MASTER'S THESIS

Master of Science in Advanced Mathematics and Mathematical Engineering

Title: Linkable Attribute-based Signature

Author: Héctor Masip Ardévol

Advisor: Paz Morillo Bosch and Ramiro Martínez Pinilla

Department: Mathematics

Academic year: 2019-2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master in Advanced Mathematics and Mathematical Engineering

Master's thesis

Linkable Attribute-based Signature

Héctor Masip Ardévol

Supervised by Paz Morillo Bosch and Ramiro Martínez Pinilla

September, 2020

Primer que res agraeixo a la Paz i al Ramiro per tota la seva ajuda i paciència en aquesta tesi, sobretot en els períodes en els quals, carinyosament, els massacrava a preguntes (algunes d'elles bastant "negligibles"). També m'agradaria reconèixer l'esforç de tots aquells mestres que han fet possible que en converteixi en la persona que sóc ara. En particular, dono les gràcies a Mamen per motivar-me a descobrir el meravellós món de les matemàtiques i a Diego, perquè, encara que no dominés l'idioma en el qual havia d'impartir l'assignatura, em va saber transmetre la seva passió per la criptografia (enganxant-me a ella).

Fora d'aquest món acadèmic, vull agrair a tots els amics, companys i "haters" per ensenyar-me tots els valors morals i socials que m'han permès desenvolupar-me en aquest món que tanta incertesa en causa a tots. Especialment valoro a Marta el suport que m'ha mostrat durant tota la tesi, sobretot quan em feia espavilar en escoltar coses del tipus "això és massa difícil, no entenc res de res, crec que me'n vaig a estudiar una altra carrera" o "vaja xorrada que em fan estudiar, prefereixo suspendre a estudiar-me això".

Per últim, i no menys important, voldria reconèixer l'esforç dels meus pares i la valentia del meu germà per ensenyar-me tot el que no es pot aprendre al cole, per dirigir-me en els moments difícils, per ajudar-me en qualsevol àmbit i per posar-me al lloc quan així ho requeria la situació. Tot això i més m'ha fet possible (en un temps rècord) arribar on estic ara i també ha aconseguit modelar aquesta personalitat de la qual tant gaudeixo.

Dedico tot aquest esforç a ma mare Isabel i al meu padrí Àngel, A.C.S.

Abstract

Attribute-based signatures, as introduced by Maji et al. [MPR11] allow to generate anonymously a signature for a message on behalf of a signing policy. To sign a message, a private attribute that satisfies the signing policy has to be used. It is furthermore not possible to identify the actual signer based on the signature. Attribute-based signatures have some important properties for electronic voting, e.g., it guarantees that a vote has been submitted by an eligible voter without revealing her identity. To be fully applicable in that setting, attribute-based signatures must allow not only to determine when some signatures have been emitted by the same person, but also it has to easily detect when the signatures have been issued by different persons.

In this thesis, we solve the aforementioned problems by constructing a linkable attribute-based signature (LABS) scheme for unbounded circuits in the random oracle model. We start by providing a generic construction of LABS schemes, where we define both the linkability and exculpability properties. We also provide a concrete instantiation of our generic LABS construction from protocols based on lattices. We do it by introducing a new zero-knowledge proof of knowledge for proving possession of a valid signature of the lattice-based signature scheme of Boschini et al. [Bos+20].

Keywords

cryptography, lattice, post-quantum cryptography, e-voting, digital signature scheme, attribute-based signature

Contents

1. Introduction

> "There are three kinds of cryptography in this world: cryptography that will stop your mother from reading your files, cryptography that will stop governments from reading your files and cryptography that makes you fall in love."

Historically, the term "cryptography" has been associated with the problem of designing and analyzing encryption schemes (i.e., schemes that provide secret communication over insecure communication media). However, since the 1970s, problems such as constructing unforgeable digital signatures and designing fault-tolerant protocols have also been considered as falling within the domain of cryptography.

Modern cryptography is concerned with the construction of schemes that should be able to withstand any abuse. Such schemes are constructed so as to maintain a desired functionality, even under malicious attempts aimed at making them deviate from their prescribed functionality.

Solving a cryptographic problem (or addressing a security concern) is a two-stage process consisting of a *definitional stage* and a *constructive stage*. First, in the definitional stage, the functionality underlying the natural concern must be identified and an adequate cryptographic problem must be defined. Trying to list all undesired situations is infeasible and prone to error. Instead, one should define the functionality in terms of operation in an imaginary ideal model and then require a candidate solution to emulate this operation in the real, clearly defined model (which will specify the adversary's abilities). Once the definitional stage is completed, one proceeds to construct a system that will satisfy the definition. Such a construction may use some simpler tools, and its security is to be proved relying on the features of these tools. (In practice, of course, such a scheme also may need to satisfy some specific efficiency requirements.)

Security of cryptographic protocols strongly rely on mathematical theory and computer science; these protocols are designed around computational hardness assumptions. It is theoretically possible to break such a protocol, but the strength comes from the fact that it is infeasible to do so by any practical means. Most of the actual popular systems rely on two well-known mathematical problems: the discrete logarithm and the integer factorization. However, these problems can be efficiently solved on a sufficiently powerful quantum computer by running Shor's algorithm [Sho99][1]. Even if a long-living quantum computer is far from being a reality, one should not avoid this possibility in the design and implementation of new cryptographic tools.

The main objective of this thesis is to construct new cryptographic protocols useful for electronic voting and prove its consistency and security against a quantum adversary. Specifically, our tool will provide voters with the ability of voting without using directly their identity, while at the same time every observer (from inside or outside the election process) will be able to distinguish between her casted vote and a different one. Additionally, everyone will have the possibility to check that all the votes have been cast by someone who had the right to vote.

[1] This algorithm works well for the general case of these problems, but one should use [PZ03] instead for the elliptic case.

1.1 Electronic voting structure and goals

Societies have conducted elections for thousands of years, but technologies used to cast and tally votes have varied and evolved tremendously over that time. In 2020 much of our communication takes place online, and many people want elections to follow this trend. Electronic voting offers several advantages over traditional elections. It needs less infrastructure, it is less expensive, it obtains faster the final results, it is more flexible and it considerably simplifies the voting procedure for electors that are abroad at the time of the election.

An ideal Internet voting election process consists in the following phases:

(1) **SETUP**. During the setup phase, election officials gather information needed to run the election. This includes:

- gathering registration information for all voters;
- identifying the issues and races that will be voted on;
- designing ballots, often in multiple languages, for all precincts participating in the election;
- sending instructions and other information about the election.

(2) **DISTRIBUTION**. Different voting systems use different mechanisms to distribute ballots to vote, such as postal mail, email, or a website that provides downloadable ballots.

(3) **VOTING**. Voters fill out their ballots, often with the help of software installed on their own devices.

(4) **CASTING**. Election officials receive the completed ballots as a previous step to perform the tallying. As with distribution, different voting systems use different ballot casting mechanisms.

(5) **TALLYING**. The tallying phase includes the remainder of the tasks that finalize the election. Counting votes and announcing the election outcome are common to almost every election system, though some include other tasks such as publishing information needed for audits.

(6) **AUDITING**. Some elections will inevitably be disputed. In such cases, there is a final phase in which interested parties look for evidences that the election outcome is correct.

As much as possible, Internet voting should be private and anonymous. It is essential that voters are free to vote the way they choose, and do not feel pressured to vote for a particular candidate or vote a particular way on any issue (coercion resistance). The fewer people who know or can find out how a voter voted, the more comfortable the voter can feel about privacy.

There are some typical properties that an electronic voting scheme must satisfy, such as usability and accessibility. The first one deals with the interface of the whole process and the second one is more concerned about expanding the amount of people that is able to vote. Even if in this thesis we will not be directly concerned about this kind of features, one typically have to have in mind if her protocols could be used for further research or for real-world applications.

It is also essential that the voter has access to a proof of her vote before the casting phase, so that she can check that her vote has not been modified or erased. However, she must not be able to show to a third party what she has voted; in order to avoid vote-selling. This is a key part in the comparison between an electronic voting process and a traditional one. When you vote in a traditional election there is no mechanism that allow you to re-vote or even to tell to you which of the envelopes is yours without breaking the anonymity. In the electronic version, this problem can be easily handled with the combination of an encryption scheme (voters "introduce" their votes in

an envelope) and a digital signature scheme. In Section 3.2 we introduce an improved version of a digital signature scheme, in the sense that it will attest not to the identity the individual who signed a message while at the same time we will be able to identify if different votes have been produced by the same person.

Figure 1: A ballot making machine tested by Microsoft in 2019.

Typically, a ballot is encrypted with a public key (which is known by everyone since the very beginning of the voting phase) and then decrypted with a private key. A natural question is to ask who is the trusted party that will hold the private key during the voting and the casting phases and then will decrypt the ballots to perform the tallying. To avoid dealing with this question, a (threshold) secret sharing system is introduced in the process. In this protocol, the private key is shared among n parties (which are assumed to have non-common interests). Each of these parties can compute a part of the decryption key, but at least t of them are needed to recover the message ($t \leq n$). The value t is called the *threshold* value, and is useful to ensure that the decryption will be possible even if a small amount of parties are dishonest.

An ideal election scheme that fulfills all the requirements consists of several parties and protocols and is still an actual research topic. The protocols presented in this thesis are intended to be key pieces of any flexible e-voting scheme.

1.2 Quantum computing

A classical computer processes the input data sequentially in order to solve a problem. Turing machines or logical circuits can be used to model classical computation. A Boolean circuit is made up of elementary gates and performs a computational task, where input bits are transformed into output bits. In each step, the system has a fixed state and, depending on the complexity of an algorithm and the size of the input data, the computation can be inefficient or infeasible: suppose that f is a vectorial Boolean function and an algorithm iterates over all $x \in \{0,1\}^n$ until $f(x)$ is equal to a fixed value y. The worst-case running time is exponential in n and the algorithm is inefficient on conventional computers, even if the individual computations of $f(x)$ run very fast.

Quantum computers can compute all 2^n values f(x) simultaneously. This sounds fantastic, but there is a catch: the computation requires a system of n qubits, the internal state is inaccessible and only a single output value can be extracted from a quantum system.

Quantum computing uses *quantum circuits* instead of Boolean circuits, and quantum algorithms take *qubits* (quantum bits) instead of classical bits as input. A qubit can assume infinitely many states between 0 and 1; the state is represented by a normalized vector in the vector space $\mathbb{C}^2$. We

fix an orthonormal basis of $\mathbb{C}^2$, for example the standard basis $e_1 = (1, 0)$ and $e_2 = (0, 1)$, and denote the basis states by $|0\rangle$ and $|1\rangle$.

Definition 1.1. The *Dirac* or *ket* notation of a vector ψ in the state space is $|\psi\rangle$. The state of a qubit $|\psi\rangle$ is a *superposition* (linear combination) of the basis states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = a|0\rangle + b|1\rangle, \text{ where } a, b \in \mathbb{C} \text{ and } |a|^2 + |b|^2 = 1$$

The coefficients of $|0\rangle$ and $|1\rangle$ can be interpreted as probabilities, and a qubit as a random variable. A measurement (observation) changes the state of a qubit and yields a classical bit, i.e., either 0 or 1. The state of a qubit determines the probability of the result of a measurement: the probability of 0 is $|a|^2$ and the probability of 1 is $|b|^2$. Thus the quantum information (a and b) is hidden and cannot be directly extracted. However, the quantum information can be used in quantum circuits that transform the state of qubits.

Now, imagine that it is ten years from now and someone announces the successful construction of a large quantum computer. All the newspapers around the world run a frontpage article reporting that all of the public-key algorithms used to protect the Internet have been broken. Users panic. What exactly will happen to cryptography? Perhaps, after seeing quantum computers destroy RSA and ECDSA (two of the most widely used cryptographic protocols nowadays), Internet users will leap to the conclusion that cryptography is dead; that there is no hope of scrambling information to make it incomprehensible to, and unforgeable by, attackers; that securely storing and communicating information means using expensive physical shields to prevent attackers from seeing the information.

A closer look reveals, however, that there is no justification for the leap from "quantum computers destroy RSA and ECDSA" to "quantum computers destroy cryptography". There are many important classes of cryptographic systems beyond the ones based on integer factorization or the discrete logarithm problem: hash based, code based, lattice based, multivariate polynomials and isogeny based (among others) are believed to be secure against a quantum capable adversary.

All of these systems are believed to resist classical computers and quantum computers attacks. Nobody has figured out a way to apply Shor's algorithm on them. Another quantum algorithm, Grover's algorithm, does have some applications to these systems; but Grover's algorithm [Gro96] is not as shockingly fast as Shor's algorithm, and cryptographers can easily compensate for it by choosing somewhat larger key sizes.

Some cryptographic systems, such as RSA with a large size key, are believed to resist attacks by classical computers but do not resist attacks by quantum computers. Some alternatives, such as McEliece (a code based encryption scheme) with a huge size key, are believed to resist attacks by classical computers and attacks by quantum computers.

So why do we need to worry now about the threat of quantum computers? Why not continue to focus on RSA? If someone announces the successful construction of a quantum computer ten year from now, why not simply switch to McEliece? There are three important drawbacks that parts of the cryptographic community are already starting to focus attention on postquantum cryptography:

- We need time to improve the efficiency of post-quantum cryptography.
- We need time to build confidence in post-quantum cryptography.
- We need time to improve the usability of post-quantum cryptography.

Anyway, what is making post-quantum cryptography so attractive for private companies and why they are starting to use it? Sensitive data. Imagine an scenario where encrypted sensitive data (such

as passwords for a bank account) is stolen from a company. Then, the stealer could just wait 10 years and use a quantum computer to decrypt it. This long-term strategy was known since it was created the concept of quantum computer, but it is nowadays when quantum computation seems to be close to us and everybody prefers to be prepared for that moment. Hence, post-quantum cryptography is not only a fresh topic to research on, but also a necessary tool to prevent "quantum intruders" to read from our data.

In short, we are not yet prepared for the world to switch to post-quantum cryptography. Maybe this preparation is unnecessary. Maybe we will not actually need post-quantum cryptography. Maybe nobody will ever announce the successful construction of a large quantum computer. However, if we do not do anything, and if it suddenly turns out years from now that users do need post-quantum cryptography, years of critical research time will have been lost.

1.3 Thesis structure

The organization of this thesis is as follows. We explain the notation, the basic primitives that we are going to use and an introduction to the theory of lattices in Section 2. We also provide in Section 2 a detailed description of the lattice-based protocols that will constitute a key part in the construction of our signature scheme. The main contribution of this thesis is the definition of a linkable attribute-based signature scheme presented in Section 3. In Section 4 we provide a general construction of this kind of signatures along with its security analysis. Moreover, we give in Section 5 an instantiation in the lattice setting showing that all the building blocks required by our generic construction are obtainable from lattices. We finish with some conclusions in Section 6.

2. Preliminaries

This section is dedicated to fix the background that will be used throughout this thesis, explains the different cryptographic primitives that are used in our scheme and provides a generalized version of a sigma protocol. We mainly devote this thesis to lattice based cryptography protocols, and hence we introduce lattices together with the principal lattice problems. Finally, we present the schemes that we use as our underlying building blocks.

2.1 Notation

Column vectors are denoted with lower-case bold letters $\mathbf{v}$, while matrices are denoted using upper-case bold letters $\mathbf{A}$. Furthermore, the *transposed* of a matrix $\mathbf{A}$ will be denoted $\mathbf{A}^T$ and given $m, k_1, k_2 \in \mathbb{N}$ we define $[m] = \{1, \dots, m\}$, $[k_1, k_2] = \{k_1, k_1 + 1, \dots, k_2 - 1, k_2\}$ and $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$. Given two vectors $\mathbf{u}, \mathbf{v}$ in an n-dimensional space, we write the standard inner product as $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i$.

When a is sampled uniformly at random from a set A we write $a \xleftarrow{\$} A$, $a \xleftarrow{\$} D$ when a is sampled according to a probability distribution D and $a \xleftarrow{\$} \mathcal{A}$ when a is the output of a probabilistic algorithm $\mathcal{A}$. Similarly, if a is the output of some deterministic algorithm $\mathcal{A}$, we will just omit the superscript \$ and write $a \leftarrow \mathcal{A}$.

For a vector $\mathbf{x} = (x_1, \dots, x_n)$ and $p \in [1, \infty]$, the ℓ_p norm is $\|\mathbf{x}\|_p = (\sum_{i \in [n]} |x_i|^p)^{1/p}$ when $p < \infty$ and $\|\mathbf{x}\|_\infty = \max_{i \in [n]} |x_i|$. Similarly, for a polynomial $f = a_0 + a_1 x + \cdots + a_n x^n$, $\|f\|_p = (\sum_{i \in [0,n]} |a_i|^p)^{1/p}$ and $\|f\|_\infty = \max_{i \in [0,n]} |a_i|$. This notation extends naturally for vectors and matrices of polynomials. Moreover, given a prime q, we choose $\left[-\left\lfloor \frac{q}{2} \right\rfloor, \dots, 0, \dots, \left\lfloor \frac{q}{2} \right\rfloor\right]$ to be the representatives of elements in $\mathbb{Z}_q$.

As for computational complexity, we use the traditional *big-O notation*. Namely, given two functions $f, g : \mathbb{N} \to \mathbb{R}$ we will use the following symbols:

- $f = O(g)$ if $\exists k > 0$ and $n_0 > 0$ such that $\forall n \geq n_0$, $|f(n)| \leq k \cdot g(n)$;
- $f = \Omega(g)$ if $g = O(f)$;
- $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$;
- $f = o(g)$ if $\forall k > 0$ $\exists n_0 > 0$ such that $\forall n \geq n_0$, $|f(n)| < k \cdot g(n)$;
- $f = \omega(g)$ if $g = o(f)$.

In addition, we will also use the *soft-O* notation to hide logarithmic factors, i.e., with $\tilde{O}(f)$ we mean $O(f \cdot \log^c f)$ for some fixed constant c.

Now, we give the definition of a negligible function. Intuitively, a negligible function is one that not only tends to zero as $n \to \infty$, but does so faster than the inverse of any polynomial. This functions capture the notion we want to achieve in practise when we talk about adversary's advantage.

Definition 2.1 (Negligible function)**.** A function $f : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for all $c \in \mathbb{R}_{\geq 0}$ there exists an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, we have

$$|f(n)| < \frac{1}{n^c}.$$

In computational hypothesis, we consider that a problem is *hard* if no Probabilistic Polynomial Time (PPT for short) adversary is able to solve it with non-negligible probability.

In cryptography, the *security parameter* is a variable that measures the input size of a computational problem. We express the security parameter in unary so that the time complexity of the cryptographic algorithm is polynomial in the size of the input: while the size of n is $\log n$, the size of 1^λ is λ. Moreover, we denote by $\mathrm{negl}(\lambda)$ the family of negligible functions on the security parameter 1^λ, i.e., functions that, whenever $\lambda \to \infty$, tend to zero faster than the inverse of any polynomial.

2.2 Cryptographic primitives

Here we present in detail some of the cryptographic primitives that take part in the construction of any robust electronic voting scheme.

2.2.1 Trapdoor functions

Informally, a *one-way function* is a function that is easy to compute on every input, but hard to invert given the image of a random input.

A trapdoor (one-way) function is a special kind of one-way function. Intuitively, a trapdoor function is a function f such that given an element x in its domain it is "easy" to compute the image $f(x)$, but given an element y in the codomain it is "infeasible"[2] to obtain a preimage $z \in f^{-1}(y)$ unless some secret information (called a *trapdoor*) is known.

Definition 2.2 (Trapdoor Function)**.** A *trapdoor function* is a function $f : X \to Y$ satisfying the following conditions:

- There exists a generator that, given a security parameter 1^λ, outputs a function f together with a trapdoor t_f. Computing $f(x)$ is an efficient process for all $x \in X$.
- There exists a PPT algorithm that, given $f(x)$ and t_f, efficiently computes $z \in X$ such that $f(z) = f(x)$. That is, f is easy to invert when t_f is known.
- For any PPT algorithm $\mathcal{A}$:

$$\Pr\left[f(z) = f(x) \mid (f, t_f) \xleftarrow{\$} \mathrm{KeyGen}(1^\lambda), x \xleftarrow{\$} X, z \xleftarrow{\$} \mathcal{A}(f, f(x))\right] \in \mathrm{negl}(\lambda)$$

To construct a trapdoor function that satisfies the above conditions, the hardness of the preimage computation should be based on some computational hard problem.

For instance, assuming that it is difficult to factorize large composite numbers we can construct a trapdoor function.

Example 2.3. Given two integers n, e such that $(e, \varphi(n)) = 1$, $d \equiv e^{-1} \pmod{\varphi(n)}$ is the trapdoor for the function

$$f(x) \equiv x^e \pmod{n}.$$

To check that, one just need to observe that $f(x)^d \equiv x^{ed} \equiv x \pmod{n}$.

[2] In our definition we will not specifically define what we mean by "easy" or "infeasible", as it will be clear by every particular case.

2.2.2 Commitment schemes

Suppose that Alice wishes to play "paper-scissor-stone" over the telephone with Bob. When conducted over the telephone we have the problem that whoever goes first is going to lose the game. One way around this for the party who goes first is to "commit" to their choice, in such a way that the other party can not determine what was committed to. Then the two parties can reveal their choices, with the idea that the other party can then verify that the revealing party has not altered its choice between the commitment and the revealing stage.

Such a scheme is called a commitment scheme, and the standard version of it is defined as follows:

Definition 2.4 (Commitment Scheme)**.** A *commitment scheme* with message space $\mathcal{M}$ and commitment space $\mathcal{C}$ is a triple of algorithms (C.Gen, C.Com, C.Ver):

- **C.Gen**: The generator algorithm takes as input the security parameter 1^λ and outputs a public commitment key pk. $\mathsf{pk} \xleftarrow{\$} \mathsf{C.Gen}(1^\lambda)$.
- **C.Com**: The commitment algorithm takes as input the commitment key pk and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening d. We denote as $\mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, m)$ the set of all possible outputs of this algorithm under fixed pk and m. $(c, d) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, m)$.
- **C.Ver**: The verification algorithm takes as input the commitment key pk, the message m, a commitment c and an opening d as inputs and outputs 1 or 0 (i.e., Valid or Invalid). We denote as $\mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m)$ the set of all possible pairs (c, d) this algorithm outputs 1 under fixed pk and m. $\{0, 1\} \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m, c, d)$.

We also require the commitment scheme to satisfy the following *correctness* notion:

- **Correctness**: For all $m \in \mathcal{M}$, $\mathsf{pk} \xleftarrow{\$} \mathsf{C.Gen}(1^\lambda)$, $(c, d) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, m)$ we have that

$$1 \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m, c, d)$$

  In other words, if the commitment has been built correctly and the valid message and opening are published the verifier algorithm always accepts.

  Note that $\mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m)$ contains not only the pairs that are generated by C.Com, but also those which are generated by a different algorithm and end up being labeled as Valid by C.Ver. In particular, this implies that $\mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, m) \subseteq \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m)$. The elements in $\mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m) \backslash \mathcal{D}_{\mathsf{Com}}(\mathsf{pk}, m)$ are the principal concern in the security notions.

  Finally, we say that a commitment scheme is *secure* if it satisfies the following two properties:

- **Binding**: A commitment can only be correctly opened to one message. This property can be perfect or computational.

  We call the scheme *perfectly binding* if:

$$1 \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m, c, d) \ \wedge \ 1 \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m', c, d') \implies m = m'$$

  We call the scheme *computationally binding* if, for all PPT adversaries $\mathcal{A}$:

$$\Pr[\mathsf{pk} \xleftarrow{\$} \mathsf{C.Gen}(1^\lambda); (c, m, m', d, d') \xleftarrow{\$} \mathcal{A}(\mathsf{pk}, 1^\lambda) : 1 \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m, c, d) \wedge 1 \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m', c, d') \ \wedge \ m \neq m'] \in \mathsf{negl}(\lambda)$$

- **Hiding**: It should not be possible to recover a message m from a commitment c.

  We call the scheme *perfectly hiding* if a commitment can be opened to any message:

$$\text{given } (c, d) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, m),\ \forall m' \exists d' \text{ such that } 1 \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m', c, d')$$

  We call the scheme *computationally hiding* if, for all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$:

$$|\Pr[\mathsf{pk} \xleftarrow{\$} \mathsf{C.Gen}(1^\lambda); (m_0, m_1, \mathsf{aux}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{pk}); b \xleftarrow{\$} \{0,1\}; (c, d) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, m_b); b' \xleftarrow{\$} \mathcal{A}_2(c, \mathsf{aux}) : b = b'] - 1/2| \in \mathsf{negl}(\lambda)$$

Note that a secure commitment scheme can not be perfectly binding and perfectly hiding at the same time, since this would lead to a contradiction. In Section 2.4.5 we define the commitment scheme that we use in our protocols, which is perfectly binding and computationally hiding.

2.2.3 Digital signatures

Now imagine a situation where Bob receives an email claiming to be from his friend Alice. Bob wants to verify that the email really is from Alice and he wants to do it without interacting more with her. Digital signatures provide a simple solution to this. When sending an email to Bob, Alice generates a signature on the message; and then she sends the message together with the signature to Bob. Now, Bob can both read the message and assure himself that the message comes from Alice.

Digital signatures are a crucial tool in electronic voting, allowing to verify the vote integrity once it is received in the voting server. They also allow the verification of the voter eligibility.

Definition 2.5 (Digital Signatures)**.** A *digital signature scheme* with message space $\{0,1\}^\ell$ is a triple of algorithms $(\mathsf{S.KeyGen}, \mathsf{S.Sign}, \mathsf{S.Verify})$:

- **S.KeyGen**: The key generation algorithm takes as input the security parameter 1^λ and the message length 1^ℓ, and outputs a verification key vk and signing key sk.
  $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$.
- **S.Sign**: The signing algorithm takes as inputs the signing key sk and a message $\mathbf{x} \in \{0,1\}^\ell$, and outputs a signature σ. $\sigma \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}, \mathbf{x})$.
- **S.Verify**: The verification algorithm takes as inputs the verification key vk, the message $\mathbf{x}$ and the signature σ, first checks that $\mathbf{x} \in \{0,1\}^\ell$, and then outputs 1 or 0 (i.e., Valid or Invalid).
  $\{0,1\} \leftarrow \mathsf{S.Verify}(\mathsf{vk}, \mathbf{x}, \sigma)$.

The digital signature scheme must satisfy the following *correctness* property:

- **Correctness**: For all $\lambda, \ell \in \mathbb{N}$ and $\mathbf{x} \in \{0,1\}^\ell$ we have that:

$$|\Pr[(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{S.KeyGen}(1^\lambda, 1^\ell); \sigma \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}, \mathbf{x}) : 1 \longleftarrow \mathsf{S.Verify}(\mathsf{vk}, \mathbf{x}, \sigma)] - 1| \in \mathsf{negl}(\lambda)$$

The common security notion for a digital signature is *existential unforgeability under an adaptive chosen message attack* (eu-acma for short). Roughly speaking, an adversary should not be able to forge a signature of a message of her choice. Even if this notion is not the strongest one, it is a

realistic one since it is not far from reality to assume that an adversary could have access to the signing algorithm as a black box.

As usual in most of security notions surrounding modern cryptography, it is modelled using the following attack game between an adversary $\mathcal{A}$ and a challenger:

1. The challenger runs $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$ and sends vk to $\mathcal{A}$.
2. $\mathcal{A}$ queries the challenger several times. For $i = 1, 2, \ldots$ the i-th *signing query* is a message $\mathbf{x}_i \in \{0,1\}^\ell$. Given $\mathbf{x}_i$, the challenger computes $\sigma_i \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}, m_i)$ and gives σ_i to the adversary $\mathcal{A}$.
3. Eventually $\mathcal{A}$ outputs a candidate forgery pair $(\mathbf{x}, \sigma)$. The adversary wins if $1 \longleftarrow \mathsf{S.Verify}(\mathsf{vk}, \mathbf{x}, \sigma)$ and $\mathbf{x}$ is not one of the messages $\mathcal{A}$ has made signature queries.

We define the *advantage of an adversary* $\mathcal{A}$ as the probability that $\mathcal{A}$ wins the above game.

Definition 2.6 (eu-acma)**.** We say that a digital signature scheme is eu-acma if the advantage of the above game is negligible for all PPT adversaries.

2.3 Zero-knowledge proofs

Now suppose that Alice wants to convince Bob that she knows something without Bob finding out exactly what Alice knows. This apparently contradictory state of affairs is dealt with using zero-knowledge proofs. In the literature of zero-knowledge proofs the role of Alice is taken by Peggy and called the prover, since she wishes to prove something, whilst the role of Bob is taken by Victor and called the verifier, since he wishes to verify that the prover actually knows something. There is a well-known story presenting the fundamental ideas of zero-knowledge proofs in the paper "How to Explain Zero-Knowledge Protocols to Your Children" [Qui+90].

To change a little bit, we present these ideas through the game *Where's Wally?*. *Where's Wally?* is a picture book game where the reader is challenged to find a small character called Wally hidden somewhere on a picture that is filled with many other characters. The pictures are designed so that it is hard to find Wally. An example of such a picture can be found in Figure 2.

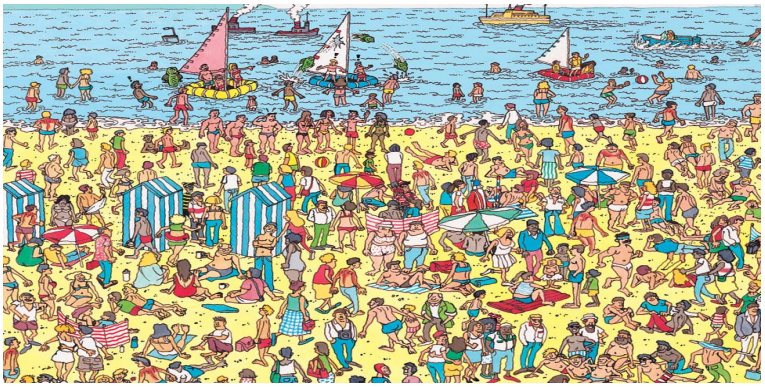


Figure 2: *Where's Wally?* average picture.

Imagine that you are a professional *Where's Wally?* solver. A company comes to you with a *Where's Wally?* book that they need to solve. The company wants you to prove that you are actually a professional *Where's Wally?* solver and thus asks you to find Wally in a picture from their

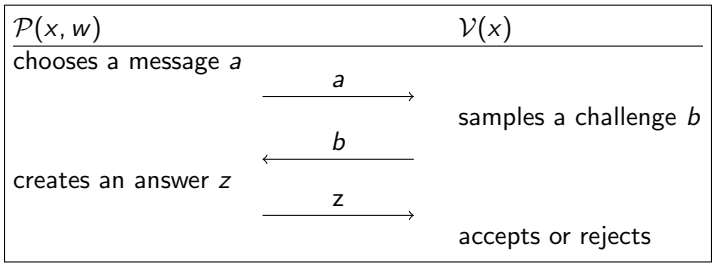
book. The problem is that you do not want to do work for them without being properly paid. Both you and the company want to cooperate, but you do not trust each other. It does not seem like it is possible to satisfy the company's demand without doing free work for them, but in fact there is a zero-knowledge proof which allows you to prove to the company that you know where Wally is in the picture without revealing to them how you found him.

The protocol goes as follows: You ask the company representative to turn around, and then you place a very large piece of cardboard over a random picture in the book such that the center of the cardboard is positioned over Wally. You cut out a small window in the center of the cardboard such that Wally is visible. You can now ask the company representative to turn around and view the large piece of cardboard with the hole in the middle, and observe that Wally is visible through the hole. The cardboard is large enough that they can not determine the page of the book under the cardboard. You can ask the representative to turn back around so you can remove the cardboard and give back the book.

As described, this proof is an illustration only, and not completely rigorous. Moreover, even if it is just a toy example, it is sufficient to understand the intuition behind zero-knowledge proofs.

Recall that a *language* $\mathcal{L} \subseteq \{0,1\}^*$ is said to have polynomial time recognizable *binary relation* $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ if $\mathcal{L} = \{x \mid \exists w \text{ s.t. } (x,w) \in \mathcal{R}\}$ where the size $|w|$ of w is at most $p(|x|)$ for some polynomial p. We call the string w a *witness* to the *statement* $x \in \mathcal{L}$ and we write $\mathcal{L}_{\mathcal{R}}$ to denote that the language $\mathcal{L}$ is induced by the binary relation $\mathcal{R}$.

A standard example of zero-knowledge proof in cryptography are Σ-protocols. Informaly, a Σ-protocol is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which, given an x, $\mathcal{P}$ tries to convince $\mathcal{V}$ that he knows a witness w such that $(x,w) \in \mathcal{R}$.



Protocol 3: Rounds in a Σ-protocol.

Definition 2.7 (Σ-protocol)**.** A protocol is said to be a Σ-protocol for the relation $\mathcal{R}$ if it satisfies:

- The protocol is of the above 3-form , and we have *completeness*: if $\mathcal{P}, \mathcal{V}$ follow the protocol on input x and private input w to $\mathcal{P}$ where $(x,w) \in \mathcal{R}$, the verifier always accepts.
- From any x and any pair of accepting conversations on input x with same message a, for instance $(a,b,z), (a,b',z')$ where $b \neq b'$, one can efficiently compute w such that $(x,w) \in \mathcal{R}$. This property is typically called *soundness*.
- There exists a polynomial-time simulator that takes as input x and a random b and outputs an accepted conversation (a,b,z) with the same probability distribution as conversations between honest $\mathcal{P}$ and $\mathcal{V}$ on input x. This property is known as *zero-knowledge*.

Example 2.8. Let p be a prime, q a prime divisor of $p-1$, and g an element of order q in $\mathbb{Z}_p^*$ (the multiplicative group of $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$). Suppose that a prover $\mathcal{P}$ has chosen $w \xleftarrow{\$} \mathbb{Z}_q$ and has published $h \equiv g^w \pmod{p}$. A verifier $\mathcal{V}$ who gets p, q, g, h can easily check that p, q are primes,

and that g, h have order q. Since there is only one subgroup of order q in $\mathbb{Z}_p^*$, this automatically means that $h \in \langle g \rangle$, i.e. there exists an w such that $h \equiv g^w$. But this does not necessarily mean that $\mathcal{P}$ knows such an w.

We can solve this problem by using the following protocol:

1. $\mathcal{P}$ chooses $r \xleftarrow{\$} \mathbb{Z}_q$ and sends $a \equiv g^r \pmod{p}$ to $\mathcal{V}$.

2. $\mathcal{V}$ chooses a challenge $b \xleftarrow{\$} \mathbb{Z}_{2^t}$ and sends it to $\mathcal{P}$. Here, t is a fixed positive integer such that $2^t < q$.

3. $\mathcal{P}$ sends $z \equiv r + bw \pmod{q}$ to $\mathcal{V}$, who checks that $g^z \equiv ah^b \pmod{p}$, that p, q are prime and that g, h have order q, and accepts iff this is the case.

We can easily check that the above protocol is a Σ-protocol:

- If the protocol is constructed and followed correctly, the verifier will always accept.
- If (a, b, z) and (a, b', z') are two accepted conversations, then we have that $b, b' \in \mathbb{Z}_q$ and $z \equiv r + bw \pmod{q}$ and $z' \equiv r + b'w \pmod{q}$. Subtracting we obtain $z - z' \equiv w(b - b') \pmod{q}$. Now, since $b \neq b'$ we have that $b - b' \not\equiv 0 \pmod{q}$, meaning that $b - b'$ has inverse mod q. Then, from two different challenges we could compute $w = (z - z')(b - b')^{-1} \pmod{q}$.
- To simulate it, simply choose at random the challenge b and z and then compute $a = g^z h^{-b} \pmod{p}$. This way, (a, b, z) is an accepted conversation and has exactly the same distribution as real conversations between the prover and the verifier.

However, such a scheme only convinces the verifier who interacts with the challenger. Then, to avoid repeating the above proof to many verifiers, normally this interactive protocol is transformed into a non interactive protocol through the Fiat-Shamir transformation [FS87]. In this method, the only who is playing a relevant role in the protocol is the prover, as the challenge of the verifier is replaced by a hash of the commitment. This allow to emulate a truly random behaviour of the verifier, so that the protocol still satisfies its three properties.

Definition 2.9 (Random Oracle)**.** A *random oracle* is modeled as a black box that responds to every query with a truly random answer chosen uniformly from its output domain. If a query is repeated, it responds with the same answer.

Definition 2.10 (Fiat-Shamir transformation)**.** Let $(\mathcal{P}, \mathcal{V})$ be a Σ-protocol for the relation $\mathcal{R}$, and $H(\cdot)$ a hash function with range equal to the verifier's challenge space. The *Fiat-Shamir transformation* of the σ-protocol is the non-interactive proof system $(\mathcal{P}^H, \mathcal{V}^H)$ defined as follows:

$\mathcal{P}^H(x, w)$: Run $\mathcal{P}(x, w)$ to obtain a message a, and compute $b \longleftarrow H(x, a)$. Then complete the run of $\mathcal{P}$ with b as the challenge to get the answer z. Finally, output the pair $\pi = (a, b, z)$ as the proof.

$\mathcal{V}^H(x, \pi)$: Return the output of $\mathcal{V}(a, b, z)$ if $b = H(x, a)$ and 0 otherwise.

It is a well known fact that in the random oracle model, the Fiat-Shamir transformation of any Σ-protocol is a non-interactive proof system.

Example 2.11. In order to make the Σ-protocol from Example 2.8 non-interactive, we just need to replace the challenge from step 2 by $b \longleftarrow H(g, h, a)$.

In this thesis we also use a variant of standard zero-knowledge proofs.

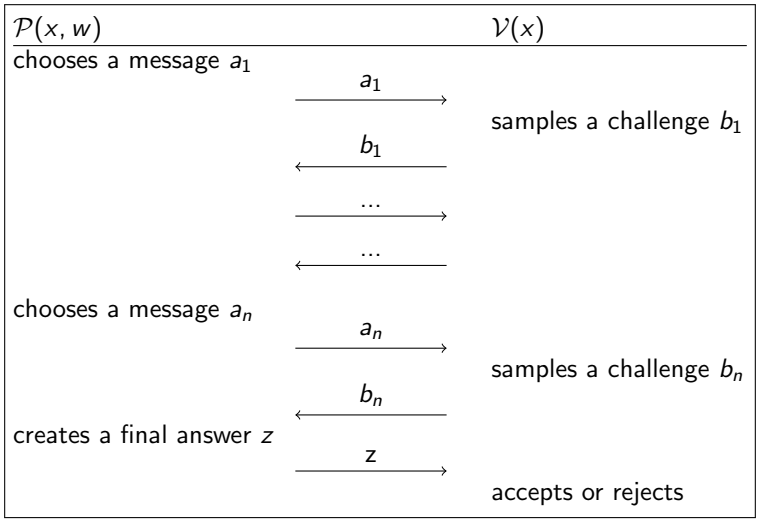
Definition 2.12 (Zero-Knowledge Proof)**.** A $(2n + 1)$-*move Public Coin Honest-Verifier Zero-Knowledge proof of Knowledge*, ZKPoK for short, is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which, given an x, $\mathcal{P}$ tries to convince $\mathcal{V}$ that he knows a witness w such that $(x, w) \in \mathcal{R}$.

$\mathcal{P}$ and $\mathcal{V}$ engage in an interaction where $\mathcal{P}$ consecutively sends a message (also referred as *commitment*) a_i answered by $\mathcal{V}$ with a random challenge b_i for $i = 1, \dots, n$. Finally $\mathcal{P}$ gives a final answer z and $\mathcal{V}$ outputs 1 or 0 (i.e., accepts or rejects the proof) checking the conversation $(x, \{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n, z)$. The conversation is called a *valid conversation* if the verifier $\mathcal{V}$ outputs 1.

The protocol verifies the following properties:

- **Completeness**: If an honest prover $\mathcal{P}$ knows a valid witness w such that $(x, w) \in \mathcal{R}$ and follows the protocol, then an honest verifier $\mathcal{V}$ always accepts the conversation.
- k-**Special Soundness**: From k valid conversations $\{(x, \{a_i^j\}_{i=1}^n, \{b_i^j\}_{i=1}^n, z^j)\}_{j=1}^k$, and $\{b_i^j\}_{i=1}^n \neq \{b_i^{j'}\}_{i=1}^n$ for all $j \neq j'$, it is possible to efficiently extract a witness w such that $(x, w) \in \mathcal{R}$.
- **Honest-Verifier Zero-Knowledge (HVZK)**: There exists a polynomial-time simulator $\mathcal{S}$ that takes $x \in \mathcal{L}_{\mathcal{R}}$ as input and random set of challenges $\{b_i\}_{i=1}^n$ and outputs a valid conversation $(x, \{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n, z)$ with the same probability distribution as conversations between honest $\mathcal{P}$ and $\mathcal{V}$. We will likely refer to this simulator as the *zero-knowledge simulator* of the protocol.

We say that the protocol is *computationally* (resp. *statistically*) *special* HVZK if the simulated transcript is computationally (resp. statistically) indistinguishable from a real transcript.



Protocol 4: $(2n + 1)$ rounds in a ZKPoK.

As usual, completeness is required to make the protocol consistent. k-Special Soundness basically means that a prover able to answer k challenges is honest, as in this case a witness could be extracted. Finally, zero-knowledge simply tells us that no verifier learns anything other than the fact that the statement is true. In other words, just knowing the statement (not the witness) is sufficient to imagine a scenario showing that the prover knows the secret.

2.4 Lattices

In this section we recall some definitions about lattices and some well-known lattice problems and properties that are inherent to our construction. In this section we assume that we are given a norm $\|\cdot\|$, and we consider it being the Euclidean norm (i.e., the ℓ_2 norm $\|\cdot\|_2$) unless stated otherwise.

2.4.1 Introduction to lattices

Definition 2.13 (Lattice)**.** A *lattice* $\mathcal{L}$ is a set of points in an n-dimensional space, usually $\mathbb{R}^n$, with a periodic structure. That is, the following conditions hold:

- $\mathbf{0} \in \mathcal{L}$ and $\forall \mathbf{x}, \mathbf{y} \in \mathcal{L}$ we have that $-\mathbf{x}, \mathbf{x}+\mathbf{y} \in \mathcal{L}$ [additive subgroup]
- $\forall \mathbf{x} \in \mathcal{L}$ we have that $B_\epsilon(\mathbf{x}) \cap \mathcal{L} = \{\mathbf{x}\}$ for some $\epsilon > 0$. [discrete]

where $B_\epsilon(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n \mid \|\mathbf{x}-\mathbf{y}\| \leq \epsilon\}$.

Although every (non-trivial) lattice $\mathcal{L}$ is infinite, it is always generated as the integer linear combinations of some linearly independent basis vectors (see Figure 5) in the following way.

Definition 2.14 (Full rank lattice)**.** Given n linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_n} \in \mathbb{R}^n$, the *full-rank lattice* generated by them is the set:

$$\mathcal{L}(\mathbf{b_1}, \ldots, \mathbf{b_n}) = \left\{ \sum_{i=1}^{n} z_i \mathbf{b_i} \mid z_i \in \mathbb{Z} \right\} = \{\mathbf{Bz} \mid \mathbf{z} \in \mathbb{Z}^n\} = \mathcal{L}(\mathbf{B})$$

where $\mathbf{B}$ is the the matrix whose columns are the vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$. The vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$ are known as the *basis* of the lattice.

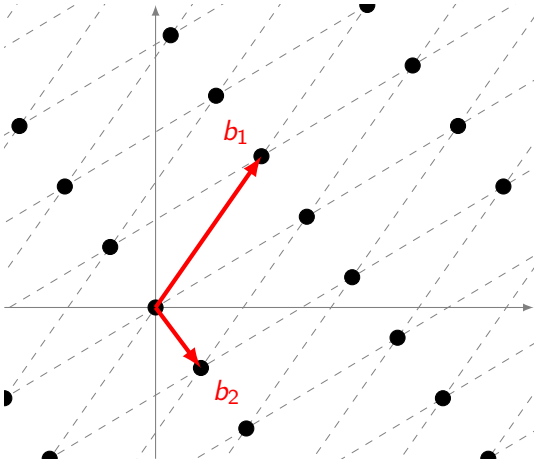


Figure 5: A lattice $\mathcal{L}(\mathbf{b_1}, \mathbf{b_2})$ in $\mathbb{R}^2$.

Since we are going to deal only with full rank lattices, from now on we will simply address to them with the term *lattices*.

Short lattice vectors play an important role in cryptographic applications. They are the basis of most of the problems whose hardness will be used in our constructions.

Definition 2.15. We define the *minimum* $\lambda_i(\mathcal{L})$ as:

(1) The norm of the shortest nonzero lattice vector $\mathbf{v} \in \mathcal{L}$ defines $\lambda_1(\mathcal{L}) = \|\mathbf{v}\|$.

(2) The i-th successive minimum is defined $\lambda_i(\mathcal{L}) = \min_S(\max_{\mathbf{v}\in S} \|\mathbf{v}\|)$, where S runs over all linearly independent sets $S \subset \mathcal{L}$ with $|S| = i$.

Lattice-based cryptography constructions are based on the presumed hardness of lattice problems, the most basic of which is the *Shortest Vector Problem* (SVP).

Definition 2.16 (SVP)**.** Given a base $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a non-zero lattice vector $\mathbf{v} \in \mathcal{L}$ for which $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

Here, we are given as input a lattice represented by its basis, and our goal is to output the shortest nonzero vector in it. In fact, one typically considers the approximation variant of SVP [3] (denoted SVP_γ) where the goal is to output a lattice vector whose length is at most some approximate factor $\gamma(n)$ times the length of the shortest nonzero vector, where n is the dimension of the lattice:

Definition 2.17 (Approximate Shortest Vector Problem (SVP_γ))**.** Given a base $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a non-zero lattice vector $\mathbf{v} \in \mathcal{L}$ for which $\|\mathbf{v}\| \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$.

The most well known and widely studied algorithm to attempt to solve this lattice problem is the *LLL algorithm*, developed by Lenstra, Lenstra and Lovász (see [LLL82]). This is a polynomial time algorithm for SVP that achieves an approximation factor of $2^{O(n)}$. Anyways, if one insists of an exact solution to SVP, or even just an approximation to within polynomial factors, the best known algorithm has a running time of $2^{O(n)}$. The space requirement for this algorithm is unfortunately also exponential which makes it essentially impractical.

This discussion leads us to the following conjecture.

Conjecture 2.18. *There is no polynomial time algorithm that approximates lattice problems to within polynomial factors.*

Less informally, it is conjectured that approximating lattice problems to within polynomial factors is a hard problem. The security of many lattice-based cryptographic constructions is based on this conjecture.

The field of lattice-based cryptography has been developed based on the assumption that lattice problems are hard. But is lattice-based cryptography suitable for a post-quantum world? Are lattice problems hard even for quantum computers? The correct answer to this questions is "probably yes". There are currently no known quantum algorithms for solving lattice problems that perform significantly better than the best known classical (i.e., non quantum) algorithms.

Attempts to solve lattice problems by quantum algorithms have been made since Shor's discovery of the quantum factoring algorithm [Sho99], but have so far met with little success within polynomial time. The main difficulty is that the periodicity finding technique, which is used in Shor's factoring algorithm and related quantum algorithms, does not seem to be applicable to lattice problems. It is therefore natural to consider the following conjecture, which justifies the use of lattice-based cryptography for post-quantum cryptography:

Conjecture 2.19. *There is no polynomial time quantum algorithm that approximates lattice problems to within polynomial factors.*

[3] This kind of variant is not only defined for the SVP problem, but they are also similarly defined for most of lattice based problems.

2.4.2 Lattice problems

Any lattice admits multiple basis, and this fact is at the core of many cryptographic applications.

Theorem 2.20. *$\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ if and only if there exists a unimodular matrix $\mathbf{U}$ (i.e., an integer square matrix with determinant ± 1) such that $\mathbf{B}' = \mathbf{BU}$.*

Proof. ($\Longrightarrow$) If $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ then each column $\mathbf{b}'_\mathbf{i}$ of $\mathbf{B}'$ (or any combination of them) is a point of the lattice $\mathcal{L}(\mathbf{B})$. That means that $\mathbf{b}'_i = \mathbf{B}\mathbf{u}_\mathbf{i}$, where $\mathbf{u}_\mathbf{i} \in \mathbb{Z}^n$, and repeating this argument for all the columns of $\mathbf{B}'$ we have that $\mathbf{B}' = \mathbf{BU}$, where $\mathbf{U}$ is an integer matrix. Analogously we have that $\mathbf{B} = \mathbf{B}'\mathbf{V}$, with $\mathbf{V}$ another integer matrix. Combining the both expressions we have that $\mathbf{B} = \mathbf{BUV}$. Then $\mathbf{B}(\mathbf{I}_n - \mathbf{UV}) = \mathbf{0}$, where $\mathbf{I}_n$ is the identity matrix. Multiplying by the inverse of $\mathbf{B}$ we have $\mathbf{I}_n = \mathbf{UV}$ which means that $\mathbf{U}$ is unimodular.

($\Longleftarrow$) $\mathbf{B}' = \mathbf{BU}$ means that the columns of $\mathbf{B}'$ are integer combinations of columns of $\mathbf{B}$ ($\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$). Multiplying the previous equality by $\mathbf{U}^{-1}$ (which is also unimodular) we have that $\mathbf{B} = \mathbf{B}'\mathbf{U}^{-1}$ meaning that the columns of $\mathbf{B}$ are integer combinations of columns of $\mathbf{B}'$ ($\mathcal{L}(\mathbf{B}) \subseteq \mathcal{L}(\mathbf{B}')$). Hence $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$. ■

This theorem leads us to define the determinant of lattices:

Definition 2.21. The *determinant* of a lattice is the absolute value of the determinant of the basis matrix

$$\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$$

Definition 2.22 (Dual Lattice)**.** The *dual* of a lattice $\mathcal{L} \subset \mathbb{R}^n$ is the lattice:

$$\mathcal{L}^* = \{\mathbf{y} \in \mathbb{R}^n \mid \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z} \text{ for all } \mathbf{x} \in \mathcal{L}\}$$

Proposition 2.23. *Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ be a basis of a lattice $\mathcal{L}$. Then $\mathcal{L}^*$ is the lattice generated by $(\mathbf{B}^T)^{-1} = (\mathbf{B}^{-1})^T$. Moreover, $\det(\mathcal{L}^*) = 1/\det(\mathcal{L})$.*

Proof. It is easy to see that $\mathcal{L}^*$ is a lattice. By definition, $\mathbf{y} \in \mathcal{L}^*$ if and only if $\mathbf{B}^T\mathbf{y} \in \mathbb{Z}^n$. This is the same as $\mathbf{y} = (\mathbf{B}^T)^{-1}\mathbf{x}$ for some $\mathbf{x} \in \mathbb{Z}^n$. This implies that the dual $\mathcal{L}^*$ of a lattice $\mathcal{L}$ is generated by the columns of $(\mathbf{B}^T)^{-1}$. Furthermore, it follows that

$$\det(\mathcal{L}^*) = \left|\det\left((\mathbf{B}^T)^{-1}\right)\right| = \frac{1}{|\det(\mathbf{B})|} = \frac{1}{\det(\mathcal{L})}$$

■

In cryptography we will be more concerned about q-ary lattices, because they are easier to work with in a computer. Intuitively, a q-ary lattice is an integer lattice where the membership of a point $\mathbf{x}$ in $\mathcal{L}$ is determined by $\mathbf{x} \pmod q$. Such lattices are in one-to-one correspondence with linear codes in $\mathbb{Z}_q^n$.

Definition 2.24 (q-ary lattice)**.** A lattice $\mathcal{L}$ is a *q-ary lattice* if $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$.

It is straightforward to see that the following are two q-ary lattices:

Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some integers q, n, m, we can define two m-dimensional q-ary lattices:

$$\Lambda_q(\mathbf{A}) = \left\{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y} = \mathbf{A}^T\mathbf{z} \pmod q : \mathbf{z} \in \mathbb{Z}^n\right\}$$

and

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{y} = \mathbf{0} \pmod q\}.$$

Note that both lattices have full rank since they contain $q\mathbb{Z}^m$.

$\Lambda_q(\mathbf{A})$ and $\Lambda_q^{\perp}(\mathbf{A})$ can also be viewed as *linear codes*: the first is generated by the rows of $\mathbf{A}$ and the second contains all vectors that are orthogonal modulo q to the rows of $\mathbf{A}$. In other words, the first q-ary lattice corresponds to the code generated by the rows of $\mathbf{A}$ whereas the second corresponds to the code whose parity check matrix is $\mathbf{A}$. It is a well-known fact that these two lattices are dual to each other, up to normalization:

$$\Lambda_q^{\perp}(\mathbf{A}) = q\Lambda_q(\mathbf{A})^* \text{ and } \Lambda_q(\mathbf{A}) = q\Lambda_q^{\perp}(\mathbf{A})^*.$$

These lattices will also allow us to define some more lattice problems.

Definition 2.25 (Short Integer Solution ($\mathrm{SIS}_{n,m,q,\beta}$))**.** Let $\mathbf{a}_i \in \mathbb{Z}_q^n$ be m uniformly random vectors forming the columns of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$. Assuming that $q > \beta$, find a non-zero integer vector $\mathbf{x} \in \mathbb{Z}^m$ of norm $\|\mathbf{x}\| \leq \beta$ such that

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{m} \mathbf{a_i} \cdot x_i = \mathbf{0} \in \mathbb{Z}_q^n.$$

Here is important to remark some observations about the SIS problem:

- Without the upper-bound on $\|\mathbf{x}\|$, one may use Gaussian elimination to find a solution. If $q \leq \beta$, then $\mathbf{z} = (0, \dots, 0, q, 0, \dots, 0) \in \mathbb{Z}^m$ would be a trivial solution.
- The norm β and the number m of vectors $\mathbf{a}_i$ must be large enough that a solution is guaranteed to exist. This is the case whenever $\beta \geq \sqrt{\bar{m}}$ and $m \geq \bar{m}$, where $\bar{m}$ is the smallest integer greater than $n \log q$, by a pigeonhole argument. There are more than q^n vectors $\mathbf{x} \in \{0,1\}^m$, there must be two distinct $\mathbf{x}, \mathbf{x}'$ such that $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}' \in \mathbb{Z}_q^n$, so their difference $\mathbf{z} = \mathbf{x} - \mathbf{x}' \in \{0, 1, -1\}^m$ is a solution of norm at most β.

Finding short vectors in the dual lattice $\Lambda_q^{\perp}(\mathbf{A})$ is finding a short solution to a set of n random equations modulo q in m variables. The SIS problem asks to find a sufficiently short non-zero vector in $\Lambda_q^{\perp}(\mathbf{A})$, where $\mathbf{A}$ is chosen uniformly at random.

We can also define an inhomogeneous version of the SIS problem.

Definition 2.26 (Inhomogeneous Short Integer Solution ($\mathrm{ISIS}_{n,m,q,\beta}$))**.** Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be an uniformly random matrix and $\mathbf{u} \in \mathbb{Z}_q^n$ an uniformly random vector. Find a non-zero vector $\mathbf{x} \in \mathbb{Z}^m$ of norm $\|\mathbf{x}\| \leq \beta$ such that $\mathbf{A}\mathbf{x} = \mathbf{u}$.

Note that finding two different solutions to the ISIS problem implies finding a solution to the SIS problem. These lattice problems are considered as hard as well known hard lattice problems (see [MR07] for the reductions).

One of the most known lattice problems to build up cryptographic protocols is the Learning With Errors (LWE).

Definition 2.27 (LWE distribution)**.** Let n, q be integers, χ a discrete probability distribution over $\mathbb{Z}$ (usually a discrete Gaussian distribution) and $\mathbf{s}$ a "secret" vector from $\mathbb{Z}_q^n$.

We denote by $\mathcal{A}_{\mathbf{s},\chi}$ the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \xleftarrow{\$} \chi$ and considering it in $\mathbb{Z}_q$ and calculating $(\mathbf{a}, b = (\langle \mathbf{a}, \mathbf{s}\rangle + e)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

There are two main versions of the LWE problem: the *search* version, which is to find the secret given LWE samples, and the *decision* version, which is to distinguish between LWE samples and uniformly random ones.

Definition 2.28 (Search-LWE$_{n,m,q,\chi}$)**.** Given m independent samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ drawn from $\mathcal{A}_{\mathbf{s},\chi}$, find $\mathbf{s}$.

Definition 2.29 (Decision-LWE$_{n,m,q,\chi}$)**.** Given m independent samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where every sample is distributed according to either $\mathcal{A}_{\mathbf{s},\chi}$ or the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$, distinguish which is the case.

Again, it is important to remark some observations about the search and decision LWE problem:

- Without the error term e, both problems are easy to solve, because we can efficiently recover $\mathbf{s}$ from LWE samples by Gaussian elimination.
- It is often convenient to combine the given samples into a matrix $\mathbf{A} \in Z_q^{n \times m}$ (whose columns are the vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$) and a vector $\mathbf{b} \in \mathbb{Z}_q^m$ (whose entries are the $b_i \in \mathbb{Z}_q$), so that for LWE samples we have
$$\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T,$$
where $\mathbf{e} \xleftarrow{\$} \chi^m$.

If we consider $\mathbf{A}$ as the matrix that defines $\Lambda_q(\mathbf{A})$ then the search version problem becomes recovering the coordinates of a point in this lattice after perturbing it with the error, while the decisional version is to distinguish random points in $\mathbb{Z}^m$ from perturbed $\Lambda_q(\mathbf{A})$ points.

It has been shown that any polynomial number of samples from the LWE look random to any polynomial-time attacker (it is as hard as the approximated SVP [LPR13]).

2.4.3 Ideal lattices

Observe that $\mathbf{A}$ has quadratic size in the dimension of the lattice, so that it yields to a high communication cost. In fact, the efficiency of lattice-based cryptography can be substantially improved replacing general matrices by matrices with a special structure.

Definition 2.30. Given a vector $\mathbf{f} = (f_0, \dots, f_{n-1}) \in \mathbb{Z}^n$ we define the *transformation matrix associated to* $\mathbf{f}$ as:
$$\mathbf{F} = \left(\begin{array}{ccc|c} & \mathbf{0}^T & & -f_0 \\ \hline \ddots & & & -f_1 \\ & \mathbf{I}_{n-1} & & \vdots \\ & & \ddots & -f_{n-1} \end{array}\right).$$

Definition 2.31 (Ideal Lattice)**.** An *ideal lattice* is a lattice $\mathcal{L}(\mathbf{A})$ generated by a block matrix $\mathbf{A} = \left(\mathbf{A}^{(1)} \mid \cdots \mid \mathbf{A}^{(m/n)}\right)$ whose blocks $\mathbf{A}^{(i)}$ are constructed from a vector $\mathbf{a}^{(i)}$ and a transformation matrix $\mathbf{F}$:
$$\mathbf{A}^{(i)} = \mathbf{F}^*\mathbf{a}^{(i)} = \left(\mathbf{a}^{(i)}, \mathbf{F}\mathbf{a}^{(i)}, \dots, \mathbf{F}^{n-1}\mathbf{a}^{(i)}\right).$$

Example 2.32. Let $\mathbf{f} = (-1, 0, \dots, 0)$. Then, we have that each block $\mathbf{A}^{(i)}$ is a circulant matrix

$$\mathbf{A}^{(i)} = \begin{pmatrix} a_1^{(i)} & a_n^{(i)} & \dots & a_3^{(i)} & a_2^{(i)} \\ a_2^{(i)} & a_1^{(i)} & \dots & a_4^{(i)} & a_3^{(i)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1}^{(i)} & a_{n-2}^{(i)} & \dots & a_1^{(i)} & a_n^{(i)} \\ a_n^{(i)} & a_{n-1}^{(i)} & \dots & a_2^{(i)} & a_1^{(i)} \end{pmatrix},$$

i.e., a matrix whose columns are all cyclic rotations of the first column $\mathbf{a}^{(i)} = (a_1^{(i)}, \dots, a_n^{(i)})$.

Ideal lattices with block matrices like the previous one are typically called *cyclic lattices*. An important implication of the circulant structure of the blocks is that it reduces the amount of space needed from nm elements from $\mathbb{Z}_q$ to just m elements, because each block $\mathbf{A}^{(i)}$ is fully specified by its first column $\mathbf{a}^{(i)} = (a_1^{(i)}, \dots, a_n^{(i)})$. No better algorithms (than those for general lattices) are known to solve some of the underlying lattice problems for such cyclic lattices. So, it is reasonable to assume that solving some of the underlying lattice problems on these lattices is as hard as the general case.

For general $\mathbf{f}$, the corresponding lattices have been called ideal lattices because they can be equivalently characterized as ideals in the polynomial ring $R = \mathbb{Z}[x]/\langle f(x)\rangle$, where $f(x) = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} + x^n \in \mathbb{Z}[x]$. The reason for the name *ideal* is that multiplying two polynomials $a, b \in R$ is equivalent to multiply the matrix $\mathbf{A}$ constructed from the vector $\mathbf{a}$ (here and whenever it is necessary, we identify a polynomial a with a vector $\mathbf{a}$ whose coordinates are the coefficients of the polynomial) with the vector $\mathbf{b}$.

An important question to remark here is asking how hard is to find short vectors in the lattice $\Lambda_q(\mathbf{A})$, where $\mathbf{A}$ is the block matrix defined above. It has been shown in [Mic07] that if $\mathbf{f} = (-1, 0, \dots, 0)$ then short vectors can be easily found in time $O(q)$. On the other hand, finding short vectors in $\Lambda_q(\mathbf{A})$ on the average is as hard as solving various lattice problems in the worst case over ideal lattices (see [LM06] for the reductions), provided that the vector $\mathbf{f}$ satisfies the following two properties:

- For any two unit vectors $\mathbf{u}, \mathbf{v}$, the vector $[\mathbf{F}^*\mathbf{u}]\mathbf{v}$ has small norm.
- The polynomial $f(x) = f_0 + f_1 x + \cdots + f_{n-1}x^{n-1} + x^n \in \mathbb{Z}[x]$ is irreducible over the integers, i.e., it does not factor into the product of integer polynomials of smaller degree.

Notice that the first property is satisfied by the vector $\mathbf{f} = (-1, 0, \dots, 0)$ corresponding to circulant matrices, but the polynomial $x^n - 1$ corresponding to $\mathbf{f}$ is not irreducible.

To solve this problem, we typically choose $\mathbf{f}$ to be $(1, 0, \dots, 0)$, so that $f(x) = x^n + 1$, with n a power of 2, as it satisfies both properties and gives us nice security reductions. This way each block of the matrix $\mathbf{A}$ is an anti-cyclic matrix:

$$\mathbf{A}^{(i)} = \begin{pmatrix} a_1^{(i)} & -a_n^{(i)} & \dots & -a_3^{(i)} & -a_2^{(i)} \\ a_2^{(i)} & a_1^{(i)} & \dots & -a_4^{(i)} & -a_3^{(i)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1}^{(i)} & a_{n-2}^{(i)} & \dots & a_1^{(i)} & -a_n^{(i)} \\ a_n^{(i)} & a_{n-1}^{(i)} & \dots & a_2^{(i)} & a_1^{(i)} \end{pmatrix}$$

so that every element above the diagonal in circulant matrices is replaced by its negative.

No algorithm is known to take advantage of the structure in anti-cyclic lattices to efficiently solve the ideal (or ring) version of the lattice problems presented before.

In the following we denote by R_q to $\mathbb{Z}_q[x]/\langle f(x)\rangle$, where $f(x)$ is a polynomial of degree n. If not stated otherwise, in what follows we choose $f(x)$ to be $x^n + 1$, with n a power of 2. Similarly as R, ideals in R_q corresponds with lattices in $\mathbb{Z}_q^n$.

Definition 2.33 (Ring Short Integer Solution (R-SIS$_{n,m,q,\beta}$))**.** Let $\mathbf{a} = (a_1, \dots, a_m) \in R_q^m$ be a uniformly random vector of polynomials. Find a non-zero vector of polynomials $\mathbf{x} = (x_1, \dots, x_m) \in R^m$ of norm $\|\mathbf{x}\| \leq \beta$ such that

$$\langle \mathbf{a}, \mathbf{x} \rangle = \sum_{i=1}^{m} a_i x_i = 0 \in R_q.$$

There are efficient reductions from the ideal version of SVP$_\gamma$ to R-SIS, so that this problem is also believed to be hard [PR06].

Now we define the main problem we often work with. It is the ideal version of LWE, which is called Ring Learning With Errors (R-LWE$_{n,m,q,\chi}$).

Definition 2.34 (R-LWE distribution)**.** Let n, q be integers, χ a discrete probability distribution over R (usually a discrete Gaussian distribution) and s a "secret" element from R_q.

We denote by $\mathcal{A}_{\mathbf{s},\chi}$ the probability distribution over $R_q \times R_q$ obtained by choosing $a \in R_q$ uniformly at random, choosing $\mathbf{e} \xleftarrow{\$} \chi$ and considering it in R_q and calculating $(a, b = (a \cdot s + e)) \in R_q \times R_q$.

Again, there are two versions of the problem:

Definition 2.35 (Search-R-LWE$_{n,m,q,\chi}$)**.** Given m independent samples $(a_i, b_i) \in R_q \times R_q$ drawn from $\mathcal{A}_{s,\chi}$, find s.

Definition 2.36 (Decision-R-LWE$_{n,m,q,\chi}$)**.** Given m independent samples $(a_i, b_i) \in R_q \times R_q$ where every sample is distributed according to either $\mathcal{A}_{s,\chi}$ or the uniform distribution over $R_q \times R_q$, distinguish which is the case.

If parameters are chosen properly, the R-LWE problem becomes as hard as well known hard ideal lattice problems such as the ideal version of the approximated Shortest Vector Problem (SVP$_\gamma$) (see [LPR13]).

2.4.4 Trapdoors from lattices

Typical operations in lattice-based cryptography are the selection of uniformly random integer matrices $\mathbf{A}$ modulo q, and then the evaluation of simple linear functions like

$$f_{\mathbf{A}}(\mathbf{x}) := \mathbf{A}\mathbf{x} \pmod q \quad \text{and} \quad g_{\mathbf{A}}(\mathbf{x}, \mathbf{e}) := \mathbf{x}^T\mathbf{A} + \mathbf{e}^T \pmod q.$$

on short integer vectors $\mathbf{x}, \mathbf{e}$. One can easily check that an adversary able to invert these functions could solve some hard problems such as ISIS (for the former) and LWE (for the latter).

All lattice cryptography is not just as simple as choosing random integer matrices $\mathbf{A}$ and the evaluation of linear functions. In fact, these operations yield only collision-resistant hash functions, public-key encryption schemes that are secure under passive attacks, and similar non-complex constructions. From a practical point of view, advanced schemes such as chosen ciphertext secure

encryption also require generating a matrix $\mathbf{A}$ together with some *trapdoor*, typically in the form of a nonsingular square matrix $\mathbf{S}$ of short integer vectors such that

$$\mathbf{AS} = 0 \pmod{q}.$$

That is, a short basis of the lattice $\Lambda_q^{\perp}(\mathbf{A})$.

Definition 2.37. We say that a matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is *primitive* if

$$\mathbf{G} \cdot \mathbb{Z}^m = \mathbb{Z}_q^n,$$

i.e., if the columns of $\mathbf{G}$ generate all of $\mathbb{Z}_q^n$.

Works before Micciancio and Peikert [MP12] focused on constructing a random matrix $\mathbf{A}$ through some specialized and complex process to obtain a trapdoor. Micciancio and Peikert changed this approach by not starting directly dealing with $\mathbf{A}$, but instead crafting a public matrix $\mathbf{G}$, for which the associated functions $f_{\mathbf{G}}$ and $g_{\mathbf{G}}$ admit very efficient and high-quality inversion algorithms. $\mathbf{G}$ is normally called a *gadget* matrix and it is built (in its non-ring version) as the tensor product of a primitive vector and the identity matrix.

Now, we extend $\mathbf{G}$ into a semi-random matrix $\mathbf{A}' = [\bar{\mathbf{A}} \mid \mathbf{G}]$, for uniformly random matrix $\bar{\mathbf{A}}$. It is easy to see how the task of preimage sampling for $f_{\mathbf{A}'}$ reduces very efficiently to the corresponding task for $f_{\mathbf{G}}$:

$$f_{\mathbf{A}'}(\mathbf{x}) = [\bar{\mathbf{A}} \mid \mathbf{G}]\mathbf{x} = \bar{\mathbf{A}}\mathbf{x}_1 + \mathbf{G}\mathbf{x}_2 = \mathbf{y};$$

$$f_{\mathbf{A}'}^{-1}(\mathbf{y}) = (\mathbf{x}_1^T \mid f_{\mathbf{G}}^{-1}(\mathbf{y} - \bar{\mathbf{A}}\mathbf{x}_1)^T); \tag{1}$$

where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ has been divided appropriately. So to invert $f_{\mathbf{A}'}$ one would just need to invert $f_{\mathbf{G}}$.

Finally, we simply apply to $\mathbf{A}'$ a transformation defined by the matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{I} & -\mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

for a random "short" secret matrix $\mathbf{R}$ that will serve as a trapdoor, to obtain

$$\mathbf{A} = \mathbf{A}' \cdot \mathbf{T} = [\bar{\mathbf{A}} \mid \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}].$$

The transformation $\mathbf{T}$ has the following properties:

- It is very easy to compute and invert. Note that $\mathbf{T}^{-1} = \left(\begin{smallmatrix} \mathbf{I} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{smallmatrix}\right)$.
- It results in a matrix $\mathbf{A}$ that is distributed statistically close to uniformly at random (see [MP12] for the details).
- For the resulting functions $f_{\mathbf{A}}$ and $g_{\mathbf{A}}$, preimage sampling and inversion reduce to the corresponding tasks for $f_{\mathbf{G}}$ and $g_{\mathbf{G}}$.

The last property follows from the fact that

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}'\mathbf{T}\mathbf{x} = \mathbf{y};$$

$$f_{\mathbf{A}}^{-1}(\mathbf{y}) = \mathbf{T}^{-1} f_{\mathbf{A}'}^{-1}(\mathbf{y});$$

and using (1) we obtain the desired property. More complex but similar methods can be used to invert the function $g_{\mathbf{A}}$ instead.

We will be more interested in the ring version of all of this methods, as we will use a signature scheme its security is based in the R-SIS problem (described in Section 2.4.6). However, Micciancio and Peiker explain in [MP12] how all their constructions and algorithms are still valid and could extend to the ring setting.

2.4.5 R-LWE based commitment scheme

We use for our construction the commitment scheme proposed by Martínez and Morillo [MM19], which is a non-relaxed version of the one proposed by Benhamouda *et al.* [Ben+14]. Specifically, the opening algorithm in [Ben+14] accepts commitments from a larger space than those generated by the commitment algorithm; while in [MM19] the set of valid openings coincides with the set of openings obtained from the commitment algorithm.

This scheme will be the appropriate one because, apart from being a quantum resistant commitment scheme, it allow us to prove in zero-knowledge linear and multiplicative relations between the committed messages. In particular, this scheme allows us to prove (in zero-knowledge) polynomial relations (which are represented by arithmetic circuits).

The commitment scheme is defined by the following three algorithms:

- **C.Gen**: The generator algorithm takes as input the security parameter 1^λ and outputs a public key $\mathsf{pk} = (\mathbf{a}, \mathbf{b}) \in (R_q^k)^2$, where $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ and k are defined so that the difficulty of solving the R-LWE problem is related to 1^λ. $\mathsf{pk} \xleftarrow{\$} \mathsf{C.Gen}(1^\lambda)$.
- **C.Com**: The commitment algorithm takes as input the commitment key pk and a message $m \in R_q$, and outputs a commitment $\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e}$ and an opening $d = (m, r, \mathbf{e})$, where $r \xleftarrow{\$} R_q$ and $\mathbf{e} \xleftarrow{\$} \chi^k$ so that $\|\mathbf{e}\|_\infty < n$, with $n = 2^\kappa$, for some appropiate κ. Here, χ is a distribution over R. $(\mathbf{c}, d) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, m)$.
- **C.Ver**: The verification algorithm takes as input the commitment key pk, the message m, a commitment $\mathbf{c}$ and an opening d as inputs and accepts if $\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e}$ and $\|\mathbf{e}\|_\infty < n$, or rejects otherwise. $\{0, 1\} \longleftarrow \mathsf{C.Ver}(\mathsf{pk}, m, \mathbf{c}, d)$.

Theorem 2.38 (Proposition 1, [MM19])**.** *If $n \geq 256$, $\gamma \geq 3$ and $k \geq \frac{8\gamma+4}{2\gamma-5}$ then the above is a secure (perfectly binding and computationally hiding) commitment scheme under the assumption that* R-LWE *is hard.*

Proof Sketch. Correctness follows immediately.

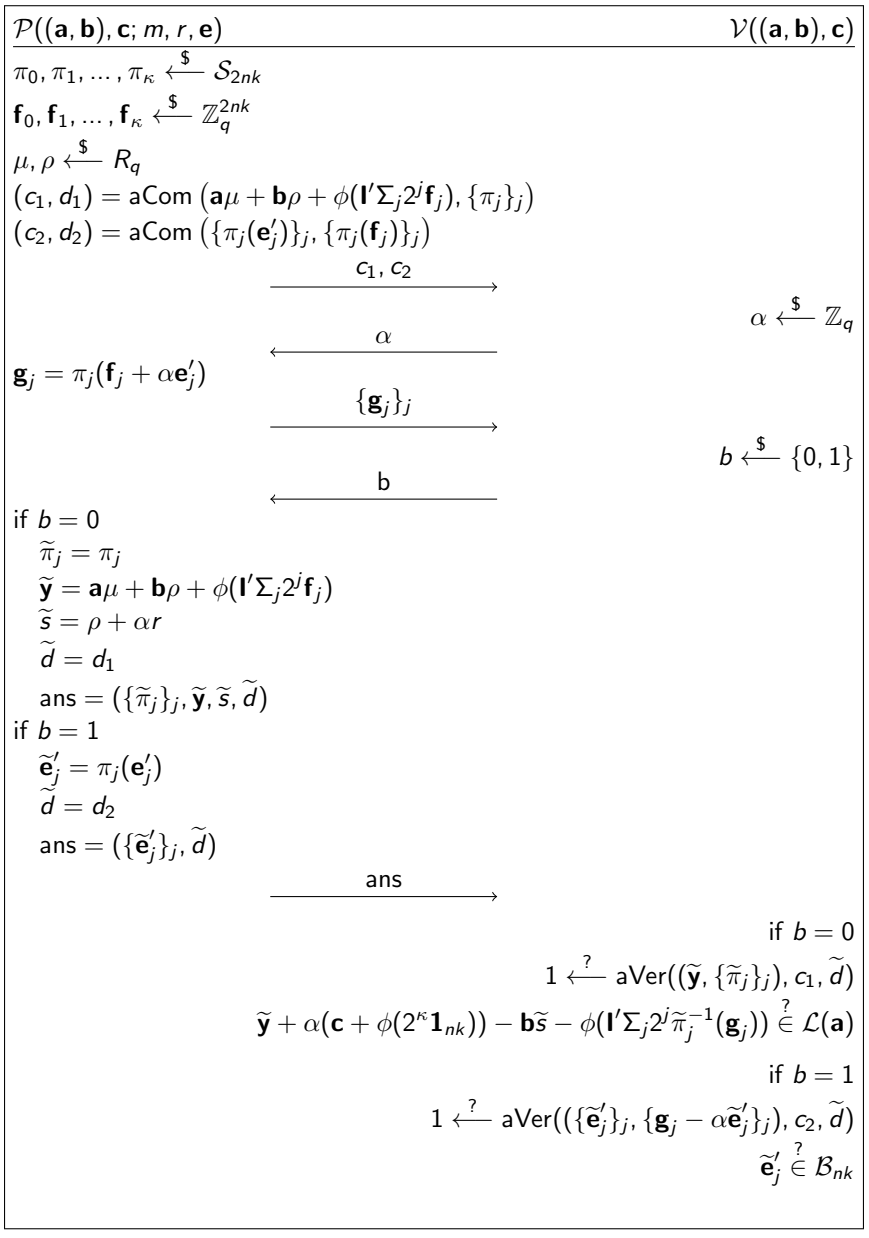
To prove the binding property, recall that two accepted openings to the same commitment would be:

$$\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e},$$
$$\mathbf{c} = \mathbf{a}m' + \mathbf{b}r' + \mathbf{e}';$$

which is the same as $\mathbf{a}(m - m') + \mathbf{b}(r - r') + (\mathbf{e} - \mathbf{e}') = \mathbf{0}$. Hence, if $q \equiv 3 \pmod 8$, with overwhelming probability over the choice of $\mathbf{a}$ and $\mathbf{b}$, there are no $\bar{m}, \bar{r} \in R_q$ and $\bar{\mathbf{e}} \in R_q^k$ small such that $\mathbf{a}\bar{m} + \mathbf{b}\bar{r} + \bar{\mathbf{e}} = \mathbf{0}$ holds and $\bar{m} \neq 0$ (the probability that this occurs is negligible). This implies that the commitment scheme is perfectly binding.

The above uses the fact that whenever $q \equiv 3 \pmod 8$, $x^n + 1$ splits into two irreducible polynomials $p_1(x), p_2(x)$ over $\mathbb{Z}_q[x]$.

Finally, since $\mathbf{b}r + \mathbf{e}$ are k R-LWE samples, indistinguishable from independent uniformly random polynomials under the R-LWE$_{n,q,\chi}$ assumption, any adversary able to break the hiding property would be able to solve the decisional R-LWE$_{n,q,\chi}$. ■



Protocol 6: ZKPoK of a valid opening.

This commitment scheme is specially interesting as allows someone knowing the openings to prove knowledge of a valid opening and linear and multiplicative relations between the messages of distinct commitments. Since it will be used later by our linkable attribute based signature scheme, we expose in Protocol 6 the ZKPoK for proving knowledge of a valid opening in the commitment

scheme[4].

The main objective of this protocol is to prove the knowledge of a valid opening for the commitment $\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e}$. They do that by also proving knowledge of some masking elements $\mu, \rho, \mathbf{f}$ and a of vector of polynomials $\mathbf{y}$ such that:

(a) $\pi_j(\mathbf{e}'_j) \in \mathcal{B}_{nk}$,

(b) $\mathbf{y} + \alpha\mathbf{c} = \mathbf{a}(\mu + \alpha m) + \mathbf{b}(\rho + \alpha r) + (\mathbf{f} + \alpha\mathbf{e})$,

where $\mathcal{B}_{nk} \subset \{0,1\}^{2nk}$ are vectors with the same number of 0's and 1's, and the $\mathbf{e}'_j$ are extensions to dimension $2nk$ of the binary representation of the vector of polynomials $\mathbf{e}$.

Since the important elements were committed in the first round (using an auxiliary commitment scheme (aCom, aVer)) before α was chosen, these two properties imply knowledge of a valid opening for the commitment. In Protocol 6 you can see the details on how they obtain this. We will refer to it as Π_{Open}.

Clearly, the commitment scheme is not homomorphic, since the sum of two commitments may not be a commitment to the sum as the errors may grow and overcome the upper bound for the norm. However it is possible to prove knowledge of openings to different commitments proving that the committed messages satisfy a given linear relation. Namely, we want to prove knowledge of valid openings for some commitments $\mathbf{c}_i = \mathbf{a}m_i + \mathbf{b}r_i + \mathbf{e}_i$ such that the messages that they commit satisfy the relation

$$m_3 = \lambda_1 m_1 + \lambda_2 m_2.$$

This can be done analogously as the previous case, just by running Protocol 5 three times imposing that the message masking elements hold the same linear relation. We will denote it as Π_{Add} and the scheme is exposed in Protocol 7.

A proof of knowledge of a multiplication relation is a little more tricky. We have three valid commitments $\mathbf{c}_i = \mathbf{a}m_i + \mathbf{b}r_i + \mathbf{e}_i$ such that the messages that they commit satisfy the relation

$$m_3 = m_1 \cdot m_2,$$

and we want to prove knowledge of valid openings for the commitments $\mathbf{c}_i$ satisfying this relation.

Similarly, we mask the messages m_1 and m_2 with random $\mu_1, \mu_2 \xleftarrow{\$} R_q$. When we multiply them, some crossed terms appear: $(m_1 + \mu_1)(m_2 + \mu_2) = m_3 + (m_1\mu_2 + m_2\mu_1) + \mu_1\mu_2$. If we want to get $m_3 = m_1 \cdot m_2$ we need to prove a similar equality involving two challenges $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$ chosen by the verifier. In fact, this protocol can also be seen as parallel executions of the first protocol, but taking into account the crossing terms. Similarly, we will expose it in Protocol 8 and denote it as Π_{Mult}.

[4] See [MM19] for particular details or notation on the protocols in this section.

$\mathcal{P}((\mathbf{a},\mathbf{b}),\mathbf{c}_i; m_i, r_i, \mathbf{e}_i)$	$\mathcal{V}((\mathbf{a},\mathbf{b}),\mathbf{c})$
$\pi_{i0}, \pi_{i1}, \dots, \pi_{i\kappa} \xleftarrow{\$} \mathcal{S}_{2nk}$	
$\mathbf{f}_{i0}, \mathbf{f}_{i1}, \dots, \mathbf{f}_{i\kappa} \xleftarrow{\$} \mathbb{Z}_q^{2nk}$	
$\mu_1, \mu_2, \rho_1, \rho_2, \rho_3 \xleftarrow{\$} R_q$	
$\mu_3 = \lambda_1\mu_1 + \lambda_2\mu_2$	
$(c_1, d_1) = \mathsf{aCom}\left(\{\mathbf{a}\mu_i + \mathbf{b}\rho_i + \phi(\mathbf{l}'\Sigma_j 2^j \mathbf{f}_{ij})\}_i, \{\pi_{ij}\}_{i,j}\right)$	
$(c_2, d_2) = \mathsf{aCom}\left(\{\pi_{ij}(\mathbf{e}'_{ij})\}_{i,j}, \{\pi_{ij}(\mathbf{f}_{ij})\}_{i,j}\right)$	
$\xrightarrow{c_1,\, c_2}$	
	$\alpha \xleftarrow{\$} \mathbb{Z}_q$
$\xleftarrow{\alpha}$	
$\mathbf{g}_{ij} = \pi_{ij}(\mathbf{f}_{ij} + \alpha\mathbf{e}'_{ij})$	
$\xrightarrow{\{\mathbf{g}_{ij}\}_{i,j}}$	
	$b \xleftarrow{\$} \{0,1\}$
$\xleftarrow{\mathsf{b}}$	
if $b = 0$	
$\widetilde{\pi}_{ij} = \pi_{ij}$	
$\widetilde{\mathbf{y}}_i = \mathbf{a}\mu_i + \mathbf{b}\rho_i + \phi(\mathbf{l}'\Sigma_j 2^j \mathbf{f}_{ij})$	
$\widetilde{s}_i = \rho_i + \alpha r_i$	
$\widetilde{d} = d_1$	
$\mathsf{ans} = (\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\mathbf{y}}_i\}_i, \{\widetilde{s}_i\}_i, \widetilde{d})$	
if $b = 1$	
$\widetilde{\mathbf{e}}'_{ij} = \pi_{ij}(\mathbf{e}'_{ij})$	
$\widetilde{d} = d_2$	
$\mathsf{ans} = (\{\widetilde{\mathbf{e}}'_{ij}\}_{i,j}, \widetilde{d})$	
$\xrightarrow{\mathsf{ans}}$	
	if $b = 0$
	$1 \xleftarrow{?} \mathsf{aVer}((\{\widetilde{\mathbf{y}}_i\}_i, \{\widetilde{\pi}_{ij}\}_{i,j}), c_1, \widetilde{d})$
	$\widetilde{\mathbf{y}}_i + \alpha(\mathbf{c}_i + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{b}\widetilde{s}_i - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_{ij}^{-1}(\mathbf{g}_{ij})) \overset{?}{\in} \mathcal{L}(\mathbf{a})$
	$\widetilde{t}_3 \overset{?}{=} \lambda_1\widetilde{t}_1 + \lambda_2\widetilde{t}_2$
	if $b = 1$
	$1 \xleftarrow{?} \mathsf{aVer}((\{\widetilde{\mathbf{e}}'_{ij}\}_{ij,}, \{\mathbf{g}_{ij} - \alpha\widetilde{\mathbf{e}}'_{ij}\}_{i,j}), c_2, \widetilde{d})$
	$\widetilde{\mathbf{e}}'_{ij} \overset{?}{\in} \mathcal{B}_{nk}$

Protocol 7: ZKPoK of a linear relation between the messages.

$\mathcal{P}((\mathbf{a},\mathbf{b}),\mathbf{c}_i; m_i, r_i, \mathbf{e}_i)$	$\mathcal{V}((\mathbf{a},\mathbf{b}),\mathbf{c})$
$\pi_{i0}, \pi_{i1}, \dots, \pi_{i\kappa} \xleftarrow{\$} \mathcal{S}_{2nk}$	
$\mathbf{f}_{i0}, \mathbf{f}_{i1}, \dots, \mathbf{f}_{i\kappa} \xleftarrow{\$} \mathbb{Z}_q^{2nk}$	
$\mu_i, \mu_+, \mu_\times, \rho_i \xleftarrow{\$} R_q$	
$m_\times = \mu_1\mu_1, \quad m_+ = \mu_1 m_2 + \mu_2 m_1$	
$(c_1, d_1) = \mathsf{aCom}\left(\{\mathbf{a}\mu_i + \mathbf{b}\rho_i + \phi(\mathbf{l}'\Sigma_j 2^j \mathbf{f}_{ij})\}_i, \{\pi_{ij}\}_{i,j}\right)$	
$(c_2, d_2) = \mathsf{aCom}\left(\mu_3, \mu_\times, \mu_+\right)$	
$(c_3, d_3) = \mathsf{aCom}\left(\{\pi_{ij}(\mathbf{e}'_{ij})\}_{i,j}, \{\pi_{ij}(\mathbf{f}_{ij})\}_{i,j},\right)$	
$(c_4, d_4) = \mathsf{aCom}\left(\mu_\times + m_+, \mu_+ + m_+\right)$	
$\xrightarrow{c_1, c_2, c_3, c_4}$	
	$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$
$\xleftarrow{\alpha}$	
$\delta_1 = \alpha, \delta_2 = \alpha, \delta_3 = \beta$	
$\mathbf{g}_{ij} = \pi_{ij}(\mathbf{f}_{ij} + \delta_i \mathbf{e}'_{ij})$	
$(c_5, d_5) = \mathsf{aCom}\left((\beta\mu_\times) + \alpha(\beta\mu_+) + \alpha^2(\mu_3)\right)$	
$\xrightarrow{\{\mathbf{g}_{ij}\}_{i,j}, c_5}$	
	$b \xleftarrow{\$} \{0,1\}$
$\xleftarrow{\mathsf{b}}$	
if $b = 0$	
$\widetilde{\pi}_{ij} = \pi_{ij}$	
$\widetilde{\mathbf{y}}_i = \mathbf{a}\mu_i + \mathbf{b}\rho_i + \phi(\mathbf{l}'\Sigma_j 2^j \mathbf{f}_{ij})$	
$\widetilde{t}_\times = \mu_\times + m_\times,\ \widetilde{t}_+ = \mu_+ + m_+,\ \widetilde{s}_i = \rho_i + \delta_i r_i$	
$\widetilde{d}_1 = d_1,\ \widetilde{d}_4 = d_4,\ \widetilde{d}_5 = d_5$	
$\mathsf{ans} = (\{\widetilde{\pi}_{ij}\}_{i,j}, \{\widetilde{\mathbf{y}}_i\}_i, \widetilde{t}_\times, \widetilde{t}_+, \{\widetilde{s}_i\}_i, \widetilde{d}_1, \widetilde{d}_4, \widetilde{d}_5)$	
if $b = 1$	
$\widetilde{\mathbf{e}}'_{ij} = \pi_{ij}(\mathbf{e}'_{ij})$	
$\widetilde{\mu}_3 = \mu_3,\ \widetilde{\mu}_\times = \mu_\times,\ \widetilde{\mu}_+ = \mu_+$	
$\widetilde{d}_2 = d_2,\ \widetilde{d}_3 = d_3,\ \widetilde{d}_5 = d_5$	
$\mathsf{ans} = (\{\widetilde{\mathbf{e}}'_{ij}\}_{i,j}, \widetilde{\mu}_3, \widetilde{\mu}_\times, \widetilde{\mu}_+, \widetilde{d}_2, \widetilde{d}_3, \widetilde{d}_5)$	
$\xrightarrow{\mathsf{ans}}$	
	if $b = 0$
	$1 \xleftarrow{?} \mathsf{aVer}((\{\widetilde{\mathbf{y}}_i\}_i, \{\widetilde{\pi}_{ij}\}_{i,j}), c_1, \widetilde{d}_1)$
	$1 \xleftarrow{?} \mathsf{aVer}((\widetilde{t}_\times, \widetilde{t}_+), c_4, \widetilde{d}_4)$
	$1 \xleftarrow{?} \mathsf{aVer}(\beta\widetilde{t}_\times + \alpha\beta\widetilde{t}_+ + \alpha^2\widetilde{t}_3 - \beta\widetilde{t}_1\widetilde{t}_2, c_5, \widetilde{d}_5)$
	$\widetilde{\mathbf{y}}_i + \delta_i(\mathbf{c}_i + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{b}\widetilde{s}_i - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_{ij}^{-1}(\mathbf{g}_{ij})) \stackrel{?}{\in} \mathcal{L}(\mathbf{a})$
	if $b = 1$
	$1 \xleftarrow{?} \mathsf{aVer}((\widetilde{\mu}_3, \widetilde{\mu}_\times, \widetilde{\mu}_+), c_2, \widetilde{d}_2)$
	$1 \xleftarrow{?} \mathsf{aVer}((\{\widetilde{\mathbf{e}}'_{ij}\}_{ij,}, \{\mathbf{g}_{ij} - \alpha\widetilde{\mathbf{e}}'_{ij}\}_{i,j}), c_3, \widetilde{d}_3)$
	$1 \xleftarrow{?} \mathsf{aVer}((\beta\widetilde{\mu}_\times) + \alpha(\beta\widetilde{\mu}_+) + \alpha^2(\widetilde{\mu}_3), c_5, \widetilde{d}_5)$
	$\widetilde{\mathbf{e}}'_{ij} \stackrel{?}{\in} \mathcal{B}_{nk}$

Protocol 8: ZKPoK of a multiplicative relation between the messages.

2.4.6 R-SIS based signature scheme

In this section we describe the variant of Boyen's signature [Boy10] by Micciancio and Peikert [MP12], adapted to have security based on hardness assumptions on ideal lattices. We begin by recalling some definitions and properties concerning the discrete Normal distribution.

Definition 2.39. The *continuous Normal distribution* over $\mathbb{R}^n$ centered at $\mathbf{v}$ with standard deviation σ is defined by the function

$$\rho^n_{\mathbf{v},\sigma}(\mathbf{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\frac{-\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2}}.$$

When $\mathbf{v} = 0$, we will just write $\rho^n_\sigma(\mathbf{x})$.

Definition 2.40. The *discrete Normal distribution* over $\mathbb{Z}^n$ centered at some $\mathbf{v} \in \mathbb{R}^n$ with standard deviation σ is defined as

$$D^n_{\mathbf{v},\sigma}(\mathbf{x}) = \frac{\rho^n_{\mathbf{v},\sigma}(\mathbf{x})}{\rho^n_\sigma(\mathbb{Z}^n)}.$$

We write $D^n_\sigma(\mathbf{x})$ whenever $\mathbf{v} = 0$. In the above definition, the quantity $\rho^n_\sigma(\mathbb{Z}^n) = \sum_{\mathbf{z}\in\mathbb{Z}^n} \rho^n_\sigma(\mathbf{z})$ is just a scaling quantity needed to make the function into a probability distribution. Also note that $\rho^n_{\mathbf{v},\sigma}(\mathbb{Z}^n) = \rho^n_\sigma(\mathbb{Z}^n)$ for all $\mathbf{v} \in \mathbb{Z}^n$, thus the scaling quantity is the same for all $\mathbf{v}$.

A sample z from a discrete Normal distribution over R_q centered at $\mathbf{v}$ with standard deviation σ, denoted $D_{R_q,\mathbf{v},\sigma}$, is generated as a sample from a discrete Normal distribution over $\mathbb{Z}^n$ and then map it into R_q using the previous mentioned embedding of coordinates into coefficients of the polynomial. Similarly, we omit the $\mathbf{0}$ and just write $\mathbf{y} = (y_1, \ldots, y_k) \xleftarrow{\$} D^k_{R_q,\sigma}$ to mean that a vector $\mathbf{y}$ is generated according to D_σ over $\mathbb{Z}^{kn}$ and then gets interpreted as k polynomials y_i. With an abuse of notation, we denote by $D^\perp_{\mathbf{A},\mathbf{u},\sigma}$ the distribution of the vectors $\mathbf{v} \in R^m_q$ such that $\mathbf{v} \xleftarrow{\$} D^m_{R_q,\sigma}$ conditioned on $\mathbf{Av} \equiv \mathbf{u} \pmod q$, where $\mathbf{A}$ is an appropriate matrix with m columns.

The following lemma will provide useful properties of this distribution.

Lemma 2.41 (Lemma 2.1, [Bos+20])**.** *Let $n > 0$. The following bounds hold:*

1. $\Pr_{\mathbf{z}\xleftarrow{\$}D^n_\sigma}(\|\mathbf{z}\| > 1.05\sigma\sqrt{n}) < (0.998)^n$,
2. $\Pr_{\mathbf{z}\xleftarrow{\$}D^n_\sigma}(\|\mathbf{z}\|_\infty > 8\sigma) < n2^{-47}$.

Recall that existential unforgeability against adaptive chosen-message attacks requires that the adversary should not be able to forge a signature on some message of her choice, even if she has access to a signing oracle.

Lemma 2.42 (Trapdoor generation, adapted from [Bos+20])**.** *Let $m = 2, k = \lceil \log q \rceil, \bar{m} = m + k$. There exists an algorithm* GenTrap *that, on input an invertible element $h \in R_q$, outputs a vector $\bar{\mathbf{a}} \in R^{\bar{m}}_q$ and a trapdoor $\mathbf{R} \in R^{m\times k}_q$ such that:*

- $\bar{\mathbf{a}} = [\mathbf{a} \mid \mathbf{aR} + h\mathbf{g}]$*, where* $\mathbf{g}$ *is the gadget vector* $\mathbf{g} = [1, 2, 2^2, \ldots, 2^{k-1}] \in R^k_q$*, and* $\mathbf{a} = [a \mid 1] \in R^m_q$*, $a \xleftarrow{\$} R_q$.*
- $\mathbf{R}$ *is distributed as a Normal $D^{m\times k}_{R,s}$ for some $s = \alpha q$, where $\alpha > 0$ is a* R-LWE *error term, $\alpha q > \omega(\sqrt{\log n})$.*

- $\bar{\mathbf{a}}$ *is computationally pseudorandom (ignoring the component set to 1) under* Decision-R-LWE*, where we set* $\chi = D_{R,s}$.

Genise and Micciancio [GM18] give an optimal sampling algorithm for the previous trapdoor.

Lemma 2.43 (Gaussian sampler, adapted from [Bos+20])**.** *Let* $R_q, m, k, \bar{m}$ *be as in Lemma 2.42,* $\mathbf{g}$ *be the gadget vector* $\mathbf{g} = [1, 2, 2^2, \dots, 2^{k-1}]$, $\mathbf{a} \in R_q^m$ *and* $\mathbf{R} \in R_q^{m \times k}$ *be the output of* GenTrap*, and* $\mathbf{v}$ *a vector in* R_q^d *for some* $d \geq 0$*. Then, there is an algorithm* SampleD *that can efficiently sample from the distribution* $D^{\perp}_{[\bar{\mathbf{a}}|\mathbf{aR}+\mathbf{g}|\mathbf{v}],\mathbf{u},s}$ *for any* $s = O(\sqrt{n \log q}) \cdot \omega(\sqrt{\log n})$ *and for any* $u \in R_q$.

Before moving on, it is worth to recall here some other kind of notions of security for a digital signature scheme than those defined in Section 2.2.3. It is also modelled using the following attack game between an adversary $\mathcal{A}$ and a challenger:

1. The challenger runs $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$ and sends vk to $\mathcal{A}$.
2. $\mathcal{A}$ queries the challenger several times. For $i = 1, 2, \dots$ the i-th *signing query* is a message $\mathbf{x}_i \in \{0,1\}^\ell$. Given $\mathbf{x}_i$, the challenger computes $\sigma_i \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}, m_i)$ and gives σ_i to the adversary $\mathcal{A}$.
3. Eventually $\mathcal{A}$ outputs a candidate forgery pair $(\mathbf{x}, \sigma)$. The adversary wins if $1 \longleftarrow \mathsf{S.Verify}(\mathsf{vk}, \mathbf{x}, \sigma)$ and σ is not one of the signatures $\mathcal{A}$ has received in the signature queries.

Note that in this case $\mathcal{A}$ is also allowed to output a forged signature for some of the queried messages $\mathbf{x}_i$ in the security game. We define the *advantage of an adversary* $\mathcal{A}$ as the probability that $\mathcal{A}$ wins the above game.

Definition 2.44 (su-acma)**.** We say that a digital signature scheme is *strongly unforgeable under an adaptive chosen message attack*, su-acma for short, if the advantage of this game is negligible for all PPT adversaries.

They are similar notions of security, but it is clear that su-acma is stronger than eu-acma since the adversary's output space is slightly larger in the former.

A different notion would be *static chosen message attack* (scma), where the adversary submits all messages $\mathbf{x}_1, \dots, \mathbf{x}_n$ before seeing the verification key vk and the corresponding signatures. This notion can also be existential or strongly unforgeable.

Definition 2.45 (scma)**.** We say that a digital signature scheme is scma if the advantage of the above game is negligible for all PPT adversaries.

Micciancio and Peikert proved their variant of [Boy10] to be strongly unforgeable against static chosen-message attack, su-scma for short, under SIS with a tighter reduction (Theorem 6.1 in [MP12]); and then made it strongly unforgeable against adaptive chosen-message attacks, su-acma for short, using a generic transformation through chameleon hash functions (Theorem 1 in [ST01]).

Definition 2.46 (Ideal Boyen's signature as in [Bos+20])**.** Let $k = \lceil \log q \rceil$, $m = 2$ and $\bar{m} = m + k$ be the length of the public matrices, and ℓ be the length of the message. Also, let $s_{\mathsf{sk}} = \sqrt{\log(n^2)} + 1$ and $s_\sigma = \sqrt{n \log n} \cdot \sqrt{\log(n^2)}$ be the standard deviations of the distributions of the signing key and of the signature respectively.

- **S.KeyGen**: Run the algorithm GenTrap to get a vector $[\mathbf{a} \mid \mathbf{b}] = [\mathbf{a} \mid \mathbf{aR} + \mathbf{g}] \in R_q^{\bar{m}}$ and a trapdoor $\mathbf{R}$. The public key is composed by $\ell + 1$ random vectors $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_\ell \xleftarrow{\$} R_q^k$, a random $u \xleftarrow{\$} R_q$ and the vector $[\mathbf{a} \mid \mathbf{b}]$ and the secret key is $\mathbf{R}$. $(\mathsf{vk}, \mathsf{sk}) = ((\mathbf{a}, \mathbf{b}, \mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_\ell, u), \mathbf{R}) \xleftarrow{\$} \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$.
- **S.Sign**: To sign a message $\mathbf{x} = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$, the signer constructs a message-dependent public vector $\mathbf{a_x} = [\mathbf{a} \mid \mathbf{b} \mid \mathbf{a}_0 + \sum_{i=1}^{\ell} (-1)^{x_i} \mathbf{a}_i]$ and then it samples a short vector $\mathbf{s} \in R_q^{\bar{m}+k}$ running the algorithm SampleD on input $(\mathbf{a_x}, u, \mathbf{R})$. $\sigma = \mathbf{s} \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}, \mathbf{x})$.
- **S.Verify**: It first checks that $\mathbf{x} \in \{0,1\}^\ell$. The verification algorithm checks that the vector $\mathbf{s}$ has small norm, i.e., $\|\mathbf{s}\|_\infty \leq 8 s_\sigma$. Then, it constructs $\mathbf{a_x} = [\mathbf{a} \mid \mathbf{b} \mid \mathbf{a}_0 + \sum_{i=1}^{\ell} (-1)^{x_i} \mathbf{a}_i]$ and checks that $\mathbf{s}$ satisfies the verification equation, i.e., $\mathbf{a_x s} \equiv u \pmod{q}$.

Correctness follows from Lemmas 2.41, 2.42, 2.43. In [Bos+20] it is proven that this scheme is eu-acma under R-SIS. They do it by proving that if there exists a PPT adversary $\mathcal{A}$ that can break the signature scheme, we can construct an algorithm $\mathcal{B}$ that can solve R-SIS or R-LWE exploiting $\mathcal{A}$. Using a more technical argument, but following the same logical steps, it could also be constructed a proof to show that this signature is strongly unforgeable.

Theorem 2.47 (Theorem 2.6, [Bos+20])**.** *If there exists a* PPT *adversary* $\mathcal{A}$ *that can break the* eu-acma *security of the above scheme, then there exists a* PPT *algorithm* $\mathcal{B}$ *that can solve either* R-SIS *or* R-LWE .

3. Voting process

Electronic voting is desired to be the modern way of conducting our democracy. One important property behind it is that it should allow voters to vote from their personal electronic devices, which directly will solve important problems arising while using traditional systems such as accessibility and usability. However, some concerns may arise about its security even if machines are used in a polling station to perform the voting process.

A typical approach for an electronic voting scheme is to combine an encryption scheme with a digital signature scheme. The former is used for providing secrecy of the votes transmitted between the voter and the bulletin board, in front of observers. The latter is used to ensure the integrity of the transmitted votes, as well as providing assurance of the origin of such votes. In other words, we desire that a malicious entity is not able to modify or forge a vote without being detected by the system.

In this approach, voters encrypt their votes before casting them, with the idea that only the electoral board is able to perform the necessary computations to decrypt the final tally. After encryption and prior to casting, voters also digitally sign their votes, in order to prove later on to the election authorities that they have been cast by eligible votes. This approach is similar to postal voting, where the voter signs the outer envelope containing her vote. Also similarly, outer envelopes are removed after verification of the signature so that single votes can not be related to a voter's identity.

Digital signature schemes allow identification of the voter who cast the vote, and therefore can also be used in order to discern whether a voter tries to cast a vote more than once. However, decrypted votes could still be related to the voter identities by checking the digital signature of the encrypted votes stored in the bulletin board in the same order.

Therefore, encrypting and signing is not enough, and variants of those cryptographic primitives should be used instead. We need a way to maintain a connection between a voter and her vote, while at the same time this connection has to reveal nothing from the voter more than the fact that she has the right to vote. Moreover, we want to provide votes with the property of being distinguishable among those cast by a different voter.

3.1 Digital signatures literature

There are several variants of digital signature schemes that would partially solve the aforementioned problem. One of those is *ring signatures* [RST01]. In a ring signature, a signature is produced anonymously among a set of possible signers. Additionally, ring signatures have no group managers, no setup procedures, no revocation procedures and no coordination: any user can choose any set of possible signers that includes himself, and sign any message by using his secret key and the other's public keys, without getting their approval or assistance. This is inappropriate for e-voting protocols because the use of this signature could led to vote selling. If a voter sets up a ring (i.e., a group of possible singers) and a signature is produced, *the signer-ambiguous* property of a ring signature ensures that the signature have been emitted by any member in the ring with the same probability. Hence, a re-voting method can not be directly applied and then by simply showing to a third party who you voted for before casting it is enough to sell it.

Another similar option is *group signatures* [CH91]. Unlike ring signatures, group signatures only allow a member of the set of possible signers to sign on behalf of the group. Moreover, a basic requirement of such schemes is unlinkability: given two votes and their signatures, we can not tell

if the signatures were from the same signer or not. This is again infeasible for our purposes.

Mesh signatures [Boy07] also look like an attractive candidate to solve our problem. They present themselves as a direct generalization of ring signatures: instead of requiring that each and everyone generate and publish their public key in a ring scheme, mesh signatures just need one trustworthy certificate authority to publish their keys in the mesh scheme. However, this signatures satisfy an analogous version of the signer-ambiguous property from ring signatures. Hence, our need seems to lie beyond the anonymity idea in which these three signatures are based.

A more appropriate primitive are *attribute-based signatures* (ABS) [MPR11]. An ABS is a versatile tool allowing a signer to anonymously authenticate a message $\mathbf{m}$ with respect to a public signing policy C only if the signer has a signing key associated to an *attribute*[5] $\mathbf{x} \in \{0,1\}^\ell$, where $\ell \in \mathbb{N}$, that satisfies C (i.e., $C(\mathbf{x}) = 1$). Here, the policy C will be represented as an arithmetic circuit and distinct signing polices will be used in different elections. An attribute-based signature scheme reveals no information on the signer's identity or the attribute other than the fact that the signature is valid.

It has some interesting applications in electronic voting, e.g., it guarantees that a vote has been submitted by an elegible voter without revealing her identity. The main problem of a standard ABS scheme for electronic voting is the non-existence of an algorithm that allow us to distinguish if two signatures have been emitted by the same person, without revealing her attribute. This is a property that we need to incorporate to an ABS in order to use it in an e-voting, as one of such a schemes needs to allow voters to change their votes (in this case, by being able to vote multiple times) prior the ending of the electoral process. In particular, this property will avoid coercion in the voting period.

Note that an ABS does not directly show the attributes to the intended recipient and an attribute collusion may occur (i.e., two different voters share the same set of attributes). Hence, it would be again impossible to discern if two signatures have been cast by the same voter or not. This is the second concern we have to deal with in order to make an ABS practical in our context.

3.2 Linkable attribute-based signature scheme

To solve both problems mentioned in the previous section, we will formally define here a *linkable attribute-based signature scheme* and provide its security notions. We will provide a practical implementation in Section 4. Before that, let us recall some definitions regarding arithmetic circuits.

Definition 3.1. An *arithmetic circuit C over a ring R* with ℓ input wires and one output wire is a directed acyclic graph. It contains ℓ wires that come from ℓ values of the ring R respectively; called the *input wires* and one wire that does not go to any node, called the *output wire*. All other wires are called *internal wires*

The *size* of an arithmetic circuit is the number of gates in it and it is typically denoted as $|C|$. The *depth* of an arithmetic circuit C, denoted depth(C), is the length of the longest directed path from an input wire to the output wire. For instance, In Figure 9 we have a circuit with size $|C| = 6$ and depth(C) $= 4$.

The nodes are called *gates* and are labelled by either $+$ (addition) or $\times$ (product). Moreover, the input wires are indexed by $1, \dots, \ell$, the the internal wires are indexed by $\ell + 1, \dots, \ell + |C| - 1$ and the output wire has index $\ell + |C|$. Here, assume that each gate takes only two incoming wires with multiple fan-out wires, where all the fan-out wires are indexed with the same index.

[5] In e-voting context, we typically define attributes as a set of general characteristics about the voter's identity. Some examples might be location, age range, gender... The elected set of attributes will vary over distinct elections.

We define $\mathcal{C}_\ell$ to be the collection of arithmetic circuits defined over a ring R, each having ℓ input wires. We also define the collection $\mathcal{C} = \{\mathcal{C}_\ell\}_{\ell \in \mathbb{N}}$.

Definition 3.2 (Bounded collection)**.** We say that the a collection of arithmetic circuits $\mathcal{C}_\ell$, for a fixed $\ell \in \mathbb{N}$, is *bounded* if for every circuit $C \in \mathcal{C}_\ell$ we have that depth$(C) \leq d$, with $d \in \mathbb{N}$.

We also specify the *topology of an arithmetic circuit* by a function topo : $\{\ell+1, \ldots, \ell+|C|\} \to \{+, \times\} \times \{1, \ldots, \ell+|C|-1\} \times \{1, \ldots, \ell+|C|-1\}$. Intuitively, this function maps a non-input wire to its first and second incoming wires in which these three wires are connected by a gate, whose label ($+$ or $\times$) is also showed in the topology. For $(\star, i_1, i_2) \leftarrow \text{topo}(i)$, we require that $i_1, i_2 < i$ where $\star \in \{+, \times\}$. For instance, in Figure 9 we have that topo$(7) = (\times, 1, 2)$, topo$(11) = (\times, 7, 9)$ and topo$(12) = (+, 10, 11)$.

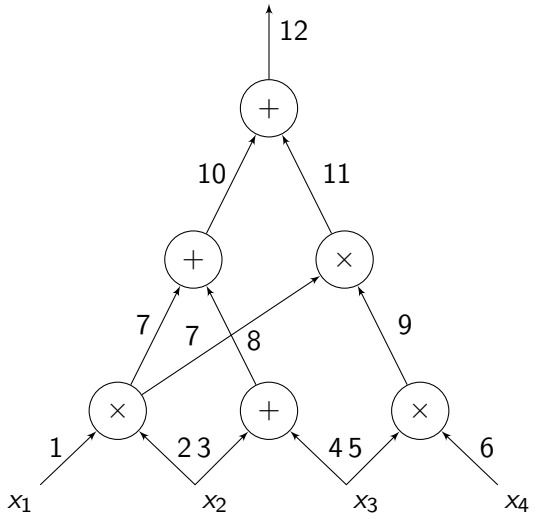


Figure 9: An arithmetic circuit with 6 input wires and 6 gates.

In what follows we formalize the notion of linkable attribute-based signature scheme. In this definition, instead of directly taking an attribute $\bar{\mathbf{x}} \in \{0,1\}^{\bar{\ell}}$ to generate a signing key, we will use an extension of it. The reason behind this extension is forcing two distinct signers to use different attributes to produce signatures, so that they can be distinguished by our algorithms. Specifically, we will take $\mathbf{x} = [\bar{\mathbf{x}} \mid \bar{\mathbf{y}}] \in \{0,1\}^{\ell}$ with $\bar{\mathbf{x}} \in \{0,1\}^{\bar{\ell}}$, $\bar{\mathbf{y}} \in \{0,1\}^{\bar{m}}$ and $\ell = \bar{\ell} + \bar{m}$. As noticed before, an attribute collusion may occur and hence we impose that the extension part $\bar{\mathbf{y}}$ is distinct for every pair of distinct signers, e.g., we could set $\bar{\mathbf{y}}$ to be a random number between 0 and $2^{\bar{m}} - 1$. However, note that in electronic voting the extension part $\bar{\mathbf{y}}$ is meaningless when we have to evaluate a public signing policy C. Hence, in this context, the relevant part of this evaluation is where the original attribute $\bar{\mathbf{x}}$ is involved (i.e., the evaluation $C(\mathbf{x})$ will output 1 or 0 depending merely on $\bar{x}$).

In the rest of this thesis, whenever it is intended that a circuit is satisfied by an attribute and it is not the case, the algorithm will implicitly output $\perp$ (i.e., the algorithm stops and outputs "fail"). Also, as we just deal with extension attributes from now on, we will refer to them simply as attributes.

Definition 3.3 (Linkable attribute-based signature scheme (LABS))**.** A *linkable attribute-based signature scheme* supporting the class of arithmetic circuits $\mathcal{C} = \{\mathcal{C}_\ell\}_{\ell \in \mathbb{N}}$ and message space $\{0,1\}^*$ is defined by the following five algorithms:

- **LABS.Setup**: The setup algorithm takes as input the security parameter 1^λ and the input length 1^ℓ of the circuits in $\mathcal{C}_\ell$, and outputs the master public key mpk and the master secret key msk. $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell)$.
- **LABS.KeyGen**: The signing key generation algorithm takes as input the master public key mpk, the master secret key msk and an attribute $\mathbf{x} \in \{0,1\}^\ell$, and outputs a signing key $\mathsf{sk}_\mathbf{x}$. $\mathsf{sk}_\mathbf{x} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$.
- **LABS.Sign**: The signing algorithm takes as input the master public key mpk, a secret key $\mathsf{sk}_\mathbf{x}$ associated with an attribute $\mathbf{x}$, a circuit $C \in \mathcal{C}_\ell$, and a message $\mathbf{m} \in \{0,1\}^*$. It checks that $C(\mathbf{x}) = 1$ and then outputs a signature Σ. $\Sigma \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_\mathbf{x}, C, \mathbf{m})$.
- **LABS.Verify**: The verification algorithm takes as input the master public key mpk, a message $\mathbf{m}$, a circuit C and a signature Σ, and outputs 0 (invalid) or 1 (valid). $\{0,1\} \longleftarrow \mathsf{LABS.Verify}(\mathsf{mpk}, \mathbf{m}, C, \Sigma)$.
- **LABS.Link**: The linking algorithm takes as input two messages $\mathbf{m}_1, \mathbf{m}_2$ as well as two signatures Σ_1, Σ_2. It checks that signatures Σ_1, Σ_2 of messages $\mathbf{m}_1, \mathbf{m}_2$ are valid respectively, (outputting $\perp$ otherwise), and outputs 1 or 0 (i.e., there is a link between the signatures or not). $\{0,1\} \longleftarrow \mathsf{LABS.Link}(\mathbf{m}_1, \mathbf{m}_2, \Sigma_1, \Sigma_2)$.

We require it to satisfy the following properties:

- **Correctness**: Let $\lambda, \ell \in \mathbb{N}$, $\mathbf{x} \in \{0,1\}^\ell$ and $C \in \mathcal{C}_\ell$. We say that the scheme has the *correctness* property if we have that $C(\mathbf{x}) = 1$ and:

$$\Pr[(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell); \mathsf{sk}_\mathbf{x} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}); \Sigma \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_\mathbf{x}, C, \mathbf{m}) : \mathsf{LABS.Verify}(\mathsf{mpk}, \mathbf{m}, C, \Sigma) = 0] \in \mathsf{negl}(\lambda)$$

- **Linkability**: Let $\lambda, \ell, t \in \mathbb{N}$ with $t \geq 2$. We formalize this property by the following game between a challenger and an adversary $\mathcal{A}$:

  1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell)$ and gives mpk to $\mathcal{A}$.
  2. $\mathcal{A}$ outputs t attributes $\mathbf{x}_1, \ldots, \mathbf{x}_t$ to the challenger. The challenger returns to $\mathcal{A}$ the secret keys $\mathsf{sk}_{\mathbf{x}_i} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}_i)$, for $i = 1, \ldots, t$.
  3. Finally, $\mathcal{A}$ outputs $t+1$ signatures $(\mathbf{m}_1, \Sigma_1), \ldots, (\mathbf{m}_{t+1}, \Sigma_{t+1})$.

  The adversary $\mathcal{A}$ wins the game if the following two conditions hold for some circuit $C \in \mathcal{C}_\ell$:

  (i) $\mathsf{LABS.Verify}(\mathsf{mpk}, \mathbf{m}_i, C, \Sigma_i) = 1$ for all $i \in [t+1]$.
  (ii) $\mathsf{LABS.Link}(\mathbf{m}_i, \mathbf{m}_j, \Sigma_i, \Sigma_j) = 0$ for every $i, j \in [t+1]$.

  The advantage of $\mathcal{A}$ is defined as the probability of $\mathcal{A}$ winning the above game. We say that the scheme has the *computational linkability* property if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.

- **Exculpability**: Let $\lambda, \ell, t \in \mathbb{N}$ with $t \geq 2$. We formalize this property by the following game between a challenger and an adversary $\mathcal{A}$:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell)$ and gives mpk to $\mathcal{A}$.
2. $\mathcal{A}$ outputs a message $\mathbf{m}$ an t distinct attributes $\mathbf{x}_1, \ldots, \mathbf{x}_t$ along with a circuit $C \in \mathcal{C}_\ell$ to the challenger. The challenger first checks that $\mathbf{x}_i \neq \mathbf{x}_j$ with $i \neq j$, for every $i, j \in [t]$ (outputting $\perp$ otherwise). Then, it runs $\mathsf{sk}_{\mathbf{x}_i} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}_i)$ for $i = 1, \ldots, t$, and returns to $\mathcal{A}$ the secret keys $\mathsf{sk}_{\mathbf{x}_1}, \ldots, \mathsf{sk}_{\mathbf{x}_{t-1}}$ along with the signature $\Sigma \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_t}, C, \mathbf{m})$.
3. At the end, $\mathcal{A}$ outputs a signature $(\mathbf{m}^*, \Sigma^*)$.

The adversary $\mathcal{A}$ wins the game if the following two conditions hold:

(i) $\mathsf{LABS.Verify}(\mathsf{mpk}, \mathbf{m}^*, C, \Sigma^*) = 1$.

(ii) $\mathsf{LABS.Link}(\mathbf{m}, \mathbf{m}^*, \Sigma, \Sigma^*) = 1$.

The advantage of $\mathcal{A}$ is defined as the probability of $\mathcal{A}$ winning the above game. We say that the scheme has the *computational exculpability* property if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.

On the one side, the linkability property is reflecting the fact that if $t+1$ signatures are issued, then at most $t+1$ signers have used the scheme to produce them. Specifically, if the same attribute is used to create a signature of the same message or two different messages, then it will be noticed by the LABS scheme with non-negligible probability. This property is the one that will allow us to implement the re-voting policy in our electronic voting system by simply replacing the "old" existing vote by the new linked one.

On the other side, the exculpability property is reflecting the other way around: if two different attributes are used to create signatures they will never be linked. Notice that we also included the situation where two different signers contribute each other to generate a third signature. Again, the signatures created independently by every signer can not be linked to this new "cooperated" one. This is a necessary condition in order to make the voting protocol consistent, since otherwise it could happen that someone maliciously wants to re-vote and in the process another's vote is replaced.

Note that in both the linkability and exculpability properties full power is given to the adversary. We are giving him total freedom and every tool that is required to forge signatures. This is an strong notion of security, since even with all this resources she should not to be able to link when signatures are emitted by different signers and not to link when signatures are emitted by the same signer.

Additionally, as in [EKK18], we define two security notions, privacy and unforgeability, for our linkable attribute-based signature scheme. However, we decided to differentiate between interior and exterior privacy as it is important in our context. Roughly speaking, the former intends to not leak any information on the signer's attribute using the same arithmetic circuit to produce signatures by distinct signers, and the latter intends the same but using different arithmetic circuits to produce signatures by the same signer.

We achieve interior privacy by testing if the distribution of $\mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_0}, C, \mathbf{m})$ is computationally indistinguishable to the distribution of $\mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C, \mathbf{m})$ for attributes $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^\ell$ such that $C(\mathbf{x}_0) = C(\mathbf{x}_1) = 1$.

Definition 3.4 (Interior privacy)**.** The security notion of *interior privacy* for a linkable attribute-based signature scheme is defined by the following game between a challenger and an adversary $\mathcal{A}$:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell)$ and gives mpk to $\mathcal{A}$.

2. $\mathcal{A}$ outputs a message $\mathbf{m} \in \{0,1\}^*$, two attribute $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^\ell$ and a circuit $C \in \mathcal{C}_\ell$ to the challenger. The challenger first checks that $C(\mathbf{x}_0) = C(\mathbf{x}_1) = 1$ and then runs $\mathsf{sk}_{\mathbf{x}_0} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}_0)$ and $\mathsf{sk}_{\mathbf{x}_1} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}_1)$. Then, it picks a random bit $\{0,1\} \xleftarrow{\$} b$ and returns to $\mathcal{A}$ the signature $\Sigma^* \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_b}, C, \mathbf{m})$.
3. $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ for b.

The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - 1/2|$ and we say that the attribute-based signature scheme is *computationally interiorly private* if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.

Similarly, we achieve exterior privacy by testing if the distribution of two signatures produced with the same secret key but distinct circuits is computationally indistinguishable to the distribution of two signatures produced with both distinct secret keys and circuits.

Definition 3.5 (Exterior privacy)**.** The security notion of *exterior privacy* for a linkable attribute-based signature scheme is defined by the following game between a challenger and an adversary $\mathcal{A}$:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell)$ and gives mpk to $\mathcal{A}$.
2. $\mathcal{A}$ outputs a message $\mathbf{m} \in \{0,1\}^*$, two attributes $\mathbf{x}_1, \mathbf{x}_2 \in \{0,1\}^\ell$ and two circuits $C_1, C_2 \in \mathcal{C}_\ell$ to the challenger. The challenger first checks that $C_1(\mathbf{x}_1) = C_2(\mathbf{x}_1) = C_2(\mathbf{x}_2) = 1$ and then runs $\mathsf{sk}_{\mathbf{x}_i} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x}_i)$ for $i = 1, 2$. Then, it picks a random bit $\{0,1\} \xleftarrow{\$} b$ and proceeds as follows:
   - If $b = 0$, the challenger returns to $\mathcal{A}$ signatures $\Sigma_1 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C_1, \mathbf{m})$ and $\Sigma_2 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C_2, \mathbf{m})$.
   - If $b = 1$, the challenger returns to $\mathcal{A}$ signatures $\Sigma_1 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C_1, \mathbf{m})$ and $\Sigma_2 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_2}, C_2, \mathbf{m})$.
3. $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ for b.

The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - 1/2|$ and we say that the attribute-based signature scheme is *computationally exteriorly private* if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.

The second notion is adaptively unforgeability, which requires that an adversary is only able to forge a signature with negligible probability even if she sees signatures on messages of her choice.

Definition 3.6 (Unforgeability)**.** The security notion of *adaptively unforgeable* for a linkable attribute-based signature scheme is defined by the following game between a challenger and an adversary $\mathcal{A}$:

1. The challenger runs $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{LABS.Setup}(1^\lambda, 1^\ell)$ and gives mpk to $\mathcal{A}$.
2. $\mathcal{A}$ may adaptively make the following queries to the challenger:
   (i) **Signing**. $\mathcal{A}$ submits a signing query on any attribute, message and circuit tuple $(\mathbf{x}, \mathbf{m}, C)$ to the challenger.
   - If the tuple $(\mathbf{x}, \mathbf{m}, C)$ has not been previously queried, the challenger first checks that $C(\mathbf{x}) = 1$.

* If $\mathbf{x}$ has not been submitted previously, the challenger runs $\mathsf{sk}_{\mathbf{x}} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$ and sends the signature $\Sigma \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}}, C, \mathbf{m})$ to $\mathcal{A}$.
* If $\mathbf{x}$ has been submitted previously, the challenger just sends the signature $\Sigma \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}}, C, \mathbf{m})$ to $\mathcal{A}$.

– If the tuple $(\mathbf{x}, \mathbf{m}, C)$ has been previously queried, the challenger resends the signature Σ that has been previously produced with that tuple to $\mathcal{A}$.

(ii) **Key reveal**. $\mathcal{A}$ submits a key reveal query on any attribute $\mathbf{x}$ to the challenger.

– If the attribute $\mathbf{x}$ has not been submitted previously, the challenger returns the signing key $\mathsf{sk}_{\mathbf{x}} \xleftarrow{\$} \mathsf{LABS.KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{x})$ to $\mathcal{A}$.

– If the attribute $\mathbf{x}$ has been submitted previously, the challenger resends the signing key $\mathsf{sk}_{\mathbf{x}}$ that has been previously produced with that attribute to $\mathcal{A}$.

3. $\mathcal{A}$ outputs a signature $(\mathbf{m}^*, C^*, \Sigma^*)$

The adversary $\mathcal{A}$ wins the game if the following three conditions hold:

(i) $\mathsf{LABS.Verify}(\mathsf{mpk}, \mathbf{m}^*, C^*, \Sigma^*) = \mathsf{Valid}$.

(ii) $\mathcal{A}$ did not submit a signing query on $(\mathbf{x}, \mathbf{m}^*, C^*)$ for any $\mathbf{x}$ such that the challenger did not output $\perp$.

(iii) $\mathcal{A}$ did not submit a key reveal query for any $\mathbf{x}$ such that $C^*(\mathbf{x}) = 1$.

The advantage of $\mathcal{A}$ is defined as the probability of $\mathcal{A}$ winning the above game. We say that the attribute-based signature scheme is *computationally adaptively unforgeable* if the advantage of any PPT algorithm $\mathcal{A}$ is negligible.

4. Generic LABS scheme

Before presenting our construction, we provide a brief overview. The main idea in the standard ABS scheme of El Kaafarani and Katsumata [EKK18] is that the attribute authority issues a signature σ on an attribute $\mathbf{x} \in \{0,1\}^{\ell}$ to certify that the intended signer is allowed to sign a message on behalf of that attribute. To sign anonymously, the signer proves the following facts in zero-knowledge: the signature issued by the attribute authority is valid and the corresponding secret attribute satisfies the public circuit $C \in \mathcal{C}_{\ell}$ attached to the message. To do so, the signer first commits to the signature and all of the values assigned to the internal wires of the circuit C on input the attribute $\mathbf{x}$. Then he proves in zero-knowledge that the values inside the commitments satisfy Eq. (2 - 4).

However, to construct our generic LABS scheme we add some extra tools to this ABS scheme. Namely, the signer evaluates a hash function on input her attribute $\mathbf{x}$ and the circuit C satisfied by this attribute, and then proves in zero-knowledge that she is in possession of such a pair $(\mathbf{x}, C)$. Then, adding both the output of this hash function and the zero-knowledge proof of possession to the ABS signature provoke our linking algorithm to be functional, as it makes both the linkability and exculpability properties to be satisfied.

The reason behind the use of this hash function is its properties:

- Its determinism provides to the administrator the ability to output the same hash when the same signer intends to produce another signature.
- Its pre-image resistance gives the possibility of making every outputted hash public so it can be used by the linking algorithm.
- Collision resistance is a key part to accomplish both the linkability and the exculpability properties. A simple look to the hash will suffice to differentiate if two signatures have been issued by the same signer or not.

It is crucial to say that an input to this hash function will be formed by an attribute and a circuit that is satisfied by this attribute. In this way, if a voter wants to re-vote in the same election, the same hash will be given to her. However, if she wants to vote later in another election, a different hash will be given to her even if the same attribute is used. From now on, we will refer as *tag* to the output of this hash function as it fits better in the e-voting context.

Therefore, the tools we need to prepare are a digital signature scheme, a commitment scheme, a ZKPoK proving the relations that are described below between committed values and a last ZKPoK proving the well-formedness of the tag.

Before moving on, recall that in Definition 3.2 we described bounded arithmetic circuits as those who have their depth bounded by some natural number. Nevertheless, in the context of attribute-based signatures it is typical to say that an arithmetic circuit is bounded if the required hardness assumptions on the R-LWE problem and/or the R-SIS problems grow exponentially in the depth of the circuit. For instance, to base the security of the attribute-based signature scheme under a polynomial R-LWE assumption, we might need to restrict the depth of the circuit to be $O(\lambda)$, where 1^{λ} is the security parameter. In this context, our building block for LABS schemes support unbounded circuits as polices, as they do not suffer from this issue.

Let x_i for $i \in [\ell+1, \ell+|C|-1]$ denote the value to the i-th internal wire of the arithmetic circuit C on input $\mathbf{x} = (x_1, \ldots, x_{\ell})$ and let $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{sk}_{\mathsf{Sign}}$ denote the verification key and the signing key of the underlying digital signature scheme, respectively, and let $\mathsf{pk}_{\mathsf{Com}}$ denote the public

commitment key of the underlying commitment scheme. Then, the relation $\mathcal{R}_{\mathsf{LABS}}$ is defined as follows:

$$\mathcal{R}_{\mathsf{LABS}} = \Big(\text{statement} = (\mathsf{vk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Com}}, C \in \mathcal{C}_\ell, c_\sigma, (c_i)_{i=1}^{\ell+|C|-1}),$$
$$\text{witness} = (\mathbf{x} = (x_1, \dots, x_\ell), \sigma, d_\sigma, (d_i)_{i=1}^{\ell+|C|-1})\Big).$$

The committed values in $c_\sigma, (c_i)_{i=1}^{\ell+|C|-1}$ satisfy the following conditions:

- $(c_\sigma, d_\sigma) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}_{\mathsf{Com}}, \sigma)$ and $(c_i, d_i) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}_{\mathsf{Com}}, x_i)$ for $i \in [\ell + |C| - 1]$;
- $\mathsf{S.Verify}(\mathsf{vk}_{\mathsf{Sign}}, \mathbf{x}, \sigma) = 1$;
- $x_i = x_{i_1} \star_i x_{i_2}$ for $i \in [\ell + 1, \ell + |C| - 1]$, where $(\star_i, i_1, i_2) \leftarrow \mathsf{topo}_C(i)$ and $\star_i \in \{+, \times\}$;
- $1 = x_{(\ell+|C|)_1} \star_{(\ell+|C|)} x_{(\ell+|C|)_2}$, where $(\star_{(\ell+|C|)}, (\ell + |C|)_1, (\ell + |C|)_2) \leftarrow \mathsf{topo}_C(\ell + |C|)$.

Here, recall that $\mathcal{D}_{\mathsf{Open}}(\mathsf{pk}_{\mathsf{Com}}, m)$ is the set of all possible pairs that are labelled as valid by the verification algorithm $\mathsf{C.Ver}(\mathsf{pk}_{\mathsf{Com}}, m)$ under a fixed public key $\mathsf{pk}_{\mathsf{Com}}$ and a message m. We simply define the corresponding language $\mathcal{L}_{\mathsf{LABS}}$ as the language induced by the relation $\mathcal{R}_{\mathsf{LABS}}$.

For simplicity, in the rest of this section we omit $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$ from the statement, since they are fixed by the LABS.Setup algorithm and all signers use the same $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$.

4.1 Construction

Now, we provide our linkable attribute-based signature scheme for unbounded arithmetic circuits. In what follows, we assume a digital signature scheme (S.KeyGen, S.Sign, S.Verify) and a commitment scheme (C.Gen, C.Com, C.Ver).

- **LABS.Setup**: On input the security parameter 1^λ and the input length 1^ℓ for the family of circuits $\mathcal{C}_\ell$, generate a verification key and a signing key $(\mathsf{vk}_{\mathsf{Sign}}, \mathsf{sk}_{\mathsf{Sign}}) \xleftarrow{\$} \mathsf{S.KeyGen}(1^\lambda, 1^\ell)$ and a public commitment key $\mathsf{pk}_{\mathsf{Com}} \xleftarrow{\$} \mathsf{C.Gen}(1^\lambda)$. Then output:

  $$\mathsf{mpk} = (\mathsf{vk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Com}}, H(\cdot), G(\cdot)) \text{ and } \mathsf{msk} = (\mathsf{sk}_{\mathsf{Sign}});$$

  where $H(\cdot)$ and $G(\cdot)$ are hash functions used by the algorithm LABS.Sign, which are programmed as random oracles in the security proof. Further, we assume the output space of $G(\cdot)$ to be $\{0,1\}^\ell$.

- **LABS.KeyGen**: On input $\mathbf{x} = (x_1, \dots, x_\ell) \in \{0,1\}^\ell$, create a signature on the attribute $\mathbf{x}$ by running $\sigma \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \mathbf{x})$. Then, output the secret key as $\mathsf{sk}_{\mathbf{x}} = (\mathbf{x}, \sigma)$.

- **LABS.Sign**: On input a message $\mathbf{m} \in \{0,1\}^*$, a secret key $\mathsf{sk}_{\mathbf{x}}$ and a circuit $C \in \mathcal{C}_\ell$ with an associating topology $\mathsf{topo}_C(\cdot)$ such that $C(\mathbf{x}) = 1$, proceeds as follows:

  1. On input the attribute $\mathbf{x}$ and the satisfied circuit C, compute the tag $T_{\mathbf{x}} \longleftarrow H(\mathbf{x}, C)$[6].
  2. Generate a ZKPoK π proving possession of an attribute $\mathbf{x}$ such that $T_{\mathbf{x}} \longleftarrow H(\mathbf{x}, C)$.

[6]We assume whenever it is necessary that we can encode C uniquely into a binary string

3. Compute $\mathbf{h} = (h_1, \dots, h_\ell) \longleftarrow G(\mathbf{m}, C)$ and create a new circuit $\hat{C} \in \mathcal{C}_\ell$ with two dummy gates connected to each of the input wires of C. Namely, to the input wires $i \in [\ell]$ of C, we add a series composition of two addition gates where one gate adds h_i and the other gate adds $-h_i$. On input x_i to the i-th input wire of $\hat{C}$, it first evaluates to $x_i + h_i$ and then evaluates back to $x_i + h_i - h_i = x_i$, on which point it gets fed to the i-th input wire of C. Further, let N to be the number of gates of $\hat{C}$: $N = |\hat{C}| = |C| + 2\ell$.
4. For all $i \in [\ell + 1, \ell + N - 1]$ compute $(\star_i, i_1, i_2) \longleftarrow \mathsf{topo}_{\hat{C}}(i)$ where $\star_i \in \{+, \times\}$, and denote the newly created values $(x_i)_{i=\ell+1}^{\ell+N-1}$ in ascending order as

$$\begin{cases} x_i = x_{i_1} + x_{i_2}, & \text{if } \star_i = + \\ x_i = x_{i_1} \cdot x_{i_2}, & \text{if } \star_i = \times \end{cases}.$$

5. Create a commitment $(c_\sigma, d_\sigma) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}_{\mathsf{Com}}, \sigma)$ of the signature σ, and for all $i \in [\ell + N - 1]$, create a commitment $(c_i, d_i) \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}_{\mathsf{Com}}, x_i)$ that commits to the value of each wire in $\hat{C}$ (except for the output one).
6. Generate a ZKPoK Π proving that the committed values satisfy the relation $\mathcal{R}_{\mathsf{LABS}}$. Concretely, it generates a proof for the following conditions:
   - The attribute $(c_i)_{i=1}^{\ell}$ committed to $\mathbf{x} = (x_1, \dots, x_\ell)$ and the signature c_σ committed to σ satisfy the following verification equation:

$$\mathsf{S.Verify}(\mathsf{vk}_{\mathsf{Sign}}, \mathbf{x}, \sigma) = 1. \tag{2}$$

   - For all $i \in [\ell + 1, \ell + N - 1]$, the value c_i committed to x_i satisfy the following equation:

$$\begin{cases} x_i = x_{i_1} + x_{i_2} & \text{if } \star_i = + \\ x_i = x_{i_1} \cdot x_{i_2} & \text{if } \star_i = \times \end{cases}. \tag{3}$$

   - The values $c_{(\ell+N)_1}$ and $c_{(\ell+N)_2}$ committed to $x_{(\ell+N)_1}$ and $x_{(\ell+N)_2}$, respectively, satisfy the following equation:

$$\begin{cases} 1 = x_{(\ell+N)_1} + x_{(\ell+N)_2} & \text{if } \star_{\ell+N} = + \\ 1 = x_{(\ell+N)_1} \cdot x_{(\ell+N)_2} & \text{if } \star_{\ell+N} = \times \end{cases}. \tag{4}$$

7. Finally, output $\Sigma_{\mathbf{x}} = (T_{\mathbf{x}}, \pi, c_\sigma, (c_i)_{i=1}^{\ell+N-1}, \Pi)$.

- **LABS.Verify**: Compute $\mathbf{h} \longleftarrow G(\mathbf{m}, C)$ and construct the circuit $\hat{C}$. Then, verify the proof Π with respect to the circuit $\hat{C}$ and the proof π with respect to the attribute $\mathbf{x}$. Output 1 if both proofs are verified correctly, and output 0 otherwise.
- **LABS.Link**: Given $\Sigma_{\mathbf{x}_1}, \Sigma_{\mathbf{x}_2}$, return 1 if $T_{\mathbf{x}_1} = T_{\mathbf{x}_2}$ and 0 otherwise.

Correctness. Observe that $\hat{C}(\mathbf{x}) = C(\mathbf{x})$ for all $\mathbf{m}, \mathbf{x}$. Also note that if the same attribute is used to generate a second or more signatures using the same circuit, then the tag will not vary. Therefore, the correctness of the scheme follow simply from the correctness of the underlying commitment scheme, digital signature scheme, the ZKPoK for the relation $\mathcal{R}_{\mathsf{LABS}}$ and the ZKPoK proving possession of a preimage for the hash function $H(\cdot)$.

Lemma 4.1 (Linkability)**.** *The above* LABS *scheme satisfies the linkability property when the hash function $H(\cdot)$ is collision resistant.*

Proof. Assume that the adversary $\mathcal{A}$ submits the attributes $\mathbf{x}_1, \ldots, \mathbf{x}_t$ as the challenge. The challenger responds by sending back the secret keys $\mathsf{sk}_{\mathbf{x}_1} = (\mathbf{x}_1, \sigma_{\mathbf{x}_1}), \ldots, \mathsf{sk}_{\mathbf{x}_t} = (\mathbf{x}_t, \sigma_{\mathbf{x}_t})$ associated to these attributes. Then, the adversary has to output $t+1$ signatures $\Sigma_{\mathbf{x}_1^*}, \ldots, \Sigma_{\mathbf{x}_{t+1}^*}$ (associated to the messages $\mathbf{m}_1, \ldots, \mathbf{m}_{t+1}$) where the first two components of these signatures are $(T_{\mathbf{x}_1^*}, \pi_{\mathbf{x}_1^*}), \ldots, (T_{\mathbf{x}_{t+1}^*}, \pi_{\mathbf{x}_{t+1}^*})$, respectively. The adversary $\mathcal{A}$ wins the game if the signatures are verified to be valid with respect to the messages and

$$\mathsf{LABS.Link}(\mathbf{m}_i, \mathbf{m}_j, \Sigma_{\mathbf{x}_i}, \Sigma_{\mathbf{x}_j}) = 0,$$

for every $i, j \in [t+1]$.

This is equivalent to test if $T_{\mathbf{x}_i} \neq T_{\mathbf{x}_j}$ for every $i, j \in [t+1]$. Now, since the same circuit C is used to produce all the signatures and the collision resistance hash function $H(\cdot)$ produces the tags (guaranteed by the soundness property of the ZKPoK π), it is clear that $T_{\mathbf{x}_i} \neq T_{\mathbf{x}_j}$ if and only if $\mathbf{x}_i \neq \mathbf{x}_j$. Moreover, for every $i \in [t+1]$, the soundness property of the proof $\Pi_{\mathbf{x}_i}$ implies that the adversary $\mathcal{A}$ knows a signature σ' of some attribute $\mathbf{x}'$, and the unforgeability of σ' implies that this signature has to be among $\sigma_{\mathbf{x}_1}, \ldots, \sigma_{\mathbf{x}_1}$.

Hence, from the collision resistance of $H(\cdot)$ and the pigeonhole principle we obtain that there exists two distinct $i^*, j^* \in [t+1]$ such that $\mathbf{x}_{i^*} = \mathbf{x}_{j^*}$ and then $T_{\mathbf{x}_{i^*}} = T_{\mathbf{x}_{i^*}}$. Hence, the advantage of the adversary $\mathcal{A}$ of winning the linkability game is 0. ■

Lemma 4.2 (Exculpability)**.** *The above* LABS *scheme satisfies the exculpability property when the hash function $H(\cdot)$ is collision resistant.*

Proof. This proof is similar to the linkability one. Assume that the adversary $\mathcal{A}$ sends the attributes $\mathbf{x}_1, \ldots, \mathbf{x}_t$ and a circuit C to the challenger. The challenger responds by sending back the secret keys $\mathsf{sk}_{\mathbf{x}_1}, \ldots, \mathsf{sk}_{\mathbf{x}_{t-1}}$ along with the signature Σ produced with the missing secret key $\mathsf{sk}_{\mathbf{x}_t}$. We denote as $T_{\mathbf{x}_t}$ the tag issued in Σ. Finally, the adversary $\mathcal{A}$ outputs a signature Σ^* together with the message m^*. Similarly, we denote as $T_{\mathbf{x}^*}$ the tag issued in Σ^*. The adversary $\mathcal{A}$ wins the game if Σ^* is verified to be valid with respect to the message $\mathbf{m}^*$ and

$$\mathsf{LABS.Sign}(\mathbf{m}, \mathbf{m}^*, \Sigma, \Sigma^*) = 1.$$

Again, this is equivalent to test if $T_{\mathbf{x}_t} = T_{\mathbf{x}^*}$. Following the same argument as in the previous proof, this equality holds if the attributes $\mathbf{x}_t$ and $\mathbf{x}^*$ coincide, where $\mathbf{x}^*$ is an attribute from the list $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$. Hence, the advantage of the adversary $\mathcal{A}$ of winning the exculpability game is negligible. ■

4.2 Security analysis

Theorem 4.3 (Interior privacy)**.** *Assuming a computationally hiding commitment scheme, computationally special* HVZK *proofs of knowledge for both the relation $\mathcal{R}_{\mathsf{LABS}}$ and a secure hash function $H(\cdot)$; we have that the previous linkable attribute-based scheme is computationally interiorly private in the random oracle model.*

Proof. Assume that the adversary $\mathcal{A}$ submits $(\mathbf{m}, \mathbf{x}_0, \mathbf{x}_1, C)$ as the challenge, and let $\hat{C}$ be the circuit created at Step 3 of the Sign algorithm that has N gates. In the actual game, which we denote by $\mathsf{Game}_{\mathsf{real}}$, the challenger picks a random bit $b \xleftarrow{\$} \{0,1\}$ and returns the signature $\Sigma^* \xleftarrow{\$} \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_b}, C, \mathbf{m})$, where $\Sigma^* = (T_{\mathbf{x}_b}, \pi^*, c_\sigma^*, (c_i^*)_{i=1}^{\ell+N-1}, \Pi^*)$. Here, the proof Π^* is the

actual ZKPoK created with the witness satisfying $\left((\hat{C}, c_\sigma^*, (c_i^*)_{i=1}^{\ell+|C|-1}), (\mathbf{x}_b, \sigma, d_\sigma^*, (d_i^*)_{i=1}^{\ell+|C|-1})\right) \in \mathcal{R}_{\mathsf{LABS}}$ as input, where $\sigma \xleftarrow{\$} \mathsf{S.Sign}(\mathsf{sk}_{\mathsf{Sign}}, \mathbf{x}_b)$, and every commitment/opening pairs are created by running $\mathsf{C.Com}(\mathsf{pk}_{\mathsf{Com}}, \cdot)$. Recall that we omit $\mathsf{vk}_{\mathsf{Sign}}$ and $\mathsf{pk}_{\mathsf{Com}}$ from the statement for simplicity.

We begin by considering the game Game_{-2}, where the proof Π^* is instead created by running the zero-knowledge simulator $\mathcal{S}_{\Pi^*}$ of the protocol itself. Since the ZKPoK Π^* is assumed to be computationally special HVZK, the proof Π^* created in $\mathsf{Game}_{\mathsf{real}}$ and Game_{-2} are computationally indistinguishable. Similarly, we consider the game Game_{-1}, where now the proof π^* is instead created by running the zero-knowledge simulator $\mathcal{S}_{\pi^*}$. Again, since the ZKPoK π^* is assumed to be computationally special HVZK, the proof π^* created in $\mathsf{Game}_{\mathsf{real}}$ and Game_{-1} are computationally indistinguishable. Finally, we consider the game Game_0, where the tag $T_{\mathbf{x}_b}$ is replaced by a random number. Hence, since $H(\cdot)$ is assumed to be a random oracle, the tag made in Game_0 and $\mathsf{Game}_{\mathsf{real}}$ are computationally indistinguishable.

Now, we consider $\ell + N - 1$ hybrid games, where in the i-th game Game_i, the challenger swaps the commitment c_i^* with a commitment to 0. Finally, the Game_i challenger outputs a challenge signature $\Sigma^* = (T_{\mathbf{x}_b}, \pi^*, c_\sigma^*, (c_i^*)_{i=1}^{\ell+N-1}, \Pi^*)$. Due to the computationally hiding property of the underlying commitment scheme, the view of the adversary in Game_{i-1} and Game_i is negligible, for $i = -1, 0, \ldots, \ell + N - 1$.

Finally, in game $\mathsf{Game}_{\ell+N}$, the challenger swaps the commitment c_σ^* with a commitment to 0. Following the same argument as above, the differences in the view of the adversary in $\mathsf{Game}_{\ell+N-1}$ and $\mathsf{Game}_{\ell+N}$ is negligible due to the computationally hiding property of the commitment scheme. Furthermore, since all the commitments are now commitments to 0 in $\mathsf{Game}_{\ell+N}$, the signature $\Sigma^* = (T_{\mathbf{x}_b}, \pi^*, c_\sigma^*, (c_i^*)_{i=1}^{\ell+N-1}, \Pi^*)$ is completely independent of the attributes $\mathbf{x}_0, \mathbf{x}_1$. Therefore we have that in $\mathsf{Game}_{\ell+N}$, the advantage of adversary $\mathcal{A}$ is 0.

Combining all the games together, we have that the advantage of any adversary $\mathcal{A}$ winning $\mathsf{Game}_{\mathsf{real}}$ is negligible. ■

Theorem 4.4 (Exterior privacy)**.** *Assuming a computationally hiding commitment scheme, computationally special* HVZK *proofs of knowledge for both the relation $\mathcal{R}_{\mathsf{LABS}}$ and a secure hash function $H(\cdot)$; we have that the previous linkable attribute-based scheme is computationally exteriorly private in the random oracle model.*

Proof. Assume that the adversary $\mathcal{A}$ submits $(\mathbf{m}, \mathbf{x}_1, \mathbf{x}_2, C_1, C_2)$ as the challenge. In the actual game, which we denote by $\mathsf{Game}_{\mathsf{real}}$, the challenger picks a random bit $b \xleftarrow{\$} \{0, 1\}$ and then proceeds as follows:

- If $b = 0$, it returns to $\mathcal{A}$ the signatures

$$\Sigma_1 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C_1, \mathbf{m}), \quad \text{and} \quad \Sigma_2 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C_2, \mathbf{m}).$$

- If $b = 1$, it returns to $\mathcal{A}$ the signatures

$$\Sigma_1 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_1}, C_1, \mathbf{m}), \quad \text{and} \quad \Sigma_2 \xleftarrow{\$} \mathsf{LABS.Sign}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{x}_2}, C_2, \mathbf{m}).$$

The rest of this proof is analogous to the previous one. Namely, we define a sequence of games in which every component of the signatures Σ_1 and Σ_2 are replaced by (computationally indistinguishable) random values; in both $b = 0$ and $b = 1$. Hence, the advantage of adversary $\mathcal{A}$ is again negligible as the cases are indistinguishable. ■

Theorem 4.5 (Adaptive unforgeability)**.** *Assume a computationally hiding and a computationally binding commitment scheme, computationally special* HVZK *proofs of knowledge for the relation* $\mathcal{R}_{\mathsf{LABS}}$ *and a secure hash function* $H(\cdot)$*, and an* eu-acma *digital signature scheme. Then, the previous linkable attribute-based scheme is adaptively unforgeable in the random oracle model.*

Proof. Note that our LABS scheme only varies with the ABS scheme defined in [EKK18] in the computation of the tag and the generation of the ZKPoK π. Also, note that our definition for adaptive unforgeability is equivalent. Hence, having in mind that π is assumed to be computationally special HVZK, this proof is omitted as is analogous to Theorem 2 in [EKK18]. ■

From Lemmas 4.1 and 4.2 and Theorems 4.3, 4.4 and 4.5 we conclude that there exists an interior private, exterior private and adaptive unforgeable linkable attribute-based signature scheme in the random oracle model.

5. LABS for unbounded circuits from lattices

In this section, we provide an instantiation of our generic LABS construction for unbounded circuits from lattices.

5.1 Preparing tools

We use the commitment scheme proposed by Martínez and Morillo [MM19] together with their ZKPoK for proving valid openings and for proving arithmetic relations between committed values (see Section 2.4.5). To prove in zero-knowledge that tags are computed properly (i.e., proving knowledge of a witness $(\mathbf{x}, C)$ such that $T_{\mathbf{x}} \longleftarrow H(\mathbf{x}, C)$) we use the general ZKPoK proposed by Jawurek et al. [JKO13]. We will comment this protocol later in this section. Finally, we use the variant of Boyen's signature [Bos+20] described in Section 2.4.6 as our underlying signature scheme. From all this building block, we obtain an associating lattice-based LABS scheme for the relation $\mathcal{R}_{\mathsf{LABS}}$.

[MM19] provides three ZKPoK for proving useful relations among committed values: Π_{Open} for proving knowledge of a valid opening and Π_{Add}, Π_{Mult} for proving arithmetic relations (over R_q) of committed values.

Theorem 5.1. *The commitment scheme in [MM19] has associating* ZKPoK *($\Pi_{\mathsf{Open}}, \Pi_{\mathsf{Add}}$ and Π_{Mult} respectively) for the following three relations*

$\mathcal{R}_{\mathsf{Open}} = \{(\mathsf{pk}, c), (m, d) \mid (c, d) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m)\}$,

$\mathcal{R}_{\mathsf{Add}} = \{(\mathsf{pk}, (c_i)_{i=1}^3), ((m_i, d_i)_{i=1}^3) \mid m_3 = m_1 + m_2 \wedge (c_i, d_i) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m_i) \text{ for } i \in [3]\}$,

$\mathcal{R}_{\mathsf{Mult}} = \{(\mathsf{pk}, (c_i)_{i=1}^3), ((m_i, d_i)_{i=1}^3) \mid m_3 = m_1 \cdot m_2 \wedge (c_i, d_i) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m_i) \text{ for } i \in [3]\}$,

as they satisfy the properties of completeness, soundness and zero-knowledge.

These protocols are described in a low level in Section 2.4.5. The above protocols additionally require internally an standard commitment scheme, which is used by the prover in the first round to send a commitment to the verifier. Although, we could use the commitment scheme of [MM19], this would not be the best idea since we would lack of efficiency. A better idea would be, for instance, to simply use a hash function as this internal commitment algorithm: on input a message m, the commitment c is the output of the hash function applied to the message m concatenated with a random string, and the opening d is the random string.

In addition to these three zero-knowledge proofs, we construct two more useful ZKPoK, that will be denoted as $\Pi_{\mathsf{EqTo}\star}, \Pi_{\mathsf{EqTo}\{0,1\}}$, for proving that a commitment opens to a specific value and a more specific one proving that a commitment opens to 0 or 1, respectively. These additional ZKPoK will be a key part in proving Eq. (2-4), as we need to prove in zero-knowledge more complex relations than those covered by the zero-knowledge proofs provided in Theorem 5.1. For instance, while we can use Π_{Add} or Π_{Mult} to prove that the commitments satisfy the relation $m_3 = m_1 \star m_2$, where $\star \in \{+, \times\}$, we can not use any of the above ZKPoK to prove that the commitments satisfy the relation $1 = m_1 \star m_2$.

Theorem 5.2. *The commitment scheme in [MM19] has associating* ZKPoK *for the following two relations*

$\mathcal{R}_{\mathsf{EqTo}\star} = \{(\mathsf{pk}, c, m), d \mid (c, d) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m)\}$,

$\mathcal{R}_{\mathsf{EqTo}\{0,1\}} = \{(\mathsf{pk}, c), (m \in \{0, 1\}, d) \mid (c, d) \in \mathcal{D}_{\mathsf{Open}}(\mathsf{pk}, m)\}$,

as they satisfy the properties of completeness, soundness and zero-knowledge.

Proof. First, we show how to construct a ZKPoK for the relation $\mathcal{R}_{\mathsf{EqTo}\star}$. We follow the proof for the relation $\mathcal{R}_{\mathsf{Open}}$ described in [MM19]. Recall Section 2.4.5 for the idea of using masking elements to hide the elements that compose the commitment.

Let $\mathbf{a} = (a_1, \dots, a_k), \mathbf{b} = (b_1, \dots, b_k) \in R_q^k$, a message $m \in R_q$, a random element r in R_q and $\mathbf{e} \in R_q^k$. The goal of the protocol is for the prover $\mathcal{P}$ to convince the verifier $\mathcal{V}$ that the commitment $\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e} \in R_q^k$, with $\|\mathbf{e}\| < 2^\kappa$, is a valid commitment of the public message m without leaking any other information.

We identify a polynomial u with a vector $\mathbf{u}$ that has as elements the coefficients of the polynomial. For convenience, we also identify a vector of polynomials with the concatenation of its associated vectors.

$$\begin{aligned} \varphi\colon \mathbb{Z}_q^n &\to R_q \\ \mathbf{u} = (u_0, u_1, \dots, u_{n-1}) &\mapsto u = u_0 + u_1 x + \cdots + u_{n-1} x^{n-1} \\ \phi\colon \mathbb{Z}_q^{nk} &\to R_q^k \\ \mathbf{u} = (u_0, u_1, \dots, u_{nk-1}) &\mapsto \mathbf{u} = (\varphi(u_0, \dots, u_{n-1}), \dots, \varphi(u_{n(k-1)}, \dots, u_{nk-1})) \end{aligned}$$

$\mathcal{P}$ first consider the vector $\bar{\mathbf{e}} = \phi^{-1}(\mathbf{e}) + 2^\kappa \mathbb{1}_{nk}$ and its binary decomposition $\bar{\mathbf{e}} = \sum_{j=0}^{\kappa} 2^j \bar{\mathbf{e}}_j$, $\bar{\mathbf{e}}_j \in \{0,1\}^{nk}$. Then, $\mathcal{P}$ appends to each vector $\bar{\mathbf{e}}_j$ a vector of bits $\mathbf{e}_j''$ such that $\mathbf{e}_j' = (\bar{\mathbf{e}}_j \mid \mathbf{e}_j'') \in \mathcal{B}_{nk}$. Recall that $\mathcal{B}_{nk} \subset \{0,1\}^{2nk}$ is the set of vectors with the same number of 0 and 1's.

Then, we have: $\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e} = \mathbf{a}m + \mathbf{b}r + \phi((\mathbf{l}' \sum_j 2^j \mathbf{e}_j') - 2^\kappa \mathbf{1}_{nk})$,
where $\mathbf{l}' = [\mathbf{I}_{nk} \mid \mathbf{0}_{nk}]$.

Let $\mathcal{S}_{2nk}$ the set of permutations over $2nk$ elements. Protocol 10 depicts the ZKPoK for the relation $\mathcal{R}_{\mathsf{EqTo}\star}$.

Completeness: It is immediate as all relations hold by construction. Note that in this case, we check the equality $\widetilde{\mathbf{y}} + \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{b}\widetilde{s} - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j)) \stackrel{?}{=} \mathbf{a}(\alpha m)$ instead of just asking if the left hand side belongs to the lattice $\mathcal{L}(\mathbf{a})$. Also note that we do not mask the message m as it is also known for the verifier $\mathcal{V}$.

Soundness: If a (possible malicious) prover $\widetilde{\mathcal{P}}$ is able to provide accepted answers to δ rounds of interaction with an honest verifier $\mathcal{V}$ with probability $(q + \frac{1}{2q})^\delta + \epsilon$, where ϵ is non-negligible, then he is able to efficiently extract a witness with probability $2(\epsilon/3)^3$. Details on how to find valid answers to the required number of possible challenges are skipped as they are proved in [MM19].

By the pigeonhole principle we can find commitments c_1, c_2, two α, α' and two $\mathbf{g}_j, \mathbf{g}_j'$ that induce accepted answers. Define $\Delta_\alpha = \alpha - \alpha' \neq 0$. The binding property ensures that openings to $\widetilde{\pi}_j, \widetilde{\mathbf{y}}$ and $\widetilde{\mathbf{e}}_j'$ are fixed. Then we have

$$
\begin{aligned}
\mathbf{a}(\alpha m) &= \widetilde{\mathbf{y}} + \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{b}\widetilde{s} - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j)) \\
\mathbf{a}(\alpha' m) &= \widetilde{\mathbf{y}} + \alpha'(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{b}\widetilde{s}' - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}'_j)) \\
\Delta_\alpha \mathbf{c} &= \mathbf{a}\Delta_\alpha m + \mathbf{b}(\widetilde{s} - \widetilde{s}') + \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j - \mathbf{g}'_j) - \Delta_\alpha 2^\kappa \mathbf{1}_{nk}) \\
\mathbf{c} &= \mathbf{a}m + \mathbf{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\Delta_\alpha^{-1}(\mathbf{g}_j - \mathbf{g}'_j)) - 2^\kappa \mathbf{1}_{nk}) \\
\mathbf{g}_j - \alpha\widetilde{\mathbf{e}}'_j &= \mathbf{g}'_j - \alpha'\widetilde{\mathbf{e}}'_j \\
\widetilde{\mathbf{e}}'_j &= \Delta_\alpha^{-1}(\mathbf{g}_j - \mathbf{g}'_j) \\
\mathbf{c} &= \mathbf{a}m + \mathbf{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\mathbf{e}}'_j) - 2^\kappa \mathbf{1}_{nk})
\end{aligned}
$$

As these elements come from accepted answers we know that $\widetilde{\mathbf{e}}_j \in \mathcal{B}_{nk} \subset \{0,1\}^{2nk}$ and therefore $\phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\mathbf{e}}'_j) - 2^\kappa \mathbf{1}_{nk})$ has norm smaller than 2^κ. Then $(m, \Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}'), \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\mathbf{e}}'_j) - 2^\kappa \mathbf{1}_{nk}))$ is a valid opening.

Zero-Knowledge:

Case $b = 0$:

$$
\begin{aligned}
&\hat{s} \xleftarrow{\$} R_q, \quad \hat{\mathbf{g}}_j \xleftarrow{\$} \mathbb{Z}_q^{2nk}, \quad \hat{\pi}_j \xleftarrow{\$} \mathcal{S}_{2nk} \\
&c_1 = \mathrm{aCom}(\mathbf{b}\hat{s} + \phi(\mathbf{l}'\Sigma_j 2^j \hat{\pi}_j^{-1}(\hat{\mathbf{g}}_j)) \\
&\qquad - \alpha(\mathbf{c} - \mathbf{a}m + \phi(2^\kappa \mathbf{1}_{nk})), \{\hat{\pi}_j\}_j)
\end{aligned}
$$

$\mathcal{P}$ reveals $\{\hat{\mathbf{g}}_j\}_j, \{\widetilde{\pi}_j = \hat{\pi}_j\}_j$, $\widetilde{\mathbf{y}} = \mathbf{b}\hat{s} + \phi(\mathbf{l}'\Sigma_j 2^j \hat{\pi}_j^{-1}(\hat{\mathbf{g}}_j)) - \alpha(\mathbf{c} - \mathbf{a}m + \phi(2^\kappa \mathbf{1}_{nk}))$ $\widetilde{s} = \hat{s}$. Indistinguishable from a real conversation with the same $\pi_j = \widehat{\pi}_j$ and where $\rho = \widehat{s} - \alpha r$ and $\mathbf{f}_j = \widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j) - \alpha \mathbf{e}'_j$.

$$
\begin{aligned}
\mathbf{g}_j &= \pi_j(\mathbf{f}_j + \alpha \mathbf{e}'_j) \\
&= \pi_j(\widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j)) + \pi_j(\alpha \mathbf{e}'_j - \alpha \mathbf{e}'_j) \\
&= \widehat{\mathbf{g}}_j
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{b}\rho + \phi(\mathbf{l}\Sigma_j 2^j \mathbf{f}_j) \\
&\quad = \mathbf{b}(\widehat{s} - \alpha r) + \phi(\mathbf{l}\Sigma_j 2^j (\widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j) - \alpha \mathbf{e}'_j)) \\
&\quad = \mathbf{b}\widehat{s} + \phi(\mathbf{l}\Sigma_j 2^j \widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j)) \\
&\qquad - \alpha(\mathbf{b}r + \phi(\mathbf{l}\Sigma_j 2^j \mathbf{e}'_j)) \\
&\quad = \mathbf{b}\widehat{s} + \phi(\mathbf{l}\Sigma_j 2^j \widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j)) \\
&\qquad - \alpha(\mathbf{c} - \mathbf{a}m + \phi(2^\kappa \mathbf{1}_{nk}))
\end{aligned}
$$

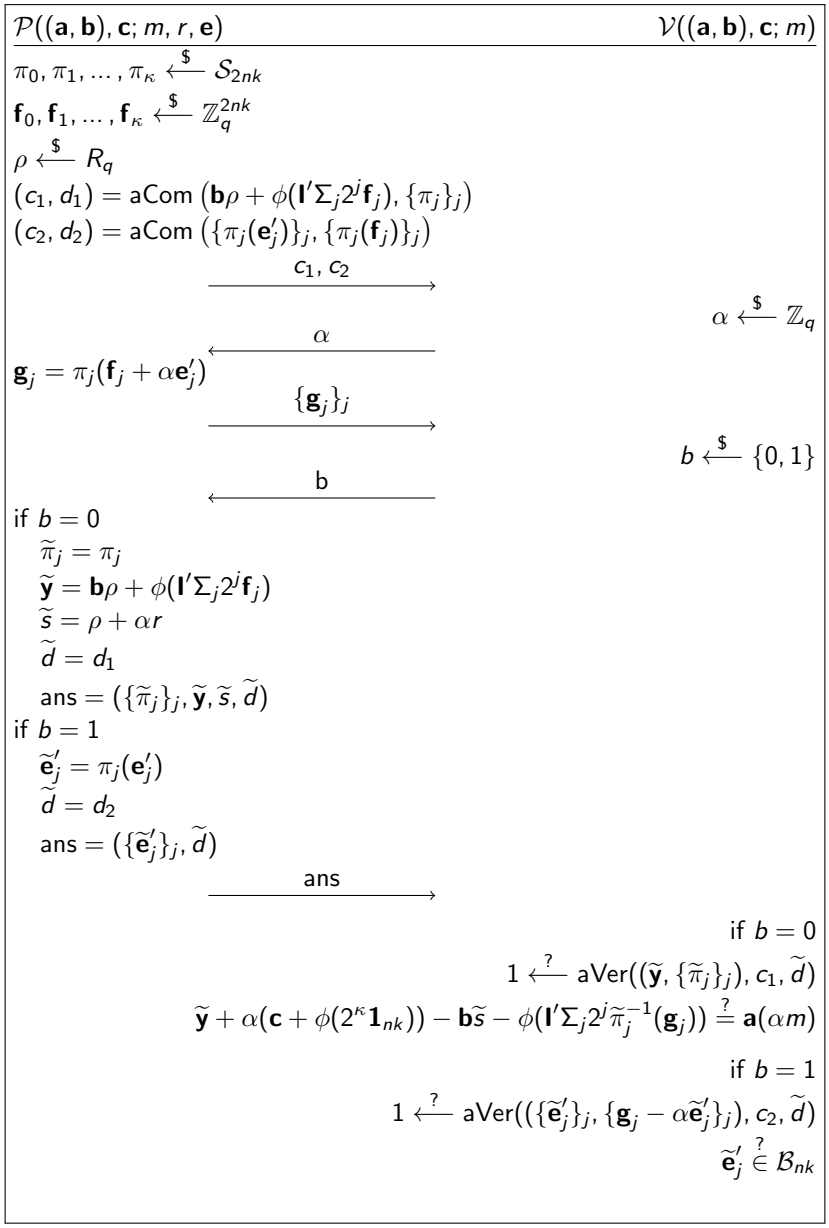
Case $b = 1$:

$$
\begin{aligned}
&\hat{\mathbf{e}}'_j \xleftarrow{\$} \mathcal{B}_{nk}, \quad \hat{\mathbf{f}}_j \xleftarrow{\$} \mathbb{Z}_q^{2nk} \quad \hat{\pi}_j \xleftarrow{\$} \mathcal{S}_{2nk} \\
&c_2 = \mathrm{aCom}(\{\hat{\pi}_j(\hat{\mathbf{e}}'_j)\}_j, \{\hat{\pi}_j(\hat{\mathbf{f}}_j)\}_j) \\
&\hat{\mathbf{g}}_j = \hat{\pi}_j(\hat{\mathbf{f}}_j + \alpha \hat{\mathbf{e}}'_j)
\end{aligned}
$$

$\mathcal{P}$ reveals $\{\widehat{\mathbf{g}}_j\}_j$, $\{\widetilde{\mathbf{e}}'_j = \widehat{\pi}_j(\widehat{\mathbf{e}}'_j)\}_j$. Equivalent to an honest conversation were π_j is such that $\pi_j(\mathbf{e}'_j) = \widehat{\pi}_j(\widehat{\mathbf{e}}'_j)$ and $\mathbf{f}_j$ is such that $\mathbf{f}_j = \pi_j^{-1}(\widehat{\pi}_j(\widehat{\mathbf{f}}_j))$.

$$
\begin{aligned}
\mathbf{g}_j &= \pi_j(\mathbf{f}_j + \alpha \mathbf{e}'_j) \\
&= \pi_j(\pi_j^{-1}(\widehat{\pi}_j(\widehat{\mathbf{f}}_j))) + \alpha \widehat{\pi}_j(\widehat{\mathbf{e}}'_j) \\
&= \widehat{\pi}_j(\widehat{\mathbf{f}}_j + \alpha \widehat{\mathbf{e}}'_j)
\end{aligned}
$$

Notice that in both cases simulated conversations follow the same distribution as honest conversations.



Protocol 10: ZKPoK for the relation $\mathcal{R}_{\mathsf{EqTo}\star}$.

Now, we show how to construct a ZKPoK for the relation $\mathcal{R}_{\mathsf{EqTo}\{0,1\}}$. As in the previous relation, we want to prove in zero-knowledge that $\mathbf{c} = \mathbf{a}m + \mathbf{b}r + \mathbf{e}$ opens to a value; but in this case we want this value to be 0 or 1. Protocol 11 depicts the ZKPoK for the relation $\mathcal{R}_{\mathsf{EqTo}\{0,1\}}$.

Completeness: It is immediate as all relations hold by construction. Note that in this case, we check the equality $\widetilde{\mathbf{y}} + \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{a}(\widetilde{\sigma}^{-1}(\mathbf{h}))_1 - \mathbf{b}\widetilde{s} - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j)) \stackrel{?}{=} \mathbf{0}$ as we again do not have exact knowledge of the message, but we know that it has to be a bit.

Soundness: Again, if a (possible malicious) prover $\widetilde{\mathcal{P}}$ is able to provide accepted answers to δ rounds of interaction with an honest verifier $\mathcal{V}$ with probability $(q + \frac{1}{2q})^\delta + \epsilon$, where ϵ is non-negligible, then he is able to efficiently extract a witness with probability $2(\epsilon/3)^3$.

By the pigeonhole principle we can find commitments c_1, c_2, two α, α' and two $\mathbf{g}_j, \mathbf{g}'_j, \mathbf{h}, \mathbf{h}'$ that induce accepted answers. Define $\Delta_\alpha = \alpha - \alpha' \neq 0$. The binding property ensures that openings to $\widetilde{\pi}_j, \widetilde{\sigma}, \widetilde{\mathbf{y}}, \widetilde{\mathbf{e}}'_j$ and $\widetilde{m}$ are fixed. Then we have

$$
\begin{aligned}
\mathbf{0} &= \widetilde{\mathbf{y}} + \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{a}(\widetilde{\sigma}^{-1}(\mathbf{h}))_1 - \mathbf{b}\widetilde{s} - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j)) \\
\mathbf{0} &= \widetilde{\mathbf{y}} + \alpha'(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{a}(\widetilde{\sigma}^{-1}(\mathbf{h}'))_1 - \mathbf{b}\widetilde{s}' - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}'_j)) \\
\Delta_\alpha \mathbf{c} &= \mathbf{a}(\widetilde{\sigma}^{-1}(\mathbf{h} - \mathbf{h}'))_1 + \mathbf{b}(\widetilde{s} - \widetilde{s}') + \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j - \mathbf{g}'_j) - \Delta_\alpha 2^\kappa \mathbf{1}_{nk}) \\
\mathbf{c} &= \mathbf{a}(\widetilde{\sigma}^{-1}(\Delta_\alpha^{-1}(\mathbf{h} - \mathbf{h}')))_1 + \mathbf{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\Delta_\alpha^{-1}(\mathbf{g}_j - \mathbf{g}'_j)) - 2^\kappa \mathbf{1}_{nk}) \\
\mathbf{g}_j - \alpha\widetilde{\mathbf{e}}'_j &= \mathbf{g}'_j - \alpha'\widetilde{\mathbf{e}}'_j \\
\widetilde{\mathbf{e}}'_j &= \Delta_\alpha^{-1}(\mathbf{g}_j - \mathbf{g}'_j) \\
\mathbf{h} - \alpha\widetilde{\mathbf{m}} &= \mathbf{h}' - \alpha'\widetilde{\mathbf{m}} \\
\widetilde{\mathbf{m}} &= \Delta_\alpha^{-1}(\mathbf{h} - \mathbf{h}') \\
\mathbf{c} &= \mathbf{a}(\hat{\sigma}^{-1}(\widetilde{\mathbf{m}}))_1 + \mathbf{b}(\Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}')) + \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\mathbf{e}}'_j) - 2^\kappa \mathbf{1}_{nk})
\end{aligned}
$$

As these elements come from accepted answers we know that $\widetilde{\mathbf{e}}_j \in \mathcal{B}_{nk} \subset \{0,1\}^{2nk}$ and therefore $\phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\mathbf{e}}'_j) - 2^\kappa \mathbf{1}_{nk})$ has norm smaller than 2^κ. For the same reason $\widetilde{\mathbf{m}} \in \mathcal{B}_1 \subset \{0,1\}^2$, and therefore $(\hat{\sigma}^{-1}(\widetilde{\mathbf{m}}))_1$ is a bit. Then $((\hat{\sigma}^{-1}(\widetilde{\mathbf{m}}))_1, \Delta_\alpha^{-1}(\widetilde{s} - \widetilde{s}'), \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\widetilde{\mathbf{e}}'_j) - 2^\kappa \mathbf{1}_{nk}))$ is a valid opening.

$\mathcal{P}((\mathbf{a},\mathbf{b}),\mathbf{c}; m \in \{0,1\}, r, \mathbf{e})$		$\mathcal{V}((\mathbf{a},\mathbf{b}),\mathbf{c}; m \in \{0,1\})$
$\pi_0, \pi_1, \ldots, \pi_\kappa \xleftarrow{\$} \mathcal{S}_{2nk}$		
$\sigma \xleftarrow{\$} \mathcal{S}_2$		
$\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_\kappa \xleftarrow{\$} \mathbb{Z}_q^{2nk}$		
$\boldsymbol{\mu} = (\mu_1, \mu_2) \xleftarrow{\$} \mathbb{Z}_q^2$		
$\rho \xleftarrow{\$} R_q$		
$(c_1, d_1) = \mathsf{aCom}\left(\sigma, \mathbf{a}\mu_1 + \mathbf{b}\rho + \phi(\mathbf{l}'\Sigma_j 2^j \mathbf{f}_j), \{\pi_j\}_j\right)$		
$(c_2, d_2) = \mathsf{aCom}\left(\sigma(\boldsymbol{\mu}), \sigma((m, 1-m)), \{\pi_j(\mathbf{e}'_j)\}_j, \{\pi_j(\mathbf{f}_j)\}_j\right)$		
	$\xrightarrow{c_1,\, c_2}$	
		$\alpha \xleftarrow{\$} \mathbb{Z}_q$
	$\xleftarrow{\alpha}$	
$\mathbf{g}_j = \pi_j(\mathbf{f}_j + \alpha \mathbf{e}'_j)$		
$\mathbf{h} = \sigma(\boldsymbol{\mu} + \alpha(m, 1-m))$		
	$\xrightarrow{\{\mathbf{g}_j\}_j,\, \mathbf{h}}$	
		$b \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{\mathsf{b}}$	
if $b = 0$		
$\widetilde{\pi}_j = \pi_j$		
$\widetilde{\sigma} = \sigma$		
$\widetilde{\mathbf{y}} = \mathbf{a}\mu_1 + \mathbf{b}\rho + \phi(\mathbf{l}'\Sigma_j 2^j \mathbf{f}_j)$		
$\widetilde{s} = \rho + \alpha r$		
$\widetilde{d} = d_1$		
$\mathsf{ans} = (\{\widetilde{\pi}_j\}_j, \widetilde{\sigma}, \widetilde{\mathbf{y}}, \widetilde{s}, \widetilde{d})$		
if $b = 1$		
$\widetilde{\mathbf{e}}'_j = \pi_j(\mathbf{e}'_j)$		
$\widetilde{\mathbf{m}} = \sigma((m, 1-m))$		
$\widetilde{d} = d_2$		
$\mathsf{ans} = (\{\widetilde{\mathbf{e}}'_j\}_j, \widetilde{\mathbf{m}}, \widetilde{d})$		
	$\xrightarrow{\mathsf{ans}}$	
		if $b = 0$
		$1 \xleftarrow{?} \mathsf{aVer}((\widetilde{\mathbf{y}}, \widetilde{\sigma}, \{\widetilde{\pi}_j\}_j), c_1, \widetilde{d})$
		$\widetilde{\mathbf{y}} + \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})) - \mathbf{a}(\widetilde{\sigma}^{-1}(\mathbf{h}))_1 - \mathbf{b}\widetilde{s} - \phi(\mathbf{l}'\Sigma_j 2^j \widetilde{\pi}_j^{-1}(\mathbf{g}_j)) \stackrel{?}{=} \mathbf{0}$
		if $b = 1$
		$1 \xleftarrow{?} \mathsf{aVer}((\{\widetilde{\mathbf{e}}'_j\}_j, \{\mathbf{g}_j - \alpha\widetilde{\mathbf{e}}'_j\}_j, \widetilde{\mathbf{m}}, \mathbf{h} - \alpha\widetilde{\mathbf{m}}), c_2, \widetilde{d})$
		$\widetilde{\mathbf{e}}'_j \stackrel{?}{\in} \mathcal{B}_{nk}, \quad \widetilde{\mathbf{m}} \stackrel{?}{\in} \mathcal{B}_1$

Protocol 11: ZKPoK for the relation $\mathcal{R}_{\mathsf{EqTo}\{0,1\}}$.

Zero-Knowledge:

Case $b = 0$:

$$\hat{s} \xleftarrow{\$} R_q, \quad \hat{\mathbf{g}}_j \xleftarrow{\$} \mathbb{Z}_q^{2nk}, \quad \hat{\mathbf{h}} \xleftarrow{\$} \mathbb{Z}_q^2$$
$$\hat{\pi}_j \xleftarrow{\$} \mathcal{S}_{2nk}, \quad \hat{\sigma} \xleftarrow{\$} \mathcal{S}_2$$
$$c_1 = \mathsf{aCom}(\hat{\sigma}, \mathbf{a}(\hat{\sigma}^{-1}(\hat{\mathbf{h}}))_1 + \mathbf{b}\hat{s}$$
$$+\phi(\mathbf{l}'\Sigma_j 2^j \hat{\pi}_j^{-1}(\hat{\mathbf{g}}_j)) - \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})), \{\hat{\pi}_j\}_j)$$

$\mathcal{P}$ reveals $\{\hat{\mathbf{g}}_j\}_j, \hat{\mathbf{h}}, \{\widetilde{\pi}_j = \hat{\pi}_j\}_j, \{\widetilde{\sigma} = \hat{\sigma}\}_j, \widetilde{\mathbf{y}} = \mathbf{a}(\hat{\sigma}^{-1}(\hat{\mathbf{h}}))_1 + \mathbf{b}\hat{s} + \phi(\mathbf{l}'\Sigma_j 2^j \hat{\pi}_j^{-1}(\hat{\mathbf{g}}_j))$ $-\alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk})), \widetilde{s} = \hat{s}$. Indistinguishable from a real conversation with the same $\pi_j = \widehat{\pi}_j$, $\sigma = \widehat{\sigma}$ and where $\boldsymbol{\mu} = \widehat{\sigma}^{-1}(\widehat{\mathbf{h}}) - \alpha(m, 1-m)$, $\rho = \widehat{s} - \alpha r$ and $\mathbf{f}_j = \widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j) - \alpha \mathbf{e}'_j$.

$$\begin{aligned}\mathbf{g}_j &= \pi_j(\mathbf{f}_j + \alpha \mathbf{e}'_j) \\ &= \pi_j(\widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j)) + \pi_j(\alpha \mathbf{e}'_j - \alpha \mathbf{e}'_j) \\ &= \widehat{\mathbf{g}}_j\end{aligned}$$

$$\begin{aligned}\mathbf{h} &= \sigma(\boldsymbol{\mu} + \alpha(m, 1-m))) \\ &= \sigma(\widehat{\sigma}^{-1}(\widehat{\mathbf{h}})) \\ &= \widehat{\mathbf{h}}\end{aligned}$$

$$\begin{aligned}&\mathbf{a}\mu_1 + \mathbf{b}\rho + \phi(\mathbf{l}\Sigma_j 2^j \mathbf{f}_j) = \\ &= \mathbf{a}(\widehat{\sigma}^{-1}(\widehat{\mathbf{h}}) - \alpha(m, 1-m))_1 \\ &\quad + \mathbf{b}(\widehat{s} - \alpha r) \\ &\quad + \phi(\mathbf{l}\Sigma_j 2^j (\widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j) - \alpha \mathbf{e}'_j)) \\ &= \mathbf{a}(\widehat{\sigma}^{-1}(\widehat{\mathbf{h}}))_1 + \mathbf{b}\widehat{s} + \phi(\mathbf{l}\Sigma_j 2^j \widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j)) \\ &\quad - \alpha(\mathbf{a}m + \mathbf{b}r + \phi(\mathbf{l}\Sigma_j 2^j \mathbf{e}'_j)) \\ &= \mathbf{a}(\widehat{\sigma}^{-1}(\widehat{\mathbf{h}})_1 + \mathbf{b}\widehat{s} + \phi(\mathbf{l}\Sigma_j 2^j \widehat{\pi}_j^{-1}(\widehat{\mathbf{g}}_j)) \\ &\quad - \alpha(\mathbf{c} + \phi(2^\kappa \mathbf{1}_{nk}))\end{aligned}$$

Case $b = 1$:

$$\hat{\mathbf{e}}'_j \xleftarrow{\$} \mathcal{B}_{nk}, \quad \hat{\mathbf{m}} \xleftarrow{\$} \mathcal{B}_1, \hat{\mathbf{f}}_j \xleftarrow{\$} \mathbb{Z}_q^{2nk}$$
$$\hat{\mu} \xleftarrow{\$} \mathbb{Z}_q^2, \quad \hat{\pi}_j \xleftarrow{\$} \mathcal{S}_{2nk}, \quad \hat{\sigma} \xleftarrow{\$} \mathcal{S}_2$$
$$c_2 = \mathsf{aCom}(\hat{\sigma}(\hat{\boldsymbol{\mu}}), \hat{\sigma}(\hat{\mathbf{m}}), \{\hat{\pi}_j(\hat{\mathbf{e}}'_j)\}_j, \{\hat{\pi}_j(\hat{\mathbf{f}}_j)\}_j)$$
$$\hat{\mathbf{g}}_j = \hat{\pi}_j(\hat{\mathbf{f}}_j + \alpha \hat{\mathbf{e}}'_j), \quad \hat{\mathbf{h}} = \hat{\sigma}(\hat{\boldsymbol{\mu}} + \alpha \hat{\mathbf{m}})$$

$\mathcal{P}$ reveals $\{\widehat{\mathbf{g}}_j\}_j$, $\widehat{\mathbf{h}}$, $\{\widetilde{\mathbf{e}}'_j = \widehat{\pi}_j(\widehat{\mathbf{e}}'_j)\}_j$, $\widetilde{\mathbf{m}} = \widehat{\sigma}(\widehat{\mathbf{m}})$. Equivalent to an honest conversation were π_j is such that $\pi_j(\mathbf{e}'_j) = \widehat{\pi}_j(\widehat{\mathbf{e}}'_j)$, $\mathbf{f}_j$ is such that $\mathbf{f}_j = \pi_j^{-1}(\widehat{\pi}_j(\widehat{\mathbf{f}}_j))$, σ is such that $\sigma((m, 1-m)) = \widehat{\sigma}(\widehat{\mathbf{m}})$ and $\boldsymbol{\mu}$ is such that $\boldsymbol{\mu} = \sigma^{-1}(\widehat{\sigma}(\widehat{\boldsymbol{\mu}}))$.

$$\begin{aligned}\mathbf{g}_j &= \pi_j(\mathbf{f}_j + \alpha \mathbf{e}'_j) \\ &= \pi_j(\pi_j^{-1}(\widehat{\pi}_j(\widehat{\mathbf{f}}_j))) + \alpha \widehat{\pi}_j(\widehat{\mathbf{e}}'_j) \\ &= \widehat{\pi}_j(\widehat{\mathbf{f}}_j + \alpha \widehat{\mathbf{e}}'_j) \\ &= \widehat{\mathbf{g}}_j\end{aligned}$$

$$\begin{aligned}\mathbf{h} &= \sigma(\boldsymbol{\mu} + \alpha(m, 1-m)) \\ &= \sigma(\sigma^{-1}(\widehat{\sigma}(\widehat{\boldsymbol{\mu}}))) + \alpha \widehat{\sigma}(\widehat{\mathbf{m}}) \\ &= \widehat{\sigma}(\widehat{\boldsymbol{\mu}} + \alpha \widehat{\mathbf{m}})\end{aligned}$$

Notice that in both cases simulated conversations follow the same distribution as honest conversations.

All omitted commitments are computed as commitments to 0, and are indistinguishable from honestly computed commitments by the hiding property of the auxiliary commitment scheme. ■

Remark 5.3. The ZKPoK provided in Theorem 5.1 and 5.2 can be composed in parallel to obtain a protocol for larger relations, e.g., provided with commitments $(c_i)_{i=1}^4$ of the values $(m_i)_{i=1}^4$ satisfying $m_4 = \sum_{i=1}^3 m_i$, we can prove this relation by creating one extra auxiliary commitment c_{aux} for $m_{\mathsf{aux}} = m_1 + m_2$ and running two Π_{Add} in parallel for the statement pairs $(\mathsf{pk}, c_1, c_2, c_{\mathsf{aux}})$ and $(\mathsf{pk}, c_{aux}, c_3, c_4)$.

5.2 ZKPoK for the LABS relation

To instantiate the generic LABS construction in Section 4 from lattices, it is sufficient to prove that the digital signature scheme from Section 2.4.6 and the commitment scheme from Section 2.4.5 are equipped with a ZKPoK for the relation $\mathcal{R}_{\mathsf{LABS}}$ and a ZKPoK proving possession of hash inputs.

The ZKPoK π is obtained by running the protocol by Jawurek et al. [JKO13], where they provide an efficient zero-knowledge proof for generic languages. In particular, they are able to construct a protocol that is able to prove (in zero-knowledge) efficiently statements such as "I know x such that $y = \text{SHA-256}(x)$" for a common input y. To this end, they make use of *garbled circuits*, a tool that enables to solve the problem of two-party computation (2PC). In the problem of 2PC (introduced by Yao in [Yao82]), two parties hold secret inputs x_1 and x_2 respectively and want to jointly compute some function $z = f(x_1, x_2)$ while keeping their inputs secret. Note that zero-knowledge proofs are a proper subset of 2PC: ZKPoK are instances of 2PC where there is an asymmetry between the parties and only the prover $\mathcal{P}$ holds an input w (the witness) and where the function $f(w)$ outputs accept iff w is a valid witness for x (here, x is a common input to both $\mathcal{P}$ and $\mathcal{V}$).

From a high-level point of view, Jawurek et al. [JKO13] protocols works as follows: the verifier $\mathcal{V}$ and the prover $\mathcal{P}$ run Yao's protocol, where $\mathcal{V}$ acts as the circuit constructor and $\mathcal{P}$ acts as the circuit evaluator. After the evaluation, $\mathcal{P}$ sends to $\mathcal{V}$ the output key. The security of Yao's protocol implies that a computationally bounded prover $\mathcal{P}$ can not guess the output key unless she has a valid witness for y.

Therefore, to instantiate our LABS we only remain to provide a ZKPoK for the relation $\mathcal{R}_{\mathsf{LABS}}$.

Below we aim at construction a ZKPoK for proving Eq. (2-4) in our LABS construction, where the attribute $\mathbf{x}$ and Boyen's variant signatures σ are commited using the commitment scheme of [MM19].

Taking Remark 5.3 into consideration, a ZKPoK for proving Eq. (3) and (4), which are essentially proving that the circuit is computed correctly, can be constructed by simply composing the ZKPoK $\Pi_{\mathsf{EqTo}\star}, \Pi_{\mathsf{Add}}, \Pi_{\mathsf{Mult}}$ in parallel. Specifically, we use $\Pi_{\mathsf{Add}}, \Pi_{\mathsf{Mult}}$ to prove that we computed each gate correctly, and use $\Pi_{\mathsf{EqTo}\star}$ to prove that the value associated to the output wire is equal to 1. Then, we only need to focus on how to construct a ZKPoK for proving Eq. (2), i.e., proving possession of a valid Boyen's variant signature.

To summarize, our goal is to construct a ZKPoK for proving possesion of a valid Boyen's variant signature (described in Section 2.4.6) $\sigma = \mathbf{s} = (s_1, \dots, s_{\bar{m}+k}) \in R_q^{\bar{m}+k}$, where $\mathbf{x} = (x_1, \dots, x_\ell) \in \{0,1\}^\ell$ is viewed as the message, provided the verification key $\mathsf{vk}_{\mathsf{Sign}}$ and the commitments to the signature σ and message $\mathbf{x}$. Then, since the protocols in Theorem 5.1 and Theorem 5.2 allows for parallel composition, our desired ZKPoK for the relation $\mathcal{R}_{\mathsf{LABS}}$ is obtained by composing the protocol for the Boyen's variant signature with the protocols for Eq. (3) and (4) together. Below, we assume the commitment $\mathbf{c}_\sigma$ of the signature is provided in the form $(\bar{\mathbf{c}}_h)_{h\in[\bar{m}+k]}$ where each $\bar{\mathbf{c}}_h$ is a commitment of the h-th element of $\mathbf{s}$, and the commitment $\mathbf{c}_\mathbf{x}$ of the message is provided in the form $(\mathbf{c}_i)_{i\in[\ell]}$ where each $\mathbf{c}_i$ is a commitment of the value $x_i \in \{0,1\}$.

Now, due to the verification algorithm of the Boyen's variant signature scheme, proving a signature is valid is equivalent to prove that the following three statements hold:

$$\mathbf{x} \in \{0,1\}^\ell \iff x_i \in \{0,1\} \text{ for } i \in [\ell], \tag{5}$$

$$\|\mathbf{s}\|_\infty \leq 8s_\sigma \iff \|s_j\|_\infty \leq 8s_\sigma \text{ for } j \in [\bar{m}+k], \tag{6}$$

$$\left[\mathbf{a} \mid \mathbf{b} \mid \mathbf{a}_0 + \sum_{i=1}^{\ell} (-1)^{x_i} \mathbf{a}_i\right] \mathbf{s} \equiv u \pmod{q}. \tag{7}$$

Below we construct ZKPoK respectively for the above equations by converting each of them into an arithmetic circuit, and using the ZKPoK provided in Theorem 5.1 and Theorem 5.2 as building blocks to prove the satisfiability of each circuit.

ZKPoK for proving Eq. (5). It is sufficient to prove that for every $i \in [\ell]$, the commitment $\mathbf{c}_i \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, x_i)$ opens to either 0 or 1. To that end, it is enough by using the ZKPoK $\Pi_{\mathsf{EqTo}\{0,1\}}$, as the functionality of the protocol $\Pi_{\mathsf{EqTo}\{0,1\}}$ is precisely the objective of Eq. (5).

ZKPoK for proving Eq. (6). Here, for simplicity of the protocol, we assume that $8s_\sigma$ can be written as $2^\zeta - 1$ for some positive integer ζ. Equivalently, $\zeta = \log(8s_\sigma + 1)$.

First, we prepare some notations. For $j \in [\bar{m} + k]$, let $\mathbf{s}_{j,h} \in \{0,1\}^n$ be the h-th vector of bits of the binary decomposition of $s_j \in R_q$ for $h \in [\zeta]$ and let $s^i_{j,h} \in \{0,1\}$ denote the i-th component of $\mathbf{s}_{j,h}$ for $i \in [n]$. Further, we set $\mathbf{w}_{j,h} = 2^{h-1}\mathbf{s}_{j,h}$ for $h \in [\zeta]$ and $\mathbf{w}_{j,[h']} = \sum_{h=1}^{h'} \mathbf{w}_{j,h}$ for $h' \in [\zeta]$, where $\mathbf{w}_{j,[1]} = \mathbf{w}_{j,1}$.

Next, we create the following auxiliary commitments for $j \in [\bar{m} + k]$:

$$\mathbf{c}_{\mathsf{zero}} \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, 0), \quad \mathbf{c}_{\mathsf{coeff},h} \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, 2^{h-1}), \quad \bar{\mathbf{c}}^i_{j,h} \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, s^i_{j,h})$$
$$\bar{\mathbf{c}}_{j,h} \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, \varphi(\mathbf{s}_{j,h})), \quad \mathbf{a}_{j,h} \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, \varphi(\mathbf{w}_{j,h})), \quad \mathbf{a}_{j,[h']} \xleftarrow{\$} \mathsf{C.Com}(\mathsf{pk}, \varphi(\mathbf{w}_{j,[h']}))$$

for $h \in [\zeta]$ and $h' \in [\zeta]$.

Then, using the commitments $(\bar{\mathbf{c}}_h)_{h \in [\bar{m}+k]}$, the auxiliary commitments and composing the ZKPoK $\Pi_{\mathsf{EqTo}\star}, \Pi_{\mathsf{EqTo}\{0,1\}}, \Pi_{\mathsf{Add}}$ and Π_{Mult} together, we construct a ZKPoK for the following statements for all $j \in [\bar{m} + k]$, $h \in [\zeta]$ and $h' \in [\zeta]$:

$$\mathbf{c}_{\mathsf{zero}} \text{ opens to } 0 \ \wedge \ \mathbf{c}_{\mathsf{coeff},h} \text{ opens to } 2^{h-1} \ \wedge \ \bar{\mathbf{c}}^i_{j,h} \text{ opens to 0 or 1} \ \wedge \ \mathbf{s}_{j,h} = (s^1_{j,h}, \ldots, s^n_{j,h}) \ \wedge$$
$$\mathbf{w}_{j,h} = 2^{h-1} \cdot \mathbf{s}_{j,h} \ \wedge \ \mathbf{w}_{j,[h']} = \mathbf{w}_{j,h'} + \mathbf{w}_{j,[h'-1]} \ \wedge \ \mathbf{0} = \varphi^{-1}(s_j) - \mathbf{w}_{j,[\zeta]}.$$

We check that the above statement is equivalent to Eq. (6). The above statements proves that $\mathbf{s}_{j,h} \in \{0,1\}^n$. Furthermore, when $\mathbf{s}_{j,h} \in \{0,1\}^n$, we have $\|s_j\|_\infty \le \sum_{h=1}^{\zeta} 2^{h-1} \|\mathbf{s}_{j,h}\|_\infty \le \sum_{h=1}^{\zeta} 2^{h-1} = 2^\zeta - 1 = 8s_\sigma$. Therefore, if the above statement holds, then we must have $\|s_j\|_\infty \le 8s_\sigma$ for all $j \in [\bar{m} + k]$.

ZKPoK for proving Eq. (7). We first prepare some notations. Let a_j (resp., $a_{i,j}$) denote the j-th (resp., $(j-m)$-th) entry of $\mathbf{a}$ (resp. $\mathbf{a}_i$), for $j \in [m]$ (resp. $j \in [m+1, \bar{m}]$) and $i \in [0, \ell]$. Similarly, let b_j denote the $(j - \bar{m})$-th entry of $\mathbf{b}$, for $j \in [\bar{m}+1, \bar{m}+k]$. Then, observe that we can rewrite Eq. (7) using the following equations:

$$\sum_{j_1=1}^{m} a_{j_1} \cdot s_{j_1} + \sum_{j_2=m+1}^{\bar{m}} b_{j_2} \cdot s_{j_2} + \sum_{j_3=\bar{m}+1}^{\bar{m}+k} \left(a_{0,j_3} + \sum_{i=1}^{\ell} (-1)^{x_i} \cdot a_{i,j_3}\right) \cdot s_{j_3} = u. \tag{8}$$

Next, we prepare some auxiliary values in order to prove the above equation composing the ZKPoK $\Pi_{\mathsf{EqTo}\star}, \Pi_{\mathsf{EqTo}\{0,1\}}, \Pi_{\mathsf{Add}}$ and Π_{Mult} together:

Firstly, we prepare values for the third component of the sum in Eq. 8:

$$\begin{aligned} y_i &= (-1)^{x_i}, \\ w_{i,j_3} &= y_i \cdot a_{i,j_3}, \\ w_{[i'],j_3} &= \Sigma_{i=1}^{i'} w_{i,j_3}, \ \text{(where } w_{[1],j_3} = w_{1,j_3}\text{), and} \\ a_{j_3} &= a_{0,j_3} + w_{[\ell],j_3}, \end{aligned}$$

for $i, i' \in [\ell], j_3 \in [\bar{m}+1, \bar{m}+k]$.

Secondly, we combine them all by forming the values:

$$\begin{aligned} v_j &= a_j \cdot s_j, \ \text{ for } j \in [m] \cup [\bar{m}+1, \bar{m}+k], \\ v_j &= b_j \cdot s_j, \ \text{ for } j \in [m+1, \bar{m}], \\ v_{[j']} &= \Sigma_{j_1=1}^{j'} v_{j_1}, \ \text{ for } j' \in [m], \ \text{(where } v_{[1]} = v_1\text{)}, \\ v_{[j']} &= \Sigma_{j_2=m+1}^{j'} v_{j_2}, \ \text{ for } j' \in [m+1, \bar{m}], \ \text{(where } v_{[m+1]} = v_{m+1}\text{), and} \\ v_{[j']} &= \Sigma_{j_3=\bar{m}+1}^{j'} v_{j_3}, \ \text{ for } j' \in [\bar{m}+1, \bar{m}+k], \ \text{(where } v_{[\bar{m}+1]} = v_{\bar{m}+1}\text{).} \end{aligned}$$

Finally, we sum up them all by forming $t = v_{[m]} + v_{[\bar{m}]} + v_{[\bar{m}+k]}$.

Next, we create auxiliary commitments for the related values:

$$\mathbf{c}_{\mathsf{mat},j_1} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, a_{j_1}), \quad \mathbf{c}_{\mathsf{mat},j_2} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, b_{j_2}), \quad \mathbf{c}_{\mathsf{mat},i,j_3} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, a_{i,j_3}),$$

for $i \in [0, \ell], j_1 \in [m], j_2 \in [m+1, \bar{m}]$ and $j_3 \in [\bar{m}+1, \bar{m}+k]$.

$$\boldsymbol{\omega}_{i,j_3} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, w_{i,j_3}), \quad \boldsymbol{\omega}_{[i'],j_3} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, w_{[i'],j_3}), \quad \boldsymbol{\alpha}_{j_3} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, a_{j_3}),$$

for $i \in [\ell], i' \in [2, \ell]$ and $j_3 \in [\bar{m}+1, \bar{m}+k]$.

$$\boldsymbol{\psi}_i \longleftarrow \mathsf{C.Com}(\mathsf{pk}, y_i), \quad , \boldsymbol{\nu}_j \longleftarrow \mathsf{C.Com}(\mathsf{pk}, v_j), \quad \boldsymbol{\nu}_{[j']} \longleftarrow \mathsf{C.Com}(\mathsf{pk}, v_{[j']}),$$

for $i \in [\ell], j \in [\bar{m}+k]$ and $j' \in [2, m] \cup [m+2, \bar{m}] \cup [\bar{m}+2, \bar{m}+k]$.

Then, using the commitments $(\mathbf{c}_i)_{i=1}^{\ell}, (\bar{\mathbf{c}}_h)_{h \in [\bar{m}+k]}$, the auxiliary commitments and composing the ZKPoK $\Pi_{\mathsf{EqTo}\star}, \Pi_{\mathsf{EqTo}\{0,1\}}, \Pi_{\mathsf{Add}}$ and Π_{Mult} together, we construct a ZKPoK for the following statement for all , $i \in [\ell], i' \in [2, \ell], j_1 \in [m], j_2 \in [m+1, \bar{m}], j_3 \in [\bar{m}+1, \bar{m}+k], j \in [\bar{m}+k], j' \in [2, m] \cup [m+2, \bar{m}] \cup [\bar{m}+2, \bar{m}+k]$:

$$\begin{gathered} \mathbf{c}_{\mathsf{zero}} \text{ opens to } 0 \ \wedge \ \mathbf{c}_{\mathsf{mat},j_1}, \mathbf{c}_{\mathsf{mat},j_2}, \mathbf{c}_{\mathsf{mat},0,j_3}, \mathbf{c}_{\mathsf{mat},i,j_3} \text{ opens to } a_{s,j}, b_{s,j}, a_{0,j}, a_{i,j} \ \wedge \\ y_i = (-2) \cdot x_i + 1 \ \wedge \ w_{i,j_3} = y_i \cdot a_{i,j_3} \ \wedge \ w_{[i'],j_3} = w_{i',j_3} + w_{[i'-1],j_3} \ \wedge \ a_{j_3} = a_{0,j_3} + w_{[\ell],j_3} \ \wedge \\ v_j = a_j \cdot s_j \ \wedge \ v_{[j']} = v_{j'} + v_{[j'-1]} \ \wedge \ t = v_{[m]} + v_{[\bar{m}]} + v_{[\bar{m}+k]} \ \wedge \ 0 = u - t \end{gathered}$$

The above statement can be checked that is equivalent to proving Eq. (8).

ZKPoK for $\mathcal{R}_{\mathsf{LABS}}$. To summarize, we obtain a ZKPoK for proving possession of a valid Boyen's variant signature by composing the ZKPoK for proving Eq. (5 - 7) together. Then, by composing this ZKPoK with the aforementioned ZKPoK for proving Eq. (3) and (4), we obtain our desired ZKPoK for the relation $\mathcal{R}_{\mathsf{LABS}}$. Thus, we obtain our lattice-based LABS scheme for unbounded circuits by instantiating the generic LABS construction in Section 4 with our ZKPoK for $\mathcal{R}_{\mathsf{LABS}}$.

6. Conclusions

Quantum computers may become a technological reality, so it is important to start basing the security of our communications (in terms of privacy, efficiency and so on) on post-quantum assumptions. Inside post-quantum cryptography, lattice-based cryptography is one of the most promising areas of research in cryptography.

In this thesis we have presented a new protocol useful to build an electronic voting scheme. This protocol allows election administrators to solve both problem of distinguishing if vote signatures have been issued by the same voter, and the problem of distinguishing if vote signatures are emitted by different voters, while at the same time preserving the anonymity of the voters.

We have equipped a standard attribute-based signature scheme with an additional linking algorithm and the definition of the linkability and exculpability properties that all together would solve the above problems. We also have defined two security notions, namely internal and external privacy, that fulfills the objective of electronic voting.

Moreover, we have constructed a general LABS schemes supporting unbounded circuits as polices. To do so, first we have associated a tag (i.e., the output of a hash function) to each voter together with a ZKPoK proving possession of a preimage for this hash function, and secondly we have proved that this general construction satisfies both the properties and the security notions mentioned in the previous paragraph.

Last but not least, we have provided an instantiation of our generic LABS for unbounded circuits from lattices in the random oracle model. We have created two ZKPoK based on the ones by Martínez and Morillo [MM19], and we have combined all these zero-knowledge proofs to prove in zero-knowledge that the Boyen's variant signature of the attribute is valid and the corresponding secret attribute satisfies the public signing policy.

This thesis intends to be a small contribution to the research area of lattice-based cryptography, and even if it has been developed for several years, this is still a very recent area of research for which there is still a lot of work to do.

References

[BBD08] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post Quantum Cryptography*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 3540887016.

[Ben+14] Fabrice Benhamouda et al. *Efficient Zero-Knowledge Proofs for Commitments from Learning With Errors over Rings*. Cryptology ePrint Archive, Report 2014/889. https://eprint.iacr.org/2014/889. 2014.

[BLO18] Carsten Baum, Huang Lin, and Sabine Oechsner. "Towards practical lattice-based one-time linkable ring signatures". In: *Information and communications security*. Vol. 11149. Lecture Notes in Comput. Sci. Springer, Cham, 2018, pp. 303–322. DOI: 10.1007/978-3-030-01950-1_1. URL: https://doi.org/10.1007/978-3-030-01950-1_1.

[Bos+20] Cecilia Boschini et al. "Efficient Post-quantum SNARKs for RSIS and RLWE and Their Applications to Privacy". In: *Post-Quantum Cryptography*. Ed. by Jintai Ding and Jean-Pierre Tillich. Cham: Springer International Publishing, 2020, pp. 247–267. ISBN: 978-3-030-44223-1.

[Boy07] Xavier Boyen. "Mesh signatures: how to leak a secret with unwitting and unwilling participants". In: *Advances in cryptology—EUROCRYPT 2007*. Vol. 4515. Lecture Notes in Comput. Sci. Springer, Berlin, 2007, pp. 210–227. DOI: 10.1007/978-3-540-72540-4_12. URL: https://doi.org/10.1007/978-3-540-72540-4_12.

[Boy10] Xavier Boyen. "Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more". In: *Public key cryptography—PKC 2010*. Vol. 6056. Lecture Notes in Comput. Sci. Springer, Berlin, 2010, pp. 499–517. DOI: 10.1007/978-3-642-13013-7_29. URL: https://doi.org/10.1007/978-3-642-13013-7_29.

[CH91] David Chaum and Eugène van Heyst. "Group Signatures". In: *Advances in Cryptology — EUROCRYPT '91*. Ed. by Donald W. Davies. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 257–265. ISBN: 978-3-540-46416-7.

[EKK18] Ali El Kaafarani and Shuichi Katsumata. "Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices". In: *Public-key cryptography—PKC 2018. Part II*. Vol. 10770. Lecture Notes in Comput. Sci. Springer, Cham, 2018, pp. 89–119. DOI: 10.1007/978-3-319-76581-5_4. URL: https://doi.org/10.1007/978-3-319-76581-5_4.

[FS87] Amos Fiat and Adi Shamir. "How to prove yourself: practical solutions to identification and signature problems". In: *Advances in cryptology—CRYPTO '86 (Santa Barbara, Calif., 1986)*. Vol. 263. Lecture Notes in Comput. Sci. Springer, Berlin, 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12. URL: https://doi.org/10.1007/3-540-47721-7_12.

[GM18] Nicholas Genise and Daniele Micciancio. "Faster Gaussian sampling for trapdoor lattices with arbitrary modulus". In: *Advances in cryptology—EUROCRYPT 2018. Part I*. Vol. 10820. Lecture Notes in Comput. Sci. Springer, Cham, 2018, pp. 174–203. DOI: 10.1007/978-3-319-78381-9_7. URL: https://doi.org/10.1007/978-3-319-78381-9_7.

[Gro96] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph].

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. *Zero-Knowledge Using Garbled Circuits: How To Prove Non-Algebraic Statements Efficiently*. Cryptology ePrint Archive, Report 2013/073. 2013. URL: https://eprint.iacr.org/2013/073.

[Kno19] Heiko Knospe. *A course in cryptography*. Vol. 40. Pure and Applied Undergraduate Texts. American Mathematical Society, Providence, RI, 2019, pp. xvii+323. ISBN: 978-1-4704-5055-7.

[LLL82] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. "Factoring polynomials with rational coefficients". In: *Math. Ann.* 261.4 (1982), pp. 515–534. ISSN: 0025-5831. DOI: 10.1007/BF01457454. URL: https://doi.org/10.1007/BF01457454.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. "Generalized compact knapsacks are collision resistant". In: *Automata, languages and programming. Part II*. Vol. 4052. Lecture Notes in Comput. Sci. Springer, Berlin, 2006, pp. 144–155. DOI: 10.1007/11787006_13. URL: https://doi.org/10.1007/11787006_13.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *J. ACM* 60.6 (2013), Art. 43, 35. ISSN: 0004-5411. DOI: 10.1145/2535925. URL: https://doi.org/10.1145/2535925.

[Mic07] Daniele Micciancio. "Generalized compact knapsacks, cyclic lattices, and efficient one-way functions". In: *Comput. Complexity* 16.4 (2007), pp. 365–411. ISSN: 1016-3328. DOI: 10.1007/s00037-007-0234-9. URL: https://doi.org/10.1007/s00037-007-0234-9.

[MM19] Ramiro Martínez and Paz Morillo. *RLWE-based Zero-Knowledge Proofs for linear and multiplicative relations*. Cryptology ePrint Archive, Report 2019/1486. https://eprint.iacr.org/2019/1486. 2019.

[MOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. With a foreword by Ronald L. Rivest. CRC Press, Boca Raton, FL, 1997, pp. xxviii+780. ISBN: 0-8493-8523-7.

[MP12] Daniele Micciancio and Chris Peikert. "Trapdoors for lattices: simpler, tighter, faster, smaller". In: *Advances in cryptology—EUROCRYPT 2012*. Vol. 7237. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2012, pp. 700–718. DOI: 10.1007/978-3-642-29011-4_41. URL: https://doi.org/10.1007/978-3-642-29011-4_41.

[MP18] Ramiro Martínez Pinilla. "Fully post-quantum protocols for e-voting, coercion resistant cast as intended and mixing networks". PhD thesis. Jan. 2018. DOI: 10.13140/RG.2.2.35061.27363.

[MPR11] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. "Attribute-based signatures". In: *Topics in cryptology—CT-RSA 2011*. Vol. 6558. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, pp. 376–392. DOI: 10.1007/978-3-642-19074-2_24. URL: https://doi.org/10.1007/978-3-642-19074-2_24.

[MR07] Daniele Micciancio and Oded Regev. "Worst-case to average-case reductions based on Gaussian measures". In: *SIAM J. Comput.* 37.1 (2007), pp. 267–302. ISSN: 0097-5397. DOI: 10.1137/S0097539705447360. URL: https://doi.org/10.1137/S0097539705447360.

[MR09] Daniele Micciancio and Oded Regev. "Lattice-based cryptography". In: *Post-quantum cryptography*. Springer, Berlin, 2009, pp. 147–191. DOI: 10.1007/978-3-540-88702-7_5. URL: https://doi.org/10.1007/978-3-540-88702-7_5.

[Pei14] Chris Peikert. "A decade of lattice cryptography". In: *Found. Trends Theor. Comput. Sci.* 10.4 (2014), pp. i–iii, 283–424. ISSN: 1551-305X. DOI: 10.1561/0400000074. URL: https://doi.org/10.1561/0400000074.

[PR06] Chris Peikert and Alon Rosen. "Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices". In: *Theory of cryptography*. Vol. 3876. Lecture Notes in Comput. Sci. Springer, Berlin, 2006, pp. 145–166. DOI: 10.1007/11681878_8. URL: https://doi.org/10.1007/11681878_8.

[PZ03] John Proos and Christof Zalka. *Shor's discrete logarithm quantum algorithm for elliptic curves*. 2003. arXiv: quant-ph/0301141 [quant-ph].

[Qui+90] Jean-Jacques Quisquater et al. "How to Explain Zero-Knowledge Protocols to Your Children". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Ed. by Gilles Brassard. New York, NY: Springer New York, 1990, pp. 628–631. ISBN: 978-0-387-34805-6.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to leak a secret". In: *Advances in cryptology—ASIACRYPT 2001 (Gold Coast)*. Vol. 2248. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, pp. 552–565. DOI: 10.1007/3-540-45682-1_32. URL: https://doi.org/10.1007/3-540-45682-1_32.

[Sho99] Peter W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM Rev.* 41.2 (1999), pp. 303–332. ISSN: 0036-1445. DOI: 10.1137/S0036144598347011. URL: https://doi.org/10.1137/S0036144598347011.

[ST01] Adi Shamir and Yael Tauman. "Improved online/offline signature schemes". In: *Advances in cryptology—CRYPTO 2001 (Santa Barbara, CA)*. Vol. 2139. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, pp. 355–367. DOI: 10.1007/3-540-44647-8_21. URL: https://doi.org/10.1007/3-540-44647-8_21.

[Yao82] Andrew C. Yao. "Protocols for secure computations". In: *23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*. IEEE, New York, 1982, pp. 160–164.