractions for some classical combinatorial quantities". In: *Discrete Mathematics* 41.2 (1982), pp. 145–153. DOI: `https://doi.org/10.1016/0012-365X(82)90201-1`.

[Fla80] P. Flajolet. "Combinatorial aspects of continued fractions". In: *Discrete Mathematics* 32.2 (1980), pp. 125–161. DOI: `https://doi.org/10.1016/0012-365X(80)90050-3`.

[FS87] A. Fiat and A. Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology — CRYPTO' 86*. Ed. by A. M. Odlyzko. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[Gam85] T. E. Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE Trans. Inf. Theory* 31.4 (1985), pp. 469–472. URL: `https://doi.org/10.1109/TIT.1985.1057074`.

[Gen+07a] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. "Secure Distributed Key Generation for Discrete-Log Based Cryptosystems". In: *J. Cryptol.* 20.1 (2007), pp. 51–83.

[Gen+07b] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. "Secure distributed key generation for discrete-log based cryptosystems". In: *Journal of Cryptology* 20 (2007), pp. 51–83.

[Gen+96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. "Robust threshold DSS signatures". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1996, pp. 354–371.

[GG18] R. Gennaro and S. Goldfeder. "Fast multiparty threshold ECDSA with fast trustless setup". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1179–1194.

[GGN16] R. Gennaro, S. Goldfeder, and A. Narayanan. "Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2016, pp. 156–174.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. "How to Play ANY Mental Game". In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. New York, NY, USA: Association for Computing Machinery, 1987, 218–229. DOI: 10.1145/28395.28420. URL: https://doi.org/10.1145/28395.28420.

[GPS22] S. Gueron, E. Persichetti, and P. Santini. "Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup". In: *Cryptography* 6.1 (2022), p. 5.

[Gri+21] A. B. Grilo, K. Hövelmanns, A. Hülsing, and C. Majenz. "Tight adaptive reprogramming in the QROM". In: *ASIACRYPT 2021*. Springer. 2021.

[Her50] C. Hermite. "Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objects de la théorie des nombres." In: *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1850.40 (1850), pp. 261–278. DOI: doi:10.1515/crll.1850.40.261. URL: https://doi.org/10.1515/crll.1850.40.261.

[HPT22] T. Haines, O. Pereira, and V. Teague. "Running the Race: A Swiss Voting Story". In: *E-Vote-ID*. Vol. 13553. LNCS. Springer, 2022, pp. 53–69.

[HS19] T. Haines and B. Smyth. "Surveying definitions of coercion resistance". In: *IACR ePrint Arch.* (2019), p. 822. URL: https://eprint.iacr.org/2019/822.

[HT15] J. A. Halderman and V. Teague. "The New South Wales iVote System: Security Failures and Verification Flaws in a Live Online Election". In: *VoteID*. Vol. 9269. LNCS. Springer, 2015, pp. 35–53.

[Idea] *Use of E-Voting Around the World*. https://www.idea.int/news-media/media/use-e-voting-around-world@misc. Accessed: 2023-05-06.

[Ifesa] *Internet Voting: Past, Present and Future*. https://www.ifes.org/news/internet-voting-past-present-and-future. Accessed: 2023-02-18.

[Ifesb] *The Negs and Regs of Continued Fractions*. https://scholarship.claremont.edu/hmc_theses/180. HMC Senior Theses.

[IO94] S. ITO and M. OHTSUKI. "Parallelogram Tilings and Jacobi-Perron Algorithm". In: *Tokyo Journal of Mathematics* 17.1 (1994), pp. 33 –58. DOI: 10.3836/tjm/1270128186. URL: https://doi.org/10.3836/tjm/1270128186.

[ISN89] M. Ito, A. Saito, and T. Nishizeki. "Secret sharing scheme realizing general access structure". In: *Electronics and Communications in Japan* (1989).

[Jac13] C. G. J. Jacobi. "Correspondance mathématique avec Legendre". In: *C. G. J. Jacobi's Gesammelte Werke: Herausgegeben auf Veranlassung der königlich preussischen Akademie der Wissenschaften*. Ed. by C. W. Borchardt. Cambridge Library Collection - Mathematics. Cambridge University Press, 2013, 385–462.

[Jam23] W. Jamroga. "Pretty Good Strategies for Benaloh Challenge". In: *Electronic Voting*. Ed. by M. Volkamer, D. Duenas-Cid, P. Rønne, P. Y. A. Ryan, J. Budurushi, O. Kulyk, A. Rodriguez Pérez, and I. Spycher-Krivonosova. Cham: Springer Nature Switzerland, 2023, pp. 106–122.

[JCJ10] A. Juels, D. Catalano, and M. Jakobsson. "Coercion-Resistant Electronic Elections". In: *Towards Trustworthy Elections*. Vol. 6000. LNCS. Springer, 2010, pp. 37–63.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. "Designated Verifier Proofs and Their Applications". In: *EUROCRYPT*. Vol. 1070. LNCS. Springer, 1996, pp. 143–154.

[KKB18] H. Kaur, N. Kumar, and S. Batra. "An efficient multi-party scheme for privacy preserving collaborative filtering for healthcare recommender system". In: *Future Generation Computer Systems* 86 (2018), pp. 297–307. DOI: https://doi.org/10.1016/j.future.2018.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0167739X17327012.

[KL20] J Katz and Y Lindell. *Introduction to modern cryptography book*. 2020.

[KMR12] M. Keller, G. L. Mikkelsen, and A. Rupp. "Efficient Threshold Zero-Knowledge with Applications to User-Centric Protocols". In: *Information Theoretic Security*. Ed. by A. Smith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 147–166.

[Kom] C. Komlo. *A Note on Various Forking Lemmas*. URL: https://www.chelseakomlo.com/assets/content/notes/Forking-Lemma-Variants.pdf.

[Kup13] G. Kuperberg. "Another Subexponential-time Quantum Algorithm for the Dihedral Hidden Subgroup Problem". In: *TQC 2013*. Ed. by S. Severini and F. G. S. L. Brandão. Vol. 22. LIPIcs. Schloss Dagstuhl, 2013.

[LHK16] P. Locher, R. Haenni, and R. E. Koenig. "Coercion-Resistant Internet Voting with Everlasting Privacy". In: *Financial Cryptography Workshops*. Vol. 9604. LNCS. Springer, 2016, pp. 161–175.

[Lic+21] N. Licht, D. Duenas-Cid, I. Krivonosova, and R. Krimmer. "To i-vote or Not to i-vote: Drivers and Barriers to the Implementation of Internet Voting". In: *E-VOTE-ID*. Vol. 12900. LNCS. Springer, 2021, pp. 91–105.

[Lin17] Y. Lindell. "Fast secure two-party ECDSA signing". In: *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37*. Springer. 2017, pp. 613–644.

[Lin22] Y. Lindell. *Simple Three-Round Multiparty Schnorr Signing with Full Simulatability*. Cryptology ePrint Archive, Paper 2022/374. https://eprint.iacr.org/2022/374. 2022. URL: https://eprint.iacr.org/2022/374.

[LQT20] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso. "VoteAgain: A scalable coercion-resistant voting system". In: *USENIX Security Symposium*. USENIX Association, 2020, pp. 1553–1570.

[Max+19] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. "Simple Schnorr multi-signatures with applications to Bitcoin". In: *Designs, Codes and Cryptography* 87 (Sept. 2019). DOI: 10.1007/s10623-019-00608-x.

[Men21] M. Mendez. "Shift-plethysm, hydra continued fractions, and m-distinct partitions". In: *European Journal of Combinatorics* 95 (2021), p. 103340. DOI: https://doi.org/10.1016/j.ejc.2021.103340.

[MKO16] M. Marwan, A. Kartit, and H. Ouahmane. "Applying secure multi-party computation to improve collaboration in healthcare cloud". In: *2016 Third International Conference on Systems of Collaboration (SysCo)* (2016), pp. 1–6.

[MR04] P. MacKenzie and M. K. Reiter. "Two-party generation of DSA signatures". In: *International Journal of Information Security* 2 (2004), pp. 218–239.

[MS81] R. J. McEliece and D. V. Sarwate. "On sharing secrets and Reed-Solomon codes". In: *Communications of the ACM* 24.9 (1981), pp. 583–584.

[Nic+03] A. Nicolosi, M. N. Krohn, Y. Dodis, and D. Mazières. "Proactive Two-Party Signatures for User Authentication". In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. The Internet Society, 2003.

[NIS17] NIST. *Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. 2017.

[NIS23] NIST. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. URL: https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals. 2023.

[Pan12] O. Panprasitwech. "Combinatorial Proofs of Some Identities for Non-regular Continued Fractions". In: *International Journal of Combinatorics* 2012 (Sept. 2012). DOI: 10.1155/2012/894380.

[PBS12] P. Pullonen, D. Bogdanov, and T. Schneider. "The Design and Implementation of a Two-Party Protocol Suite for SHAREMIND 3". In: 2012.

[Ped91] T. P. Pedersen. "Distributed Provers with Applications to Undeniable Signatures". In: *Advances in Cryptology — EUROCRYPT '91*. Ed. by D. W. Davies. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 221–242.

[Ped92] T. P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *Advances in Cryptology — CRYPTO '91*. Ed. by J. Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 129–140.

[Per07] O. Perron. "Grundlagen fur eine Theorie des Jacobischen Kettenbruchalgorithmus". In: *Mathematische Annalen* 64 (1907), pp. 1–76. URL: https://api.semanticscholar.org/CorpusID:120134247.

[PR97] E. Petrank and R. Roth. "Is code equivalence easy to decide?" In: *IEEE Transactions on Information Theory* 43.5 (1997), pp. 1602–1604. DOI: 10.1109/18.623157.

[PS21] M. Pétréolle and A. D. Sokal. "Lattice paths and branched continued fractions II. Multivariate Lah polynomials and Lah symmetric functions". In: *European Journal of Combinatorics* 92 (2021). DOI: https://doi.org/10.1016/j.ejc.2020.103235.

[PS23] E. Persichetti and P. Santini. "A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures". In: *Cryptology ePrint Archive* (2023).

[PS96] D. Pointcheval and J. Stern. "Security Proofs for Signature Schemes". In: *Advances in Cryptology — EUROCRYPT '96*. Ed. by U. Maurer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 387–398.

[Røn+20] P. B. Rønne, A. Atashpendar, K. Gjøsteen, and P. Y. A. Ryan. "Short Paper: Coercion-Resistant Voting in Linear Time via Fully Homomorphic Encryption". In: *Financial Cryptography and Data Security*. Ed. by A. Bracciali, J. Clark, F. Pintore, P. B. Rønne, and M. Sala. Cham: Springer International Publishing, 2020, pp. 289–298.

[Rot06] R. M. Roth. *Introduction to coding theory*. Vol. 47. 18-19. IET, 2006, p. 4.

[RRI16] P. Y. A. Ryan, P. B. Rønne, and V. Iovino. "Selene: Voting with Transparent Verifiability and Coercion-Mitigation". In: *Financial Cryptography Workshops*. Vol. 9604. LNCS. Springer, 2016, pp. 176–192.

[RS60] I. S. Reed and G. Solomon. "Polynomial codes over certain finite fields". In: *Journal of the society for industrial and applied mathematics* 8.2 (1960), pp. 300–304.

[Sch91] C. Schnorr. "Efficient signature generation by smart cards". In: *Journal of Cryptology* 4 (Jan. 1991), pp. 161–174. DOI: 10.1007/BF00196725.

[Smy19] B. Smyth. "Athena: A verifiable, coercion-resistant voting system with linear complexity". In: *IACR ePrint Arch.* (2019), p. 761. URL: https://eprint.iacr.org/2019/761.

[SRA81] A. Shamir, R. L. Rivest, and L. M. Adleman. "Mental Poker". In: *The Mathematical Gardner*. Ed. by D. A. Klarner. Prindle, Weber, and Schmidt, 1981, pp. 37–43.

[ST04] B. Schoenmakers and P. Tuyls. "Practical Two-Party Computation Based on the Conditional Gate". In: *ASIACRYPT*. Vol. 3329. LNCS. Springer, 2004, pp. 119–136.

[SW18] D. Stinson and R. Wei. "Combinatorial Repairability for Threshold Schemes". In: *Designs, Codes and Cryptography* 86 (Jan. 2018). DOI: 10.1007/s10623-017-0336-6.

[SZ10] H. Shin and J. Zeng. "The q-tangent and q-secant numbers via continued fractions". In: *European Journal of Combinatorics* 31.7 (2010), pp. 1689–1705. DOI: https://doi.org/10.1016/j.ejc.2010.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0195669810000491.

[Sze68] G. Szekeres. "A Combinatorial Interpretation of Ramanujan's Continued Fraction". In: *Canadian Mathematical Bulletin* 11.3 (1968), 405–408. DOI: 10.4153/CMB-1968-046-x.

[Tso+17] R. Tso, A. Alelaiwi, S. M. M. Rahman, M.-E. Wu, and M. S. Hossain. "Privacy-Preserving Data Communication Through Secure Multi-Party Computation in Healthcare Sensor Cloud". In: *Journal of Signal Processing Systems* 89 (Oct. 2017). DOI: 10.1007/s11265-016-1198-2.

[Unr17] D. Unruh. "Post-quantum security of Fiat-Shamir". In: *Advances in Cryptology–ASIACRYPT 2017*. Springer. 2017.

[WAB07] S. G. Weber, R. Araujo, and J. Buchmann. "On Coercion-Resistant Electronic Elections with Linear Work". In: *The Second International Conference on Availability, Reliability and Security (ARES'07)*. 2007, pp. 908–916. DOI: 10.1109/ARES.2007.108.

[Wik04] D. Wikström. "A Universally Composable Mix-Net". In: *Theory of Cryptography*. Ed. by M. Naor. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 317–335.

[Wik09] D. Wikström. "A Commitment-Consistent Proof of a Shuffle". In: *ACISP*. Vol. 5594. LNCS. Springer, 2009, pp. 407–421.

[Yao82] A. C. Yao. "Protocols for secure computations". In: *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*. 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38.

[Yao86] A. C. Yao. "How to generate and exchange secrets". In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[ZH17] R. Zhu and Y. Huang. "Efficient Privacy-Preserving General Edit Distance and Beyond". In: 2017.

Appendix A

Combinatorial properties of multidimensional continued fractions

The results of this appendix are published in [BMS22].

A.1 Introduction

Multidimensional continued fractions were introduced by Jacobi [Jac13] (and then generalized by Perron [Per07]) in the attempt to answer a problem posed by Hermite [Her50] who asked for an algorithm that provides periodic representations for algebraic irrationals of any degree, in the same way as continued fractions are periodic if and only if they represent quadratic irrationals. Unfortunately, the Jacobi–Perron algorithm does not solve the problem, which is still a beautiful open problem in number theory, but opened a new and rich research field. Indeed, there are many studies about multidimensional continued fractions and their modifications, aiming to generalize the results and properties of classical continued fractions.

Continued fractions have been widely studied from different points of view. Several works explore the combinatorial properties of continued fractions giving many interesting interpretations. In the book of Benjamin and Quinn [BQ03], one chapter is devoted to continued fractions showing that numerators and denominators of convergents count some particular tilings, reporting also some results proved in [ATBQ00]. In [BJ14], the author provided further results regarding the properties of continued fractions in terms of counting tilings and giving also a combinatorial interpretation to the expansion of e. A different approach to the combinatorial

aspects of continued fractions is given in [Fla80], where they are connected to some labelled paths. Recently, in [CS20a], the authors described a combinatorial interpretation of continued fractions as quotients of the number of perfect matchings of snake graphs. Further interesting works in this field are [BS22; EK13; F82; Men21; Pan12; PS21; SZ10; Sze68].

Regarding multidimensional continued fractions, there are just few works about their combinatorial properties. In [IO94], the Jacobi–Perron algorithm is used for giving a generating method of the so-called stepped surfaces. In [BL11], the authors used multidimensional continued fractions for obtaining a method of generation of discrete segments in the three-dimensional space. Finally, in [Arn+01], multidimensional continued fractions have been exploited for obtaining results about tilings, discrete approximations of lines and planes, and Markov partitions for toral automorphism.

In this paper, we propose an elementary approach to the study of combinatorial properties of multidimensional continued fractions, obtaining a natural interpretation in terms of counting tilings of a board using tiles of length one, two or three, where we can also stack such tiles. We also give an interpretation to negative conditions for the height of the stacks. In particular, Section A.2 is devoted to the preliminary definitions and properties of multidimensional continued fractions, where we also introduce them from a formal point of view. Section A.3 presents the main results.

A.2 Preliminaries

Multidimensional continued fractions (of degree two) represent a pair of real numbers (α_0, β_0) by means of two sequences of integers $(a_i)_{i\geq 0}$, $(b_i)_{i\geq 0}$ as follows:

$$\alpha_0 = a_0 + \cfrac{1}{b_1 + \cfrac{1}{b_3 + \cfrac{1}{\ddots \\ a_2 + \cfrac{1}{\ddots \\ a_3 + \cfrac{1}{\ddots}}}}}}{\cfrac{1}{b_2 + \cfrac{1}{\ddots \\ a_3 + \cfrac{1}{\ddots \\ a_1 + \cfrac{1}{b_3 + \cfrac{1}{\ddots \\ a_2 + \cfrac{1}{\ddots \\ a_3 + \cfrac{1}{\ddots}}}}}}}}}, \quad \beta_0 = b_0 + \cfrac{1}{b_2 + \cfrac{1}{\ddots \\ a_3 + \cfrac{1}{\ddots \\ a_1 + \cfrac{1}{b_3 + \cfrac{1}{\ddots \\ a_2 + \cfrac{1}{\ddots \\ a_3 + \cfrac{1}{\ddots}}}}}}}$$

where the a_i's and b_i's are called *partial quotients* and they can be obtained by the Jacobi algorithm in the following way:

$$\begin{cases} \alpha_i = \lfloor a_i \rfloor \\ \beta_i = \lfloor b_i \rfloor \\ \alpha_{i+1} = \dfrac{1}{\beta_i - b_i} \\ \beta_{i+1} = \dfrac{\alpha_i - a_i}{\beta_i - b_i} \end{cases} \qquad i = 0, 1, 2, ...$$

We can introduce multidimensional continued fractions also in a formal way, where the partial quotients are not in general obtained by an algorithm and the numerators are not necessarily equals to 1, as well as in the classical case, given two sequences $(a_i)_{i\geq 0}$, $(b_i)_{i\geq 0}$, one can introduce and study the continued fraction

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \ddots}}.$$

Definition A.1. Given the sequences of integers $(a_i)_{i\geq 0}$, $(b_i)_{i\geq 0}$ and $(c_i)_{i\geq 0}$ (with $c_0 = 1$), called *partial quotients*, we define the multidimensional continued fraction (MCF) as the following couple of objects:

$$a_0 + \cfrac{c_2}{b_1 + \cfrac{c_4}{b_3 + \cfrac{}{a_2 + \cfrac{}{a_3 + \cfrac{}{\ddots}}}}}, \quad \cfrac{c_1}{b_0 + \cfrac{c_3}{b_2 + \cfrac{}{a_1 + \cfrac{c_4}{b_3 + \cfrac{}{a_2 + \cfrac{}{a_3 + \cfrac{}{\ddots}}}}}}}. \tag{A.2.1}$$

In the following, we will write shortly $[(a_0, a_1, ...), (b_0, b_1, ...), (1, c_1, ...)]$ for such a MCF.
We also call *complete quotients* the elements of the sequences of real numbers $(\alpha_i)_{i\geq 0}$, $(\beta_i)_{i\geq 0}$ and $(\gamma_i)_{i\geq 0}$ defined by the following relations:

$$\alpha_i = a_i + \frac{\beta_{i+1}}{\alpha_{i+1}}, \quad \beta_i = b_i + \frac{\gamma_{i+1}}{\alpha_{i+1}}, \quad \gamma_i = c_i$$

for $i = 0, 1, 2, ...$, so that $(\alpha_0, \beta_0) = [(a_0, a_1, ...), (b_0, b_1, ...), (1, c_1, ...)]$.

In the following, we will use $\mathbf{a}_i^j$ for denoting the finite sequence $(a_i, a_{i+1}, ..., a_j)$, for $i \leq j$ integers. Thus, using this notation the finite MCF

$$[(a_0, a_1, ...a_n), (b_0, b_1, ..., b_n), (1, c_1, ..., c_n)]$$

can be also written as $[\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n]$.

We define the n–th convergent of a MCF, similarly to the convergents of classical continued fractions, as the following pair of rationals:

$$\left(\frac{A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)}{C(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)}, \frac{B(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)}{C(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)}\right) := [\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n].$$

Sometimes, when there is no possibility of confusion, we will also use the notation $\left(\frac{A_n}{C_n}, \frac{B_n}{C_n}\right)$ without making explicit the dependence on the partial quotients.

Observation A.1. We would like to observe that the partial quotient c_0 does not appear in the expansion of the MCF described in (A.2.1). However, we set it equals to 1 because the convergents can be also evaluated in the following way:

$$\begin{pmatrix} a_0 & 1 & 0 \\ b_0 & 0 & 1 \\ c_0 & 0 & 0 \end{pmatrix} \cdots \begin{pmatrix} a_n & 1 & 0 \\ b_n & 0 & 1 \\ c_n & 0 & 0 \end{pmatrix} = \begin{pmatrix} A_n & A_{n-1} & A_{n-2} \\ B_n & B_{n-1} & B_{n-2} \\ C_n & C_{n-1} & C_{n-2} \end{pmatrix}.$$

Since $\frac{A_0}{C_0} = a_0$ and $\frac{B_0}{C_0} = b_0$, it is a natural choice to set $c_0 = 1$. Moreover, we would like to highlight that A_n does not depend on b_0, c_0, c_1 and B_n does not depend on a_0, b_1, c_0, c_2.

Proposition A.1. *Given the sequences of integers $(a_i)_{i\geq 0}$, $(b_i)_{i\geq 0}$, $(c_i)_{i\geq 0}$, then for all $n \geq 3$ we have*

$$A(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n) = a_0 A(\boldsymbol{a}_1^n, \boldsymbol{b}_1^n, \boldsymbol{c}_1^n) + b_1 A(\boldsymbol{a}_2^n, \boldsymbol{b}_2^n, \boldsymbol{c}_2^n) + c_2 A(\boldsymbol{a}_3^n, \boldsymbol{b}_3^n, \boldsymbol{c}_3^n) \tag{A.2.2}$$

and

$$A(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n) = a_n A(\boldsymbol{a}_0^{n-1}, \boldsymbol{b}_0^{n-1}, \boldsymbol{c}_0^{n-1}) + b_n A(\boldsymbol{a}_0^{n-2}, \boldsymbol{b}_0^{n-2}, \boldsymbol{c}_0^{n-2}) + c_n A(\boldsymbol{a}_0^{n-3}, \boldsymbol{b}_0^{n-3}, \boldsymbol{c}_0^{n-3}). \tag{A.2.3}$$

Proof. By definition we have that

$$\frac{A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)}{C(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)} = a_0 + \frac{B(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n)}{A(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n)}, \quad \frac{B(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)}{C(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)} = b_0 + c_1 \frac{C(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n)}{A(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n)}.$$

Thus, we have the following equalities:

$$\begin{aligned} C(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n) &= A(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n) \\ B(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n) &= c_1 C(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n) + b_0 A(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n) \\ A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n) &= a_0 A(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n) + B(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n). \end{aligned} \tag{A.2.4}$$

By substitution we get

$$A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n) = a_0 A(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n) + b_1 A(\mathbf{a}_2^n, \mathbf{b}_2^n, \mathbf{c}_2^n) + c_2 A(\mathbf{a}_3^n, \mathbf{b}_3^n, \mathbf{c}_3^n).$$

Equation (A.2.3) can be proved by induction. The basis of the induction is trivial. By inductive hypothesis we have that

$$A(\mathbf{a}_1^n,\mathbf{b}_1^n,\mathbf{c}_1^n) = a_n A(\mathbf{a}_1^{n-1},\mathbf{b}_1^{n-1},\mathbf{c}_1^{n-1}) + b_n A(\mathbf{a}_1^{n-2},\mathbf{b}_1^{n-2},\mathbf{c}_1^{n-2}) + c_n A(\mathbf{a}_1^{n-3},\mathbf{b}_1^{n-3},\mathbf{c}_1^{n-3})$$

$$A(\mathbf{a}_2^n,\mathbf{b}_2^n,\mathbf{c}_2^n) = = a_n A(\mathbf{a}_2^{n-1},\mathbf{b}_2^{n-1},\mathbf{c}_2^{n-1}) + b_n A(\mathbf{a}_2^{n-2},\mathbf{b}_2^{n-2},\mathbf{c}_2^{n-2}) + c_n A(\mathbf{a}_2^{n-3},\mathbf{b}_2^{n-3},\mathbf{c}_2^{n-3})$$

$$A(\mathbf{a}_3^n,\mathbf{b}_3^n,\mathbf{c}_3^n) = a_n A(\mathbf{a}_3^{n-1},\mathbf{b}_3^{n-1},\mathbf{c}_3^{n-1}) + b_n A(\mathbf{a}_3^{n-2},\mathbf{b}_3^{n-2},\mathbf{c}_3^{n-2}) + c_n A(\mathbf{a}_3^{n-3},\mathbf{b}_3^{n-3},\mathbf{c}_3^{n-3})$$

Substituting and factoring out a_n, b_n and c_n we get

$$\begin{aligned} A(\mathbf{a}_0^n,\mathbf{b}_0^n,\mathbf{c}_0^n) = & \\ = a_n[a_0 A(\mathbf{a}_1^{n-1},\mathbf{b}_1^{n-1},\mathbf{c}_1^{n-1}) + b_1 A(\mathbf{a}_2^{n-1},\mathbf{b}_2^{n-1},\mathbf{c}_2^{n-1}) + c_2 A(\mathbf{a}_3^{n-1},\mathbf{b}_3^{n-1},\mathbf{c}_3^{n-1})] + & \\ + b_n[a_0 A(\mathbf{a}_1^{n-2},\mathbf{b}_1^{n-2},\mathbf{c}_1^{n-2}) + b_1 A(\mathbf{a}_2^{n-2},\mathbf{b}_2^{n-2},\mathbf{c}_2^{n-2}) + c_2 A(\mathbf{a}_3^{n-2},\mathbf{b}_3^{n-2},\mathbf{c}_3^{n-2})] + & \\ + c_n[a_0 A(\mathbf{a}_1^{n-3},\mathbf{b}_1^{n-3},\mathbf{c}_1^{n-3}) + b_1 A(\mathbf{a}_2^{n-3},\mathbf{b}_2^{n-3},\mathbf{c}_2^{n-3}) + c_2 A(\mathbf{a}_3^{n-3},\mathbf{b}_3^{n-3},\mathbf{c}_3^{n-3})]. & \end{aligned}$$

Finally, by using again (A.2.2) we get the thesis. □

A.3 Counting the number of tilings using multidimensional continued fractions

In this section, we give a combinatorial interpretation to the convergents of a MCF in terms of tilings of some boards, extending the approaches of Benjamin [BQ03; ATBQ00] and Balof [BJ14] for the classical continued fractions.

In the following, a $(n+1)$–board is a $1 \times (n+1)$ chessboard, a *square* is a 1×1 tile, a *domino* is a 1×2 tile and a *bar* is a 1×3 tile. The $n+1$ cells of the $(n+1)$–board are labeled from 0 to n (i.e., we refer to the cell of position 0 for the first cell and so on). A tiling of a n–board is a covering using squares, dominoes and bars that can be also stacked. In particular, the height conditions for stacking them are given by finite sequences like $(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$, where

- the element a_i of $\mathbf{a}_0^n$ denotes the number of stackable squares in the i–th position (e.g., a_0 is the number of stackable squares in the cell of position 0 of the $(n+1)$–board);

- the element b_i of $\mathbf{b}_0^n$ denotes the number of stackable dominoes covering the positions $i-1$ and i (e.g., b_1 is the number of stackable dominoes covering the positions 0 and 1 of the $(n+1)$–board);
- the element c_i of $\mathbf{c}_0^n$ denotes the number of stackable bars covering the positions $i-2$, $i-1$ and i (e.g., c_2 is the number of stackable dominoes covering the positions 0, 1, and 2 of the $(n+1)$–board).

At first glance the first element in $\mathbf{b}_0^n$ does not give height conditions, as well as the first two elements of $\mathbf{c}_0^n$, however their role will be important later when discussing different types of tilings. We will denote by $M(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$ the number of possible tilings of a $(n+1)$–board with height conditions $(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$.

Example A.1. Consider $n = 5$ with the following height conditions

$$(1,2,3,2,3,2), \quad (b_0,6,5,4,3,2), \quad (c_0,c_1,1,2,3,1),$$

where we do not explicit the values of b_0, c_0, c_1 since, in this case, they are not relevant for the tilings. Examples of valid tilings are represented in fig. A.1, while in fig. A.2 is represented a non-valid tiling for these height conditions: in this case there are too many bars covering the last three cells. Given the above sequences of partial quotients, the sequence of the convergents $\left(\frac{A_0^n}{C_0^n}\right)$ is the following:

n	0	1	2	3	4	5
Convergents	1	4	$\frac{30}{11}$	$\frac{47}{16}$	$\frac{44}{15}$	$\frac{202}{69}$

In this case, we have not specified the value of b_0, since it does not provide any height condition, and consequently we can not write explicitly the sequence of convergents $\left(\frac{B_0^n}{C_0^n}\right)$.

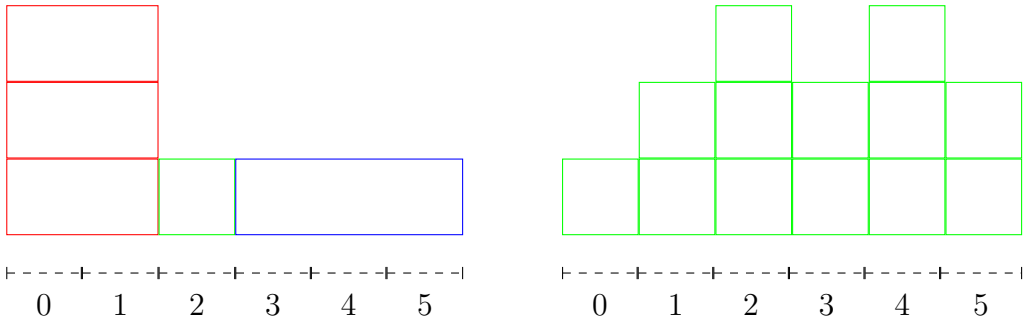


Figure A.1: Examples of valid tilings.

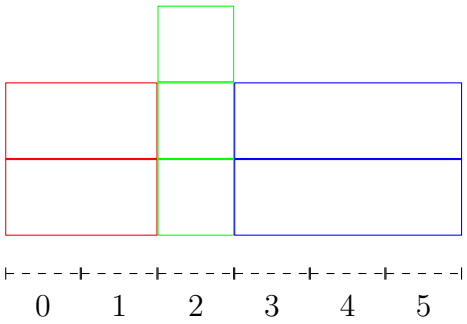


Figure A.2: Example of a non-valid tiling.

Theorem A.1. *Let* $\left(\frac{A(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}{C(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}, \frac{B(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}{C(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}\right) := [\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n]$. *Then we have the following:*

- $A(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$ *counts the number of possible tilings of a* $(n+1)$*–board with height conditions* $(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$.
- $B(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$ *counts the number of possible tilings of a* $(n+2)$*–board, where only in this case the first cell is labelled with -1 (i.e., we add a cell on the left to a* $(n+1)$*–board), with height conditions* $(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$*, such that the first tile of the tiling is a domino or a bar.*
- $C(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$ *counts the number of possible tilings of a* n*–board with height conditions* $(\boldsymbol{a}_1^n, \boldsymbol{b}_1^n, \boldsymbol{c}_1^n)$.

Proof. We want to show that the number of tilings $M(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$ and $A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$ have the same initial values and recurrence formula. Clearly, for a 1–board we have

$$M(a_0, b_0, c_0) = a_0 = A(a_0, b_0, c_0),$$

and for a 2–board

$$M(\mathbf{a}_0^1, \mathbf{b}_0^1, \mathbf{c}_0^1) = a_0 a_1 + b_1 = A(\mathbf{a}_0^1, \mathbf{b}_0^1, \mathbf{c}_0^1).$$

Then for a 3–board we can have tilings with 3 stacks of squares, or 1 stack of squares in the first position and 1 stack of dominoes in the second and third position, or one stack of dominoes in the first and second position and one stack of squares in the third position, or 1 stack of bars:

$$M(\mathbf{a}_0^2, \mathbf{b}_0^2, \mathbf{c}_0^2) = a_0 a_1 a_2 + a_0 b_2 + a_2 b_1 + c_2 = A(\mathbf{a}_0^2, \mathbf{b}_0^2, \mathbf{c}_0^2).$$

For a $(n+1)$–board, with $n > 2$, we can observe that the number of tilings satisfies the following recursive formula:

$$M(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n) = a_0 M(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n) + b_1 M(\mathbf{a}_2^n, \mathbf{b}_2^n, \mathbf{c}_2^n) + c_1 M(\mathbf{a}_3^n, \mathbf{b}_3^n, \mathbf{c}_3^n),$$

since we can count the tilings dividing them in three sets: tilings that start with a stack of squares, tilings that start with a stack of dominoes and tilings that start with a stack of bars. Thus, the number of tilings of a $(n+1)$–board starting with a stack of squares is $a_0 M(\mathbf{a}_1^n, \mathbf{b}_1^n, \mathbf{c}_1^n)$ and similarly for the other two situations. So we have the first point. The third point follows immediately from the first equality in (A.2.4).

About the second point we can observe that if the board has only one cell (i.e. the -1 cell) there are no possible tilings (a_{-1} is implicitly set at 0), and this is consistent with $B_{-1} = 0$ (see the matricial representation of the convergents in observation A.1. Moreover, $M(\mathbf{a}_{-1}^0, \mathbf{b}_{-1}^0, \mathbf{c}_{-1}^0) = b_0 = B(a_0, b_0, c_0)$, since we can tile the 2–board only with a domino. Similarly, $M(\mathbf{a}_{-1}^1, \mathbf{b}_{-1}^1, \mathbf{c}_{-1}^1) = b_0 a_1 + c_1 = B(\mathbf{a}_0^1, \mathbf{b}_0^1, \mathbf{c}_0^1)$, because we only have two possibilities: a tile composed by one domino and one square or a tile composed by one bar. Now, we can complete the proof by induction with the same argument used above.

□

With the same notation as theorem A.1 we have the following corollary for a $(n+1)$–circular board, which is a $(n+1)$–board where the first and last tile are bordering.

Corollary A.1. *The number of tilings of a $n+1$–circular board with height condition $(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$ with $c_0 = 0$ (i.e. we forbid bars covering the cells $0, n, n-1$) is $A(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n) + B(\boldsymbol{a}_0^{n-1}, \boldsymbol{b}_0^{n-1}, \boldsymbol{c}_0^{n-1})$*

Proof. $A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$ counts the number of all the possible tilings where the cells 0 and n are not covered by the same stack of dominoes or bars. The tilings that are missing are the ones where a stack of dominoes covers 0 and n or a stack of bars covers $1, 0, n$, (the only other possible case, where a stack of bars covers $0, n, n-1$) is impossible since $c_0 = 0$). In particular we notice that in both case the stack begins in the cell n. By the previous theorem $B(\mathbf{a}_0^{n-1}, \mathbf{b}_0^{n-1}, \mathbf{c}_0^{n-1})$ counts the number of tilings of a $n+1$ board starting from cell -1, starting with a stack of dominoes or bars. We can notice that this is the same of saying that the board starts with cell n followed by the cell 0. □

In the following proposition we show that the numerators of convergents of a MCF can be also seen in terms of permutations.

Proposition A.2. *If* $a_0 = 4, b_1 = 1, c_2 = 1$ *and* $a_i = i+1$ *for* $i > 0$, $b_i = i-1$ *for* $i > 1$, *and* $c_i = i-2$ *for* $i > 2$, *then* $A_n = (n+2)! + (n+1)! + n!$, *i.e.* $A_0 = 4, A_1 = 9, A_2 = 32, \ldots$.

Proof. We prove the identity by induction. It is straightforward to check the thesis for A_0, A_1, A_2. We will now suppose $A_k = (k+2)! + (k+1)! + k!$ for every $k < n$ and prove the property for n. By (A.2.3) we have

$$A_n = a_n A_{n-1} + b_n A_{n-2} + c_n A_{n-3}.$$

From the definition of a_i, b_i, c_i and the inductive hypothesis we get

$$\begin{aligned} A_n = (n+1)[(n+1)! + n! + (n-1)!] + (n-1)[n! + (n-1)! + (n-2)!] + \\ + (n-2)[(n-1)! + (n-2)! + (n-3)!]. \end{aligned}$$

We will deal with the three addends separately:

$$\begin{aligned} (n+1)[(n+1)! + n! + (n-1)!] &= (n+1)(n+1)! + (n+1)! + n! + (n-1)! \\ &= (n+2)! - (n+1)! + (n+1)! + n! + (n-1)! = (n+2)! + n! + (n-1)!, \end{aligned}$$

$$(n-1)[n! + (n-1)! + (n-2)!] = (n-1)(n-2)!(n(n-1) + (n-1) + 1) = n!n,$$

$$(n-2)[(n-1)! + (n-2)! + (n-3)!] = (n-2)![n^2 + -2n + 1] = (n-1)!(n-1).$$

Summing all three equation we get

$$A_n = (n+2)! + n! + (n-1)! + n!n + (n-1)!(n-1) = (n+2)! + (n+1)! + n!$$

□

Observation A.2. The MCF of the previous proposition is

$$[(4,2,3,4,5,6,...),(b_0,1,1,2,3,4,...),(1,c_1,1,1,2,3,...)]$$

and the first sequence of convergents $\left(\frac{A_n}{C_n}\right)_{n\geq 0}$ appears to be convergent to the real number 4.54752..., but we were not able to explicitly determine this real number. In the case of classical continued fraction a similar situation happens for the continued fraction

$$2+\cfrac{1}{1+\cfrac{1}{2+\cfrac{2}{3+\cfrac{3}{4+\ddots}}}}$$

whose convergents have as numerator the sequence $((n+1)!+n!)_{n\geq 0}$ and in this case the continued fraction converges to e.

A.3.1 Negative Dominoes and Bars

Now, we want to generalize Theorem A.1 in order to allow negative b_i, c_i, following the ideas of [Ifesb].

We notice that a positive b_i adds b_i number of ways to tile cells $i-1, i$. So a natural way to explain negative coefficient is to impose some restrictions such that a negative b_i give us $|b_i|$ less way to cover the cells $i-1, i$. An analogous argument can be done for c_i.

Definition A.2 (Mixed Tiling)**.** Let $(a_i)_{i\geq 0}$ be a sequence of positive integers and $(b_i)_{i\geq 0}$, $(c_i)_{i\geq 0}$ be sequences of integers such that

- if $b_i < 0$ and $c_i > 0$, then $a_i > |b_i|$;
- if $b_i > 0$ and $c_i < 0$, then either $a_i > |c_i|$ or $b_i > |c_i|$;
- if $b_i < 0$ and $c_i < 0$, then $a_i > |b_i| + |c_i|$.

Then we define a *mixed tiling* of an $(n+1)-$board with height condition respectively given by $\mathbf{a}_0^n$, $\mathbf{b}_0^n$ and $\mathbf{c}_0^n$ as follows: for any $k \in \mathbb{N}$,

(i) if $b_k \geq 0$ and $c_k \geq 0$, we fall back in the same case defined at the beginning of Appendix A.3;

(ii) if $b_k < 0$ and $c_k > 0$, when there is a stack of a_{k-1} squares in the cell $k-1$, we discard the tilings having up to $|b_k|$ squares in the cell k and we refer to them as inadmissible tilings;

(iii) if $c_k < 0$ and $b_k > 0$, we have two cases:

  (i) if $a_k > |c_k|$, when there is a stack stack of a_{k-2} squares in the cell $k-2$ and a stack of a_{k-1} squares in the cell $k-1$, then we consider as inadmissible all the tilings having up to $|c_k|$ squares in the cell k;

  (ii) otherwise, necessarily $b_k > |c_k|$. In this case when there is a stack of a_{k-2} squares in the cell $k-2$, the inadmissible tilings are those with up to $|c_k|$ dominoes covering the cells $k-1, k$;

(iv) if $c_k < 0$ and $b_k < 0$, we have two cases:

  (i) when at the same time there is a stack of a_{k-2} squares in the cell $k-2$ and a stack of a_{k-1} squares in the cell $k-1$, we discard all the tilings having up to $|c_k| + |b_k|$ squares in the cell k;

  (ii) when there is a stack of a_{k-1} squares in the cells $k-1$, the inadmissible tilings have up to $|b_k|$ squares in the cell k.

Observation A.3. Notice that the last condition applies when there are less than a_{k-2} squares in the cell $k-2$ to compensate the negative b_k as in the case 4a.

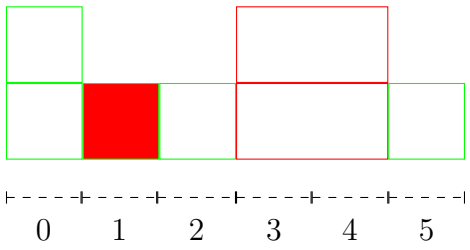
Example A.2. Consider the height conditions given by

$$(2,3,1,2,2,3), \quad (b_0,-1,3,3,2,-1), \quad (c_0,c_1,-2,2,1,-1).$$

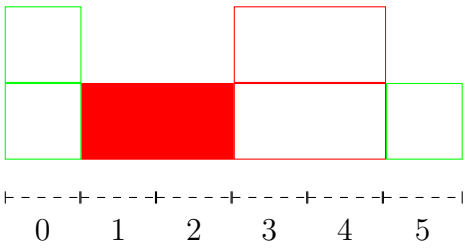
In this case there are several restrictions given by these choice of conditions:

- Since $b_1 = -1$, then when we have $a_0 = 2$ squares in position 0, we need to exclude all the tilings having one square in the cell in position 1 (see fig. A.3a).
- Since $c_2 = -2$ and $a_2 = 1 < |c_2|$, then we are in the case 3b and we need to exclude the tilings having two squares in position 0 and 1 or two dominoes in the positions 1 and 2 (see fig. A.3b).

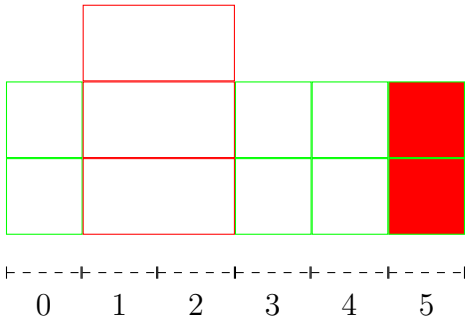
- Finally, $b_5 = c_5 = -1$ so we are in the fourth case. Therefore the negligible tilings are those having $a_3 = 2$ squares in position 3, $a_4 = 2$ squares in position 4 and one or two squares in position 5 (see fig. A.3c). Moreover we also need to discard the tilings having $a_4 = 2$ squares in position 4 and $|b_5| = 1$ square in position 5 (see fig. A.3d).



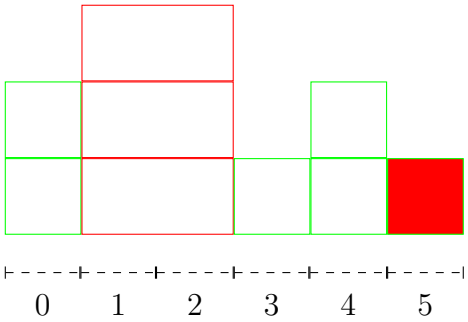
(a) Negligible tiling (case 2).



(b) Negligible tiling (case 3b).



(c) Negligible tiling (case 4a).



(d) Negligible tiling (case 4b).

Figure A.3: Some examples of negligible tilings.

Theorem A.2. *Consider the height conditions given by* $\left(\frac{A(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}{C(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}, \frac{B(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}{C(\boldsymbol{a}_0^n,\boldsymbol{b}_0^n,\boldsymbol{c}_0^n)}\right) := [\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n]$ *such that the conditions in definition A.2 hold. Then* $A(\boldsymbol{a}_0^n, \boldsymbol{b}_0^n, \boldsymbol{c}_0^n)$ *is the number of mixed tilings with height conditions* $\boldsymbol{a}_0^n$, $\boldsymbol{b}_0^n$, *and* $\boldsymbol{c}_0^n$.

Proof. In the following proof we will exclude the case of b_n and c_n being both not negative, since it follows easily by theorem A.1.

First we want to show that the number of mixed tiling $M(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$ satisfies the same initial condition and recurrence relations of $A(\mathbf{a}_0^n, \mathbf{b}_0^n, \mathbf{c}_0^n)$.

- If $n = 0$ we trivially have $M(a_0, b_0, c_0) = a_0 = A(a_0, b_0, c_0)$.

- If $n = 1$ we have $M(\mathbf{a}_0^1, \mathbf{b}_0^1, \mathbf{c}_0^1) = a_0a_1 + b_1 = A(\mathbf{a}_0^1, \mathbf{b}_0^1, \mathbf{c}_0^1)$, since $b_1 < 0$ we may cover using only squares, that are a_0a_1, but we need to subtract $|b_1|$ inadmissible tilings, when we have a_0 squares in the cell in position 0 and less than $|b_1| + 1$ squares in the cell in position 1.

- If $n = 2$ we have $M(\mathbf{a}_0^2, \mathbf{b}_0^2, \mathbf{c}_0^2) = a_0a_1a_2 + a_0b_2 + a_2b_1 + c_2 = A(\mathbf{a}_0^2, \mathbf{b}_0^2, \mathbf{c}_0^2)$, indeed $a_0a_1a_2$ is the total number of tilings consisting in only squares, a_0b_2 and b_1a_2 are the number of tilings involving a stack of squares and a stack of dominoes that we need to add (when $b_i \geq 0$) or subtract (when $b_i < 0$). Finally c_2 is the number of tiling using only bars we need to add (when $c_2 \geq 0$)) or the number of tilings we need to subtract ($c_2 < 0$).

We now need to prove that M has the same recurrence property expressed in Proposition A.1.

- If $b_n < 0$ and $c_n > 0$, then every tiling must finish either with a stack of squares or a stack of bars. By induction there are $c_nM(\mathbf{a}_0^{n-3}, \mathbf{b}_0^{n-3}, \mathbf{c}_0^{n-3})$ tilings that end with a stack of bars and $a_nM(\mathbf{a}_0^{n-1}, \mathbf{b}_0^{n-1}, \mathbf{c}_0^{n-1})$ tilings that end with a stack of squares, ignoring the condition stated in definition A.2. Among these, we need to subtract $|b_n|M(\mathbf{a}_0^{n-2}, \mathbf{b}_0^{n-2}, \mathbf{c}_0^{n-2})$ inadmissible tiling, namely those having a stack of a_{n-1} squares in the cell $n-1$ and less than $|b_n| + 1$ squares in the cell n.

- If $b_n > 0$ and $c_n < 0$ then every tiling must finish either with a stack of squares or a stack of dominoes. By induction these are respectively $a_nM(\mathbf{a}_0^{n-1}, \mathbf{b}_0^{n-1}, \mathbf{c}_0^{n-1})$ and $b_nM(\mathbf{a}_0^{n-2}, \mathbf{b}_0^{n-2}, \mathbf{c}_0^{n-2})$. Now we need to distinguish two possible cases:

  - If $a_n > |c_n|$, then we need to subtract $|c_n|M(\mathbf{a}_0^{n-3}, \mathbf{b}_0^{n-3}, \mathbf{c}_0^{n-3})$ inadmissible tilings, i.e. those having a stack of a_{n-1} squares in cell $n-1$, a_{n-2} squares in the cell $n-2$ and less than $|c_n| + 1$ squares in cell n.
  - If $a_n \leq |c_n|$, then $b_n > |c_n|$ by hypothesis and so we need to subtract $|c_n|M(\mathbf{a}_0^{n-3}, \mathbf{b}_0^{n-3}, \mathbf{c}_0^{n-3})$ inadmissible tiling, which in this case are those having stack of a_{n-2} squares in the cell $n-2$ and less than $|c_n| + 1$ dominoes covering the cells in positions $n-1$, n.

- Finally, if $c_n < 0$ and $b_n < 0$ then every tiling must finish with a stack of squares. By induction there are $a_nM(\mathbf{a}_0^{n-1}, \mathbf{b}_0^{n-1}, \mathbf{c}_0^{n-1})$ tilings that end with

a stack of squares. From this we need to subtract $|b_n|M(\mathbf{a}_0^{n-2}, \mathbf{b}_0^{n-2}, \mathbf{c}_0^{n-2})$ inadmissible tilings, those when there is a stack of a_{n-1} squares in the cell $n-1$ and less than $|b_n|+1$ squares in the cell n. Moreover we also need to subtract $(|b_n|+|c_n|)M(\mathbf{a}_0^{n-3}, \mathbf{b}_0^{n-3}, \mathbf{c}_0^{n-3})$ inadmissible tilings, i.e. when there is are stacks of a_{n-1} and a_{n-2} squares in the cells $n-1$ and $n-2$ respectively and less than $|c_n|+|b_n|+1$ squares in the cell n. However in this way we have counted twice the tilings having full stacks of a_{n-1} and a_{n-2} squares in the cells $n-1$ and $n-2$, and less than $|b_n|+1$ squares in the last cell, so we have to add up this coverings again. These are a total of $|b_n|M(\mathbf{a}_0^{n-3}, \mathbf{b}_0^{n-3}, \mathbf{c}_0^{n-3})$ tiling, obtaining the result stated by the thesis.

□

Observation A.4. The Jacobi algorithm has been generalized to higher dimensions by Perron [Per07] as follows:

$$\begin{cases} a_n^{(i)} = \lfloor \alpha_n^{(i)} \rfloor \\ \alpha_{n+1}^{(1)} = \dfrac{1}{\alpha_n^{(m)} - a_n^{(m)}} \\ \alpha_{n+1}^{(i)} = \dfrac{\alpha_n^{(i-1)} - a_n^{(i-1)}}{\alpha_n^{(m)} - a_n^{(m)}} \end{cases} \qquad n = 0, 1, 2, ...$$

starting from m real numbers $\alpha_0^{(1)}, ..., \alpha_0^{(m)}$ and providing a MCF

$$[(a_0^{(1)}, a_1^{(1)}, ...), ..., (a_0^{(m)}, a_1^{(m)}, ...)]$$

of degree m which is defined by the following relation

$$\begin{cases} \alpha_n^{(i-1)} = a_n^{(i-1)} + \dfrac{\alpha_{n+1}^{i}}{\alpha_{n+1}^{(1)}}, \quad i = 2, ..., m \\ \alpha_n^{(m)} = a_n^{(m)} + \dfrac{1}{\alpha_{n+1}^{(1)}} \end{cases} \qquad n = 0, 1, 2, ...$$

Our results about the MCF of degree 2 easily extends to a MCF of degree m by considering $m+1$ different tiles of length $1, 2, ..., m+1$. In this paper we deal with the case of degree 2 for the seek of simplicity about the notation.