UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Long-term privacy in electronic voting systems

Núria Costa Mirada

Long-term privacy in electronic voting systems

Núria Costa Mirada

Supervisor: Dra. Paz Morillo Bosch

March 2021

Contents

List of Figures

Agraïments

Aquests agraïments només poden començar d'una manera i és donant les gràcies a la Paz. Gràcies per tot el suport, la paciència i l'acompanyament, per totes les hores de feina (i no feina) compartides no només durant aquests anys de doctorat sinó des que ens vam conèixer durant el meu projecte final de carrera. Però, sobretot, gràcies per la passió que poses en tot allò que fas.

També vull donar les gràcies al Jordi per confiar en mi fa sis anys per formar part del seu equip de recerca i seguir-hi confiant a dia d'avui. Gràcies també a tots els meus companys, tant als que he trobat a Scytl com a la UPC, perquè he tingut i tinc la sort de treballar amb gent de molt talent i d'aprendre'n molt de tots ells.

Un gràcies especial també per a en Ramiro, que ha estat clau en tot aquest procés. I gràcies als meus dos revisors externs, Vanessa Daza i Enrique Larraia, pels seus comentaris que han contribuït a millorar la qualitat de la tesi.

Finalment, gràcies a la meva família, a tota. Al meu pare i a la meva mare, per ensenyar-me que tot el que val la pena en aquesta vida requereix d'un esforç. A l'Anna, per entendre'm millor que ningú. Al Lluís, per creure sempre en mi i no deixar que mai em rendeixi. I a tu, Júlia, per donar-me l'última empenta que necessitava.

Preface

This thesis is the result of an industrial PhD done at Scytl in close collaboration with Dr. Paz Morillo, from the Department of Applied Mathematics at UPC and Ramiro Martínez, PhD student. The main objective of this kind of PhD is to do an applied research, by analyzing the needs of the company and proposing solutions.

As a member of the Research and Security team at Scytl, the author of this thesis has participated in the design of several electronic voting systems as well as in their implementation, by providing support to the development team. This work has allowed her to obtain an in-depth knowledge of the electronic voting field and to learn which are the existing solutions for satisfying both the customer requirements and those established by the security guidelines. An important part of her work consists also on thinking how to improve current voting systems based on the market needs but also on new security recommendations given by the experts.

One of the main concerns nowadays is how to be prepared for the appearance of quantum computers and the risk they suppose for the long-term privacy of the online voting systems. Currently, Scytl's technology ensures the privacy of voters in front of attacks done by classical computers but it will not ensure privacy in the future if a quantum computer is used to perform the same attack. Hence, from here the following question arises: is it possible to build a quantum-safe online voting system which provides long-term privacy? With the aim of giving a positive answer to this question this research started.

This thesis consists on a first important part which is the research done on the basics of post-quantum cryptography and, more concretely, on lattice-based cryptography. Since Scytl was not working on post-quantum cryptography when this work started and the author has not any experience on this field, this has been a mandatory step before being able to contribute to the state of the art of lattice-based crytographic primitives. These contributions are essential building blocks of the online voting system presented as part of this thesis and allow to provide privacy even in the presence of a quantum adversary.

It is worth to say that the research done for this PhD has allowed Scytl to participate in the European Union PROMETHEUS project which aims to provide post-quantum signature schemes, encryption schemes and privacy-preserving protocols relying on lattice. In this context, the implementation of a post-quantum online voting system which is mostly based on that presented in this thesis, is already ongoing.

Chapter 1

Introduction

Electronic voting (e-voting) is defined by the Council of Europe as the use of information and communication technology (ICT) to cast and/or count the votes [3, 60]. There are different types of e-voting systems depending on the environment where they are conducted. If it is a controlled environment, such as the polling station, the casting of the vote is done in a place supervised by the election administration. An example of this type of e-voting is the usage of Direct Recording Electronic (DRE) voting machines. On the other hand, in an uncontrolled environment the voting devices cannot be supervised by the election administration, voters cast their votes using personal devices such as mobile phones and the vote is transferred through the Internet to a central voting server. These systems are known as online voting systems although they are also called remote electronic voting systems or internet voting systems. E-voting in a controlled environment can be seen as the electronic equivalent of traditional paper-based voting and in an uncontrolled environment as the equivalent of postal voting. While there is a number of countries that have only experimented with e-voting, there are some others that are using it for binding elections or referendum since long time ago [1, 18]. In 2000, the United States was the first one to use online voting for a binding election, followed by the UK in 2002 for local government elections, Canada, France and Switzerland in 2003 and the Netherlands in 2004. In 2005 Estonia was the first country in the world to hold a nation-wide election for the entire electorate and in 2008 the Swiss Canton of Neuchâtel used an online voting trial for the citizens living abroad (although Geneva was the first offering online voting in Switzerland in 2003). Switzerland is one of the main references on the introduction of online voting[1][73], having one of the permanent online voting platforms in the world until 2019.

Some of the common motivations for introducing e-voting in countries are the following ones: reduce fraud during the election process, speed up the processing of results, increase the accessibility of voters with disabilities, facilitate the voting process to citizens living abroad and reduce the costs associated to the electoral processes; nevertheless, there are some inherent challenges that must be addressed such as the lack of transparency for voters, the complexity of the system that is only fully understood by a small number of experts or the conflict with the existing legal

[1] In the portal of the Swiss government there is a summary of the e-voting milestones in Switzerland: https://www.bk.admin.ch/bk/en/home/politische-rechte/e-voting/chronik.html

and regulatory framework. What makes e-voting systems different from other ICT systems such as banking or e-commerce, is precisely the number of requirements that they must fulfill in order to provide a solution for all their inherent challenges or, as pointed out in [75], the interaction between these requirements. Things that are inherent to traditional paper-based voting becomes a challenge in e-voting. An e-voting system should check that all the votes stored in the ballot box were cast by eligible voters while at the same time must preserve voter's anonymity. This is easily done in a polling station where the election officer manually checks voter's identity and once the vote is inside the ballot box any relation between it and the voter disappears. On the other hand, voters want to be sure that their votes are taken into account during the counting phase but they do not want anyone to know their voting intention. Finally, due to the lack of transparency of the process, e-voting systems must offer public mechanisms in order to verify that the integrity of the election was not manipulated neither by outsiders nor by the system operators. Depending on the context where an election is run it is required that the e-voting system satisfies some requirements or others.

In online voting systems, in which this thesis is focused on, privacy and verifiability are two of the fundamental ones. Privacy requires that the link between the vote and the voter who has cast it must remain secret during the whole process (vote anonymity) and that the voting options selected by the voter must be private (vote confidentiality), while verifiability requires that all the steps of the electoral process - vote casting, vote storage and vote counting - can be checked by the voters, the auditors or external observers. There must be a compromise between these requirements so the election information published to achieve verifiability does not compromise privacy. This information is usually published protected by cryptographic means whose security, which is based on well-known computational problems such as the discrete logarithm or the factorization, cannot be broken in a reasonable amount of time with the computing devices that we have nowadays. But, what would it happen if powerful machines appear in the future? Could this be a problem for the security of an e-voting system? The answer is yes and is precisely the problem we try to solve in this thesis.

The National Institute of Standards and Technology (NIST) published on 2016 a report on post-quantum cryptography [43] to share their understanding about the status of quantum computing and post-quantum cryptography, give recommendations on how to move forward and inform about their desire to initiate a standardization process for post-quantum cryptography [2]. As the report explains, post-quantum cryptography has become more and more important in the last years due to the increase of research on quantum computers, which can be used to solve certain computational problems faster than classical computers. This means that any public-key cryptosystem that is built on top of these problems, mainly the factorization and the discrete logarithm problem, is vulnerable to quantum attacks, and can be easily broken by a quantum computer. This poses a risk on the security of most of the applications we use nowadays, in which public-key cryptography

[2]Full details of the Post-Quantum Cryptography Standardization process can be found in the following website: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization

is an indispensable component. In contrast, the impact of quantum computers in the security of symmetric cryptography is not as dramatic. These primitives make no computational assumptions and if the key sizes are large enough, they are information-theoretically secure [3]. The research on quantum and post-quantum cryptography focuses on solving the problem with public-key cryptosystems.

1.1 Quantum and Post-Quantum cryptography

Quantum computing uses the principles of quantum physics to do things that classical computers cannot, such as breaking RSA efficiently. Nevertheless, a quantum computer is not a super-fast normal computer, so they cannot solve any problem that is too hard for a classical computer such as NP-complete problems.

While classical computers operate with bits which are either 0 or 1, quantum computers use *quantum bits* (qubits) which can take both values at the same time. This ambiguous state is called *superposition*. The idea is that before we observe a qubit it does not take a definite value, it is in a state of superposition, and is only when we observe it, i.e., when we measure it, that it stops in a concrete value. A good example to better understand what this status means is to think on a coin. If we spin it, there is a chance that the coin lands on heads or on tails, it can be either, but it is not until we stop it that we know the value.

We can express a qubit state in the following way: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers called *amplitudes* such that $|\alpha|^2 + |\beta|^2 = 1$. The probability of seeing a 0 when we observe a qubit is $|\alpha|^2$ and the probability of seeing 1 is $|\beta|^2$. If instead of a single qubit we have a group of them (for example a *qbyte* which is formed by 8 qubits), we know that their states are somehow connected. This phenomenon is known as *quantum entanglement* which means that several qubits can exist in a single quantum state and changing the state of one will change the state of the others. We define this single quantum state as: $|\psi\rangle = \alpha_0|0\ldots0\rangle + \alpha_1|0\ldots1\rangle + \alpha_2|0\ldots10\rangle + \ldots + \alpha_{2^n-1}|1\ldots1\rangle$ where n is the number of qubits and $|\alpha_0|^2 + \ldots |\alpha_{2^n-1}|^2 = 1$.

Let us consider an example with 2 qubits. While in a classical computer with two bits we can have only one the following four states $\{00, 01, 10, 11\}$, in a quantum computer 2-qubits can represent these four values at the same time, they can be in a superposition of the four states (the 2-qubits state is represented as $|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$). So informally we can say that n qubits can store more information than n bits and can process also more data since they can consider a large number of combinations simultaneously.

These special properties of quantum computers allow to efficiently solve computational problems which are considered hard to break by classical computers. Two of the most important problems broadly used nowadays to provide security to our systems are the factorization and the discrete logarithm problem, which can be solved

[3] Grover's algorithm [89] executed in a quantum computer can provide a quadratic speedup on finding the symmetric key, which decreases the brute force attack time. For example, this algorithm allows finding an AES256 private key in 2^{128} quantum operations given several encrypted messages using this key

by Shor's quantum algorithm [137] in polynomial time. This is why some organizations such as NIST are recommending to transitionate to quantum-safe algorithms. Indeed the main problem comes when we want to provide long-term privacy to the information. Data encrypted using a quantum insecure algorithm may be stored by an adversary until quantum computers are available, and then use them to break the privacy of this data. If this happens for example in the e-voting context, the adversary can learn how a person voted some years ago which may have political, as well as personal implications (e.g. in case of family coercion). There are two possible solutions to the problems inherent to the appearance of quantum computers: either use quantum cryptography or post-quantum cryptography.

Quantum cryptography uses the principles of quantum mechanics to perform cryptographic operations. The best-known example is the Quantum Key Distribution that allows two parties to exchange a secret and detect any interception of it during the communication. This is due to the fact that it is not possible to measure the quantum state of the system without disturbing it. Nevertheless, quantum cryptography needs special requirements such as its own infrastructure and does not cover all the needs of secure-communications and secure e-voting systems, e.g., digital signatures, public-key encryption, zero-knowledge proofs, etc. Due to this, quantum cryptography is not suitable for the purpose of this thesis.

On the other hand, post-quantum cryptography uses classical computational problems and algorithms to build quantum-resistant cryptographic primitives, hence they can be implemented in classical computers. Some of the main families of post-quantum primitives are lattice-based cryptography, code-based cryptography, multivariate polynomial cryptography and hash-based signatures [43]. Their security is based on computational problems for which there is currently no quantum algorithm that can break them. Code-based cryptography is based on *error-correcting codes* which add redundancy to transmitted data so that the receiver can correct the errors that occurred during the communication. Multivariate polynomial cryptography consists on building cryptographic schemes which security is based on the difficulty of solving systems of multivariate equations or equations involving multiple unknowns. The security of hash-based cryptography is based on the well-studied hash functions, whose collision resistance property ensures that the probability of obtaining the same hash value using two different inputs, is negligible. Finally, from all of them lattice-based cryptography is which have received more attention and is a great promise to get cryptosystems that will remain secure in the post-quantum era [113]. It allows to build several cryptosystem such as digital signatures, public-key encryption or zero-knowledge proofs. The cryptographic protocols and the online voting system presented in this thesis uses lattice-based cryptography to achieve long-term privacy.

1.2 Our contribution

The contribution of this thesis is mainly on the fields of online voting and lattice-based cryptography. More concretely, we propose two distinct lattice-based proof of a shuffle which are used to build a post-quantum verifiable mix-net. Then, we also propose a post-quantum online voting system which uses the post-quantum verifiable

mix-net to provide anonymity and a lattice-based coercion-resistant protocol, which is also one of the contributions of this thesis, to provide cast-as-intended verifiability.

In the last years, several countries have been introducing electronic voting systems to improve their democratic processes: e-voting systems provide more accurate and fast vote counts, reduce the logistic cost of organizing an election and can offer specific mechanisms for voters with disabilities to be able to cast their votes independently. In particular, internet voting systems provide voters with the chance to cast their votes from anywhere: their homes, hospitals, or even from foreign countries in case they are abroad at the time of the election. As we have explained at the beginning of this introduction, privacy and verifiability are two fundamental requirements for internet voting systems that seem to be contradictory. Privacy requires that the link between the vote and the voter who has cast it must remain secret during the whole process (anonymity) and that the vote content is only known by the voter who cast it (confidentiality), while verifiability requires that all the steps of the electoral process - vote casting, vote storage and vote counting - can be checked by the voters, the auditors or external observers.

The different techniques used by the actual internet voting systems to achieve anonymity can be classified in three categories: blind signatures, homomorphic tallying and mixing, which will be explained in detail in Section 2.3.2. For the purpose of this thesis we are interested on the latter technique. During a mixing process the ciphertexts are transformed in such a way that the correlation between the input and output of the process is hidden and it is not possible to trace it back, i.e., ciphertexts at the output look completely different as those at the input. This operation is called a *shuffle* and it is executed in a mixing network (*mix-net*) composed of mixing nodes (*mix-nodes*) each one performing in turns the same operation. This is done in order to be able to preserve the privacy of the process even if some nodes are dishonest: as long as one of the mix-nodes remains faithful and does not reveal the secret values used for computing the shuffle, unlinkability is preserved. Notice that this method requires to provide a *proof of a shuffle* to demonstrate that the contents of the output are the same as the contents of the input, i.e., ciphertexts have not been manipulated nor added or deleted.

On the other hand, in order to build verifiable systems one key instrument is the Bulletin Board: a public place where all the audit information of the election (encrypted votes, election configuration, ...) is published by authorized parties and can be verified by anyone: voters, auditors or third parties. However, once published in the Bulletin Board, it is not possible to ensure that all the copies are deleted after the election and the audit period ends, and long-term privacy may not be ensured by the cryptographic algorithms used nowadays, for example due to the efficient quantum algorithm given by Shor [137] that breaks computational problems such as the discrete logarithm or the integer factorization problems. This means that if our online voting system uses a mix-net to preserve privacy but also publishes the encrypted votes and the proof of a shuffle in the bulletin board to give verifiability, we need to ensure that the published information does not break long-term privacy. Since lattice-based cryptography seems to be one of the main alternatives to achieve post-quantum security, we consider interesting to focus our research on mix-nets capable of shuffling lattice-based encryptions and on computing lattice-based proofs

of a shuffle. In this way, we will be able to build a post-quantum online voting system in which long-term privacy is preserved since voting options are encrypted using a lattice-based cryptosystem and the resulting ciphertexts are anonymized using a lattice-based mix-net. Publishing the audit information in the bulletin board will not suppose any risk for long-term privacy since the cryptographic primitives used are known to be secure in front of a quantum adversary.

Lattice-based proof of a shuffle. We propose two proofs of a shuffle based on lattices. The former is the first universally verifiable mix-net for a post-quantum cryptosystems and follows Wikström's technique [153], who proposes an offline pre-computation technique to reduce the online computation complexity and a provably secure technique to prove the correctness of a cryptographic shuffle. Our proposal, although is based in [153], it is not a direct adaptation of it since it introduces two significant differences: during the offline part the random elements used to re-encrypt the ciphertexts are committed using the generalized version of Pedersen commitment and it is proved that these elements belong to a certain interval using zero-knowledge proofs. We show how to permute and re-encrypt lattice-based encryptions and give the first proof of a shuffle that works for a lattice-based cryptosystem. As we have mentioned before, for building the proof we use Pedersen commitments, which are perfectly hiding and computationally binding. The former means that the commitment does not reveal any information about the message committed and the latter that once committed, the message cannot be changed. For long-term privacy we are mainly interested on the first property. Since the commitment perfectly hides the committed message, its privacy does not depend on any computational assumption whose strength may be eroded in the future, so the scheme achieves our goal, which is to construct a proof of shuffle which ensures long-term privacy. Nevertheless, since the binding property of the commitment relies on the discrete logarithm problem which is already broken by Shor's quantum algorithm, the proof cannot be considered fully post-quantum. Moreover, there is no formal definition of security, necessary to precisely know how it can be embedded in a larger construction.

The second proof of a shuffle proposal tries to improve the previous one. It is fully based on lattices and we also give a definition of security and provide a proof of security for the mix-node. The proof is based on Bayer and Groth's technique [22], who use a different approach from Wikström to demonstrate the correctness of the shuffle and significantly improves the efficiency compared with previous schemes. Again, our proposal is not a direct adaptation of [22] since working with lattices requires different techniques to be applied. In order to build the proof we use a lattice-based commitment scheme and lattice-based zero-knowledge proofs, which makes the proof of a shuffle fully post-quantum. We use this lattice-based proof of a shuffle to provide anonymity to our post-quantum online voting system.

Post-quantum online voting system. This system uses a lattice-based encryption scheme to encrypt votes in the voting device, signs them using a lattice-based signature scheme and computes a coercion-resistant cast as intended proof as the one proposed in [91] but using lattice-based primitives. This proof allows the voter to

check that the options selected have not been modified by their voting device and, in addition, it prevents the voter from being coerced. The system also provides recorded-as-cast verifiablity which allows the voter to verify that their vote was successfully stored in the ballot box. With the description of this system we achieve the main goal of this PhD thesis which was to design a quantum-safe online voting system which provides long-term privacy.

1.3 Organization

The organization of this thesis is as follows:

- In Chapter 2 we give the background needed for better understanding the following chapters. There is an introduction to cryptography in which we present basic cryptographic primitives such as encryption or signatures but also some more advanced such as zero-knowledge proofs. We also define what does it mean for a cryptographic scheme to be secure and how we can demonstrate this security. Then, we introduce the reader to online voting, by explaining which are the security requirements an ideal online voting system should satisfy and how we can ensure they are fulfilled by cryptographic means. Finally, and probably the most important part of this chapter since is the result of our earliest research, we give an introduction to lattices. We explain some of the basics concepts and which are the computational problems we work with. Then, we describe a special class of lattices which allows to build more efficient lattice-based cryptographic schemes and finally we describe some of these schemes, focusing specially on those that will be used for building our proof of a shuffle.

- Chapter 3 corresponds to our first contribution to state of the art on lattice-based mixing protocols, which are a key component in online voting systems for providing anonymity. We show how to demonstrate that a list of RLWE ciphertexts, i.e., messages encrypted using a lattice-based encryption scheme, has been successfully shuffled without modifying them. This new protocol, which as far as we know is the first universally verifiable mix-net for a post-quantum cryptosystem, was published in the NordSec conference[4] in 2017:

  – *Proof of shuffle for lattice-based cryptography.* Núria Costa, Ramiro Martínez, Paz Morillo. In: Lipmaa H., Mitrokotsa A., Matulevicius R. (eds) Secure IT Systems. NordSec 2017. Lecture Notes in Computer Science, vol 10674, pp. 280-296. Springer International Publishing (2017).

  The full version of the paper can be found at ePrint[5].

- Chapter 4 corresponds to our second contribution to state of the art on lattice-based proofs of a shuffle and addresses some of the problems detected in our previous work. It is fully based on lattices and it is the first fully post-quantum

[4] `http://www.nordsec.org/conferences/`
[5] `https://eprint.iacr.org/2017/900.pdf`

proof of a shuffle for a RLWE encryption scheme. This proof was published in the International Conference on Financial Cryptography and Data Security in 2019:

  - *Lattice-based proof of a shuffle*. Núria Costa, Ramiro Martínez, Paz Morillo. In: Bracciali A., Clark J., Pintore F., Rønne P., Sala M. (eds) Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol 11599, pp. 330-346. Springer International Publishing (2020).

  The full version of the paper can be found at ePrint[6].

- In Chapter 5 we use most of the cryptographic primitives explained in previous chapters to build our post-quantum online voting system. We define the protocol by describing each of the algorithms involved in each of the system phases and we informally discuss which are the security requirements fulfilled by the system. Finally, we propose some research lines that could be followed in future work in order to improve the post-quantum online voting system.

- We end this PhD thesis with Chapter 6, in which we share with the reader the conclusions of our research, focusing on which have been our contributions to both the academic and industry world but also which are the topics we leave open for future work.

[6] `https://eprint.iacr.org/2019/357.pdf`

Chapter 2

Preliminaries

In this chapter we introduce first the notation that will be used throughout the document (Section 2.1). Specific notation, for example regarding lattices, will be introduced in the corresponding section. Then, we give a background on cryptography (Section 2.2) focusing on those primitives that are interesting for our work and we explain what is online voting, which are the security requirements an online voting system should satisfy and which are the existing techniques for achieving privacy and verifiability (Section 2.3). Finally, in Section 2.4 we introduce the reader to lattices and ideal lattices, explaining which are the main lattice-based computational problems and some of the cryptosystems built upon them.

As its name indicates, this is a preliminary chapter which will give us the necessary background to understand the following chapters.

2.1 Notation

Standard notation regarding vectors and matrices will be used. Column vectors will be represented by boldface lowercase roman letters (such as $\mathbf{v}$ or $\mathbf{w}$) and matrices will be represented by boldface uppercase roman letters (such as $\mathbf{M}$ or $\mathbf{A}$). Given two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^N$, we define the standard inner product in $\mathbb{Z}_q^N$ as $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{N} v_i w_i$. In addition, we define the l_∞ norm of a vector $\mathbf{v}$ as $\|\mathbf{v}\|_\infty = \max_{1 \leq i \leq N} |v_i|$ and the general norm l_p as $\|\mathbf{v}\|_p = (\sum_{i=1}^{N} |v_i|^p)^{1/p}$ for $p \geq 1$.

For a real number $x \in \mathbb{R}$, we let $\lfloor x \rfloor$ denote the largest integer not greater than x, and $\lfloor x \rceil := \lfloor x + 1/2 \rfloor$ denote the integer closest to x, with ties broken upward.

Finally, we write $a \xleftarrow{\$} A$ when a is sampled uniformly at random from a set A, and $a \xleftarrow{\$} D$ if it is drawn according to the distribution D. We also write $a \leftarrow \mathcal{A}(x)$ when on input x the deterministic algorithm $\mathcal{A}$ outputs a and, $a \xleftarrow{\$} \mathcal{A}(x)$ if $\mathcal{A}$ is a probabilistic algorithm.

2.2 Basic cryptography

Cryptography has a long history, starting with the use of hieroglyphs in Egypt and ending nowadays where cryptography plays a crucial role in most of the applications

since it provides security to the transmitted and stored information. Although the first goal of cryptography was to provide confidentiality in order to ensure the secrecy of the transmitted messages, modern cryptography is concerned also about data integrity, authentication and non-repudiation. Taking as a reference the *Handbook of applied cryptography* [110], we can define these four cryptography goals in the following way:

- Confidentiality: it prevents unauthorized parties to learn content of information. It is sometimes referred as privacy or secrecy.
- Integrity: it prevents unauthorized parties from modifying data.
- Authentication: it allows to verify that information comes from where it claims.
- Non-repudiation: it prevents an entity from denying the validity of some information or action.

In order to ensure each one of these goals there exist different cryptographic primitives: *encryption* transforms data to make it incomprehensible for all those who are unauthorized to access it, thus providing confidentiality. Nevertheless encryption does not provide integrity so if the encrypted data is modified no one will notice it. In order to provide integrity we can use *signatures* which also provides authentication and non-repudiation (other primitives such as *hash functions* or *message authentication code functions* are also used to provide integrity). The receiver of a signed message can check by verifying the signature that the information was not altered during its transmission and that the sender is who claims to be. It is also possible to combine encryption and signatures, i.e., the sender of a message signs it after encrypting it, thus ensuring the four goals.

Although these two primitives are probably the most well-known, there are also others which are considered more advanced, such a *zero-knowledge proofs* or *commitments*, that apart from ensuring some of the goals mentioned above they are also used to demonstrate other properties that we will see in more detail in the following sections.

The main goal of this section is to present the cryptographic primitives used as building blocks for the online voting system described in Chapter 5. We are going to explain what does it mean for a cryptographic primitive to be secure, how can we prove its security and which are the algorithms that define each one them.

There are some common concepts that will be used throughout all the explanations and that we describe below.

Definition 1 (Probabilistic polynomial-time algorithm (PPT algorithm) [96])**.** An algorithm A is said to run in *polynomial time* if there exists a polynomial $p(\cdot)$ such that, for every input $x \in \{0,1\}^*$, the computation $A(x)$ terminates within at most $p(\|x\|)$ steps ($\|x\|$ denotes the length of the string x). A *probabilistic* algorithm is one that has access to a source of randomness that yields unbiased random bits that are each independently equal to 1 with probability 1/2 and 0 with probability 1/2.

Definition 2 (Negligible function [96])**.** A function $f(\kappa)$ is negligible (negl) if it decreases faster than the inverse of every positive polynomial:

$$\forall c > 0 \quad \exists \kappa_0 \in \mathbb{N} \quad | \quad \forall \kappa \geq \kappa_0 \quad |f(\kappa)| \leq \frac{1}{\kappa^c}$$

where κ is the security parameter.

Definition 3 (One-way function)**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function if it can be evaluated in polynomial time and for every PPT algorithm A there is a negligible function ϵ such that

$$\Pr[A(f(x)) \in f^{-1}(f(x))] \leq \epsilon(n) \quad \forall n \in \mathbb{N}$$

Definition 4 (Trapdoor one-way function)**.** A function $f : \{0,1\}^* \to \{0,1\}^*$ is a trapdoor one-way function if it is a one-way function (see Definition 3) but it becomes easy to invert with the knowledge of a trapdoor.

2.2.1 Security

When trying to define what does it mean for a cryptographic scheme to be secure we need first to define which are the conditions under it is secure, i.e., against whom or from what it is safe. For this reason, we define the following two concepts:

- The attack model: specifies the information available to the adversary when performing the attack.
- The security goal of the adversary: is the adversary's intention when attacking the cryptographic scheme.

Then, we can define the *security notion* of a cryptographic scheme as a combination of a goal and an attack and we can claim that *a cryptographic scheme achieves a certain security notion if any attacker working in a given attack model cannot achieve their goal.*

Nevertheless, is it impossible for an attacker to achieve their goal or depending on the computational resources available it will be hard to achieve it but not impossible? and, what does it mean hard, how much work it will take for an attacker to break a system? Following we answer these questions.

The *security level* of a cryptographic scheme defines the amount of work that it takes to break the cryptosystem. We distinguish between two types of security levels:

- Information-theoretic security (or unconditional security): even if an adversary has unlimited computational resources (computationally unbounded adversary) is theoretically impossible to break the cryptographic scheme. This is a desired security level but is useless in practice. An example of information-theoretically secure cryptosystem is the *one-time pad* symmetric encryption scheme.

- Computational security: this is a more relaxed security level than the previous one so it is also weaker. While unconditional security says that a cryptographic scheme cannot be broken, computational security gives a limit on how hard it is to break it. An efficient adversary (computationally bounded adversary) with reasonable resources and in a reasonable amount of time can only succeed on breaking the cryptographic scheme with some very small probability. We can express computational security in terms of two values: a limit t on the number of operations that an attacker can carry out, and a limit ϵ on the probability of success of an attack. Then, we can say that a scheme is $(t, \epsilon)-$secure if an attacker performing at most t operations has a probability of success on breaking the scheme no higher than ϵ. We are going to consider both t and ϵ as functions of the *security parameter* of the scheme [96] and the adversary as a PPT algorithm (see Definition 1).

The security level we are going to require for our cryptographic schemes is computational security since it suffices for practical proposals. Nevertheless, it is important to recall that a scheme that is computationally secure can always be broken with enough time and enough resources.

Taking all of these into consideration, we can define more precisely what does it mean for a cryptosystem to be secure in the following way:

Definition 5 (Secure cryptosystem)**.** A cryptographic scheme is secure if every PPT adversary $\mathcal{A}$ working in a given attack model achieves their goal with only negligible probability.

Apart from giving a definition of security for a cryptographic scheme we also want to demonstrate that it is indeed secure. For doing it there are mainly two approaches:

- Provable security: it shows that if the security of a cryptographic scheme is compromised then either some simple logical contradiction occurs (information-theoretic security) or some well-studied problem can be solved efficiently (computational security) [17]. In order to demonstrate computational security we can use either formal methods or reductionist proofs [79] in which is it demonstrated that any efficient adversary breaking the cryptographic scheme can be used as a subroutine by another adversary to solve the problem that is believed to be hard.

- Probable security: there are some cryptographic schemes (for example, most of symmetric encryption schemes) for which it is not possible to demonstrate that they are as hard to break as a well-studied problem. In this case, we are confident that the scheme is (probably) secure because it has not been found yet any efficient attack that breaks the scheme.

The approach we are going to follow to demonstrate that a cryptographic scheme is secure is reductionist proofs and more concretely the game-playing technique [138]. In this game there are two probabilistic processes, the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ which communicate with each other in an attack game which usually consists on four phases: init, request, challenge and response phase. During the

init phase the challenger configures the environment in which the game will be executed, i.e., generates all the public and private parameters required to setup the cryptosystem that is going to be attacked. Then, in the **request phase** the adversary performs queries to the challenger and uses the answers to try to be successful in the next phase. Intuitively, the more flexible the attacker is in this phase, the more probability will have to be successful. When the **challenge phase** is reached the adversary cannot perform more queries and it is challenged by $\mathcal{C}$ to solve a problem (for example, to distinguish between the encryption of two different messages). Finally, in the **response phase** the adversary answers the challenge and $\mathcal{A}$ wins if the answer is correct. Indeed, we normally say that $\mathcal{A}$ wins the game if some particular event $\mathcal{S}$ happens which corresponds to successfully answering to the challenge. In this context, and taking as a reference Definition 5, security means that for every PPT adversary $\mathcal{A}$, the probability that the event $\mathcal{S}$ occurs is negligible close to some other *target probability* (either 0, 1/2 or the probability that some other event occurs in other game where the same adversary is playing with another challenger). In order to demonstrate the security of a cryptosystem usually it is required not only one game but a sequence of them: Game 0, Game 1,..., Game n; where Game 0 is the original game in which the adversary plays against the original challenger. From one Game to another only small modifications are done such that adjacent games are indistinguishable, thus if $\mathcal{S}_i$ denotes the event in game G_i that makes the adversary win, $|\Pr[\mathcal{S}_i]|$ is negligible close to $|\Pr[\mathcal{S}i+1]|$. In the last game the probability that the adversary wins, i.e., the probability that the event $\mathcal{S}_n$ occurs, is negligible close to the target probability, since usually the $\mathcal{S}_n$ implies that some well-studied hard problem is broken. If it is demonstrated that G_0 is indistinguishable from G_1, G_1 from G_2 and so on, we can argue that the first game is indistinguishable from the last one and security is proved. As explained in [138] this transition from one game to another can be based on indistinguishability, on failure events or on conceptual changes.

These proofs sometimes make use of an assumption in which the hash functions are ideal, i.e., they behave as a truly random functions. This idealized model introduced by Bellare and Rogaway [25] (we say that is an idealized model because truly random functions cannot be handled by polynomial time algorithms) is called the **Random Oracle Model** (ROM) in which it is assumed that exists a random oracle $\mathcal{O}$ to which both the adversary and the challenger have access. This oracle can be seen as a black box which for each input produces an output in such a way that if the same input is provided the same output is given. Reductionist proofs that make no assumptions are said to be in the *standard model*, nevertheless they are not considered in this thesis.

2.2.2 Encryption

Encryption is the principal application of cryptography since it allows users to exchange messages in a secure way through a insecure channel, so any unauthorized user cannot learn the content of these messages. In an encryption scheme the *plaintext* is the unencrypted message and the *ciphertext* refers to the encrypted message. The set of all possible plaintexts is denoted as $\mathcal{M}$ and the set of ciphertexts as $\mathcal{C}$.

The operation that converts a plaintext into a ciphertext is called encryption and the one that turns a ciphertext back to a plaintext is called decryption. In order to execute these two operations it is necessary to use a *key*. Depending on how many keys do we need we distinguish between *symmetric encryption* (the same secret key is used to encrypt and decrypt) and *asymmetric encryption* or *public-key encryption* (a public key is used to encrypt and a private key to decrypt).

As we have explained in the previous section the definition of the security of a cryptosystem starts by specifying which are the attack model and the goal of the adversary. In an encryption scheme we distinguish between the following ones:

- Attack model:
  - *Ciphertext-only attack* (COA): this is the most basic scenario in which the attacker has access only to some ciphertexts and does not have any information about the plaintexts that were encrypted. This is a passive attacker model since $\mathcal{A}$ is just an observer.
  - *Known-plaintext attack* (KPA): in this scenario the attacker has access to one or more pairs of ciphertext and plaintext, not chosen by him. As in the previous model, this is a passive attacker model since $\mathcal{A}$ is just an observer.
  - *Chosen-plaintext attack* (CPA): in this attack model the attacker can make queries to the challenger asking for the encryption of some previously selected message. The attacker can chose these messages considering the information sent by the challenger.
  - *Chosen-ciphertext attack* (CCA)[115]: in this attack model the attacker can make queries to the challenger asking for the decryption of some ciphertexts. As in the previous model, the selection of the ciphertexts can be done considering the information sent by the challenger, i.e., the relation between a ciphertext and the corresponding decryption. If the attacker is allowed to make queries after the challenge phase, the attack model is called *Adaptative Chosen Ciphertext Attack* or *CCA2* [127].
- Adversary goal:
  - Distinguish: given a ciphertext c which encrypts one of two messages previously chosen by $\mathcal{A}$, the objective of the adversary is to determine which message was encrypted by $\mathcal{C}$ during the challenge phase. If the adversary is not successful the encryption scheme achieves the security goal of *indistinguishability* (IND) [79]. What we will require for our encryption scheme is that the probability of the adversary of being successful is close to 1/2, so we can guarantee that the only possibility $\mathcal{A}$ has to guess the correct message is to randomly select one of them.
  - Tampering: given a ciphertext c_1 that encrypts m_1 the objective of the adversary is to generate another ciphertext c_2 such that m_2 is related to m_1. If the adversary is not successful the encryption scheme achieves the security goal of *non-malleability* (NM) [59].

From these models and goals we can define several security notions but the mains ones are IND-CPA and IND-CCA. We say that our encryption scheme is IND-CPA secure if an adversary performing a CPA attack is not able to distinguish between the encryption of two different messages except with a negligible probability. This is also called *semantic security* [78]. On the other hand, an encryption scheme in IND-CCA secure if an adversary performing a CCA attack is not able to distinguish between the encryption of two different messages except with a negligible probability. If the attacker is allowed to make decryption queries also after the challenge phase, we say that the scheme in IND-CCA2 secure.

2.2.2.1 Symmetric key encryption

In a symmetric encryption (also called private-key encryption), the symmetry lies on the fact that both parties hold the same key which is used for both encryption and decryption.

Definition 6 (Symmetric key encryption scheme)**.** A symmetric encryption scheme is defined by the following PPT algorithms:

- $\mathsf{KeyGen}^{\mathsf{S}}_{\mathsf{E}}(1^{\kappa})$: the randomized key generation algorithm takes as input the security parameter 1^{κ} and outputs the secret key $k \in \mathcal{K}$. It also defines the message space $\mathcal{M}_{\kappa}$ and the ciphertext space $\mathcal{C}_{\kappa}$.
- $\mathsf{Enc}^{\mathsf{S}}(m, k)$: the encryption algorithm takes as input a message $m \in \mathcal{M}_{\kappa}$ and the secret key $k \in \mathcal{K}$ and produces a ciphertext $c \in \mathcal{C}_{\kappa}$.
- $\mathsf{Dec}^{\mathsf{S}}(c, k)$: the decryption algorithm takes as input the ciphertext $c \in \mathcal{C}_{\kappa}$ and the secret key $k \in \mathcal{K}$ and outputs the corresponding plaintext m or $\perp$ in case of error.

We use the superscript S to distinguish the $\mathsf{Enc}^{\mathsf{S}}$ and $\mathsf{Dec}^{\mathsf{S}}$ algorithms of a symmetric key encryption scheme from those in the asymmetric scheme. Following the same approach, the subscript E used in the key generation algorithm allows to distinguish it from that algorithm in the commitment scheme (see Definition 15).

Definition 7 (Correctness)**.** A symmetric key encryption scheme is correct if for all $m \in \mathcal{M}_{\kappa}$, all $k \leftarrow \mathsf{KeyGen}^{\mathsf{S}}_{\mathsf{E}}(1^{\kappa})$ and all $c \leftarrow \mathsf{Enc}^{\mathsf{S}}(m, k)$ we have that $\mathsf{Dec}^{\mathsf{S}}(c, k) = m$.

The best known attack for a symmetric-key encryption scheme is a brute-force attack, which consists on an exhaustive search of the secret key that was used to encrypt the plaintext. In order to consider a symmetric-key encryption scheme computationally secure it must not be vulnerable to this kind of attack. On the other hand, a necessary condition to be unconditionally secure is that the key should be at least as long as the message, such as in the one-time pad scheme.

The main limitation of a symmetric-key encryption primitive is the key distribution since the secret key must be shared between the sender and the receiver of the ciphertext. One way of solving this issue is using a public-key encryption scheme in which it is not necessary to share any key for executing the encryption and decryption algorithms.

2.2.2.2 Public-key encryption

In 1976, Diffie and Hellman [58] proposed a new encryption technique which allowed two parties to communicate privately without the need to share any secret key. This technique is called public-key encryption or asymmetric encryption. The asymmetry lies on the fact that the encryption of a message is done using a *public key* and the decryption using a *private key*. As their names suggest, the public key is known by everyone thus anybody can encrypt a message and the private key is only known by the receiver of the message so only them can execute the decryption. Although these two keys are different, they are not independent from each other: usually, the public key is computed from the private key using a one-way function (see Definition 3), thus from the public key it is computationally hard to retrieve the private key. As a consequence the encryption process is based on trapdoor one-way functions (see Definition 4): the encryption is the easy operation (anyone can do it) and the decryption is only easy for the user who knows the private key, i.e., the trapdoor.

Definition 8 (Public-key encryption scheme)**.** A public-key encryption scheme is defined by the following PPT algorithms:

- $\mathsf{KeyGen}_{\mathsf{E}}(1^\kappa)$: the randomized key generation algorithm takes as input the security parameter 1^κ and outputs a key pair (pk, sk). We refer to pk as the public key and to sk as the private key. It also defines the message space $\mathcal{M}_\kappa$, the ciphertext space $\mathcal{C}_\kappa$ and a randomness space $\mathcal{R}_\kappa$ (if it is a probabilistic encryption scheme).

- $\mathsf{Enc}(pk, m)$: the encryption algorithm takes as input a message $m \in \mathcal{M}_\kappa$ and the public key pk and produces a ciphertext $c \in \mathcal{C}_\kappa$. If the algorithm is probabilistic, it also uses a randomness $r \in \mathcal{R}_\kappa$ to compute the ciphertext. For simplicity, when working with a probabilistic encryption scheme we will denote this algorithm as $\mathsf{Enc}_{pk}(m, r)$.

- $\mathsf{Dec}(sk, c)$: the decryption algorithm takes as input the ciphertext $c \in \mathcal{C}_\kappa$ and the private key sk and outputs the corresponding plaintext m or $\perp$ in case of error.

We use the subscript E in the key generation algorithm allows to distinguish it from that algorithm in the commitment scheme (see Definition 15).

Definition 9 (Correctness)**.** A public-key encryption scheme is correct if for all $m \in \mathcal{M}_\kappa$, all key pairs $(pk, sk) \leftarrow \mathsf{KeyGen}_{\mathsf{E}}(1^\kappa)$ and all $c \leftarrow \mathsf{Enc}(pk, m)$ we have that $\mathsf{Dec}(sk, c) = m$.

The first cryptosystem implementing a public-key encryption scheme was the RSA cryptosystem by Rivest, Shamir and Adleman [130] in 1978, whose strength is based on the hardness of solving the RSA problem which is related to the problem of factoring large composite integers. Another well-known public-key encryption scheme is the ElGamal cryptosystem [62] defined by Taher ElGamal in 1985. The security of this scheme is based on the hardness of solving the discrete logarithm problem over finite fields.

Both the raw RSA (where the message is not padded before being encrypted thus the encryption operation is deterministic) and ElGamal encryption schemes are homomorphic (see Definition 10), which informally means that it possible to operate with the plaintexts without decrypting the ciphertexts. This will be a very useful property when constructing online voting schemes and, more concretely, when trying to ensure voters' anonymity.

Definition 10 (Homomorphic encryption scheme)**.** A public-key encryption scheme is homomorphic if $\mathsf{Enc}(pk, m_1)\phi\mathsf{Enc}(pk, m_2) \equiv \mathsf{Enc}(pk, m_1\theta m_2)$ for some operations ϕ and θ.

There are two types of homomorphism depending on the operation θ done over the plaintexts. If θ is the multiplication operation we talk about **multiplicative** homomorphism and if it is the sum, the homomorphism is **additive**.

Public-key encryption schemes solves the key-management problem inherent from symmetric encryption schemes, nevertheless it is much less efficient. As shown in the ECRYPT standard [1] or by the NIST[2], in order to achieve the same security strength public-key cryptography must use larger keys than symmetric cryptography. This is due to the fact that in public-key schemes the attacker has access to an extra piece of data that can leak information about the private key, i.e., the public key. In order to break the security of the scheme, instead of trying a brute-force attack the attacker will try to recover the private key from the public key, which becomes an easy task if we use the same key length as in symmetric-key schemes.

In real applications what is commonly used is a combination of both schemes in which we use the flexibility of the public-key cryptosystem to distribute the secret key and the efficiency of the symmetric-key cryptosystem to encrypt data. The secret key k is encrypted using the public key pk: $c_1 = \mathsf{Enc}(pk, k)$ and the message m is encrypted using the secret key k: $c_2 = \mathsf{Enc}^{\mathsf{S}}(m, k)$. The information sent is both ciphertexts. When the receiver wants to decrypt the message, they first decrypts the secret key and finally the message: $m = \mathsf{Dec}^{\mathsf{S}}(c_2, \mathsf{Dec}(c_1, sk))$.

2.2.2.3 Threshold public-key encryption scheme

In some applications such as online voting it is important that the private key used for the decryption is not owned by a single user but a group of them. In a threshold public-key encryption scheme [57] the private key is shared among a group of participants and the ciphertext can only be decrypted if a threshold number of them collaborate. In order to share the private key a *(t,n)-threshold sharing scheme* in used, where n is the number of participants and t is the threshold.

The most famous construction of a $(t, n)-$threshold sharing scheme is the Shamir Threshold Scheme [136] which we briefly explain hereunder:

1. Define a_0 as the private key that is going to be shared: $a_0 = sk$.

2. Choose $a_1, \ldots, a_{t-1}$ at random from $\mathbb{Z}_p$, where p is a prime.

[1] https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf
[2] https://www.keylength.com/en/4/

3. Construct the polynomial: $f(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1}$.
4. Choose $x_1, x_2, \ldots, x_n \in \mathbb{Z}_p$ and evaluate the polynomial in each value x_i.
5. The $i-$th participant is given the share $f(x_i)$.
6. In order to recover the secret sk only t shares are required. Then, the polynomial is reconstructed using, for example, Lagrange interpolation and the secret is computed as the evaluation of the polynomial at 0.

2.2.3 Digital signatures

As we have seen in the previous section, public-key encryption schemes are used to provide confidentiality since only the sender and the receiver of the ciphertext can learn its content. Nevertheless, since the key used to generate the ciphertext is public, any attacker could encrypt a different message and substitute the initial ciphertext by the new one without the receiver noticing it. In order to avoid this situation one solution is to use digital signatures. A digital signature [58] is a cryptographic tool that is used to ensure the integrity of the messages exchanged between two parties. The receiver of the signed message can check by verifying the signature whether the message has been modified during its transmission or not. In addition, digital signatures also provide authenticity and non-repudiation, which allows the receiver to check that the sender is who claims to be and avoids the sender to deny that they signed the message.

As public-key encryption schemes, digital signatures are also asymmetric key schemes: the *private key* only known by the sender of the message is used to compute the signature, and the *public key* which is known by everyone is used for verifying it.

Definition 11 (Digital signature scheme)**.** A digital signature scheme is defined by the following PPT algorithms:

- $\mathsf{KeyGen_S}(1^\kappa)$: the key generation algorithm takes as input the security parameter 1^κ and outputs a key pair (pk_s, sk_s). We refer to pk_s as the verification key and to sk_s as the signing key. It also defines the message space $\mathcal{M}_\kappa$ and the signature space $\mathcal{S}_\kappa$.
- $\mathsf{Sign}(m, sk)$: the signing algorithm takes as input a message $m \in \mathcal{M}_\kappa$ and the signing key sk and produces a signature $\sigma \in \mathcal{S}_\kappa$.
- $\mathsf{SignVer}(m, \sigma, pk)$: the verification algorithm takes as input the message $m \in \mathcal{M}_\kappa$, the signature $\sigma \in \mathcal{S}_\kappa$ and the verification key pk and outputs 1 if the verification succeeds or 0 otherwise.

Definition 12 (Correctness)**.** A digital signature scheme is correct if for all $m \in \mathcal{M}_\kappa$, all key pairs (pk, sk) and all signatures $\sigma \leftarrow \mathsf{Sign}(m, sk)$ we have that $\mathsf{SignVer}(m, \sigma, pk) = 1$.

We define the following attack models and adversary goals for a digital signature scheme:

- Attack model:
  - *Key-only attack* (KOA): in this scenario the adversary only knows the public key corresponding to the entity that signs the messages and they are not allowed to do signing requests nor private key requests.
  - *Known message attack* (KMA): the adversary can make queries to the challenger asking for the signature of some messages but without controlling which messages are going to be signed.
  - *Chosen message attack* (CMA): the adversary can choose which are the messages that are going to be signed by the challenger. If these queries are adaptive, i.e., they depend on previously obtained signatures and messages, the model is called *adaptive chosen message attack*.
- Adversary goal: the main goal of the adversary is to forge a signature, i.e., to generate a signature on behalf of other entity. If they achieve their objective we say that the scheme is broken. Nevertheless, depending on how the adversary forges the signature, the meaning of breaking the system varies:
  - Total break: the adversary is able to compute the private key or to find an efficient algorithm equivalent to the valid signing algorithm.
  - Selective forgery: the adversary is able to compute a signature for one message or a set of them previously chosen.
  - Existential forgery: the adversary is able to forge a signature for at least one message. The adversary has no control over the messages for which a signature is requested.

Digital signature schemes have the same drawback as public-key encryption schemes, efficiency. For this reason, what it is usually done is to sign a hash of the message instead of the message itself. Informally, a hash function is a one-way function (see Definition 3) that given an input of an arbitrary length it produces an output of fixed and short length. More formally,

Definition 13 (Hash function)**.** A hash function $h : \mathcal{D} \to \mathcal{R}$ is an unkeyed function that has the following properties [131]:

- *Compression*: h maps and input x of arbitrary finite bit-length to an output $h(x)$ of fixed bit-length n.
- *Preimage resistance*: given an output of the hash function y it is computationally hard to find the corresponding input such that $h(x) = y$.
- *2nd-preimage resistance*: given an input x and the corresponding output $h(x)$ it is computationally hard to find a second input x' which has the same output, i.e., $x' \in \mathcal{D}$ such that $x' \neq x$ and $h(x') = h(x)$.
- *Collision resistance*: it is computationally hard to find two distinct inputs x and x' (which can be freely chosen) which hash to the same output.

The compression property of the hash functions allows to improve the efficiency of the signature and verification operations.

2.2.3.1 Blind signatures

In a digital signature scheme the signer knows the message that is signing but in some scenarios (such as online voting systems) it is required that the signer generates a signature without knowing the message. Blind signature schemes, proposed by David Chaum in 1982 [40], are signature schemes in which the message is blinded before being signed, thus the signer will not learn the message content. Then, the signed message is unblinded and the signature can be publicly checked against the original message.

To illustrate how a blind signature scheme works, we show an example using the RSA scheme:

1. Alice wants Bob to sign a message m, but she does not want him to learn the content. Bob's RSA public key is (n, e) and private key (n, d).

2. Alice generates a *blinding factor* r such that $gcd(r, n) = 1$.

3. Alice computes $x = (mr^e) \mod n$ and sends x to Bob. The value m is blinded by r.

4. Bob signs x using his private key $t = x^d \mod n$ and sends t to Alice.

5. Since $t = x^d = (mr^e)^d = m^d r^{de} \mod n = m^d r \mod n$, Alice can retrieve the signature of m computing $\sigma = r^{-1}t \mod n$.

2.2.4 Zero-knowledge proofs

Interactive zero-knowledge proofs (ZKP) were introduced by Goldreich, Micali and Rackoff in 1989 [80]. An interactive ZKP is a protocol between a prover P and a verifier V in which P convince V that exists a *witness* for which some statement is true without leaking any other information. If in addition the prover also demonstrates that they know the *witness*, not just its existence, we talk about zero-knowledge proofs of knowledge (ZKPoK). For example, if we define $\mathcal{R}$ as the set of discrete logarithms problems and their solutions, the prover will demonstrate using a ZKPoK that they know a solution w to the statement $x = (p, q, g, h)$ such that $h = g^w$ without revealing any information about w (p and q are primes, $g, h \in \mathbb{Z}_p^*$ and $w \in \mathbb{Z}_q$).

A zero-knowledge proof should satisfy the following properties:

- Completeness: if the statement is true, an honest prover succeeds in convincing an honest verifier.

- Soundness: if the statement is false, a dishonest prover does not succeed in convincing an honest verifier except with negligible probability. For a ZKPoK the definition is formalized by introducing an efficient knowledge extractor that is supposed to extract a witness from the proof and that succeeds on convincing an honest verifier.

- Zero-knowledge: if the statement is true, a dishonest verifier does not learn anything more than the assertion of the statement. For ZKPs this definition is formalized by saying that there exists a simulator $\mathcal{S}$ which outputs an accepting conversation with the same probability distribution as the conversation between P and V. If the distribution is exactly the same we talk about *perfect zero-knowledge* and if it is statistically close to the original distribution we talk about *statistically zero-knowledge*. Finally, if it is not possible to distinguish between both distributions in polynomial time, we talk about *computational zero-knowledge*. There is a variant of this property called *honest-verifier zero-knowledge (HVZK)* in which the verifier is expected to perform according to the protocol.

In order to clarify the concept of interactive zero-knowledge proof of knowledge we show below a concrete example, the Schnorr protocol [135] which describes how to prove knowledge of a discrete logarithm.

Protocol 2.1: Schnorr protocol

$\mathcal{P}(g, h = g^w; w)$		$\mathcal{V}(g, h = g^w)$
$k \in \mathbb{Z}_q$ $a = g^k$		
	$\xrightarrow{a}$	
		$c \in [0, q-1]$
	$\xleftarrow{c}$	
$z = k + wc \mod q$		
	$\xrightarrow{z}$	
		$g^z \stackrel{?}{=} a \cdot h^c$

If the prover knows w and is honest, the honest verifier is convinced after the interaction, i.e., $g = a \cdot h^c = g^k \cdot (g^w)^c = g^{k+wc}$ (*completeness*). In order to demonstrate the *soundness* property we will use the fact that if the prover does not know the witness w, they cannot answer more than one challenge correctly. Given a fixed value a and two challenges c and c' (and their corresponding z and z'), if P could answer in such a way that V accepts both transcripts, then P will be able to extract the discrete logarithm w. This is demonstrated in the following way: if P answers correctly to both challenges then $g^z = a \cdot h^c$ and $g^{z'} = a \cdot h^{c'}$. So,

$$\begin{aligned} g^{z-z'} &\equiv_p h^{c-c'} \equiv_p g^{w(c-c')} \\ z - z' &\equiv_q w(c - c') \\ w &\equiv_q \frac{z - z'}{c - c'} \end{aligned}$$

Since the discrete logarithm problem is known to be computationally hard to solve, we can conclude that it is not possible for a prover to answer correctly to more than one challenge without knowing w. There exist at most one challenge c for which a honest verifier will accept a false transcript from a dishonest prover ($a = g^z h^{-c}$), but the probability of correctly guessing c given a challenge space $\mathcal{C}$ large enough, is negligible.

Finally, in order to demonstrate that the protocol is *honest-verifier zero-knowledge* we need to ensure that it can be simulated. Without knowing w and for each challenge c, it is possible to compute a triple $(a = g^z h^{-c}, c, z)$ with the same probability distribution as it occurs in the real protocol.

2.2.4.1 Sigma protocols

The Schnorr protocol is an example of Sigma (Σ) protocol [53], which is a 3-move interactive protocol between P and V, where P wants to prove knowledge of a witness w for a statement x. During the first move, the prover P sends the commitment a (see Definition 15) to the verifier, then V replies with a challenge c and finally P sends the answer z to V. The verifier decides to accept or to reject the answer based on the information they have seen, i.e., (x, a, c, z).

Definition 14 (Sigma protocol)**.** A protocol $\mathcal{P}$ is said to be a $\Sigma-$protocol if is of the above 3-move form and the following properties are satisfied:

- Completeness: if P and V are honest, V always accepts.
- Special soundness: there exists a knowledge extractor $\mathcal{E}$ which from any x and any pair of accepting conversations (a, c, z),(a, c', z') where $c \neq c'$ can efficiently compute w.
- Special honest-verifier zero-knowledge: there exists an efficient simulator $\mathcal{S}$ which on input x and a random c, outputs an accepting transcript (a, c, z) with the same probability distribution as a real conversation between an honest P and V on input x.

As we have explained before the first move of a $\Sigma-$protocol consists on computing a commitment. A commitment scheme is a cryptographic primitive which allows to commit to a message by generating a commitment, which hides the message to other parties. At a later stage, the commitment can be opened thus revealing the hidden message.

Definition 15 (Commitment scheme)**.** A commitment scheme is an interactive protocol between two parties which is defined by the following PPT algorithms:

- $\mathsf{KeyGen}_{\mathsf{C}}(1^\kappa)$: the key generation algorithm takes as input the security parameter 1^κ and outputs the public commitment key ck.
- $\mathsf{Com}_{ck}(m, d)$: the commitment algorithm takes as input a message m, the commitment key and an opening d and outputs a commitment c.

- $\mathsf{ComVer}_{ck}(m, c, d)$: the verification algorithms takes as input the message m, the commitment key ck, the commitment c and the opening d. It outputs 1 if the verification succeeds or 0 otherwise.

The subscript C in the key generation algorithm allows to distinguish it from that algorithm in the public-key encryption scheme (see Definition 8).

The scheme should satisfy the following requirements:

- Correctness: if both the sender of the commitment and the verifier are honest the algorithm ComVer always outputs 1.
- Binding: a commitment c cannot be opened to different messages, i.e., no adversary can generate c, $m \neq m'$ and d, d' such that both $\mathsf{ComVer}_{ck}(m, c, d)$ and $\mathsf{ComVer}_{ck}(m', c, d')$ output 1.
- Hiding: the commitment c does not reveal any information about the message m.

The binding and hiding requirements can be satisfied perfectly/unconditionally or computationally depending on the computational resources available to the adversary. Nevertheless, a commitment scheme cannot be perfectly hiding and binding at the same time, so it is either perfectly binding and computationally hiding or viceversa.

2.2.4.2 Non-interactive zero-knowledge proofs

Interaction between the prover and the verifier is not always possible or desirable: delays on the communication may affect the usability of the system (such as online voting systems) or errors on the order how messages are delivered may affect the security of ZKPs.

Non-interactive zero-knowledge proofs (NIZKPs) eliminate this interaction between P and V and give them some common information which depends on the security model where the proof is built.

Fiat and Shamir [66] proposed a technique to build NIZK-PoK from $\Sigma-$protocols with ROM-based security. The idea of this transformation is that the challenge (generated by the verifier in an interactive protocol) is computed as a secure hash on the public information, the statement and the commitment. Since in the Random Oracle Model hash functions are identified with truly random functions, the output of this function is a uniformly distributed random value as in the interactive proof, where the challenge is chosen at random from the challenge space.

In order to give a concrete example of this kind of proof we take as a reference Protocol 2.1 and convert it into a non-interactive protocol:

- The public information to which both P and V have access is the values p, q, g, h and a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
- The prover computes the commitment $a = g^k$.
- The prover computes the challenge as $c = H(g, h, a)$.

- The prover computes $z = k + wc$.
- The prover sends (a, z) to the verifier.
- The verifier computes $c = H(g, h, a)$ and accepts if $g^z = a \cdot h^c$

The security obtained in the ROM is heuristic and sometimes is interesting to build protocols which security is not based on an idealized model. This is the case of the Common Reference String (CRS) model which states that there exists a common reference string generated by a trusted party to which both the prover and the verifier has access. This CRS is given as input to both P and V and is used to generate the proof and verify it. The main advantage of this security model is that proofs have a standard reduction-based security proof so they are not heuristic. Nevertheless, the disadvantage is that the CRS must be created in a trusted manner, which is not always feasible.

2.3 Online Voting

In recent years several countries have been introducing online voting systems as a way to improve their democratic processes: e-voting allows more accurate and fast vote counts, reduces the logistic cost of organizing an election and also offers specific mechanisms for voters to cast their votes from anywhere. Nevertheless, one of the main drawback of these systems is the lack of transparency of the process, which generates mistrust and slows down their introduction into the electoral system. In traditional paper-based elections every citizen can observe how their vote is introduced in the ballot box and they are convinced that the counting is done correctly due to the number of observers. We trust the process because we can understand each of its phases and how they are implemented. In an online voting system, how can we be sure that our computer has not modified our vote before sending it or how can we know that our vote was counted? what happens from the moment when we click the "Send" button in our device until the results are published? We cannot observe all the operations done by the voting software (or if we can observe it probably we cannot understand it) and we need to trust a small group of experts who are managing the election and the system. In order to address all these problems and provide guidance and assistance to governments and organizations who are considering introducing online voting, some organizations such as the Council of Europe give their recommendations on legal, operational and technical standards for e-voting.

In this section we are going to define first which are the security requirements that an ideal online voting system should fulfill (Section 2.3.1) focusing on verifiability and privacy. Then, we are going to describe which are the existing techniques that allow to meet these requirements and which are their advantages and disadvantages (Sections 2.3.2 and 2.3.3). Finally in Section 2.3.4, we are going to introduce the syntax used in the definition of the voting system proposal presented in Chapter 5.

2.3.1 Security requirements

A simplified version of an online voting scheme could be that presented in Figure 2.1.

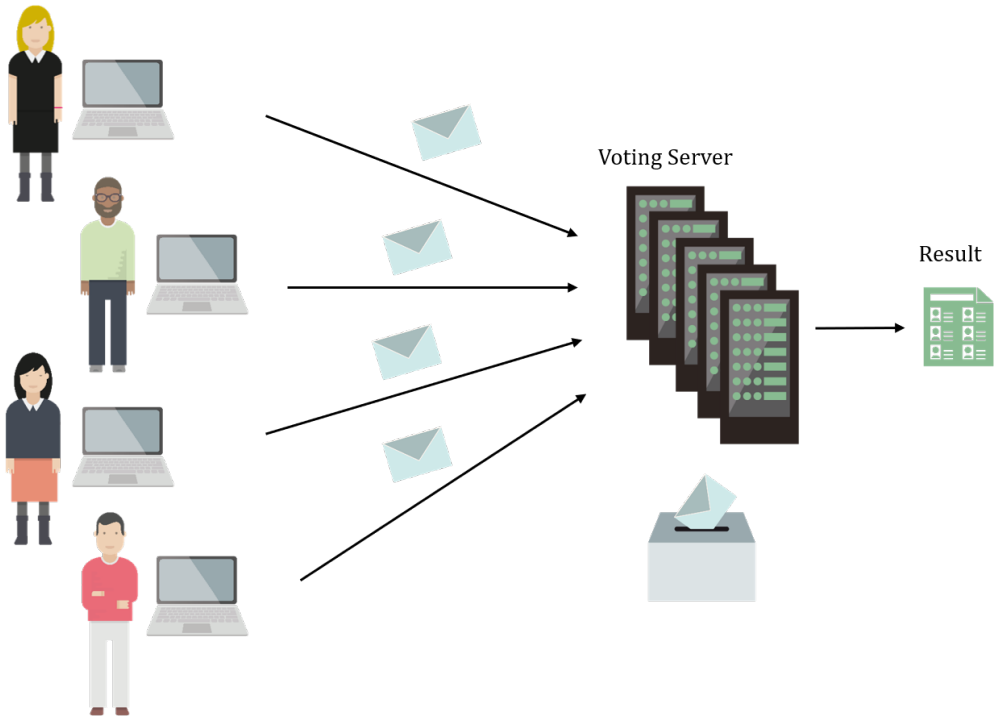


Figure 2.1: Simplified version of an online voting system

Each voter uses their own electronic device to send their vote through the Internet to the voting server, which stores the votes in an electronic ballot box. Then, once the voting phase is over the votes are counted and the final result is published. Without having any security measure implemented there are several risks that cannot be mitigated: the voting device could try to modify the voting options selected by the voters, an attacker can eavesdrop the communication channel and learn how the voter voted or a malicious system administrator could delete or add votes in the ballot box thus altering the result of the election.

For this reason, there are some security requirements that an ideal online voting system should satisfy, apart from those that are common to other online applications. We have not found any formal classification of these requirements thus the list presented in this section is the result of analyzing several e-voting papers, the legislation and lower regulations of Switzerland [38, 39] and the Council of Europe standards for e-voting [3].

- Vote authenticity: it has to be ensured that all the votes are cast by eligible voters (*eligibility*) and that only one vote per voter is taken into account during the counting phase (*vote unicity*).
- Voter privacy: the voting options selected by the voter must be private (*vote confidentiality*) and they cannot be linked to the voter who cast them (*vote anonymity*).

- Tally accuracy: the result of the election must accurately reflect the intention of the voters. Therefore, it should not be possible to modify the content of the votes (*vote integrity*), erase them from the ballot box or add fake votes.

- Election fairness: it should not be possible to provide intermediate results before the end of the election in order to give to all the candidates a fair decision.

- Verifiability: the voting system should provide enough evidences to allow voters and any third party to check that everything works as expected. Voters should be able to verify that the vote stored in the ballot box indeed represents their intention and that it has been taken into account during the counting phase. An auditor should be able to verify that only votes cast by eligible voters have been included in the tally.

- Receipt-freeness: the system should not provide to the voter with any information that allows them to demonstrate to a third party how they voted. This is done in order to prevent coercion or vote selling.

- Traceability and accountability: all the components of the voting system must leave traces of their operations in order to allow their verification. In addition, in case of any malfunction, it has to be possible to identify the responsible component.

As we can see there are several requirements that seem to be contradictory: how can the system provide enough evidences to the voter that their vote was cast-as-intended but at the same time prevent them from being coerced? or, how can the system ensure that all the votes included in the tally were cast by eligible voters without breaking voters' privacy? In order to fulfill all these security requirements we use cryptographic techniques.

As we have already seen in Section 2.2, by encrypting a message we ensure that only the sender and the intended recipient can learn the content, and signing it we protect its integrity and we can verify the authenticity of the sender. Therefore it seems that two security measures that could be implemented in order to satisfy requirements such as vote confidentiality or vote integrity are the encryption and the signature of the vote.

Unlike the online voting system presented at the beginning of this section, in the system shown in Figure 2.2 the voting options are encrypted and then digitally signed by a software executed in the voting device. Once encrypted and signed, the vote is sent to the voting server which verifies the signature before storing it in the ballot box. This allows the server to verify that the encrypted vote was not manipulated during its transmission. Once the voting phase is over and before sending the votes to the electoral board, the signatures are removed in order to avoid linking a decrypted vote with the voter who cast it. Usually each electoral board member holds a share of the private key and only if all the members collaborate votes can be decrypted. If threshold decryption techniques are used (see Section 2.2.2.3), only a threshold number of members is needed to perform the operation.

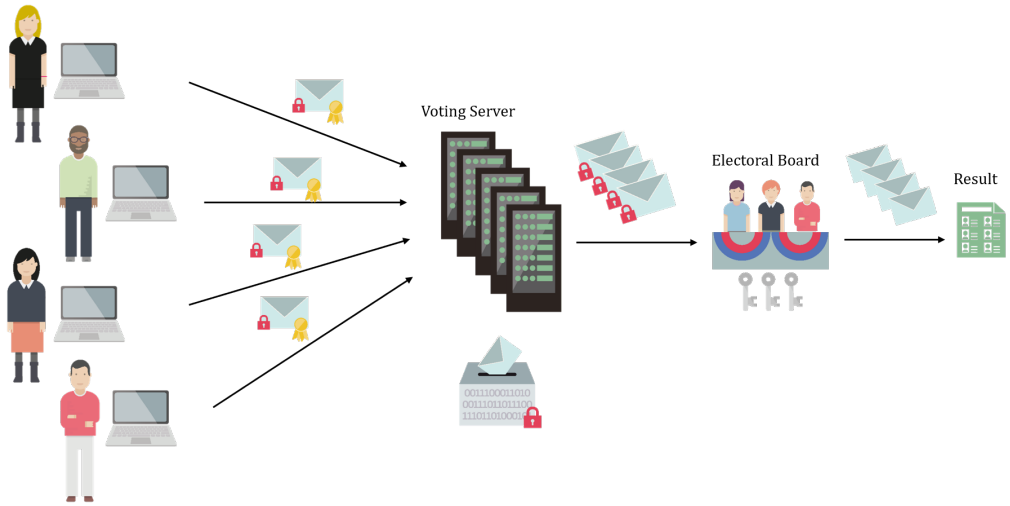


Figure 2.2: Basic online voting system. The voting options are encrypted and digitally signed in the voter's device.

Nevertheless, although with the encryption and the signature of the votes it is possible to fulfill some of the requirements, it is not enough. The voter cannot verify neither that their vote was successfully stored in the ballot box nor that it has been taken into account when computing the election result. On the other hand, although before decrypting the signatures are removed, any attacker knowing the order how the votes were cast and then analyzing the order they are decrypted, can link the content of a vote with a specific voter.

In order to address these issues which are mainly related to verifiability and privacy, we are going to describe in the following sections which are the existing techniques that allow to fulfill these security requirements.

2.3.2 Privacy in online voting systems

As we have already explained, one of the security requirements that an online voting system should satisfy is voter privacy, which include vote confidentiality and vote anonymity. The former is ensured by encrypting the vote and for the latter there are different approaches:

- Pollsterless or code voting
- Two agencies model
- Homomorphic tally
- Mixing

Each category is defined by a different anonymization procedure implemented in a distinct phase.

2.3.2.1 Pollsterless or code voting

As explained in [108, 144] a *pollster* is a piece of software that interacts with the voters during the voting phase in order to generate the vote and send it later to the

voting server. In order to preserve the voter privacy, the *pollster* must be trusted since it knows how the voter voted (for example, a *pollster* could be the voter's mobile phone). The online voting systems that implement the pollsterless protocol use pre-encryted ballots generated during the configuration phase and do not require any cryptographic operation to be done on the user's side (the software can be as simple as possible) hence the voting client does not need to be trusted.

In more detail, this kind of systems works in the following way: before the voting phase starts, each voters receives a Voting Card (see Figure 2.3) with the list of voting options available in the election and a voting code associated to each one of them. These codes are unique per voter, i.e., two different voters will have different voting code for the same voting option. In order to vote, the voter should introduce in the voting client the codes corresponding to the voting options selected. In this way, we do not need the voting client to have computational power to perform cryptographic operations, since it does not need to encrypt the vote nor to sign it. In addition, the voting card could also have a verification code for each voting option. When the server receives the voting codes, it uses them to compute the verification codes, which will be sent back to the voter, who will check using the voting card that they correspond to the voting options selected. At the end of the election, once the voting period has finished, the electoral board retrieves the voting cards in order to do the counting and provide the results.

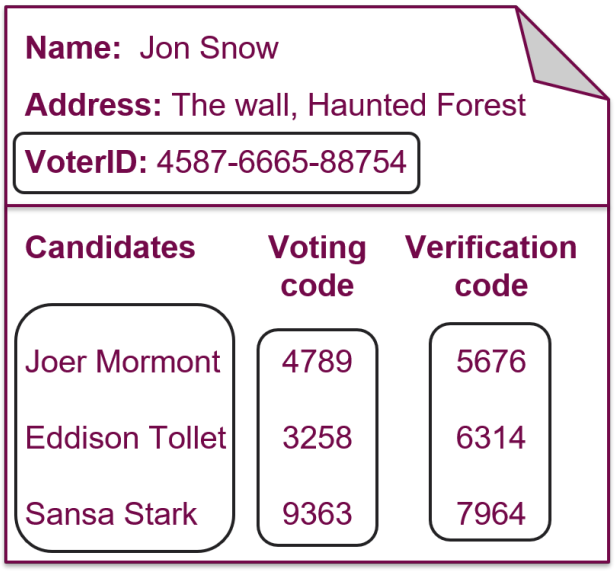


Figure 2.3: Voting card used in a pollsterless voting system

From an anonymity point of view these protocols allow us to send anonymous votes since they are not signed by the voter. They also protect the secrecy of the vote since the voting codes themselves do not give any information about the voting option they represent without having the voting card. Nevertheless, there is still a possibility of breaking voter's privacy if the voting card is compromised. For example, an attacker could know which are the voting options selected by the voter if they intercept the vote and compare the codes with those in the stolen voting card.

Apart from voter privacy, pollsterless online voting systems also satisfies verifiability, since the voter can verify that their voting intention has not been manipulated using the verification codes; and election fairness because the voting cards are only used by the electoral board once the voting phase has finished to provide the results. Nevertheless, they do not provide receipt-freeness since voters can demonstrate to a coercer how they voted using their voting card and the verification code sent by the voting server. Finally, due to the usage of voting codes this technique cannot be used in elections that allow write-ins.

Examples of voting system using this methodology are SureVote [41], Pretty Good Democracy [132] and Pretty Understandable Democracy [34].

2.3.2.2 Two agencies model

Unlike the previous protocol, the two agencies model [42, 68, 118] anonymize the votes when casting them. This protocol has two independent servers:

- Validation server: it authenticates voters, verifies their eligibility and, in case they are allowed to vote in that election, it sends them an anonymous token which will allow them to send an anonymous vote.

- Voting server: it receives the encrypted votes with the corresponding anonymous token. Only votes with tokens issued by the validation server will be accepted.

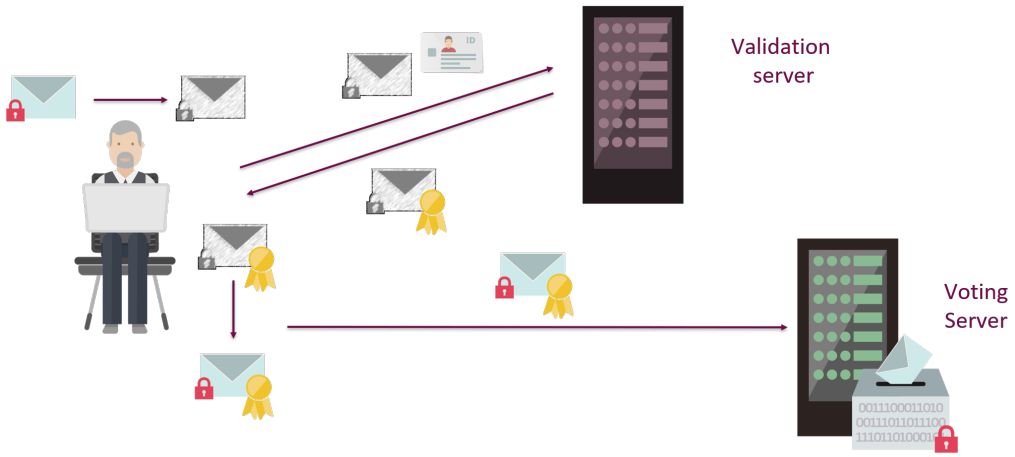


Figure 2.4: Two agencies model

This kind of protocols use blind signature already explained in Section 2.2.3.1. In more detail, the voting process is the following:

- The voting client, after the voter selects the voting options, encrypts them using the election public key, blinds the ciphertext and sends it to the validation server together with the voter's credentials. In this scenario, the voting client should have enough computational power to perform cryptographic operations.

- The validation server receives the blind vote, checks that the voter who sends it is an eligible voter and signs it. Finally, it sends back to the voter the signed blind vote.

- The voting client removes the blinding factors and obtains the encrypted vote signed by the validation server.

- The voting client sends the signed and encrypted vote to the voting server, that validates the signature and if the validation is successful, it stores the vote.

This protocol ensures anonymity because the validation server, who knows the identity of the voter, never sees the encrypted vote in clear but a blind version. In addition, the voting server stores an anoynymous encrypted vote, since its signature has been done by the validation server, not by the voter.

Nevertheless, this is not enough to provide anonymity, we should also assume that the servers are not going to collaborate. If they do collaborate, they could share some voter information, such as the IP address, that will allow them to link votes with voters. Furthermore, if the validation server is compromised, it could generate valid encrypted and signed votes that will be successfully accepted by the voting server, thus manipulating the election results.

Since this protocol does not pose any restriction on the encoding of the voting options, it supports any type of election including those having write-ins.

2.3.2.3 Homomorphic tally

Homormophic tally was first proposed in [45]. In this kind of systems voting options are encrypted using a cryptographic scheme that has homomorphic properties (see Definition 10) such as ElGamal or Paillier [119] and the anonymization procedure consists on aggregating the encrypted votes once the election has finished and decrypt only the result of the aggregation. From the two types of homomorphism (see Section 2.2.2.2), the additive one is the most used in electronic voting since the result of the aggregation is directly the sum of all the votes. In order to better understand these systems, we give below an example using the exponential version of the ElGamal cryptosystem.

$$c = (\gamma, \delta) = (g^k, h^k \cdot g^m) = (g^k, (g^x)^k \cdot g^m)$$

Recall that in the ElGamal encryption scheme the public key is $pk = (g, h)$, where $h = g^x$ and x is the private key ($sk = x$). When using additive homomorphism, the voting options are encoded using either g^1 or g^0 depending whether the voting option has been selected by the voter or not. After the encoding is done we have an array with as many elements as voting options available in the election: $(g^{m_1}, \ldots, g^{m_l})$

where $m_i \in \{0, 1\}$ and each one of these elements is encrypted independently:

$$\begin{aligned} V_1 : c_1 &= (g^{k_1}, (g^x)^{k_1} \cdot g^{m_1}) \\ V_2 : c_2 &= (g^{k_2}, (g^x)^{k_2} \cdot g^{m_2}) \\ &\vdots \\ V_l : c_l &= (g^{k_l}, (g^x)^{k_l} \cdot g^{m_l}) \end{aligned}$$

Once the voting phase has finished, the ciphertexts corresponding to the same option are aggregated:

$$\begin{aligned} V_1 :&(g^{\sum_{i=1}^{n} k_{1,i}}, (g^x)^{\sum_{i=1}^{n} k_{1,i}} \cdot g^{\sum_{i=1}^{n} m_{1,i}}) \\ V_2 :&(g^{\sum_{i=1}^{n} k_{2,i}}, (g^x)^{\sum_{i=1}^{n} k_{2,i}} \cdot g^{\sum_{i=1}^{n} m_{2,i}}) \\ &\vdots \\ V_l :&(g^{\sum_{i=1}^{n} k_{l,i}}, (g^x)^{\sum_{i=1}^{n} k_{l,i}} \cdot g^{\sum_{i=1}^{n} m_{l,i}}) \end{aligned}$$

where n is the number of voters in the election. Note that after decrypting each ciphertext the value obtained is g in power of the number of votes each voting option has received. In order to obtain this value it is necessary to compute the discrete logarithm, which is a problem assumed to be hard to solve. This poses a limitation on the complexity of the elections that can be supported since the bigger the exponent is the more expensive the computation of the discrete logarithm is.

Figure 2.5 shows an example of vote aggregation in an election with 4 options and 3 voters.

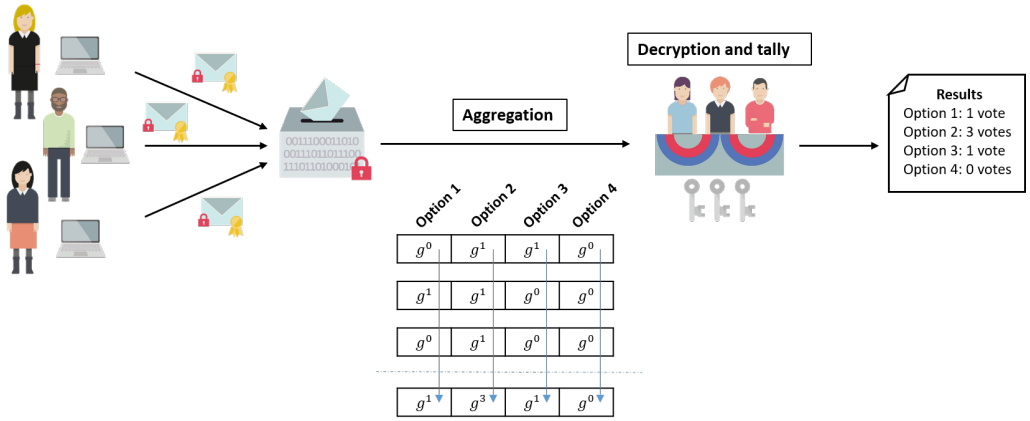


Figure 2.5: Homomorphic tally.

Summarizing, homomorphic tally systems ensures voter's privacy because:

- Votes are encrypted in the voting client and they are not decrypted until the end of the election.
- Individual votes are not decrypted but the aggregation of them, thus it is not possible to relate a decrypted vote with the voter who cast it.

Due to the encoding of the voting options, it is easy for a malicious voting client to cast an incorrect vote, for example, voting twice (g^2) for one option. Since votes are aggregated before being decrypted this attack will never be detected and the election result will be affected. For this reason, homomorphic tally systems require the generation of zero-knowledge proofs [50, 85], commonly known as OR-proofs, in the voting client in order to provide vote correctness. These proofs allow to demonstrate that the encrypted vote contains either g^0 or g^1, without revealing which one of both is encrypted.

Let us clarify these concepts with an example. The prover will encrypt $m_b \in \{m_0, m_1\} = \{0, 1\}$ using the exponential version of the ElGamal cryptosystem, and will obtain the ciphertext $c = (\gamma, \delta) = (g^k, h^k \cdot g^{m_b}) = (g^k, (g^x)^k \cdot g^{m_b})$. Then, they will compute an OR-proof in order to demonstrate the following relation:

$$\log_g \gamma = \log_h \left(\frac{\delta}{g^0}\right) \bigvee \log_g \gamma = \log_h \left(\frac{\delta}{g^1}\right)$$

Protocol 2.2: OR proof

$\mathcal{P}\left((\gamma, \delta); x\right)$		$\mathcal{V}\left((\gamma, \delta)\right)$
$r_b \leftarrow \mathbb{Z}_q$		
$(a_b, b_b) = (g^{r_b}, h^{r_b})$		
$z_{1-b}, c_{1-b} \leftarrow \mathbb{Z}_q$		
$(a_{1-b}, b_{1-b}) =$		
$(g^{z_{1-b}} \gamma^{c_{1-b}}, h^{z_{1-b}} (\delta / g^{m_{1-b}})^{c_{1-b}})$		
	$a_1, b_1, a_{1-b}, b_{1-b} \longrightarrow$	
		$c \xleftarrow{\$} \mathbb{Z}_q$
	$\longleftarrow c$	
$c_b = c - c_{1-b}$		
$z_b = r_b + x c_b$	$\longrightarrow z_b, z_{1-b}, c_b, c_{1-b}$	
		$c \stackrel{?}{=} c_b + c_{1-b}$
		$a_b \stackrel{?}{=} g^{z_b} \gamma^{c_b}$
		$b_b \stackrel{?}{=} h^{z_b} (\delta / g^{m_b})^{c_b}$
		$a_{1-b} \stackrel{?}{=} g^{z_{1-b}} \gamma^{c_{1-b}}$
		$b_{1-b} \stackrel{?}{=} h^{z_{1-b}} (\delta / g^{m_{1-b}})^{c_{1-b}}$

Note that only one equality of the two above will be true, that equality corresponding to the encrypted message. For example if $m_b = m_0 = 0$ the left-hand side equality will be true since $\log_g g^k = \log_h \left(\frac{h^k \cdot g^0}{g^0}\right)$. Nevertheless, the right-hand side one is false since we have a factor g multiplying h^k: $\log_g g^k \neq \log_h \left(\frac{h^k \cdot g^1}{g^0}\right)$. We show a sketch of the proof in Protocol 2.2 where we use the subscript b to refer to

the message selected by the prover (could be either 0 or 1) and $1 - b$ to refer to the non-selected message. The idea is that for m_b the prover can compute a real proof but for m_{1-b} they have to fake it.

Each encrypted voting option has its corresponding OR-proof that should be verified before computing the aggregation of ciphertexts. This makes homomorphic tally systems practical only for elections where the number of options is limited. Besides that, due to the specific encoding used for the voting options, write-ins or complex electoral systems are not supported by these systems.

2.3.2.4 Mixing

The mixing protocols [7] are based on trying to emulate real elections when, at the end of the election, the ballot boxes are shaken in order to break the order how the votes were cast.

The voting options selected by the voter are encrypted and signed by the voting client and stored in the voting server until the end of the voting period. Then, during the counting phase, the votes are validated and the signatures are removed in order to separate the vote from the voter who cast it. Nevertheless, this is not enough to anonymize the votes since they will be decrypted in the same order they were cast. For this reason, the anonymization procedure consists on sending the encrypted votes through a mixing process, that applies a permutation and a transformation over them, i.e., a shuffle [42], which makes the output of the process looks completely independent from the input. Thanks to this, votes cannot be linked to the voters who cast them, and they can be decrypted without breaking the anonymity.

Mixing protocols are built using mixing networks (mix-nets) that are formed by several mixing nodes (mix-nodes) each one performing in turns the shuffle. Depending on which transformation they apply, we distinguish between to types of mix-net:

- **Decryption mix-net**[42]: the voting client encrypts the vote as many times as mix-nodes using the public key of each one of them, starting from the last one to the first one. When votes are mixed, each node permutes the votes and removes the encryption layer using its private key. When the process finishes, the result is the list of votes permuted and decrypted. In order to avoid computing as many encryption operations as mix-nodes, we can use an encryption scheme such as ElGamal. In this scenario each mix-node has a key pair $(pk_i, sk_i) = (g^{x_i}, x_i)$ and votes are encrypted using a public key which is a combination of the mix-nodes' public keys: $pk = \prod_i pk_i = g^{\sum_i x_i}$. When a mix-node receives a list of ciphertexts (each ciphertext of the form $(g^k, pk^k \cdot m)$) it permutes them and removes the corresponding encryption layer by computing a partial decryption: $(g^k, pk^k \cdot (g^k)^{-x_j} \cdot m) = (g^k, (\prod_{i \neq j} pk_i)^k \cdot m)$. When the ciphertexts reach the last node, all the nodes' private keys except the last one have been removed and the messages can be recovered.

- **Re-encryption mix-net**: in this type of mix-net the transformation applied by each mix-node is the re-randomization. One of the algorithms that is suitable for doing this operation is the ElGamal encryption scheme, since it is

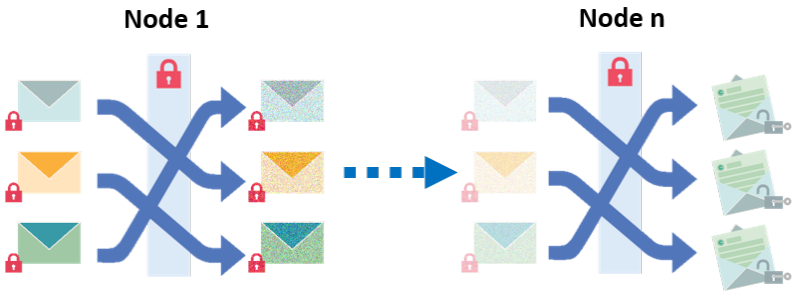


Figure 2.6: Decryption mix-net.

possible to re-encrypt the votes using the same public key and just modifying the randomness used during the encryption:

$$\text{Encryption} : (g^k, h^k \cdot m)$$
$$\text{Re-encryption} : (g^k, h^k \cdot m) \cdot (g^{k'}, h^{k'}) = (g^{k+k'}, h^{k+k'} \cdot m)$$

When the votes go through the mix-net, each node applies a secret permutation, re-randomizes the votes and sends them to the next node. After the last node, the new ciphertexts contain the same messages as those at the input of the process, but encrypted using a new randomness which is the sum of the mix-nodes' randomness: $(g^{k+\sum_i k'_i}, h^{k+\sum_i k'_i} \cdot m)$. Finally, votes are decrypted.

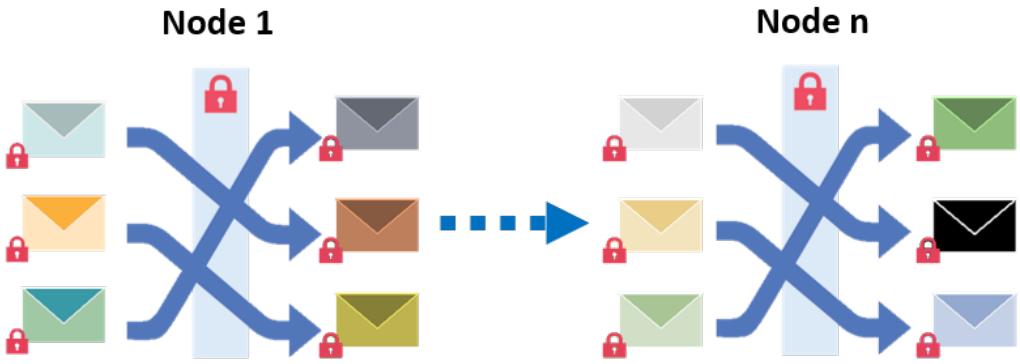


Figure 2.7: Re-encryption mix-net.

Independently of which type of mix-net is used, votes at the output look completely different from votes at the input and this opens a door to several attacks. How do we know that the mix-nodes have behaved properly? or, how can we ensure that votes have not been modified, added or removed during the process? For the anonymity procedure to work as expected, we need to assume that at least one of the mix-nodes is honest and will not leak any information neither about the permutation nor about the re-encryption parameters or private keys. On the other hand, it is important to provide verification methods to demonstrate that the mix-nodes have behaved properly. Proofs of a shuffle are zero-knowledge proofs (see Section 2.2.4)

that are used to demonstrate that the ciphertexts at the output of the mix-node are those at its input without revealing any secret information. Two of the most known verifiable mix-nets are explained in Chapters 3 and 4.

In contrast to homomorphic tally based systems, mix-net based systems do not have any vote correctness method implemented, i.e., the voting client does not need to compute a zero-knowledge proof for each voting option. Since votes are individually decrypted, invalid voting options can be detected during the counting phase. In addition, voting options do not require any special encoding and this makes mixing protocols more suitable for complex electoral processes such as elections having write-ins. In order to decide which anonymization procedure is better in terms of computational cost it should be analyzed how many voting options exist in an election and how complex it is. For example, if the election is a referendum in which the answers are usually three (yes, no, blank) most probably the best option would be to use an homomorphic-tally based system.

From the anonymization procedures explained in this section, this work is going to focus on mix-net based online voting system.

2.3.3 Verifiability in online voting systems

In the previous section we have already seen that in order to demonstrate the correctness of some procedures it is important to provide a verification method. In this section we are going to explain which types of verifiability [74] we distinguish depending on who is going to do the verification and what is going to be verified.

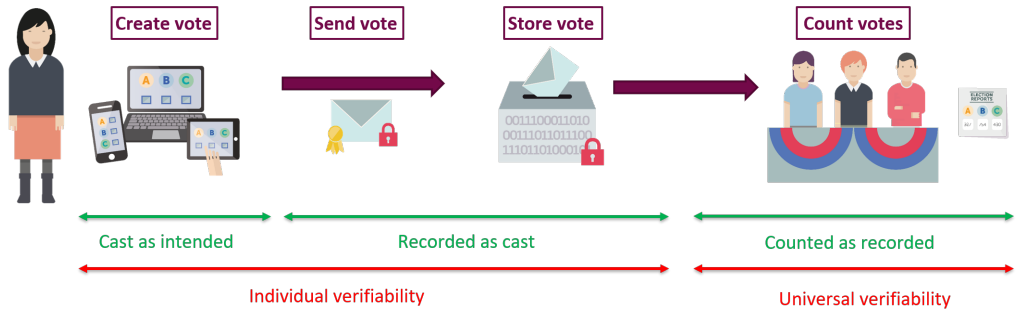


Figure 2.8: Individual and universal verifiability in online voting systems.

- Individual verifiability: voters can check that their vote contains the voting options they have selected (cast-as-intended), i.e., the voting client has not modified their selections, and that the vote has been successfully stored in the ballot box (recorded-as-cast).
- Universal verifiability: anyone can check that all the votes successfully stored in the ballot box during the voting phase has been taken into account for computing the results (counted-as-recorded), and that these votes were cast by eligible votes (eligibility verifiability).

Systems that have both individual and universal verifiability are said to be *end-to-end verifiable* [28]. It is important to remark than even if a system is end-to-end

verifiable it is still possible for an attacker to manipulate the integrity of the election result, but it will be detected.

We want also to emphasize here the complexity of ensuring both verifiability and vote privacy at the same time. As it is discussed in [74], it is not possible for an online voting system to provide unconditional verifiability and unconditional vote privacy of the vote simultaneously. For example, if the system guarantees recorded-as-cast verifiability, it implies that the encrypted vote stored in the ballot box should be linked to the identity of the voter, which makes the vote not anonymous.

A key component when talking about verifiability is the *bulletin board*. A bulletin board, first introduced by Benaloh et al. [45, 27], is a public information dissemination channel, such a web page, that has special properties: only authorized users can publish information and, once published, it cannot be erased nor tampered. A bulletin board can contain, for example, a list of encrypted votes or a list of crytographic proofs. The information published will depend on the verifiability provided by the system and the chosen implementation.

2.3.3.1 Cast-as-intended

In an online voting system voters use their own voting device to select their choices. Then, the voting device encrypts the selections and sends the encrypted vote to the voting server. From the encrypted vote it is not possible to infer any information about the voting options encrypted thus it is not possible for the voter to check if the encrypted vote contains indeed the options they have selected. A malicious voting device could modify the voter's selections without anyone noticing.

The implementation of individual verification mechanisms allows the voter to check that their vote has been cast-as-intended, i.e., to detect if the voting device has changed their selections before encrypting them.

In the last years there have been several proposals of online voting systems providing cast-as-intended that implement different individual verification mechanisms. The classification done by Guasch [90] is the most complete we have found in the literature and considers the following categories:

- **Verification with codes**: this mechanism is based on the so-called return codes. The voting server, using the vote sent by the voting device, computes the return codes and send them back to the voter who checks against their voting card that they correspond to the voting options selected. Traditionally these systems are pollsterless systems, already explained in Section 2.3.2.1, but there are other systems such as that used in Norway [76, 126] or Neuchâtel [71], that also use return codes. Unlike the pollsterless systems, in these ones voters are not required to enter any voting code, they are presented with the voting options available in the election and they just select them from the voting interface.

- **Challenge or cast**: this verification mechanism was first proposed in 2006 by Benaloh [28]. After the voting device encrypts the selected voting options, voters have two alternatives: (1) challenge the voting device in order to check if the encrypted options correspond to their selections or, (2) cast the encrypted

vote. Independently of the alternative chosen, after the voting device encrypts the voting options it shows a commitment of the vote to the voter, i.e., a hash of the vote. After this step, if the voter chooses not to challenge the voting device the encrypted vote is directly sent to the voting server. Later, the voter is able to check if the hash shown by the voting device is also present in the bulletin board (this is recorded-as-cast verification and will be explained in more detail in Section 2.3.3.2). On the contrary, if voters want to challenge the voting device, they are provided with the randomness used for encrypting their selections. Then, using a verification device and the randomness, they will recalculate the encrypted vote and check if its hash matches with that previously shown by the voting device. If the voting device has modified the voter's selections, it has negligible probability of showing the correct hash. If the verification is successful, the voting device generates a new encryption of the voting options and the voter has again the possibility of either challenge the voting device or cast the vote. Therefore, the vote that is cast is never audited. This is done in order to prevent vote selling attacks, since the voter could share the randomness with the vote buyer who will access to the bulletin board in order to check the contents of the encrypted vote. In this kind of systems it is recommended to challenge the voting device multiple times before casting the vote since this increases the probability of catching a cheating voting device. Helios [8, 9, 46], Wombat [2, 26], VoteBox [134] and STAR Vote [23] are examples of voting systems that implement this verification mechanism although only the first one is designed to be used remotely.

- **Decryption-based**: this individual verification mechanism consists on decrypting the vote that is stored in the voting server in order to check that it contains the voting options selected by the voter. The Estonian voting system [93] and the iVote 2015 system [33] implement this verification mechanism. As in the challenge or cast mechanism, the voter needs a second device to perform the verification, i.e., a smartphone. After the vote is cast the voting device shows a QR to the voter that contains the randomness used for the encryption and an identifier of the vote. The voter uses the smartphone to read the QR and requests the encrypted vote to the voting server using the voter identifier. Finally, the smartphone uses the randomness to brute-force the encrypted vote and shows the encrypted options to the voter.

- **Hardware-based verification**: this mechanism requires a trusted hardware device (for example, a smartcard) in order to be implemented. In the online voting protocols that implement this individual verification mechanism [101, 83] the voting device should interact with the hardware device, which will perform some basic cryptographic operations in order to send the vote and generate the verification information for the voter.

We have omitted from this list the category *verifiable optical scanning* since it is not a cast-as-intended verification mechanism used in online voting system.

Nevertheless, for our post-quantum online voting system we are not going to use any of the cast-as-intended approaches presented above but a variation of the

challenge or cast mechanism called *challenge and cast*, that is proposed by Guasch and Morillo in [91]. The details of this scheme are given in Section 5.2.1.

2.3.3.2 Recorded-as-cast

Online voting systems that implement mechanisms to provide recorded-as-cast allow voters to check that their votes were correctly received and stored by the voting server. For example, the decryption-based verification mechanism used to provide cast-as-intended verifiability also gives recorded-as-cast verifiability. Since the verification requires the vote to be downloaded from the voting server, the voter can be sure that the vote stored in the ballot box is the vote that was cast.

Another technique frequently used to provide recorded-as-cast verifiability is the generation of *vote receipts*. These receipts are values that uniquely identifies the vote and that are used by the voters, usually once the voting phase has ended, to check that their votes are stored in the ballot box . In the implementation used in Norway [126] and Neuchâtel [71] the receipt is a hash computed by the voting server after receiving the vote. This receipt is sent to the voting device which shows it to the voter. After the voting phase ends, a list of vote receipts is published in the bulletin board and voters can verify that their receipts are in that list. In addition, in order to prevent voters from cheating, i.e., to come up with a fake receipt and claim that something went wrong because it is not in the list, the voting server also signs the receipts and sends the signature to the voting device. Any valid receipt must have its corresponding signature.

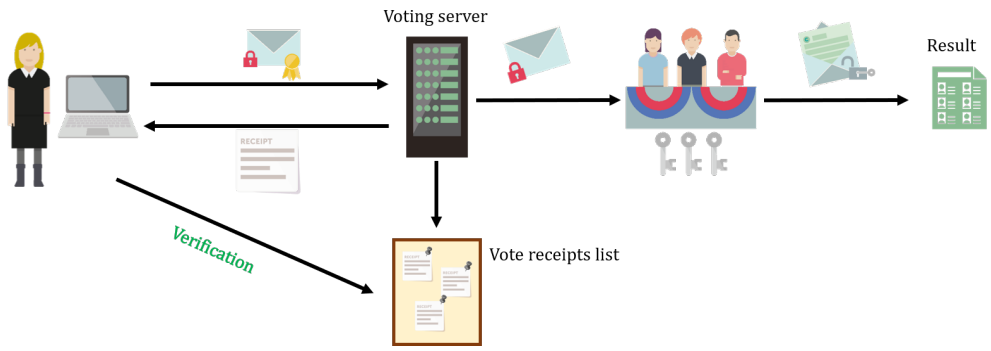


Figure 2.9: Recorded-as-cast verifiability using receipts.

2.3.3.3 Counted-as-recorded

The counted-as-recorded verifiability allows voters to check that their votes were included in the final tally, i.e., that were properly decrypted; and allow any observer to verify that all the votes cast by eligible voters were properly tallied. The easiest way to provide this verifiability would be to publish the list of encrypted votes in the bulletin board together with the output of the decryption and the decryption key. Although this will allow anyone to verify that the decryption process was properly done, it will also break voters' privacy. In order to avoid this, there are two main cryptographic techniques (already presented in Section 2.3.2) that can be applied:

- **Mix-nets**: the encrypted and signed votes are published in the bulletin board so anyone chan check that they were cast by eligible voters by verifying the signatures. Then, during the mixing process each mix-node re-encrypts and permutes the encrypted votes in turns and computes a zero-knowledge proof of the correct shuffle, which is published in the bulletin board together with the input and the output of each mix-node. Since the verification of the proof does not require any secret value to be revealed, it is universally verifiable with the information already published. Then, after the encrypted votes are shuffled they are decrypted. In order for the decryption process to be verifiable it must compute a zero-knowledge proof of correct decryption, that will be also published in the bulletin board as well as the input and the output of the process.

- **Homomorphic tally**: as in the previous technique, the encrypted and signed votes are published in the bulletin board. Then, after the aggregation of votes is done, the result is also published. Since this operation does not require any private information in order to be computed, it can be repeated by anyone who wants to verify that the operation was done correctly. Finally, as it is done when using mix-nets, votes are decrypted and the required information is published in order to verify the decryption process.

Any of these two techniques allow to verify that the decrypted votes correspond to the encrypted votes stored in the ballot box but without breaking voter's privacy.

2.3.4 Online voting syntax

In this section we give a general overview of the participants of an online voting system, the algorithms they execute and in which phase they do it. It is worth to say that this is an informal description since our goal here is only to introduce the syntax and both the algorithms and the phases will be formally presented in Chapter 5. We use as a reference the syntax described in [49] and [90, 91].

- *Voter*: A person who is entitled to cast a vote in a particular election or referendum.

- *Electoral authority*: They are in charge of configuring the election, decrypting the votes and generating the results.

- *Registration authority*: They are in charge of registering the voters and sending them the information that they need in order to cast their votes, for example, the voting card.

- *Ballot box*: Is the component in which the votes cast by eligible voters are stored. Usually it is managed by the voting server.

- *Ballot*: The legally recognized means by which the voter can express their choice of voting option.

- *Bulletin board*: Public information dissemination channel in which only authorized users can publish information that cannot be erased nor tampered.

- *Voting device*: Is the device used by the voters to cast their votes. It shows them the voting options available in the election, generates the vote from voter's selections and sends it to the voting server.
- *Voting server*: Is the component that receives, processes and stores the votes sent by the voting device. It also publishes information in the bulletin board and performs additional operations such as generating the receipt.
- *Auditor*: A person, internal or external, responsible for assessing the condition, reliability and security of the system.
- *Attacker*: A human or process, both internal or external, mounting an attack to the system or to parts of it. Also a subject authenticated as such but acting outside its role. The main goal of an attacker is to access, modify or insert sensitive information or to disrupt services.

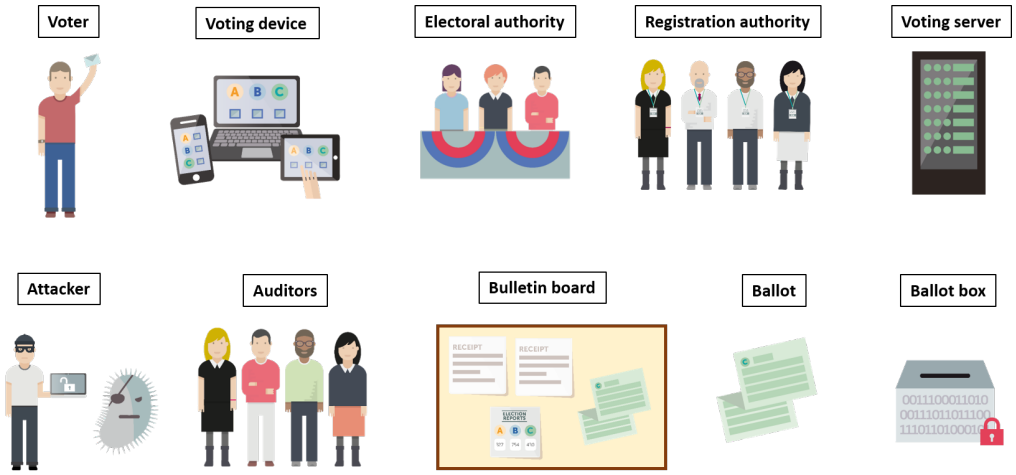


Figure 2.10: Participants of an online voting system.

Our online voting protocol $\mathcal{V} = \{\mathsf{Setup}, \mathsf{Register}, \mathsf{CreateVote}, \mathsf{AuditVote}, \mathsf{CastVote}, \mathsf{ProcessBallot}, \mathsf{VerifyVote}, \mathsf{Tally}, \mathsf{VerifyTally}\}$ consists of nine algorithms which are executed by the participants in the different phases of an election in the following way:

- **Configuration phase**: during this phase the electoral authority sets up the voting options and runs the Setup algorithm in order to generate the election information such as the election key pair. The public information is sent to the bulletin board so it can be used for verification and the private information is sent to whom belongs to (either the registration authority or the voter).
- **Registration phase**: in this phase the registration authority defines the electoral roll (voters that are eligible to vote in the election) and executes the Register algorithm to generate all the information that voters need in order to cast their vote, i.e., the credentials. As in the previous phase the public information is published in the bulletin board and the credentials are sent to the voters through a private channel.

- **Voting phase**: voters select their preferred voting options and the voting device runs the CreateVote algorithm which encrypts them and generates the information needed by the voter to check that their vote is cast as intended. Then, the voter executes the AuditVote algorithm with the aid of an audit device to check that the encrypted vote contains the selected voting options. If the verification is successful, the voter introduces their credentials into the voting device which uses them to digitally sign the vote by running the CastVote algorithm. The signed and encrypted vote is sent to the voting server which validates it using the ProcessBallot algorithm. If none of the validations fails, the vote is stored in the ballot box and a hash of it is sent to the bulletin board. Finally, the voter uses the VerifyVote algorithm to check that their vote is in the bulletin board, i.e., that their vote has been recorded-as-cast.

- **Counting phase**: when the voting phase ends, the electoral authority obtains the votes from the ballot box and executes the Tally algorithm. This algorithm first cleanses the votes, which consists on validating them and separating the ciphertexts from their signatures. Once the cleansing is done, it anonymizes the ciphertexts by running a protocol such as the mixing or homomorphic tally. Then, the election private key is reconstructed (if needed) and the ciphertexts are decrypted. Finally, the tally of the decrypted votes is computed and the results are published in the bulletin board. In order to verify the integrity of the operations executed by the Tally algorithm, the auditors runs the VerifyTally algorithm which first performs the same validations over the votes than the cleansing, and then verifies that the proofs generated during the process are valid.

In Chapter 5 we give details about which are the inputs, operations and outputs of each one of these algorithms.

2.4 Lattices

Lattices were employed in early 1980s for breaking cryptosystems but it was not until late 1900s when they were first used in the design of cryptographic schemes. In 1982, Lenstra, Lenstra and Lovász presented a lattice reduction algorithm called *LLL algorithm* [99] which has many applications in cryptanalysis [139] such as factoring polynomials or solving the knapsack problem. More than 10 years later, in 1996, Miklós Ajtai [12] presented the first lattice-based cryptographic construction, a family of one-way functions which security is based on the worst-case hardness of the Shortest Vector Problem (SVP). From then until nowadays, lattice-based cryptography [113] has become a very active area of research since it is maybe the most promising approach to get cryptosystems that will remain secure in the post-quantum era. It enjoys strong security guarantees from worst-case hardness, meaning that breaking their security implies finding an efficient algorithm for solving any instance of the underlying lattice problem. Furthermore, these constructions mainly involve linear operations such as matrix and vector sum or multiplication modulo integers, which make them highly parallelizable and consequently faster

in certain contexts. Given the interest aroused by this type of cryptography, several lattice-based protocols have been proposed like public key encryption schemes, digital signatures schemes, hash functions, identity-based encryption schemes or Zero-Knowledge Proofs of Knowledge.

In this section we give an introduction to lattices, focusing on those concepts that will be relevant for the understanding of further chapters. The organization is as follows: in Section 2.4.1 we define some basic concepts related with lattices. Then, in Section 2.4.2 we briefly explain what are Gaussian functions and distributions, since they play a central role in lattice-based cryptography. Finally, in order to describe some lattice-based cryptosystems in Section 2.4.5, we need first to introduce which are the computational problems we are going to work with to demonstrate the security of lattice-based constructions. This is done in Section 2.4.3.

2.4.1 Lattice basics

A lattice is a set of points in a n-dimensional space with a periodic structure (see Figure 2.11). More formally:

Definition 16 (Lattice)**.** A lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$:

- Discrete: $\exists \epsilon > 0$ s.t. $\forall \mathbf{x} \neq \mathbf{y} \in \mathcal{L}, \|\mathbf{x} - \mathbf{y}\| \geq \epsilon$
- Additive subgroup: $\forall \mathbf{x}, \mathbf{y} \in \mathcal{L}, \mathbf{x} - \mathbf{y} \in \mathcal{L}$

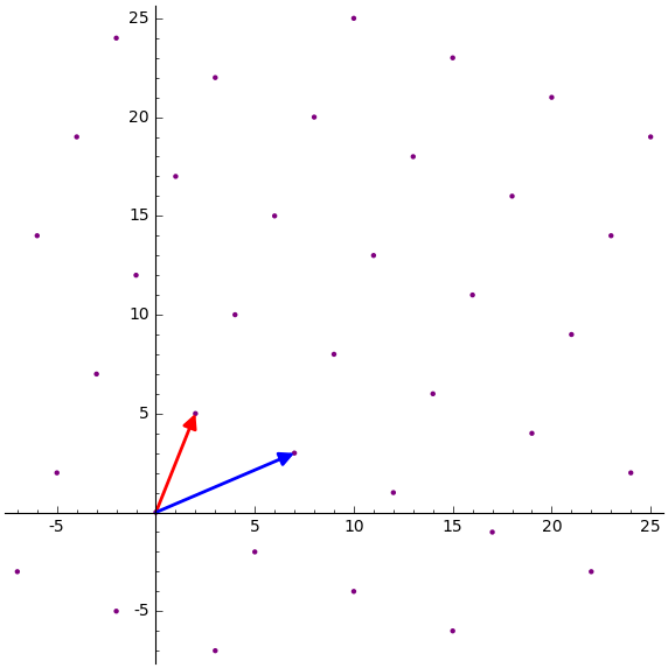


Figure 2.11: A two dimensional lattice generated by $\mathbf{b}_1 = (2, 5)$ and $\mathbf{b}_2 = (7, 3)$

Given n linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_n} \in \mathbb{R}^{m \times n}$, the lattice generated by them is the set

$$\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b_i} : x_i \in \mathbb{Z}\} = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}$$

of all integer linear combinations of the columns of $\mathbf{B}$. The matrix $\mathbf{B}$ is the **basis** of $\mathcal{L}$, and the integers m and n are called the **dimension** (number of components of each vector) and the **rank** (number of vector in the basis) of the lattice respectively. If $m = n$, $\mathcal{L}(\mathbf{B})$ is a **full-rank** lattice.

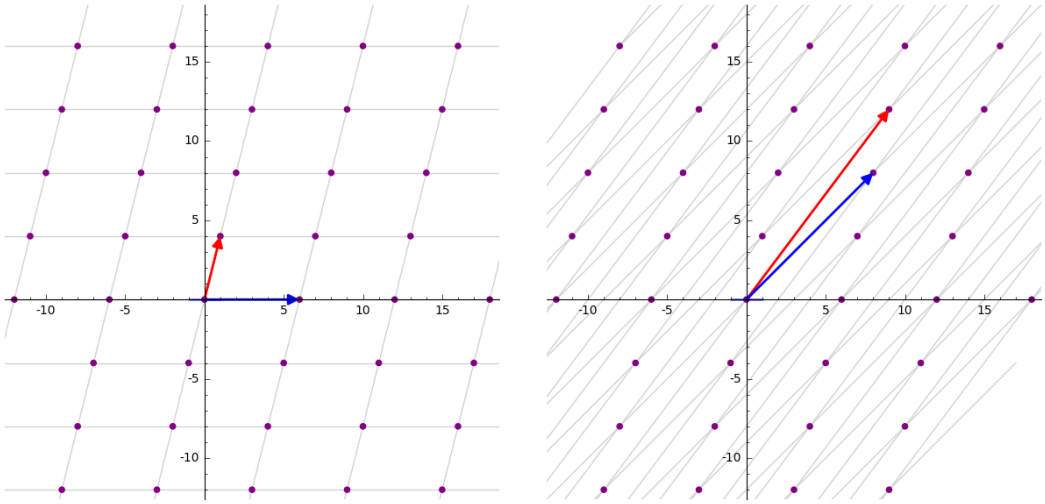
If instead of combining the columns of $\mathbf{B}$ using integers we use arbitrary real coefficients, we obtain the vector space generated by $\mathbf{B}$:

$$span(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{R}^n\}$$

The same lattice can be generated using different bases related by an unimodular transformation. In fact, the hardness of some lattice problems is based on the difficulty of transforming one basis to another of the same lattice.

Definition 17 (Unimodular matrix)**.** A square matrix $\mathbf{U} \in \mathbb{Z}^{n \times n}$ is unimodular if $det(\mathbf{U}) = \pm 1$.

Lemma 2.4.1 (Equivalent bases)**.** Two bases $\mathbf{B_1}, \mathbf{B_2} \in \mathbb{R}^{m \times n}$ are equivalent if and only if $\mathbf{B_2} = \mathbf{B_1} \cdot \mathbf{U}$ (see Figure 2.12).



(a) $\mathbf{b}_1 = (6, 0), \mathbf{b}_2 = (1, 4)$ (b) $\mathbf{b}_1 = (8, 8), \mathbf{b}_2 = (9, 12)$

Figure 2.12: A two dimensional lattice with two equivalent bases.

Each specific lattice basis is characterized by a fundamental parallelepiped:

Definition 18 (Fundamental parallelepiped)**.** Given a basis $\mathbf{B}$, we define the fundamental parallelepiped as:

$$\mathcal{P}(\mathbf{B}) = \{\sum_{i=1}^{n} c_i \mathbf{b_i} : 0 \leq c_i < 1\}$$

Or, equivalently, if we center the parallelpiped in the origin:

$$\mathcal{P}_{1/2}(\mathbf{B}) = \{\sum_{i=1}^{n} c_i \mathbf{b_i} : -1/2 \leq c_i < 1/2\}$$

Lemma 2.4.2. Let $\mathcal{L}$ be a full-rank lattice and let $\mathbf{b_1}, \ldots, \mathbf{b_n}$ be a set of n independent linear vectors of $\mathcal{L}$. Then $\mathbf{B}$ is a basis of the lattice if and only if $\mathcal{P}(\mathbf{B})$ does not contain any non-zero lattice point: $\mathcal{P}(\mathbf{B}) \cap \mathcal{L} = \{0\}$.

Figure 2.13 shows the fundamental parallelepiped corresponding to the lattice on Figure 2.11. The fundamental parallelepipeds of different basis of the same lattice are related by the determinant.

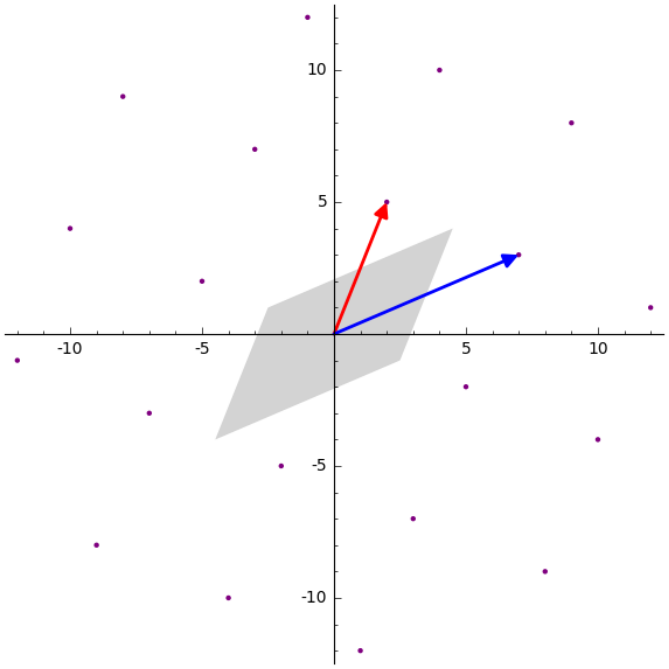


Figure 2.13: In grey, the fundamental parallelepiped corresponding to a two dimensional basis $\mathbf{b}_1 = (2, 5)$ and $\mathbf{b}_2 = (7, 3)$.

Definition 19 (Determinant)**.** The determinant $det(\mathcal{L}(\mathbf{B}))$ of a lattice is defined as the n-dimensional volume of the fundamental parallepiped of $\mathbf{B}$:

$$det(\mathcal{L}) := \sqrt{det(\mathbf{B}^T\mathbf{B})}$$

If $\mathcal{L}$ is a full-rank lattice, $det(\mathcal{L}) = |det(\mathbf{B})|$. We say that the determinant is well-defined since its value is unique per lattice, i.e., it does not depend on the choice of the basis and, consequently, the volume of all the fundamental parallelepipeds is the same.

$$\sqrt{det(\mathbf{B_1}^T\mathbf{B_1})} = \sqrt{det((\mathbf{B_2}\cdot\mathbf{U})^T\mathbf{B_2}\cdot\mathbf{U})} = \sqrt{det(\mathbf{U}^T\mathbf{B_2}^T\mathbf{B_2}\mathbf{U})} = \sqrt{det(\mathbf{B_2}^T\mathbf{B_2})}$$

The determinant of a lattice can also be represented using the Gram-Schmidt orthogonalization of $\mathbf{B}$:

$$det(\mathcal{L}(\mathbf{B})) = \prod_i \|\mathbf{b}_i^*\| \tag{2.1}$$

Definition 20 (Gram-Schmidt orthogonalization)**.** This basic procedure in linear algebra takes an ordered set of linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$ and creates

the set of n vectors orthogonal to them $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ via an iterative process: the first vector is defined as $\mathbf{b}_1^* = \mathbf{b}_1$ and for $i = 2, \ldots, n$, $\mathbf{b}_i^*$ is defined as the component of $\mathbf{b}_i$ orthogonal to $span(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}) = span(\mathbf{b}_1^*, \ldots, \mathbf{b}_{i-1}^*)$, i.e., $\langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle = 0$ for $i \neq j$:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* \quad \text{for} \quad \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$$

If we define the orthogonal basis $\mathbf{B}^*$ as the matrix with columns $\{\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*\}$, it is easy to see that $\mathbf{B}^*$ satisfies that $\mathbf{B} = \mathbf{B}^*\mathbf{R}$, where $\mathbf{R}$ is

$$\mathbf{R} = \begin{pmatrix} 1 & \mu_{2,1} & \mu_{3,1} & \ldots & \mu_{n,1} \\ 0 & 1 & \mu_{3,2} & \ldots & \mu_{n,2} \\ 0 & 0 & 1 & \ldots & \mu_{n,3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

If $\mathcal{L}$ is a full-rank lattice ($n = m$), $\mathbf{R}$ is an upper triangular square matrix with 1's on the diagonal so $|det(\mathbf{R})| = 1$ and consequently $|det(\mathbf{B})| = |det(\mathbf{B}^*)|$. Since $(\mathbf{B}^*)^\top \mathbf{B}^*$ is diagonal because the columns of $\mathbf{B}^*$ are orthogonal, the determinant of $\mathbf{B}^*$ can be computed as the product of the diagonal elements $\prod_i \|\mathbf{b}_i^*\|$, and from there we can conclude 2.1.

An upper bound on the determinant is given by the Hadamard inequality:

Theorem 2.4.3 (Hadamard inequality)**.** For any lattice $\mathcal{L}(\mathbf{B})$, $det(\mathcal{L}(\mathbf{B})) \leq \prod_{i=1}^{n} \|\mathbf{b}_i\|$.

$$\|\mathbf{b}_i\|^2 = \sum_{j=1}^{i-1} \mu_{j,i}^2 \|\mathbf{b}_j^*\|^2 + \|\mathbf{b}_i^*\|^2$$

$$\|\mathbf{b}_i\|^2 \geq \|\mathbf{b}_i^*\|^2$$

$$\|\mathbf{b}_i\| \geq \|\mathbf{b}_i^*\|$$

$$det(\mathcal{L}(\mathbf{B})) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\| \leq \prod_{i=1}^{n} \|\mathbf{b}_i\|$$

In addition to the concepts seen so far in this section, there is one basic parameter of a lattice which is the *minimum distance* $\lambda = \lambda_1$.

Definition 21 (Minimum distance)**.** The minimum distance of a lattice $\mathcal{L}$ corresponds to the length of the shortest vector of the lattice (shown in Figure 2.14):

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \{0\}} \|\mathbf{v}\|$$

It can be equivalently defined as the minimum distance of two lattice points:

$$\lambda(\mathcal{L}) = inf\{\|\mathbf{x} - \mathbf{y}\| : \mathbf{x}, \mathbf{y} \in \mathcal{L}, \mathbf{x} \neq \mathbf{y}\}$$

Given the definition of the determinant, the Gram-Schmidt orthogonalization, the Hadamard inequality and the minimum distance, we can proceed to define some parameters that will be useful to measure the quality of a basis.

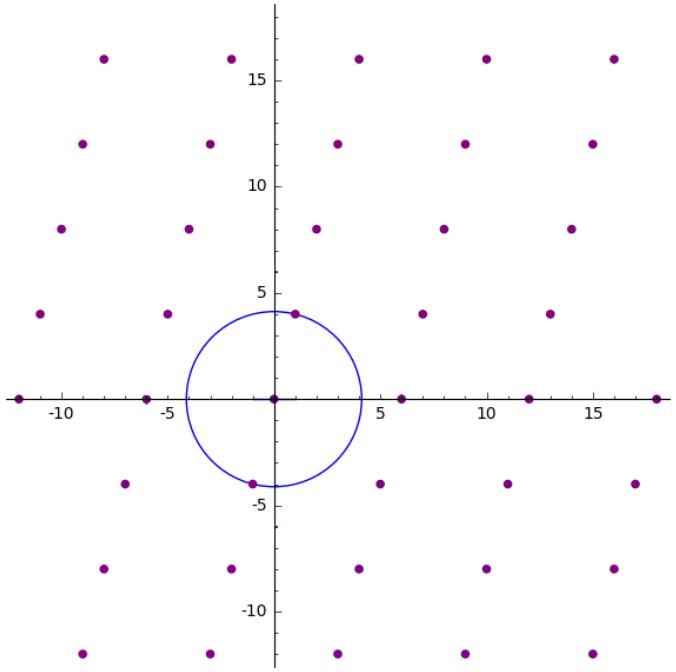


Figure 2.14: In blue the length of the shortest vector of the lattice $\lambda_1(\mathcal{L})$.

Definition 22 (Orthogonality defect)**.** Given a basis $\mathbf{B} = [\mathbf{b_1}, \ldots, \mathbf{b_n}]$ of a lattice, the orthogonality defect is defined as $\delta(\mathbf{B}) = \frac{\prod_i \|b_i\|}{|det(\mathbf{B})|}$ and is used to quantify the orthogonality of a lattice basis.

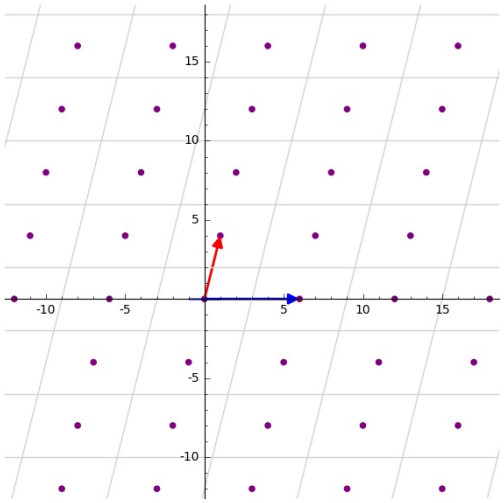
Informally, we can say that this parameter indicates how close is a basis from its orthogonal. Note that $\delta(\mathbf{B}) \geq 1$ and $\delta(\mathbf{B}) = 1$ if and only if $\mathbf{b_1}, \ldots, \mathbf{b_n}$ are pairwise orthogonal. The difficulty of solving most lattice problems is proportional to the orthogonality of its basis. As we have explained before, one lattice can be represented by several basis, all of them equivalent but not with the same orthogonality defect. The more orthogonality the basis has, the better is (see Figure 2.15). It is said that a basis with short highly orthogonal vectors, i.e., with low orthogonality defect, is a *good basis*, and when the defect is high it is a *bad basis*. In order to show an application of these concepts, we use as an example the GGH cryptosystem [77]. In this encryption scheme the public and private key are two basis of the same lattice. The private key is a good basis and allows to efficiently solve the Closest Vector Problem (CVP, Definition 32) thus allowing to decrypt the messages. On the other hand, the public key is a bad basis and from it no information about the private key can be extracted.

Another parameter that is useful to measure the quality of a basis is the *Hermite factor* $\delta_{\mathcal{L}}^n$:

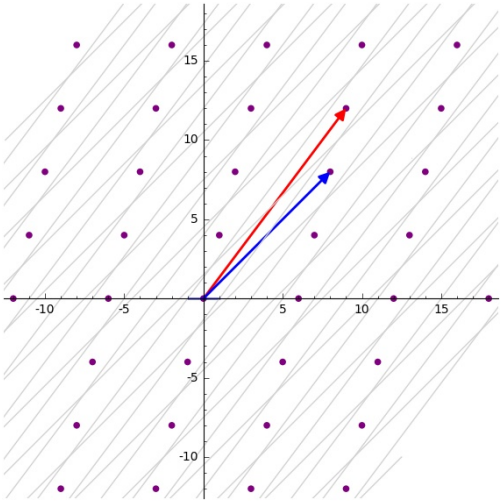
$$\delta_L^n = \frac{\lambda_1(\mathcal{L})}{|det(\mathcal{L}(\mathbf{B}))|^{1/n}}$$

or its n-th root $\delta_L = (\delta_L^n)^{1/n}$, where $\lambda_1(\mathcal{L})$ is the shortest vector of the lattice basis $\mathbf{B}$.

There is a special basis of a lattice, called the *Hermite Normal Form*, which can be efficiently computed from any other basis.



(a) Good basis: Lattice basis consisting of short lattice vectors. Basis with low orthogonality defect.



(b) Bad basis: Lattice basis consisting of long and highly non-orthogonal lattice vectors. Basis with high orthogonality defect.

Figure 2.15: Lattice represented using a good basis and a bad basis

Definition 23 (Hermite Normal Form)**.** A non-singular squared matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{n \times n}$ is in its Hermite Normal From if and only if:

- $\mathbf{B}$ is lower triangular ($b_{i,j} \neq 0 \rightarrow i \geq j$).
- Off-diagonal elements are reduced modulo the diagonal element on the row they are in $\forall i > j, 0 \leq b_{i,j} < b_{i,i}$.

We can also generalize the definition for non-squared matrices:

Definition 24. A non-singular matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_m] \in \mathbb{R}^{m \times n}$ is in its Hermite Normal From if and only if:

- Exist $1 \leq i_1 < \ldots < i_h \leq m$ such that $b_{i,j} \neq 0 \rightarrow (j < h) \wedge (i \geq i_j)$.
- Elements belonging to rows i_j are reduced modulo $b_{i_j,j}$: $\forall k > j, 0 \leq b_{i_j,k} < b_{i_j,j}$.

The index h is the number of non-zero columns and i_j corresponds to the row of the first non-zero element in column j. Every integer lattice $\mathcal{L}(\mathbf{B})$ has a unique basis in Hermite Normal Form and it is useful when solving some theoretic problems such as equality between two lattices.

The last two concepts we are going to define in this section are the dual lattice and the q-ary lattice, which are used in most of lattice-based cryptosystems.

Definition 25 (Dual lattice)**.** For a full-rank lattice $\mathcal{L}$, the dual lattice $\mathcal{L}^*$ of $\mathcal{L}$ is defined as:

$$\mathcal{L}^* = \{\mathbf{y} \in \mathbb{R}^n | \forall \mathbf{x} \in \mathcal{L}, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$$

The dual lattice is indeed a lattice formed by the set of points whose inner products with the vectors in $\mathcal{L}$ are all integers.

Definition 26 (Dual basis)**.** Given a basis $\mathbf{B} \in \mathbb{R}^{m \times n}$ of $\mathcal{L}$, the dual basis $\mathbf{D}$ of $\mathcal{L}^*$ is defined as $\mathbf{D} = \mathbf{B}(\mathbf{B^T B})^{-1}$.

We claim that:

- If $\mathbf{D}$ is the dual basis of $\mathbf{B}$, then $(\mathcal{L}(\mathbf{B}))^* = \mathcal{L}(\mathbf{D})$.
- The dual of the dual of a lattice is the original lattice: $(\mathcal{L}^*)^* = \mathcal{L}$.
- For any lattice $\mathcal{L}$, $det(\mathcal{L}^*) = 1/det(\mathcal{L})$.

Definition 27 (q-ary lattice)**.** A q-ary lattice is a lattice $\mathcal{L}$ that satisfies that $q\mathbb{Z} \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$ for some prime q.

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

$$\mathcal{L}_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{A}^T\mathbf{s} \mod q \quad \mathbf{s} \in \mathbb{Z}_q^n\}$$

$$\mathcal{L}_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = 0 \mod q\}$$

$\mathcal{L}_q(\mathbf{A})$ is generated by the rows of $\mathbf{A}$ and $\mathcal{L}_q^{\perp}(\mathbf{A})$ is the parity-check lattice because it contains all vectors that are orthogonal modulo q to the rows of $\mathbf{A}$ ($\mathbf{A}$ acts as a parity-check matrix that defines the lattice $\mathcal{L}_q^{\perp}(\mathbf{A})$). These two lattices are periodic modulo q, i.e., we can take a finite set Q of lattice points with coordinates in $0, \ldots, q-1$ and recover the whole lattice by generating copies of Q as: $Q + q\mathbb{Z}^n$.

Note that usually the matrix $\mathbf{A}$ is not a lattice basis. For example, an integer linear combination of its columns may not generate all the elements of the form $q\mathbb{Z}^n$. If for example we consider a two dimension q-ary lattice with vectors $\mathbf{b}_1 = (2, 5), \mathbf{b}_2 = (7, 3)$ (same vectors as those used in Figure 2.11), there is no integer solution to the following equation will allows us to compute the point $[q, 0]$:

$$17 = 2z_1 + 7z_2$$
$$0 = 5z_1 + 3z_2$$

Note also that every q-ary lattice is a full rank lattice since it contains $q\mathbb{Z}^n$. It is easy to see that, with high probability, a q-ary lattice defined by n vectors linearly independent in $\mathbb{Z}_q^n$ with q prime is the same as $\mathbb{Z}^n$. We illustrate this fact in Figure 2.16.

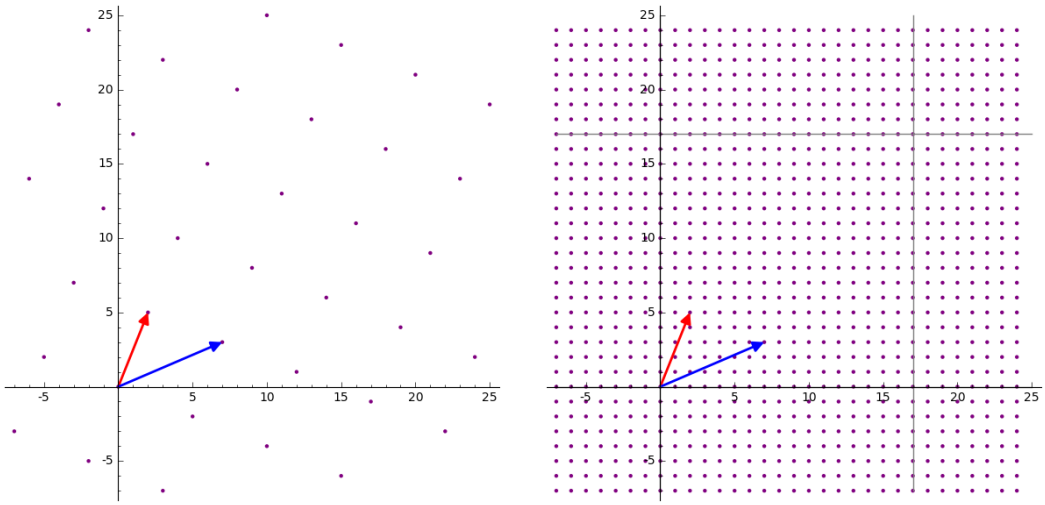
2.4.2 Gaussian Functions and Distributions

Gaussian distributions play a central role in lattice-based cryptography since they are used to build most of the cryptosystems.

The continuous Gaussian distribution over $\mathbb{R}$ is defined by the density function in the following way:

$$\rho_{\sigma,\mu}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where $\mu \in \mathbb{R}$ is the mean and σ the standard deviation. Then, the discrete Gaussian distribution over $\mathbb{Z}$ centered at $\mu = 0$ is defined as:



(a) Using $\mathcal{L}$ form (b) Using $\mathcal{L}_q(\mathbf{A})$ form with $q = 17$

Figure 2.16: Lattices generated using the same pair of vectors but the one on the right has been generated using the $\mathcal{L}_q(\mathbf{A})$ form with $q = 17$.

$$D_\sigma(x) = \frac{\rho_\sigma(x)}{\rho_\sigma(\mathbb{Z})}$$

where $\rho_\sigma(\mathbb{Z}) = \sum_{z\in\mathbb{Z}} \rho_\sigma(z)$. Note that when $\mu = 0$ it is omitted in the subscript. These definitions are also formulated using the Gaussian parameter $s = \sigma\sqrt{2\pi}$.

An example of the usage of Gaussian distributions is found in the RLWE encryption scheme (Section 2.4.5.3). This scheme requires to sample some error from a discrete Gaussian distribution in order to generate the public and private keys and also to encrypt a message. If instead of a single error we need a vector of errors to be chosen from the Gaussian distribution, as it is the case for the commitment scheme (Section 2.4.5.4) we write $\mathbf{e} \xleftarrow{\$} D_\sigma^k$ so each component of is chosen independently from D_σ.

Finally, there is a parameter related to Gaussians measures on lattices called *smoothing parameter*, which is a key concept in the best known worst-case/average-case reductions for lattice problems and several lattice-based cryptosystems. Imagine that we take a lattice $\mathcal{L}$ and we add a Gaussian with a certain standard deviation σ to each lattice point. As shown in Figure 2.17, once σ becomes large enough the Gaussian distribution is statistically close to a uniform distribution. But what does it mean that σ is large enough? Precisely this is what is quantified by the smoothing parameter. Informally, this parameter tells us how large $s = \sigma\sqrt{2\pi}$ should be in order for the distribution to become close to uniform. More formally:

Definition 28 (Smoothing parameter[112])**.** For an n-dimensional lattice $\mathcal{L}$ and positive real $\epsilon > 0$, we define its smoothing parameter $\eta_\epsilon(\mathcal{L})$ to be the smallest s such that $\rho_{1/s}(\mathcal{L}^*\backslash\{\mathbf{0}\}) \leq \epsilon$.

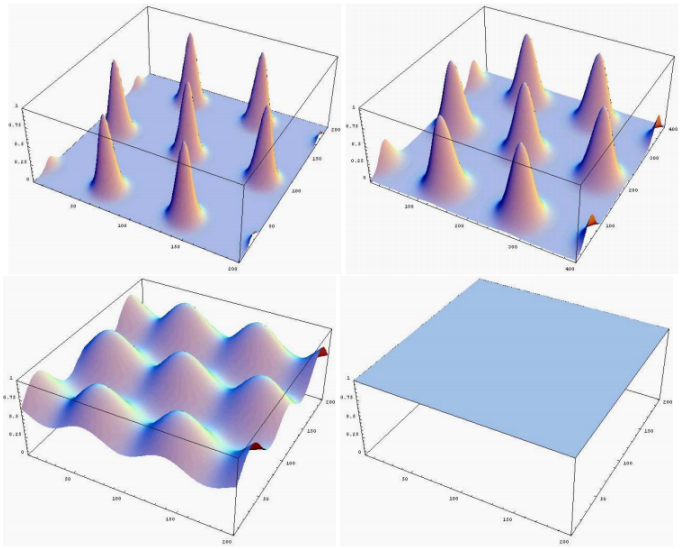


Figure 2.17: A lattice distribution perturbed with Gaussian noise using four different values of σ.

2.4.3 Lattice problems

The security of a lattice-based cryptosystem relies on the hardness of solving some computational problems on lattices which are considered secure against quantum computers, e.g., Shortest Vector Problem (SVP), Closest Vector Problem (CVP), Shortest Independent Vector Problem (SIVP) or Bounded Distance Decoding Problem (BDD). These problems are hard to solve in the worst-case, meaning that in order for an adversary to break them it must succeed on solving the problem on all its instances with non-negligible probability. Nevertheless, cryptographic schemes requires average-case hardness instead of worst-case hardness, i.e., problems for which random instances (a non-negligible portion) are hard to solve. In order to demonstrate that a lattice-based cryptographic protocol enjoys strong security guarantees, it is shown that the average-case problem is at least as hard as the arbitrary instances of a worst-case problem. Ajtai [12] was the first proposing a worst-case to average-case reduction for a lattice problem.

Most of the computational problems we describe hereunder exist in their exact and approximate version. We are going to define the approximate version whenever is possible since the exact one is just a particularization when the approximation factor $\gamma(n)$ is equals to 1 (being n the dimension of the lattice). It is demonstrated that if $\gamma(n)$ is a polynomial of the dimension of the lattice, computational problems are hard to solve.

Definition 29 (Approximate Shortest Vector Problem ($\gamma - SVP$))**.** Given a basis $\mathbf{B}$ of a n-dimensional lattice $\mathcal{L}(\mathbf{B})$, find a lattice vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $0 < \|\mathbf{v}\| \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$.

When this problem is defined in terms of the dual lattice, i.e., finding short vectors in the dual lattice $\mathcal{L}_q^{\perp}(\mathbf{A})$, it is called Short Integer Solution (SIS) (see Definition 34).

Definition 30 (Decisional Approximate Shortest Vector Problem ($GapSVP_\gamma$))**.** Given a basis $\mathbf{B}$ of a n-dimensional lattice $\mathcal{L}(\mathbf{B})$ where either $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$, decide which is the case.

Definition 31 (Approximate Shortest Independent Vector Problem ($SIVP_\gamma$))**.** Given a basis $\mathbf{B}$ of a full-rank n-dimensional lattice $\mathcal{L}(\mathbf{B})$, find a set $\mathbf{S} = \{\mathbf{s}_i\} \subset \mathcal{L}$ of n linearly independent lattice vectors such that $\|\mathbf{s}_i\| \leq \gamma(n) \cdot \lambda_n(\mathcal{L})$ for all i.

Definition 32 (Approximate Closest Vector Problem (CVP_γ))**.** Given a basis $\mathbf{B}$ of a n-dimensional lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t} \in \mathbb{R}^n$, find a lattice point $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{v}\| \leq \gamma(n) \cdot dist(\mathbf{t}, \mathcal{L})$.

We define $dist(\mathbf{t}, \mathcal{L})$ as the distance from the point $\mathbf{t}$ to the closest point in the lattice $\mathcal{L}$. There is a variant of this problem called Bounded Distance Decoding Problem in which the target point is guaranteed to be close to the lattice.

Definition 33 (Bounded Distance Decoding Problem (BDD_γ))**.** Given a basis $\mathbf{B}$ of a n-dimensional lattice $\mathcal{L}(\mathbf{B})$ and a target point $\mathbf{t} \in \mathbb{R}^n$ with the guarantee that $dist(\mathbf{t}, \mathcal{L}) < d = \lambda_1(\mathcal{L})/(2\gamma(n))$, find the unique lattice vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{t} - \mathbf{v}\| < d$.

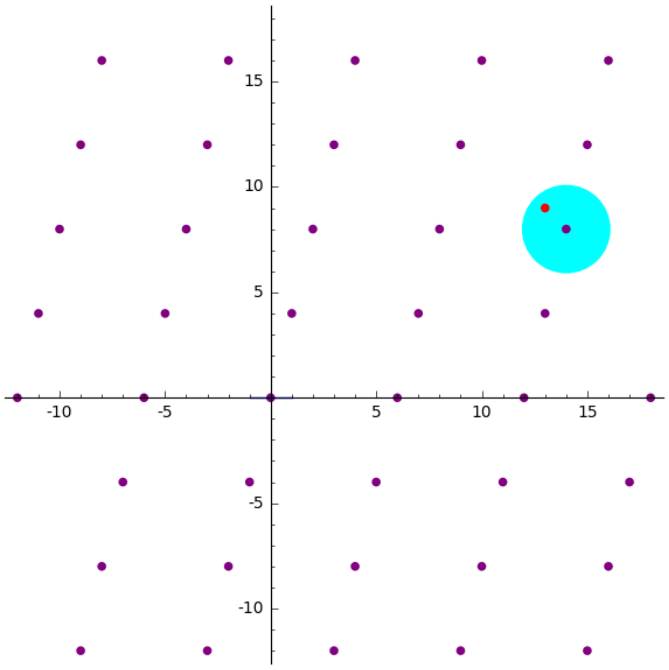


Figure 2.18: Example where $dist(\mathbf{t}, \mathcal{L}) < d = \lambda_1(\mathcal{L})/2$. The red point is the target point.

There are two main average-case problems when working with lattices which have a reduction from one of the problems presented above, i.e., the existence of an adversary that can break the average-case problem can be directly translated to breaking the worst-case problem. The first problem is called the Short Integer Solution (SIS), introduced by Ajtai [12] and used in the construction of cryptographic

primitives such as one-way and collision-resistant hash functions or digital signatures. The second one, which is considered as the 'dual' of the SIS problem, is the Learning With Errors problem that was introduced by Regev [128] in 2005. It has been used as the basis of public-key encryption schemes, identity-based encryption schemes and more. We are going to work mainly with this problem.

Definition 34 (Short Integer Solution (SIS))**.** Given m uniformly random vectors $a_i \in \mathbb{Z}_q^n$ as columns of the matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero integer vector $\mathbf{z} \in \mathbb{Z}^m$ with norm $\|\mathbf{z}\| \leq \beta$ such that

$$f_{\mathbf{A}(\mathbf{z})} = \mathbf{A}\mathbf{z} = \sum_i \mathbf{a}_i \cdot z_i = \mathbf{0} \in \mathbf{Z}_q^n$$

Definition 35 (Learning With Errors (LWE) distribution)**.** Let n and q (possibly prime) be two positive integers, χ an error distribution over $\mathbb{Z}$ (usually a discrete Gaussian distribution) and $\mathbf{s} \in \mathbb{Z}_q^n$ a secret vector. We denote $\mathcal{L}_{s,\chi}$ as the probability distribution over $\mathbb{Z}_q^n \times \mathbb{Z}$ sampled by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, $e \leftarrow \chi$ and outputting $(\mathbf{a}, b = \langle s, \mathbf{a} \rangle + e \mod q)$.

It is often convenient to represent several LWE samples in a compact manner by using vectors and matrices in the following way: $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$.

There are two versions of the LWE problem: the *search* version which consists on finding the secret vector $\mathbf{s}$ given several LWE samples, and the *decision* version, where the goal is to distinguish between LWE samples or samples chosen uniformly at random. For any version of the problem, the number m of samples available is a polynomial of n.

Definition 36 (Search-LWE)**.** Given $m = \mathsf{poly}(n)$ independent samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ drawn from $\mathcal{L}_{s,\chi}$ for a uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$, find $\mathbf{s}$.

Definition 37 (Decisional-LWE)**.** Given $m = \mathsf{poly}(n)$ independent samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, decide if they are distributed according to $\mathcal{L}_{s,\chi}$ for a uniformly random $\mathbf{s} \in \mathbb{Z}_q^n$ or the uniform distribution $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

These two versions of the problem are equivalent under certain conditions over the modulus q and the Gaussian parameter s. The search to the decision version reduction is trivial since if the secret $\mathbf{s}$ is found it can be used to verify if the component b belongs to a LWE sample by checking that $\mathbf{e} = \mathbf{b} - \mathbf{A}\mathbf{s}$ is small. The reduction from decision to search is not trivial and it is demonstrated using the following lemma:

Lemma 2.4.4. [128] Given $n \geq 1$ an integer, $2 \leq q \leq \mathsf{poly(n)}$ a prime and χ an error distribution in $\mathbb{Z}_p$. Assume that we have access to an algorithm $\mathcal{W}$ that for all $\mathbf{s}$ accepts with probability exponentially close to 1 given samples of the distribution $\mathcal{L}_{s,\chi}$, and rejects with probability exponentially close to 1 given some samples drawn from the uniform distribution $\mathcal{U}$. Then, there exist an efficient algorithm $\mathcal{W}'$ that, given samples from $\mathcal{L}_{s,\chi}$ for some $\mathbf{s}$, outputs $\mathbf{s}$ with probability exponentially close to 1.

There is a quantum reduction [128] and a classical reduction [122] from the worst-case hardness of the GapSVP problem to the search version of the LWE problem. The latter only works when the modulus q is exponential in the dimension of the lattice. Additionally, the decisional version of the LWE problem becomes no easier to solve even if the secret $\mathbf{s}$ is chosen from χ rather than uniformly. To prove this the following lemma is used:

Lemma 2.4.5. [16] Given access to an oracle $\mathcal{L}_{s,\chi}$ returning samples of the form $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $e \leftarrow \chi$ and $\mathbf{s} \in \mathbb{Z}_q^n$, we can construct samples of the form $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{e} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\mathbf{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$, $e \leftarrow \chi$ and $\mathbf{e} \leftarrow \chi^n$ at the loss of n samples overall. This is also called the normal form in [123].

Finally, if we see $\mathbf{A}$ as the basis of a lattice the search problem consists on recovering the coordinates of a lattice point after adding it some error, and the decision problem is to distinguish between uniformly random points in $\mathbb{Z}^n$ and perturbed lattice points.

2.4.4 Ideal lattices

Some lattice-based cryptographic schemes tend to require key sizes on the order n^2 due to the dimension of the basis $\mathbf{A}$, making working with lattices not desirable from a practical point of view. One way to solve this issue is to use *ideal lattices*, that have some extra algebraic structure and introduce some redundancy in the basis of the lattice, allowing a more compact representation and thus reducing significantly the storage space. Ideal lattices are a generalization of *cyclic lattices* which are defined as follows:

Definition 38 (Cyclic lattice)**.** $\mathcal{L}$ is a *cyclic lattice* if it is a discrete set and for any $\mathbf{v}, \mathbf{w} \in \mathcal{L}$: 1) $\mathbf{v} + \mathbf{w} \in \mathcal{L}$, 2) $-\mathbf{v} \in \mathcal{L}$ and 3) a cyclic shift of $\mathbf{v}$ is also in $\mathcal{L}$.

For a prime positive integer q, let $\mathbb{Z}_q^n = (\mathbb{Z}/q\mathbb{Z})^n$ denote the quotient ring of vectors whose coefficients are integers modulo q. Working on cyclic lattices in $\mathbb{Z}_q^n$ is equivalent of working on ideals in $R_q = \mathbb{Z}_q[x]/\left(x^n - 1\right)$, that is, the ring of all integer polynomials modulo $f(x) = x^n - 1 \in \mathbb{Z}[x]$, where x^n is identified with 1. Let us briefly explain this equivalence: the basis $\mathbf{A}$ of a cyclic lattice in $\mathbb{Z}_q^n$ can be constructed by taking a vector $\mathbf{a} \in \mathbb{Z}_q^n$ and making it the first column of $\mathbf{A}$. The next $n-1$ columns are generated by applying consecutive rotation of $\mathbf{a}$ in the following way:

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

So $\mathbf{A}$ is the matrix whose columns are cyclic rotations of $\mathbf{a}$. Then, the operation of multiplying the circulant matrix $\mathbf{A}$ by a vector $\mathbf{s} \in \mathbb{Z}_q^n$ is equivalent to multiplying the polynomial $a(x) = \sum_{i=1}^n a_i x^{i-1}$ by $s(x) = \sum_{i=1}^n s_i x^{i-1}$ modulo $f(x) = x^n - 1$.

This can be seen in the following example for $n = 3$:

$$\mathbf{As} = \begin{pmatrix} a_1 & a_3 & a_2 \\ a_2 & a_1 & a_3 \\ a_3 & a_2 & a_1 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} a_1 s_1 + a_3 s_2 + a_2 s_3 \\ a_2 s_1 + a_1 s_2 + a_3 s_3 \\ a_3 s_1 + a_2 s_2 + a_1 s_3 \end{pmatrix}$$

We do now the same operation but with polynomials and considering that if we use $f(x) = x^3 - 1$ as the modulus, this element will be 0 thus $x^3 = 1$:

$$\begin{aligned} &a(x)s(x) \mod f(x) = \\ &(a_1 + a_2 x + a_3 x^2)(s_1 + s_2 x + s_3 x^2) \mod (x^3 - 1) = \\ &a_1 s_1 + a_1 s_2 x + a_1 s_3 x^2 + a_2 s_1 x + a_2 s_2 x^2 + a_2 s_3 x^3 + \\ &\quad + a_3 s_1 x^2 + a_3 s_2 x^3 + a_3 s_3 x^4 \mod (x^3 - 1) = \\ &(a_1 s_1 + a_2 s_3 + a_3 s_2) + (a_1 s_2 + a_2 s_1 + a_3 s_3)x + (a_1 s_3 + a_2 s_2 + a_3 s_1)x^2 \end{aligned}$$

So given this equivalence we can represent an element $a \in \mathcal{L}$ either as a vector $(a_1, \ldots, a_n) \in \mathbb{Z}_q^n$ or as a polynomial $a_1 + a_2 x + \ldots + a_n x^{n-1} \in R_q$.

After giving an intuition of what an ideal lattice is, now we are going to define it formally:

Definition 39 (Ideal lattice)**.** $\mathcal{L}$ is an ideal lattice if it has as a basis a matrix $\mathbf{A}$ constructed from a vector $\mathbf{a}$ iteratively multiplied by a transformation matrix $\mathbf{F} \in \mathbb{Z}^{n \times n}$ defined from a vector $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) \in \mathbb{Z}^n$:

$$\mathbf{A} = \mathbf{F}^* \mathbf{a} = [\mathbf{a}, \mathbf{Fa}, \ldots, \mathbf{F}^{n-1}\mathbf{a}] \quad \text{where} \quad \mathbf{F} = \left[\begin{array}{ccc|c} 0 & \ldots & 0 & -f_0 \\ \ddots & & & -f_1 \\ & I & & \vdots \\ & & \ddots & -f_{n-1} \end{array}\right]$$

These lattices can be seen as ideals in the polynomial ring $R_q = \mathbb{Z}_q[x]/(f(x))$, where $f(x) = x^n + f_{n-1}x^{n-1} + \ldots + f_0 \in \mathbb{Z}_q[x]$ is a polynomial given by the vector $\mathbf{f}$. As we have seen before, if we choose $f(x)$ to be $f(x) = x^n - 1$, this is equivalent as working on cyclic lattice in $\mathbb{Z}_q^n$, nevertheless some ring versions of lattice problems are easy to solve in rings where $f(x) = x^n - 1$, since it is factorizable. For this reason and following what is proposed in [107], we are going to work with $f(x) = x^n + 1$ for n a power of 2, which makes the polynomial irreducible over the rationals. This is a *cyclotomic* polynomial and the ring $R_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ generates the family of the so-called *anti-cyclic integer lattices*, i.e., lattices in $\mathbb{Z}_q^n$ that are closed under the operation that cyclically rotates the coordinates and negates the cycled elements.

Notice that the polynomial $f(x) = x^n + 1$ can be expressed also as the vector $\mathbf{f} = (f_0, \ldots, f_{n-1}) = (1, 0, \ldots, 0)$ and we can define from it a transformation matrix $\mathbf{F} \in \mathbb{Z}^{n \times n}$ as:

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & \ldots & 0 & -1 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \vdots & 1 & 0 \end{bmatrix}$$

Using this transformation matrix, the basis $\mathbf{A}$ is pretty similar to that for cyclic lattices, i.e., its first column is the vector $\mathbf{a}$ and the following columns are the previous one with the coordinates cyclically rotated, but with the difference that the cycled element is also negated:

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

It is important to remind that given two polynomials, $a \in R_q$ and $p \in R_q$, the product $a \cdot p \in R_q$ is equivalent to the product of the matrix $\mathbf{A}$ with the vector $\mathbf{p} = (p_1, \dots, p_n)^\top$.

Note that using ideal lattices we only need n values to express a rank n ideal lattice, i.e., we can generate the matrix $\mathbf{A}$ just using the vector $\mathbf{a}$, rather than the $n \times n$ values needed for general lattices. This allows a more compact representation that requires less space.

Moreover, working with the polynomial representation in R_q with certain modules allows a speed-up in operations commonly used in lattice-based schemes: polynomial multiplication can be performed in $\mathcal{O}(n \log n)$ scalar operations, and in parallel depth $\mathcal{O}(\log n)$, using the Fast Fourier Transform (FFT).

There is currently no known way to take advantage of the extra structure introduced by ideal lattices, and the running time required to solve lattice problems on such lattices is the same as that for general lattices.

2.4.4.1 RLWE problem

Lyubashebsky, Peikert and Regev [107] introduced in 2010 the ring-based variant of learning with errors problem: Ring-LWE (RLWE). This was motivated by the necessity of constructing efficient LWE-based cryptosystems. Analogously to LWE, the goal will be either to distinguish random linear equations, perturbed by a small amount of noise, from truly uniform pairs or recover the secret $s \in R_q$ from arbitrarily many noisy products.

Definition 40 (RLWE Distribution)**.** For a secret $s \in R_q$, the RLWE distribution $\mathcal{A}_{s,\chi}$ over $R_q \times R_q$ is sampled choosing $a \in R_q$ uniformly at random, $e \xleftarrow{\$} \chi^n$ (i.e., $e \in R_q$ with its coefficients drawn from χ), and outputting samples of the form $(a, b = a \cdot s + e \mod q) \in R_q \times R_q$.

Using the definition of the matrix $\mathbf{A}$ for an ideal lattice: $\mathbf{A} = \mathbf{F}^*\mathbf{a}$, we can express an RLWE sample in the following way:

$$(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}) = (\mathbf{F}^*\mathbf{a}, \mathbf{b} = \mathbf{F}^*\mathbf{a} \cdot \mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$$

Given that $\mathbf{F}^*\mathbf{a} = [\mathbf{a}, \mathbf{F}\mathbf{a}, \dots, \mathbf{F}^{n-1}\mathbf{a}]$, we can divide the RLWE sample in n samples:

$$(\mathbf{F}^{(i)}\mathbf{a}, b^{(i)} = \mathbf{F}^{(i)}\mathbf{a} \cdot \mathbf{s} + e^{(i)}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \text{ where } i \in \{0, \dots, n-1\}$$

The result are LWE samples and we conclude that each pair $(a, b) \in R_q \times R_q$ of an RLWE distribution replaces n samples $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ of an standard LWE distribution.

Similarly to LWE, certain instantiations of RLWE are supported by worst-case hardness theorems [107], related to the Shortest Vector Problem (SVP). For the error distribution χ where the standard deviation $\sigma \geq \omega(\sqrt{\log n})$, and for any ring, there exist a quantum reduction from the $\gamma(n)$-SVP problem to the RLWE problem to within $\gamma(n) = \mathcal{O}(\sqrt{n} \cdot q/\sigma)$. Additionally, RLWE becomes no easier to solve even if the secret s is chosen from the error distribution rather than uniformly [16].

2.4.5 Lattice-based cryptosystems

There are several cryptographic constructions which are built upon the computational problems presented in Section 2.4.3. This gives strong security guarantees since breaking the cryptosystems implies an efficient algorithm for solving a lattice problem in the worst-case, i.e., for solving any instance of the underlying lattice problem. In this section we are going to focus on some of the cryptographic constructions that are necessary to build the lattice-based e-voting scheme.

It is worth to mention that the National Institute of Standards and Technology (NIST) initiated in December 2016 a process to solicit, evaluate and standardize one or more quantum-resistant public-key cryptographic algorithms. There have been two rounds of this process already completed and they are currently on the third round. Lattice-based candidates to be reviewed for consideration for standardization at the conclusion of the third round are: CRYSTALS-KYBER [31], NTRU [94] and SABER [54] as public-key encryption schemes or KEMs; and CRYSTALS-DILITHIUM [61] and FALCON [67] as digital signature schemes.

2.4.5.1 One-way functions

As we have already explained in Section 2.2 (Definition 3), a one-way function is a function that is easy to compute but hard to invert in terms of computations complexity. When working with lattices we distinguish between two lattice-based one way functions depending on the average-case problem their security is based on: SIS or LWE problem.

Given a public matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and the parameters $q = \mathsf{poly}(n)$ and $m = \Omega(n \log q)$, the one-way function $f_A(\mathbf{x})$ based on the SIS problem is defined as:

$$f_A(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \in \mathbb{Z}_q^n$$

where $\mathbf{x}$ is a short integer vector. Note that $f_A(\mathbf{x})$ is a surjective function since it is compressing. Indeed, $f_A(\mathbf{x})$ is a collision resistant hash function (see Definition 13) assuming the hardness of the SIS problem, as shown by Ajtai in [12].

On the other hand, the one-way function g_A based on the LWE problem, is defined as

$$g_A(\mathbf{s}, \mathbf{e}) = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \mod q \in \mathbb{Z}_q^m$$

where $\mathbf{e}$ is a short integer vector. Unlike f_A, g_A is an injective function and it cannot be used to build a collision resistance hash function. For this reason we are

going to focus only on f_A. For our cryptographic applications (the lattice-based online voting system) we will be interested on inverting f_A. Since the function is compressing there are many pre-images given one image, so in order to invert it, i.e. compute $f_A^{-1}(\mathbf{u})$, we will need to sample random pre-images according to a discrete Gaussian distribution (see [72]). This type of function was defined in [72] as a Preimage Sampleable Trapdoor Functions.

2.4.5.2 Trapdoor functions

As explained in Section 2.2 (Definition 4), a trapdoor function is a one-way function that is easy to invert with the knowledge of a secret, the trapdoor.

In lattice-based cryptography there are problems that are hard to solve given an arbitrary basis but some of them become easy if we are given a good basis (the concept of good and bad basis of a lattice is explained in Section 2.4.1). In the literature there are two different notions of what is a lattice trapdoor: (1) a *short (good) basis* [13, 15], i.e., a basis made up of short lattice vectors; or (2) a *gadget-based* trapdoor [111]. We are going to focus on the latter since, as mentioned by the authors, is simpler and faster than prior constructions. In [111] Micciancio and Peikert propose a new method for generating strong trapdoors in cryptographic lattices and they also give specialized algorithms for inverting g_A and preimage sampling for f_A. As previously mentioned we are interested on functions like f_A since they are collision-resistant hash functions (we will see in Chapter 5 how to use them in the lattice-based online voting system). The main idea of this trapdoor generation method is that the matrix $\mathbf{A}$ is generated from a matrix $\mathbf{G}$ that is public and for which we know that the associated function $f_{\mathbf{G}}$ is easy to invert (admit efficient preimage sampling). Given $\mathbf{G}$ and applying some transformations we obtain $\mathbf{A}$, that is distributed uniformly at random. Preimage sampling for $f_{\mathbf{A}}$ reduce to the corresponding tasks for $f_{\mathbf{G}}$. In more detail, the trapdoor generation method for hard random lattices $\mathcal{L}^{\perp}(\mathbf{A})$ works as follows:

1. Construct a fixed gadget matrix $\mathbf{G}$. This matrix is public and the associated function f_G can be efficiently inverted.

   $$f_G(\mathbf{x}) = \mathbf{G} \cdot \mathbf{x} \mod q$$

   Given the primitive vector $\mathbf{g}$[3], the matrix $\mathbf{G}$ is built as a tensor product of an identity matrix and the primitive vector $\mathbf{g}$.

   $$\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^{\top}$$

   The simplest gadget matrix is when q is a power of a small prime, such as $q = 2^k$. Then, the vector $\mathbf{g}$ is defined as $\mathbf{g} = (1, 2, 4, \ldots, 2^{k-1}) \in \mathbb{Z}_q^k$ where $k = \lceil \log_2 q \rceil$.

   $$\mathbf{G} = \begin{pmatrix} \cdots \mathbf{g}^{\top} \cdots & & & \\ & \cdots \mathbf{g}^{\top} \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g}^{\top} \cdots \end{pmatrix}$$

[3] A primitive vector $\mathbf{g}$ is a vector such that $\mathsf{gcd}(g_1, \ldots, g_k, q) = 1$. We refer the reader to [111] for the details about what is a primitive lattice and its corresponding primitive matrix.

2. Extend $\mathbf{G}$ into a semi-random matrix $\mathbf{A'} = [\mathbf{B}|\mathbf{HG}]$, for a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times \overline{m}}$ chosen uniformly at random and an invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$ called the *tag* of the trapdoor. For completeness we define the tag $\mathbf{H}$ but it can be set as the identity matrix most of the time. As shown in [37] inverting $f_{\mathbf{A}'}$, reduces very efficiently to inverting $f_{\mathbf{G}}$ and we give a brief explanation here. We want to invert the function $f_{\mathbf{A}'}(\mathbf{x}) = \mathbf{A'} \cdot \mathbf{x} = \mathbf{y}'$, i.e., to sample a preimage from $f_{\mathbf{A}'}^{-1}(\mathbf{y}')$. Note that we can also write $f_{\mathbf{A}'}$ as (for simplicity we assume that $\mathbf{H}$ is the identity matrix):

$$f_{\mathbf{A}'}\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} = [\mathbf{B}|\mathbf{G}]\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix} = \mathbf{B}\mathbf{x}_1 + \mathbf{G}\mathbf{x}_2 = \mathbf{y}'$$

First we choose a random $\mathbf{x}_1$ from the discrete Gaussian distribution and compute $f_{\mathbf{B}}(\mathbf{x}_1) = \mathbf{B}\mathbf{x}_1 = \overline{\mathbf{y}}$. Then, we sample a random preimage $\mathbf{x}_2$ from $f_{\mathbf{G}}^{-1}(\mathbf{y}' - \overline{\mathbf{y}}) = f_{\mathbf{G}}^{-1}(\mathbf{G}\mathbf{x}_2)$ under the appropriate Gaussian distribution using that $f_{\mathbf{G}}$ is easy to invert. Finally we define $\mathbf{x}$ as $\mathbf{x} = \begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{pmatrix}$.

3. Apply a random unimodular transformation $\mathbf{T}$ to $\mathbf{A'}$ and obtain the matrix $\mathbf{A}$.

$$\mathbf{A} = \mathbf{A'} \cdot \mathbf{T} = \mathbf{A'} \cdot \begin{pmatrix}\mathbf{I} & -\mathbf{R}\\ \mathbf{0} & \mathbf{I}\end{pmatrix} = [\mathbf{B}|\mathbf{HG} - \mathbf{BR}]$$

The transformation matrix $\mathbf{T}$ includes a short secret matrix $\mathbf{R}$ that will server as the trapdoor. We recall the definition of this trapdoor matrix $\mathbf{R}$ given in [111].

Definition 41 (Lattice-based trapdoor)**.** Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ be matrices with $m \geq w \geq n$. A $\mathbf{G}$-*trapdoor* for $\mathbf{A}$ is a matrix $\mathbf{R} \in \mathbb{Z}_q^{(m-w) \times w}$ such that $\mathbf{A}\begin{bmatrix}\mathbf{R}\\ \mathbf{I}\end{bmatrix} = \mathbf{HG}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. We refer to $\mathbf{H}$ as the *tag* or *label* of the trapdoor. The *quality* of the trapdoor is measured by its largest singular value $s_1(\mathbf{R})$.

As long as $\mathbf{R}$ has the right dimension, by the leftover hash lemma [92], ($\mathbf{B}$, $\mathbf{BR}$) is indistinguishable from ($\mathbf{B}$, $\mathbf{U}$), where $\mathbf{U} \in \mathbb{Z}_q^{n \times w}$, hence $\mathbf{A}$ is uniform. Observe that the inverse of $\mathbf{T}$ is $\mathbf{T}^{-1} = \begin{pmatrix}\mathbf{I} & \mathbf{R}\\ \mathbf{0} & \mathbf{I}\end{pmatrix}$ and we can invert $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A'} \cdot \mathbf{T} \cdot \mathbf{x} = \mathbf{y}$ by first inverting $f_{\mathbf{A}'}$, and then $\mathbf{T}$: $f_{\mathbf{A}}^{-1}(\mathbf{y}) = \mathbf{T}^{-1} f_{\mathbf{A}'}^{-1}(\mathbf{y}')$. Note that without the knowledge of the trapdoor $\mathbf{R}$ it is difficult to obtain preimages of $f_{\mathbf{A}}$.

2.4.5.3 Public-key encryption schemes

As explained in [113], there are several methods that have been proposed to build public-key encryption schemes based on the hardness of solving some lattice problems. Some of them are interesting from a theoretical point of view, since they admit security proofs which prove that breaking the scheme is as hard as solving lattice problems in the worst-case; nevertheless they are not efficient enough to be used in

practice. On the other hand, there are also public-key encryption schemes which are much more efficient than theoretical proposals but they often lack the support of a security proof.

Although there have been several lattice-based encryption schemes proposed so far, e.g., GGH/HNF [77], NTRU [94] or Ajtai-Dwork [14], we are going to focus on the RLWE encryption scheme proposed by Lyubashevsky *et al.* in [107], since it is that used for encrypting the voting options in the lattice-based e-voting scheme we propose in Chapter 5.

RLWE encryption scheme. The additive homomorphic RLWE encryption scheme proposed in [107] consists of three algorithms ($\mathsf{KeyGen_E}$, Enc, Dec) defined below. Let $R_q = \mathbb{Z}_q[x]/f(x)$ be the ring of integer polynomials modulo both $f(x) = (x^n+1)$ and q, n the security parameter which is a power of 2, $q = 1 \mod 2n$ a sufficiently large public prime modulus and κ the security parameter:

- $\mathsf{KeyGen_E}(1^\kappa)$: Given a uniformly random $a_\mathsf{E} \in R_q$ and two *small* elements $s, e \in R_q$ drawn from the error distribution χ, the public key is a RLWE sample $(a_\mathsf{E}, b_\mathsf{E}) = (a_\mathsf{E}, a_\mathsf{E} \cdot s + e) \in R_q \times R_q$ and the secret key is s.

- $\mathsf{Enc}_{a_\mathsf{E}, b_\mathsf{E}}(z, r_\mathsf{E}, e_{\mathsf{E},u}, e_{\mathsf{E},v})$: Given three random small elements $r_\mathsf{E}, e_{\mathsf{E},u}, e_{\mathsf{E},v} \in R_q$ drawn from the error distribution χ, the encryption of an n-bit message $z \in \{0,1\}^n$ (identified as a polynomial of degree $n-1$ with coefficients 0 or 1) is $(u, v) = (a_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},u} \mod q, b_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},v} + \lfloor \frac{q}{2} \rceil z \mod q) \in R_q \times R_q$.

- $\mathsf{Dec}(s, (u, v))$: Given the secret key and the ciphertext this algorithm computes:

$$\begin{aligned} v - u \cdot s &= r_\mathsf{E} \cdot b_\mathsf{E} + e_{\mathsf{E},v} - s(a_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},u}) + \lfloor \frac{q}{2} \rceil z \mod q \\ &= r_\mathsf{E} \cdot (a_\mathsf{E} \cdot s + e) + e_{\mathsf{E},v} - s(a_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},u}) + \lfloor \frac{q}{2} \rceil z \mod q \\ &= r_\mathsf{E} \cdot e - s \cdot e_{\mathsf{E},u} + e_{\mathsf{E},v} + \lfloor \frac{q}{2} \rceil z \mod q \\ &\approx \lfloor \frac{q}{2} \rceil z \end{aligned}$$

Notice that in case of lack of error the decryption would always be correct since the algorithm will return directly 0 or $\lfloor \frac{q}{2} \rceil$ depending on the encrypted bit. Given that, a decryption error will occur if the coefficients of $(r_\mathsf{E} \cdot e - s \cdot e_{\mathsf{E},u} + e_{\mathsf{E},v})$ have magnitude greater than $q/4$.

The cryptosystem can be generalized in order to encrypt messages with elements bigger than a bit, i.e., $z \in \{0, 1, \ldots, k-1\}^n$. In order to do so, we will map the n-symbol message z to a polynomial of R_q by scaling it with a factor of $\lfloor \frac{q}{k} \rceil$, instead of $\lfloor \frac{q}{2} \rceil$. In the decryption step, the symbols of z can be recovered by rounding each coefficient of $v - u \cdot s$ back to $i\lfloor \frac{q}{k} \rceil$ for $i = \{0, \ldots, k-1\}$ whichever is closest modulo q. Analogously to the case where $k = 2$, the symbols will be properly decrypted whenever the coefficients of $(r_\mathsf{E} \cdot e - s \cdot e_{\mathsf{E},u} + e_{\mathsf{E},v}) \in R_q$ have magnitude less than $q/2k$.

Due to the homomorphic property of the scheme we can compute the re-encryption just adding to the original ciphertext the encryption of the element 0, that is, the polynomial whose coefficients are all 0.

- $\mathsf{Re\text{-}enc}_{a_\mathsf{E},b_\mathsf{E}}((u,v), r'_\mathsf{E}, e'_{\mathsf{E},u}, e'_{\mathsf{E},v})$: Given the small elements $r'_\mathsf{E}, e'_{\mathsf{E},u}, e'_{\mathsf{E},v}$ drawn from the error distribution χ, the re-encryption of a ciphertext (u,v) is $(u',v') = (u,v) + \mathsf{Enc}_{a_\mathsf{E},b_\mathsf{E}}(0, r'_\mathsf{E}, e'_{\mathsf{E},u}, e'_{\mathsf{E},v}) \in R_q \times R_q$.

Explicitly, the re-encrypted ciphertext is (for simplicity we omit here the modulo q):

$$(u',v') = (u,v) + (a_\mathsf{E} \cdot r'_\mathsf{E} + e'_{\mathsf{E},u}, b_\mathsf{E} \cdot r'_\mathsf{E} + e'_{\mathsf{E},v})$$
$$(u',v') = (a_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},u}, b_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},v} + \left\lfloor \frac{q}{2} \right\rceil z) + (a_\mathsf{E} \cdot r'_\mathsf{E} + e'_{\mathsf{E},u}, b_\mathsf{E} \cdot r'_\mathsf{E} + e'_{\mathsf{E},v})$$
$$(u',v') = (a_\mathsf{E} \cdot (r_\mathsf{E} + r'_\mathsf{E}) + (e_{\mathsf{E},u} + e'_{\mathsf{E},u}), b_\mathsf{E} \cdot (r_\mathsf{E} + r'_\mathsf{E}) + (e_{\mathsf{E},v} + e'_{\mathsf{E},v}) + \left\lfloor \frac{q}{2} \right\rceil z)$$

So we can see that in (u',v') the randomness used for encrypting the message z is the sum of the randomness used during the encryption and the re-encryption.

RLWE encryption scheme and consequently the RLWE re-encryption scheme are semantically secure based on the RLWE assumption. It is demonstrated in [107] that if there exists a polynomial time algorithm that distinguishes between encryption of 0 and 1 then there exists a distinguisher that distinguishes between $\mathcal{A}_{s,\chi}$ and $\mathcal{U}(R_q)$ for a non-negligible fraction of all possible s.

Finally, as we have mention before, the RLWE encryption scheme is homomorphic so it allows to sum several ciphertexts and obtain one unique ciphertext which encrypts the sum of all the messages. If we represent each ciphertext in the following way:

$$u^{(i)} = a \cdot r^{(i)} + e_1^{(i)}$$

$$v^{(i)} = b \cdot r^{(i)} + e_2^{(i)} + \left\lfloor \frac{q}{2} \right\rceil z^{(i)}$$

The sum of all of them is:

$$u^{(\Sigma)} = \sum_i u^{(i)} = a \cdot \sum_i r^{(i)} + \sum_i e_1^{(i)} = a \cdot r^{(\Sigma)} + e_1^{(\Sigma)}$$

$$v^{(\Sigma)} = \sum_i v^{(i)} = b \cdot \sum_i r^{(i)} + \sum_i e_2^{(i)} + \sum_i \left\lfloor \frac{q}{2} \right\rceil z^{(i)} = b \cdot r^{(\Sigma)} + e_2^{(\Sigma)} + \left\lfloor \frac{q}{2} \right\rceil z^{(\Sigma)}$$

As we can see, the resulting ciphertext contains the sum of all messages but also the sum of all the individual errors. This means that the error linearly grows with the number of ciphertexts that are being added. For this reason, if we want to use the homomorphic properties of the encryption scheme, we should know which is the maximum number of ciphertexts we are going to aggregate, so we can choose in advance the appropriate parameters (s and q) in order to be able to do the sum but also to have a low or negligible error probability when decrypting.

2.4.5.4 Commitment scheme

From the proposals of lattice-based commitment schemes we choose that proposed by Benhamouda *et al.* [29] since apart from allowing us to commit to a message, it allows to prove knowledge of the committed messages and also relations between them. This scheme needs that the prime q is $q \equiv 3 \mod 8$. This implies $x^n + 1$ splits into two irreducible polynomials of degree $n/2$ [29], and every polynomial of degree smaller than $n/2$ can be inverted. There are other proposals for a lattice-based commitment scheme and the corresponding proofs such as that from Baum [20], but we leave their analysis for future work.

The commitment scheme consists of the following three algorithms:

- $\mathsf{KeyGen}_{\mathsf{C}}(1^{\kappa})$: this algorithm generates the public commitment key $ck = (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})$ where $\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}} \xleftarrow{\$} (R_q)^k$, $q \equiv 3 \mod 8$ is prime and n is a power of 2.

- $\mathsf{Com}_{ck}(m; r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}})$: in order to commit to a message $m \in R_q$, the algorithm chooses $r_{\mathsf{C}} \xleftarrow{\$} R_q$ and $\mathbf{e}_{\mathsf{C}} \xleftarrow{\$} D^k_{\sigma_e}$, where σ_e is the standard deviation of the error used for computing the commitment, conditioned on $\|\mathbf{e}_{\mathsf{C}}\|_\infty \leq n$ and computes:

$$\mathbf{c} = \mathsf{Com}_{\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}}}(m; r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}}) = \mathbf{a}_{\mathsf{C}} m + \mathbf{b}_{\mathsf{C}} r_{\mathsf{C}} + \mathbf{e}_{\mathsf{C}}$$

  The opening of the commitments is defined as $(m, r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}}, 1)$

- $\mathsf{ComVer}_{ck}(\mathbf{c}, m', r'_{\mathsf{C}}, \mathbf{e}'_{\mathsf{C}}, f')$: given $(\mathbf{c}, m', r'_{\mathsf{C}}, \mathbf{e}'_{\mathsf{C}}, f')$ the verification algorithm accepts if and only if:

$$\mathbf{a}_{\mathsf{C}} m' + \mathbf{b}_{\mathsf{C}} r'_{\mathsf{C}} + f'^{-1} \mathbf{e}'_{\mathsf{C}} = \mathbf{c} \wedge \|\mathbf{e}'_{\mathsf{C}}\|_\infty \leq \left\lfloor \frac{n^{4/3}}{2} \right\rfloor \wedge \|f'\|_\infty \leq 1 \wedge \deg f' \leq \frac{n}{2}$$

This commitment scheme satisfies the security requirements of correctness, perfectly binding and computational hiding that are explained below:

- **Correctness**: if the commitment is computed by an honest party, the verification algorithm always accepts.

- **Perfectly binding**: a commitment cannot be opened to different messages. Given two distinct openings, $(\mathbf{c}, m, r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}}, f)$ and $(\mathbf{c}, m', r'_{\mathsf{C}}, \mathbf{e}'_{\mathsf{C}}, f')$, for the same commitment, the verification algorithm with overwhelming probability accepts both of them if and only if $m = m'$.

- **Computationally hiding**: given two messages, m_0 and m_1, and the commitment to one of them $\mathbf{c}_b = \mathsf{Com}_{\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}}}(m_b; r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}})$ with $b \xleftarrow{\$} \{0, 1\}$, for every PPT adversary there is a negligible probability that the adversary guess the committed message. It is argued that by the RLWE assumption, $\mathbf{b}_{\mathsf{C}} r_{\mathsf{C}} + \mathbf{e}_{\mathsf{C}}$ is pseudorandom and thus so is $\mathbf{c}_b$.

The proof of these three properties is well explained in [29] and we omit the details here.

Notice that with this construction of the commitment scheme they are relaxing the opening such that they also accept openings of the form $\mathbf{a}_{\mathsf{C}} m + \mathbf{b}_{\mathsf{C}} r_{\mathsf{C}} + f^{-1}\mathbf{e}_{\mathsf{C}}$, where $f \in R_q$ is an additional small polynomial. This relaxation is done in order to overcome the knowledge-error "barrier" from the original commitment scheme presented in [154]. It is proved that this modification does not affect the binding property of the scheme.

There exist efficient zero-knowledge proofs to prove knowledge of an opening of a given commitment or to prove that the messages inside some commitments satisfy any polynomial relation. This proofs are well described in [29] and are also explained in Chapter 4.

Chapter 3

Post-quantum mix-net

3.1 Introduction

As we have seen in Section 2.3.2, mix-nets are of paramount importance in an online voting scenario. They provide anonymity by permuting and re-encrypting or partially decrypting the votes so the output of the process cannot be correlated with the input, i.e., the ciphertexts at the output look completely different from the ciphertexts at the input. If we want the mixing process to be universally verifiable, it is necessary that each mix-node computes a proof of a shuffle in order to demonstrate that the encrypted messages have not been modified during the operation. In addition, if we want to use the mix-net to build a post-quantum online voting system we need to base its security on quantum-resistant computational problems, such as lattice problems.

The proposal described in this chapter is a proof of shuffle based on lattices [47] which is used to build the first universally verifiable mix-net for a post-quantum cryptosystem. The proof is based on that proposed by Wikström in [153].

Although this proof of a shuffle based on lattices is not used as a building block of our post-quantum online voting system (Chapter 5), it has served as a preliminary work for designing the fully post-quantum proof of a shuffle presented in next chapter.

3.1.1 Related work

The first mix-net was introduced by Chaum [42] in 1981 who proposed a decryption mix-net using RSA onions with random padding (the padding added to the message before being encrypted). The idea is that each mix-node M_k has its public key and the corresponding private key. The messages that are going to be shuffled are encrypted as many times as mix-nodes using a different public key and random padding each time. The ciphertexts at the input of the mix-net are represented as:

$$C_i = \mathsf{Enc}(pk_1, \mathsf{Enc}(pk_2, \ldots \mathsf{Enc}(pk_l, m))))$$

begin l the number of mix-nodes.

In turns, each mix-node decrypts the outer layer of C_i using the corresponding private key, removes the random padding and sends the remaining *onion* to the next

node. If decryptions are done in the correct order, the last node obtains the original message m. Nevertheless, Chaum did not give any method for guaranteeing the correctness of the shuffle.

Nine years later, Pfitzmann and Pfitzmann [125] discovered an attack on Chaum's proposal which consists on performing two sequential shuffles. The first shuffle is done using the honest input, i.e., the ciphertexts without being manipulated. Then, for the second shuffle the attacker uses an input that is related to the honest input which yields in a relationship between the corresponding output ciphertexts and plaintexts. Finally, this allows the attacker to trace an input of the first shuffle.

In 1993, Park *et al.* noticed that Chaum's mix-net required a ciphertext size proportional to the number of mix-nodes and they proposed the first re-encryption mix-net [120], where each mix-node re-randomized the ciphertexts using a homomorphic cryptosystem like ElGamal. Recall that in the ElGamal cryptosystem it is possible to re-encrypt the messages using the same public key and just modifying the randomness used during the encryption by multiplying the initial ciphertext by the encryption of 1: $(g^k, h^k \cdot m) \cdot (g^{k'}, h^{k'}) = (g^{k+k'}, h^{k+k'} \cdot m)$. Each mix-node permutes and re-encrypts each ciphertext and decryption occurs after all shuffles have been done. Note that in this scenario the size of the final ciphertext does not depend on the number of times it is re-encrypted. Finally, the authors also proposed a different mix-net in the same paper where each node performs partial decryption besides the shuffling. Again, Pfitzmann found an attack in 1995 against both proposals [124] which uses the fact that ElGamal used over all $\mathbb{Z}_p^*$ is not semantically secure. He proposed a solution which consists on using a subgroup of order q of $\mathbb{Z}_p^*$. This approach was formalized by Tsiounis and Yung in [147].

None of the mix-nets proposed until the date were universal verifiable, i.e., the correctness of the shuffle cannot be verified. It was Sako and Kilian in 1995 who first defined the property of *universal verifiability* and proposed the first universally verifiable mix-net that provides a zero-knowledge proof of correct mixing [133]. The mix-net uses the partial decryption and re-encryption approach presented in [120] and also applies Pfitzmann's countermeasure. In addition, each mix-node after performing the shuffle provides a proof of partial decryption and a proof of a shuffle. Michels and Horster pointed out in [114] that if only one mix-node is honest, which is a common assumption when working with mix-nets, the privacy can be compromised.

Achieving efficient mixing proofs was the challenge of the late 1990s, when two solutions were proposed for building an efficient universally verifiable mix-net [5, 109]. Both proposals are based on permutation networks, use a semantically secure ElGamal cryptosystem and perform threshold decryption after the mixing. They differ on some of the zero-knowledge proofs generated. In 2001, Furukawa and Sako [70] proposed a proof of correct mixing more efficient than the previous ones. In this scheme each node uses a matrix to do the ciphertexts permutation and proves in zero-knowledge that this matrix is a permutation matrix. In the same year, Neff [116] introduced the fastest, fully-private, universally verifiable mix-net shuffle proof known so far, optimized and generalized by Groth in [84]. The proof requires three protocols which consists of proving in zero-knowledge several equalities.

There is another type of universally verifiable mix-net which is called *Optimistic*

Mixing and that was first proposed by Golle *et al.* [82] in 2002. The proof generated by this mix-net is significantly faster than the others if all the mix-nodes behave honestly, and is only when an error is detected that a proof like Neff's is generated.

It was also Golle two years later who proposed a mix-net with universal re-encryption [81] which does not require that each mix-node participates on the key generation process. This can be done with homomorphic cryptosystems like ElGamal. In the same year, Wikström [151] gave the first mix-net provably secure in the Universally Composable (UC) framework [36], unlike previous proposals that give ad-hoc definitions of security and most of them provide proofs in heuristic models. Proving security on this framework allows to precisely determine which are the security properties of a mix-net and ensures that the system will remain secure even if it runs alongside others. One year later, in 2005, Wikström [152] presented a slightly different mix-net approach from previous ones in which re-encryption is not necessary, each mix-node just partially decrypts and permutes its input. This allows the sender to verify that its message was successfully processed, i.e., the scheme is *sender verifiable*. He also gives the first proof of a partial-decryption and permutation shuffle of ElGamal ciphertexts and demonstrates that the mix-net is provably secure in the UC framework.

Motivated by the complexity of using mix-nets in elections, Adida and Wikström introduced a different mix-net approach [10, 11]. They proposed an offline pre-computation technique in order to reduce the online computation complexity. However, the scheme [10] was quite inefficient while the construction in [11] was very efficient but reduced to a relatively small number of senders. In 2009 [153], Wikström presented a mix-net based on homomorphic cryptosystems using the idea of permutation matrices. In the proposal, a proof of a shuffle is split in an offline and online phase following the approach proposed in [11]. More precisely, in the offline part the mix-node computes a commitment to the permutation matrix and proves in zero knowledge that it knows an opening for that commitment. In the online part, the node computes a commitment-consistent proof of a shuffle to demonstrate that the committed matrix has been used to shuffle the input. One year later, Terelius and Wikström [146] proposed a provably secure technique to prove the correctness of a cryptographic shuffle using simple shuffle arguments which allow the restriction of the shuffles to certain classes of permutations. Then, in 2012, Bayer and Groth [22] proposed an honest verifier zero-knowledge argument for the correctness of a shuffle of homomorphic encryptions that, compared with previous work, matches the lowest computation cost for the verifier.

Nevertheless, as these non-interactive proofs are known to be secure in the random oracle model which is only heuristic, several works have studied how to construct non-interactive zero-knowledge (NIZK) shuffle arguments in the Common Reference String (CRS) model. The two first shuffle arguments in the CRS model were proposed by Groth and Lu [87] in 2007 and by Lipmaa and Zhang [102] in 2012. Nevertheless they were significantly slower than the fastest arguments in the random oracle model. The former only provides culpable soundness which informally means that if a malicious mix-node can produce an acceptable shuffle for an invalid statement and has also access to the secret key, the underlying security assumption can be broken. The latter improves the efficiency of [87] but it provides a weaker secu-

rity notion. Both schemes suggest a NIZK argument for the correctness of a shuffle of BBS ciphertexts [30]. The BBS cryptosystem, proposed by Boneh, Boyen and Shacham works in bilinear groups. Later, Lipmaa and Fauzi [63] proposed a pairing based NIZK shuffle argument in the CRS model which achieves culpable soundness and when compared with the previous proposals [87, 102] is faster both proving and verifying, and it is based on ElGamal cryptosystem over bilinear groups. The efficiency of this proposal is improved by Lipmaa *et al.* [65] by proving knowledge-soundness (there exists a PPT extractor which is able to compute a witness from the proof and succeeds in convincing an honest verifier) in the Generic Bilinear Group Model (GBGM). Finally, the most efficient known pairing-based NIZK shuffle argument is also given by Lipmaa in 2017 [64].

However, given that these CRS-based proposals are constructed for bilinear groups, we are going to follow the approach presented in [146, 153] to build our proof of a shuffle.

To the best of our knowledge, the concept of using mix-nets for lattice-based cryptography is very new in the research literature, and as such, there are not many proposed schemes. Until 2017 there have been proposals for a lattice-based universal re-encryption for mix-nets [140, 141] but none of them proposes a proof of a shuffle, which is essential for verifiable protocols.

The most recent work on lattice-based mix-nets is that presented by Boyen *et al.* in 2020 [32]. They propose a plain decryption mix-net, i.e., a mix-net which is not verifiable, with trip wires thus achieving high level of verifiability and accountability in the presence of fully malicious mix-nodes. They also claim that re-encryption mix-nets are currently impractical for defending against quantum attackers. Their verifiability approach, consisting on using the trip wire technique, is not based on computing a proof of a shuffle but on a set of auditors who introduce fake votes at the input and reveal them at the output. Since the mix-nodes cannot distinguish between fake and real votes, if they have modified a set of votes with high probability some of them will be those introduced by the auditors, and everyone will be able to check that they are not part of the output.

3.1.2 Our proposal

We propose the first universally verifiable mix-net for a post-quantum cryptosystem [47]. The mix-net receives at its input a set of messages encrypted using a RLWE encryption scheme [107] whose security is based on the hardness of solving the Learning With Errors problem over rings (RLWE problem) [129]. In the proposal, we show how to permute and re-encrypt RLWE encryptions and we also give the first proof of a shuffle that works for a lattice-based cryptosystem. This proof is based on what is proposed in [153] but it is not a direct adaptation of it, since we introduce a new technique to implement the last part of the proof that differs from what is presented in that article.

We split the proof of a shuffle into two protocols following Wikström's technique. In the offline part, the permutation and re-encryption parameters used to shuffle the ciphertexts are committed and it is demonstrated using zero knowledge proofs that these values meet certain properties and that the openings for the commitments are

known. The zero-knowledge proofs used in this part satisfy special soundness and special honest verifier zero-knowledge (see Section 2.2.4). The first property means that given two accepting conversations with identical first messages but different challenges, it is possible to extract a valid witness. Regarding the second property, it means that for a given challenge the verifier can be simulated.

In the online part, instead of computing a commitment-consistent proof of a shuffle, each mix node should compute a commitment to its output using the commitments calculated in the offline protocol taking advantage of the homomorphic property of both the commitment and encryption schemes. Finally, the node should reveal the opening of the output commitment in order to demonstrate that it has used the committed permutation and re-encryption values to do the shuffle. It is important to notice that we are not opening the commitments directly to the secret permutation neither to the secret re-encryption values but the commitments to a linear combination of them. The openings revealed by each node perfectly hide the secret values and no information is leaked that could compromise the privacy of the process. Commitments used to construct the proof are generalized versions of the Pedersen commitment, which is perfectly hiding and computationally binding under the discrete logarithm assumption and it is widely used to provide everlasting privacy. Although it might seem a contradiction to use these commitments since its binding property relies on an assumption that is broken in a quantum scenario, the property we are interested on in order to provide long-term privacy is the hiding property, which is quantum-resistant. In addition, the reason why we use Pedersen commitments is for efficiency and simplicity. Nevertheless, since our protocol only requires a commitment that allows us to prove linear relations between committed elements, the protocol presented in this paper could be modified in order to use the commitment scheme proposed by Benhamouda *et al.* in [29]. This would allow us to construct a mix-net totally based on post-quantum cryptography, which is presented in Chapter 4. As this is a non-trivial modification we first show how to mix RLWE ciphertexts using Pedersen commitments and how to do it universally verifiable.

3.2 A commitment-consistent proof of a shuffle

In this section we give an overview of Wikström's proposal and we refer the reader to [153] for the exact details of how the protocol works. In this mix-net approach Wikström shows how to split the shuffle proof into two protocols: offline and online, which allows to reduce the online computation complexity. This proof is later improved and generalized in [146].

In a re-encryption mix-net based on a homomorphic cryptosystem, each mix-node M_k chooses a secret permutation π_k and one re-encryption parameter $\rho_{k,i}$ for each ciphertext C_i to be mixed. The output of each node is then a list of ciphertexts that have been permuted and re-encrypted: $L_k = \{C''_{k,1}, \ldots, C''_{k,N}\}$ where $C''_{k,i} = C'_{k,i}\mathsf{Enc}_{pk}(1, \rho_{k,i})$ and $C'_{k,i} = C_{k-1,\pi_k(i)}$. During the *offline protocol* each mix-node commits to the permutation, proves knowledge of how to open the commitment and proves certain properties about the committed elements. Then, during the *online protocol*, it proves that it has used the committed permutation to perform the shuffle.

Before continuing with the overview of the proof, we describe which is the commitment scheme used by Wikström that will be also the commitment scheme used in our proposal.

Pedersen commitment. Let p and q be large primes, $\mathbb{Z}_p^*$ a group of integers modulo $p = 2q + 1$ and $G_q \subset \mathbb{Z}_p^*$ a subgroup of order q where the discrete logarithm assumption holds. Given two independent generators $ck = \{g, g_1\}$ of G_q, to commit to a message $x \in \mathbb{Z}_q$ using the Pedersen commitment scheme [121], choose a random $\alpha \xleftarrow{\$} \mathbb{Z}_q$ and output $\mathsf{Com}_{ck}(x, \alpha) = g^\alpha g_1^x$. In order to open this commitment simply reveal the values α and x. This scheme is perfectly hiding and computationally binding as long as the discrete logarithm problem is hard in G_q.

In our proposal we are going to work with the *extended version* of the Pedersen commitment scheme, that allows committing to more than one message at once. Given $N + 1$ independent generators $ck = \{g, g_1, \ldots, g_N\}$ of G_q and a randomnes $\alpha \xleftarrow{\$} \mathbb{Z}_q$, the commitment to N messages $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{Z}_q^N$ is computed as:

$$\mathsf{Com}_{ck}(\mathbf{x}, \alpha) = g^\alpha \prod_{i=1}^{N} g_i^{x_i}$$

We use this version of the Pedersen commitment to commit to a matrix $\mathbf{M} \in \mathbb{Z}_q^{N \times N}$. We do that just computing a commitment to each of its columns $(\mathbf{m_1}, \ldots, \mathbf{m_N})$ where $\mathbf{m_j} = (m_{1j}, m_{2j}, \ldots, m_{Nj})^\top$ for $j = 1, \ldots, N$. This means that a matrix commitment is a vector whose components are the commitments to the matrix columns:

$$\mathsf{Com}_{ck}(\mathbf{M}, \alpha_1, \alpha_2, \ldots, \alpha_N) = (\mathsf{Com}_{ck}(\mathbf{m_1}, \alpha_1), \ldots, \mathsf{Com}_{ck}(\mathbf{m_N}, \alpha_N)) = (c_{\mathbf{m_1}}, \ldots, c_{\mathbf{m_N}}) \tag{3.1}$$

where $c_{\mathbf{m_j}} = g^{\alpha_j} \prod_{i=1}^{N} g_i^{m_{ij}}$. Finally, due to the homomorphic property of the Pedersen commitment we can compute a commitment to the product of a matrix $\mathbf{M}$ by a vector $\mathbf{x}$ from the commitment to the matrix $\mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}) = (c_{\mathbf{m_1}}, \ldots, c_{\mathbf{m_N}})$.

$$\mathsf{Com}(\mathbf{Mx}, \alpha_{\mathbf{Mx}}) = \prod_{j=1}^{N} c_{\mathbf{m_j}}^{x_j} = \prod_{j=1}^{N} \left(g^{\alpha_j} \prod_{i=1}^{N} g_i^{m_{ij}} \right)^{x_j} = g^{\langle \boldsymbol{\alpha}, \mathbf{x} \rangle} \prod_{i=1}^{N} g_i^{\langle (m_{i1}, \ldots, m_{iN}), (x_1, \ldots, x_N) \rangle} \tag{3.2}$$

where $\alpha_{\mathbf{Mx}} = \langle \boldsymbol{\alpha}, \mathbf{x} \rangle$. Note that if we try to directly compute the product of $\mathbf{M}$ by $\mathbf{x}$ and we commit to the result, we obtain the same outcome.

$$\mathbf{Mx} = \begin{pmatrix} m_{11} & \ldots & m_{1N} \\ m_{21} & \ldots & m_{2N} \\ \vdots & \vdots & \vdots \\ m_{N1} & \ldots & m_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{N} m_{1j} \cdot x_j \\ \sum_{j=1}^{N} m_{2j} \cdot x_2 \\ \vdots \\ \sum_{j=1}^{N} m_{Nj} \cdot x_N \end{pmatrix}$$

$$\mathsf{Com}(\mathbf{Mx}, \alpha_{\mathbf{Mx}}) = g^{\alpha_{\mathbf{Mx}}} \prod_{i=1}^{N} (g_i^{\sum_{j=1}^{N} m_{ij} \cdot x_j}) = g^{\alpha_{\mathbf{Mx}}} \prod_{i=1}^{N} g_i^{\langle \mathbf{m_i}, \mathbf{x} \rangle}$$

In order to construct the proof of shuffle batch techniques are used. As pointed out by Neff [116] and Furukawa and Sako [70], batch proofs are in some sense invariant under permutation and this allows to construct efficient shuffle proofs. Given the set of group elements $(y_1 = g_1^{x_1}, \ldots, y_N = g_N^{x_N})$ it will be expensive to prove knowledge of each logarithm x_i independently. Nevertheless, using a batch proof we can do it simultaneously: the prover P will demonstrate that it knows a w such that $y = g^w$. During the proof the verifier V will select $e_1, \ldots, e_N \in \mathbb{Z}_p$ and will send them to P. Then, both P and V will compute $y = \prod_{i=1}^{N} y_i^{e_i}$.

This idea is used for constructing the proof of a shuffle (note that this is just a simplified description of the proof and some of the details are omitted here. We refer the reader to [24] for the details):

1. Both P and V will use $(e_1, \ldots, e_N)$ and $C_1, \ldots, C_N$ for computing $C = \prod_{i=1}^{N} C_i^{e_i}$.

2. The prover also computes $C'' = \prod_{i=1}^{N} (C_i'')^{e_{\pi(i)}}$ and convinces the verifier that the original exponents, re-ordered using a fixed permutation are used to form C''.

3. P proves knowledge of ρ such that $C'' = C \cdot \mathsf{Enc}_{pk}(1, \rho)$

The second step is the most expensive, so Wikström design it in such a way that almost all of it can be moved to the offline phase. During this phase each mix-node computes a permutation and commits to its matrix (the *permutation matrix*). Then, the node proves that it knows how to open the commitment. The concept of *permutation matrix* is explained below.

Permutation matrix. A matrix $\mathbf{M}$ is the permutation matrix corresponding to a permutation function π if

$$\mathbf{M}_{ij} = \begin{cases} 1 \mod q & \text{if } \pi(i) = j \\ 0 \mod q & \text{otherwise} \end{cases} \tag{3.3}$$

In other words, $\mathbf{M}$ is a permutation matrix if it has exactly one non-zero element in each column and each row, and the elements of each column sum up to one, that is, the non-zero element is the number one. Notice that in this case the commitment to each column of the matrix will be $c_{\boldsymbol{m_j}} = \mathsf{Com}(\boldsymbol{m_j}, \alpha_j) = g^{\alpha_j} \prod_{i=1}^{N} g_i^{m_{ij}} = g^{\alpha_j} g_{\pi^{-1}(j)}$ and the Equation 3.1 translates to:

$$\mathsf{Com}(\mathbf{M}, \alpha_1, \alpha_2, \ldots, \alpha_N) = \left(g^{\alpha_1} g_{\pi^{-1}(1)}, \ldots, g^{\alpha_N} g_{\pi^{-1}(N)}\right) \tag{3.4}$$

and consequently Equation 3.2 translates to:

$$\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{x_j} = \prod_{j=1}^{N} \left(g^{\alpha_j} g_{\pi^{-1}(j)}\right)^{x_j} = g^{\langle \boldsymbol{\alpha}, \mathbf{x} \rangle} \prod_{j=1}^{N} g_j^{x_{\pi(j)}} \tag{3.5}$$

So we can see that given a commitment to a permutation π and a vector $\mathbf{x} = (x_1, \ldots, x_N)$ we can transform this commitment into a commitment of $\mathbf{x}_\pi$ where the elements x_j are in a different order defined by π.

Note that the commitment to the permutation allows to publicly computing the commitment to all $\langle \mathbf{m_j}, \mathbf{e} \rangle$ as shown in Equation 3.2:

$$\prod_{j=1}^{N} c_{\mathbf{m_j}}^{e_j} = g^{\langle \boldsymbol{\alpha}, \mathbf{e} \rangle} \prod_{i=1}^{N} g_i^{\langle \mathbf{m_i}, \mathbf{e} \rangle}$$

During the offline phase each mix-node computes $\mathsf{Com}_{ck}(\mathbf{M}, \boldsymbol{\alpha})$ and proves knowledge of both $\boldsymbol{\alpha}$ and $\mathbf{M}$. It also demonstrates that $\mathbf{M}$ is a permutation matrix in the following way:

1. Shows that $\prod_{i=1}^{N} \langle \mathbf{m_i}, \mathbf{e} \rangle = \prod_{i=1}^{N} e_i$, which is true only if $\mathbf{M}$ has exactly one non-zero element in each column and each row. This can be tested by using the Schwarz-Zippel's lemma which is used to prove polynomial equalities (see Lemma 3.2.1). The idea is that we can construct two non-zero multi-variate polynomials $p(x_1, \ldots, x_N)$ and $q(x_1, \ldots, x_N)$ such that $p(x_1, \ldots, x_N) = \prod_{i=1}^{N} x_i$ and $q(x_1, \ldots, x_N) = \prod_{i=1}^{N} \langle \mathbf{m_i}, \mathbf{x} \rangle$ (where $\mathbf{m_i}$ denotes the ith row of the permutation matrix). If we evaluate them at a random point and we obtain the same result, there is a high probability of the two polynomials being the same.

2. Shows that the non-zero element equal to one. This is demonstrated by showing that the elements of each row sum up to one $\mathbf{M} \cdot \mathbf{1} = \mathbf{1}$. Indeed, it suffices to prove that $\prod_{j=1}^{N} c_{\boldsymbol{m_j}} / \prod_{j=1}^{N} g_j$ is of the form g^{α} for some α. Note that this is the same as showing that $\mathsf{Com}(\mathbf{1}, v) = \langle \mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}), \mathbf{1} \rangle$ where $v = \langle \boldsymbol{\alpha}, \mathbf{1} \rangle$.

Lemma 3.2.1 (Schwartz-Zippel lemma)**.** Let $f \in \mathbb{Z}_p[x_1, \ldots, x_N]$ be a non-zero multivariate polynomial of total degree $d \geq 0$ over $\mathbb{Z}_q$, let $S \subset \mathbb{Z}_q$, and let $e_1, \ldots, e_N$ be chosen randomly from S. Then

$$\Pr[f(e_1, \ldots, e_N) = 0] \leq \frac{d}{|S|}$$

Formally, the zero-knowledge proof computed is of the following form [103]:

$$\Sigma\text{-proof} \left[\begin{array}{c|c} v, w \in \mathbb{Z}_q \\ \mathbf{e'} \in \mathbb{Z}_q^N \end{array} \middle| \begin{array}{c} \mathsf{Com}(\mathbf{1}, v) = \langle \mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}), \mathbf{1} \rangle \wedge \\ \mathsf{Com}(\mathbf{e'}, w) = \langle \mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}), \mathbf{e} \rangle \wedge \\ \prod_{j=1}^{N} e'_i = \prod_{j=1}^{N} e_i \end{array} \right]$$

where $w = \langle \boldsymbol{\alpha}, \mathbf{e} \rangle$ and $\mathbf{e'} = (e_{\pi(1)}, \ldots, e_{\pi(N)})$. Recall that $\mathbf{e} = (e_1, \ldots, e_N)$ is selected and sent by the verifier V.

During the online phase each node proves that its output is its input re-encrypted and permuted and that the permutation used is that committed in the offline phase. We are not going to enter in more details since our proposal for the online protocol differs from [153, 146].

3.3 Mixing protocol overview

In this section we present an overview of our mixing protocol but before entering into the details we want to recall some of the lattice concepts we have already introduced in Section 2.4.

Let R_q be the ring of integer polynomials $R_q = \mathbb{Z}_q[x]/\langle x^n+1\rangle$ where n is a power of 2 and q is a prime; and let χ_σ be a discretized Gaussian distribution (see Section 2.4.2). The RLWE public key is $pk = (a_\mathsf{E}, b_\mathsf{E}) = (a_\mathsf{E}, a_\mathsf{E} \cdot s + e) \in R_q \times R_q$ and the private key $sk = s$. The ciphertext obtained from encrypting a message $z \in \{0,1\}^n$ (which is identified as a polynomial in R_q with $0-1$ coefficients) with the public key pk is $(u, v) = (a_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},u}, b_\mathsf{E} \cdot r_\mathsf{E} + e_{\mathsf{E},v} + \lfloor \frac{q}{2} \rceil z) \in R_q \times R_q$, where $r_\mathsf{E}, e_{\mathsf{E},u}, e_{\mathsf{E},v} \in R_q$ are drawn from the error distribution χ_σ. Although it is not strictly necessary to use the subscript E in this chapter, we are going to maintain it in order to be aligned with the next chapter, where the subscript is important to differentiate between the ciphertext and the commitment elements.

If instead of using polynomials we use vector and matrix notation, the public key is $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$ where $\mathbf{A}$ and $\mathbf{B}$ are constructed from $\mathbf{a}$ and $\mathbf{b}$ correspondingly, in the following way:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{pmatrix}$$

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \dots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{pmatrix} = \begin{pmatrix} b_1 & -b_n & -b_{n-1} & \dots & -b_2 \\ b_2 & b_1 & -b_n & \dots & -b_3 \\ b_3 & b_2 & b_1 & \dots & -b_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_n & b_{n-1} & b_{n-2} & \dots & b_1 \end{pmatrix}$$

Finally, we can express the ciphertext as a vector of $2n$ elements $(\boldsymbol{u}, \boldsymbol{v}) = (u_1, \dots, u_n, v_1, \dots, v_n) \in \mathbb{Z}_q^{2n}$:

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} (\boldsymbol{r}_\mathsf{E}) + \begin{pmatrix} \boldsymbol{e}_{\mathsf{E},u} \\ \boldsymbol{e}_{\mathsf{E},v} \end{pmatrix} + \left\lfloor \frac{q}{2} \right\rceil \begin{pmatrix} \mathbf{0} \\ \boldsymbol{z} \end{pmatrix}$$

and its re-encryption as:

$$\begin{pmatrix} \boldsymbol{u}' \\ \boldsymbol{v}' \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} + \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix} (\boldsymbol{r}'_\mathsf{E}) + \begin{pmatrix} \boldsymbol{e}'_{\mathsf{E},\boldsymbol{u}} \\ \boldsymbol{e}'_{\mathsf{E},\boldsymbol{v}} \end{pmatrix}$$

Following this notation and given a permutation π characterized by the matrix $\mathbf{M}$ and a set of re-encryption parameters $\left(\boldsymbol{r}'^{(\boldsymbol{i})}_\mathsf{E}, \boldsymbol{e}'^{(\boldsymbol{i})}_{\mathsf{E},\boldsymbol{u}}, \boldsymbol{e}'^{(\boldsymbol{i})}_{\mathsf{E},\boldsymbol{v}}\right)$ for each one of the messages i (for all $i \in [1, \dots, N]$), we can express the shuffling of N RLWE encryptions as:

$$\underset{N\times 2n}{\begin{pmatrix} u_1''^{(1)} & \cdots & u_n''^{(1)} & v_1''^{(1)} & \cdots & v_n''^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_1''^{(N)} & \cdots & u_n''^{(N)} & v_1''^{(N)} & \cdots & v_n''^{(N)} \end{pmatrix}} = \underset{N\times N}{\begin{pmatrix} m_{11} & \cdots & m_{1N} \\ \vdots & \ddots & \vdots \\ m_{N1} & \cdots & m_{NN} \end{pmatrix}} \underset{N\times 2n}{\begin{pmatrix} u_1^{(1)} & \cdots & u_n^{(1)} & v_1^{(1)} & \cdots & v_n^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_1^{(N)} & \cdots & u_n^{(N)} & v_1^{(N)} & \cdots & v_n^{(N)} \end{pmatrix}}$$

$$+ \underset{N\times n}{\begin{pmatrix} r'^{(1)}_{\mathsf{E},1} & \cdots & r'^{(1)}_{\mathsf{E},n} \\ \vdots & \ddots & \vdots \\ r'^{(N)}_{\mathsf{E},1} & \cdots & r'^{(N)}_{\mathsf{E},n} \end{pmatrix}} \underset{n\times 2n}{\begin{pmatrix} a_1 & \cdots & a_n & b_1 & \cdots & b_n \\ -a_n & \cdots & a_{n-1} & b_2 & \cdots & b_{n-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -a_2 & \cdots & a_1 & -b_2 & \cdots & b_1 \end{pmatrix}} + \underset{N\times 2n}{\begin{pmatrix} e'^{(1)}_{\mathsf{E},u,1} & \cdots & e'^{(1)}_{\mathsf{E},u,n} & e'^{(1)}_{\mathsf{E},v,1} & \cdots & e'^{(1)}_{\mathsf{E},v,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e'^{(N)}_{\mathsf{E},u,1} & \cdots & e'^{(N)}_{\mathsf{E},u,n} & e'^{(N)}_{\mathsf{E},v,1} & \cdots & e'^{(N)}_{\mathsf{E},v,n} \end{pmatrix}}$$

$$\begin{pmatrix}\mathbf{U}'' & \mathbf{V}''\end{pmatrix} = \mathbf{M}\begin{pmatrix}\mathbf{U} & \mathbf{V}\end{pmatrix} + \mathbf{R}'_{\mathsf{E}}\begin{pmatrix}\mathbf{A}^T & \mathbf{B}^T\end{pmatrix} + \begin{pmatrix}\mathbf{E}'_{\mathsf{E},\mathbf{u}} & \mathbf{E}'_{\mathsf{E},\mathbf{v}}\end{pmatrix} \tag{3.6}$$

Matrices $\mathbf{M}, \mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}, \mathbf{E}'_{\mathsf{E},\mathbf{v}}$ are selected and kept secret by the mix-node, and $\begin{pmatrix}\mathbf{U}'' & \mathbf{V}''\end{pmatrix}$, $\begin{pmatrix}\mathbf{U} & \mathbf{V}\end{pmatrix}$, $\begin{pmatrix}\mathbf{A} & \mathbf{B}\end{pmatrix}$ are public values since they are the output and the input of the mix-node, and the public key of the encryption scheme respectively. A mix-node should prove that it knows $\mathbf{M}, \mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}, \mathbf{E}'_{\mathsf{E},\mathbf{v}}$ such that the output of the node $\begin{pmatrix}\mathbf{U}'' & \mathbf{V}''\end{pmatrix}$ is the input $\begin{pmatrix}\mathbf{U} & \mathbf{V}\end{pmatrix}$ re-encrypted and permuted, without revealing any information about $\mathbf{M}, \mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v}}$.

$$\Sigma\text{-proof}\left[\begin{array}{c|c} \begin{array}{c}\pi\\ \boldsymbol{r}'^{(1)}_{\mathsf{E}}, \ldots, \boldsymbol{r}'^{(N)}_{\mathsf{E}}\\ \boldsymbol{e}'^{(1)}_{\mathsf{E},\boldsymbol{u}}, \ldots, \boldsymbol{e}'^{(N)}_{\mathsf{E},\boldsymbol{u}}\\ \boldsymbol{e}'^{(1)}_{\mathsf{E},\boldsymbol{v}}, \ldots, \boldsymbol{e}'^{(N)}_{\mathsf{E},\boldsymbol{v}}\end{array} & \begin{array}{c}\left(\left(\boldsymbol{u}''^{(1)}, \boldsymbol{v}''^{(1)}\right), \ldots, \left(\boldsymbol{u}''^{(N)}, \boldsymbol{v}''^{(N)}\right)\right)^T\\ =\\ \begin{pmatrix}\mathsf{Re\text{-}enc}\left(\left(\boldsymbol{u}^{\pi(1)}, \boldsymbol{v}^{\pi(1)}\right), \boldsymbol{r}'^{(1)}_{\mathsf{E}}, \boldsymbol{e}'^{(1)}_{\mathsf{E},\boldsymbol{u}}, \boldsymbol{e}'^{(1)}_{\mathsf{E},\boldsymbol{v}}\right)^T\\ \ldots\\ \mathsf{Re\text{-}enc}\left(\left(\boldsymbol{u}^{\pi(N)}, \boldsymbol{v}^{\pi(N)}\right), \boldsymbol{r}'^{(N)}_{\mathsf{E}}, \boldsymbol{e}'^{(N)}_{\mathsf{E},\boldsymbol{u}}, \boldsymbol{e}'^{(N)}_{\mathsf{E},\boldsymbol{v}}\right)^T\end{pmatrix}\end{array}\end{array}\right]$$

Following Wikström's proposal we are going to split the proof into two protocols (in order to simplify the equations we omit the subscript ck in the algorithm Com).

Offline phase

1. The mix-node M_k chooses a random permutation π_k characterized by the matrix $\mathbf{M}_k \in \mathbb{Z}_q^{N\times N}$, computes a matrix commitment $\mathsf{Com}(\mathbf{M}_k, \boldsymbol{\alpha}_{m,k})$ and publishes it. It also proves knowledge of the committed permutation (see Section 3.4).

2. M_k randomly chooses the re-encryption matrices: $\mathbf{R}'_{\mathsf{E},\mathrm{k}} \in \mathbb{Z}_q^{N\times n}$,$\mathbf{E}'_{\mathsf{E},\mathbf{u},\mathrm{k}} \in \mathbb{Z}_q^{N\times n}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v},\mathrm{k}} \in \mathbb{Z}_q^{N\times n}$. It computes the corresponding matrix commitments, publishes them and prove that the committed elements are *small* (see Section 3.5).

Recall that the commitments are calculated in the following way (we omit here the subscript k which refers to a specific mix-node):

$$\begin{aligned}
\mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}_m) &= (c_{\mathbf{m_1}}, \ldots, c_{\mathbf{m_N}})\\
\mathsf{Com}(\mathbf{R}'_{\mathsf{E}}, \boldsymbol{\alpha}_{r'}) &= (c_{\boldsymbol{r}'_{\mathsf{E},\mathbf{1}}}, \ldots, c_{\boldsymbol{r}'_{\mathsf{E},\boldsymbol{n}}})\\
\mathsf{Com}(\mathbf{E}'_{\mathsf{E},\mathbf{u}}, \boldsymbol{\alpha}_{e'_u}) &= (c_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{u},\mathbf{1}}}, \ldots, c_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{u},\boldsymbol{n}}})\\
\mathsf{Com}(\mathbf{E}'_{\mathsf{E},\mathbf{v}}, \boldsymbol{\alpha}_{e'_v}) &= (c_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{v},\mathbf{1}}}, \ldots, c_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{v},\boldsymbol{n}}})
\end{aligned}$$

where each element of each vector is defined as the commitment to a matrix column j:

$$c_{\mathbf{m_j}} = \mathsf{Com}(\mathbf{m_j}, \alpha_{m_j}) = g^{\alpha_{m_j}} \prod_{i=1}^{N} g_i^{m_{ij}} \qquad \forall j \in 1 \div N$$

$$c_{\boldsymbol{r}'_{\mathsf{E},\boldsymbol{j}}} = \mathsf{Com}(\boldsymbol{r}'_{\mathsf{E},\boldsymbol{j}}, \alpha_{\boldsymbol{r}'_{\mathsf{E},\boldsymbol{j}}}) = g^{\alpha_{\boldsymbol{r}'_{\mathsf{E},\boldsymbol{j}}}} \prod_{i=1}^{N} g_i^{r'^{(i)}_{\mathsf{E},j}} \qquad \forall j \in 1 \div n$$

$$c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}} = \mathsf{Com}(\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}, \alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}) = g^{\alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}} \prod_{i=1}^{N} g_i^{e'^{(i)}_{\mathsf{E},u,j}} \qquad \forall j \in 1 \div n$$

$$c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}} = \mathsf{Com}(\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}, \alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}) = g^{\alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}} \prod_{i=1}^{N} g_i^{e'^{(i)}_{\mathsf{E},v,j}} \qquad \forall j \in 1 \div n$$

Online phase

1. Given a list of N input ciphertexts, the mix-node M_k permutes and re-encrypts the list using Equation 3.6.

2. In order to prove that the committed matrices have been used to perform the mixing, M_k computes the commitment to its output using those commitments calculated during the online phase, and finally reveals its opening (see Section 3.6)

3.4 Proof of Knowledge of a Permutation Matrix

The first step of the offline phase consists on selecting a random permutation, committing to it and finally demonstrating in zero-knowledge that the value committed is indeed a permutation. The permutation matrix is characterized by the following theorem.

Theorem 3.4.1. Given a matrix $\mathbf{M} \in \mathbb{Z}_q^{N\times N}$ and a vector $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{Z}_q^N$ of N independent variables, $\mathbf{M}$ is a permutation matrix if and only if $\mathbf{M1} = \mathbf{1}$ and $\prod_{i=1}^{N} x_i = \prod_{i=1}^{N} x'_i$ where $\mathbf{x}' = \mathbf{Mx}$.

We refer the reader to [146] for the details about the theorem's proof.

Given a commitment to a matrix $\mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}_m) = (c_{\mathbf{m_1}}, \ldots, c_{\mathbf{m_N}})$ and a vector $\mathbf{x} = (x_1, \ldots, x_N)$, we can compute a commitment to the product of the matrix by a vector $\mathsf{Com}(\mathbf{Mx}, \langle \boldsymbol{\alpha_m}, \mathbf{x}\rangle)$ using Equation 3.2. In the special case where the vector $\mathbf{x} = \mathbf{1}$ the commitment above is $\mathsf{Com}(\mathbf{1}, t)$ where $t = \sum_{j=1}^{N} \alpha_{m_j}$. Another important observation is that given a vector $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_N)$ we can express a commitment to the product of the elements of $\mathbf{x}'$ in a recursive way $\hat{c}_i = g^{\hat{r}_i} \hat{c}_{i-1}^{x'_i}$ for $i = 1, \ldots, N$ and $\hat{c}_0 = g_1$.

$$\begin{aligned}
\hat{c}_0 &= g_1 \\
\hat{c}_1 &= g^{\hat{r}_1} \hat{c}_0^{x'_1} = g^{\hat{r}_1} g_1^{x'_1} \\
\hat{c}_2 &= g^{\hat{r}_2} \hat{c}_1^{x'_2} = g^{\hat{r}_2} g^{\hat{r}_1 x'_2} g_1^{x'_1 x'_2} \\
&\vdots \\
\hat{c}_N &= g^{\hat{r}_N} \hat{c}_{N-1}^{x'_N} = g^{\hat{r}} g_1^{\prod_{i=1}^{N} x'_i}
\end{aligned}$$

where $\hat{r} = \sum_{i=1}^{N} \hat{r}_i \sum_{j=i+1}^{N} x_j$.

Applying the second condition for a permutation matrix ($\prod_{i=1}^{N} x_i = \prod_{i=1}^{N} x'_i$), it is possible to obtain a commitment $\hat{c}_N$ such that $\hat{c}_N = g^{\hat{r}} g_1^{\prod_{i=1}^{N} x'_i} = g^{\hat{r}'} g_1^{\prod_{i=1}^{N} x_i}$, and prove that we know two different valid openings $(\hat{r}, \prod_{i=1}^{N} x'_i)$ and $(\hat{r}', \prod_{i=1}^{N} x_i)$. Due to the binding property of the commitments we know that if someone is able to open a commitment to two different openings, this means that either both openings are the same or the discrete logarithm, $g_1 = g^z$ where $z = (\hat{r} - \hat{r}') \,/\, \left(\prod_{i=1}^{N} x_i - \prod_{i=1}^{N} x'_i\right)$, can be computed in the following way:

$$g^{\hat{r}} g_1^{\prod_{i=1}^{N} x'_i} = g^{\hat{r}'} g_1^{\prod_{i=1}^{N} x_i}$$

$$g^{\hat{r}-\hat{r}'} = g_1^{\prod_{i=1}^{N} x_i - \prod_{i=1}^{N} x'_i}$$

$$\log_g g_1 = \frac{\hat{r} - \hat{r}'}{\prod_{i=1}^{N} x_i - \prod_{i=1}^{N} x'_i}$$

Observe that using the Schwartz-Zippel lemma (see Lemma 3.2.1) we can prove that the polynomial equality $\prod_{i=1}^{N} x_i = \prod_{i=1}^{N} x'_i$ holds with overwhelming probability just verifying that the equation holds for a point $(\lambda_1, \ldots, \lambda_N)$ randomly chosen from $\mathbb{Z}_q^N$.

Given these preliminaries we can construct a Σ-proof to prove that the mix-node knows an opening for the commitment and that the element committed is a permutation matrix. This proof follows the approach given by Wikström which has been already explained in Section 3.2.

$$\Sigma\text{-proof}\left[\begin{array}{l|c} & \left(\mathsf{Com}(\mathbf{1}, t) = \prod_{j=1}^{N} c_{\mathbf{m_j}}\right) \\ \boldsymbol{\lambda}' \in \mathbb{Z}_q^N, t, k, z \in \mathbb{Z}_q & \wedge \left(\mathsf{Com}(\boldsymbol{\lambda}', k) = \prod_{j=1}^{N} c_{\mathbf{m_j}}^{\lambda_j}\right) \\ & \wedge \left(\prod_{i=1}^{N} \lambda_i = \prod_{i=1}^{N} \lambda'_i \vee g_1 = g^z\right) \end{array}\right]$$

This protocol (shown in detail in the next page) meets the requirements of completeness, special soundness and special honest-verifier zero-knowledge defined in Section 2.2.4. We refer the reader to [153] for the details about the demonstration of these requirements.

We let n_v, n_c and n_r denote the bitsize of components in random vectors, challenges, and random paddings respectively. The security parameters 2^{-n_v}, 2^{-n_c} and 2^{-n_r} must be negligible in n. We can construct a simulator selecting $B_1, \ldots, B_N \in G_q$, $\mathbf{d}, \mathbf{d}' \in \mathbb{Z}_q^N$ and $d_\alpha, d_\gamma, d_\delta \in \mathbb{Z}_q$ randomly, and computing $\alpha, \beta_i, \gamma, \delta$ using the verification equations. In order to prove the consistency we have to undo the built recurrences in the same way that Wikström explains in his article.

$$
\begin{aligned}
\mathbf{r}, \mathbf{s} &\xleftarrow{\$} \mathbb{Z}_q^N \\
s_\alpha &\xleftarrow{\$} \mathbb{Z}_q \\
s_\gamma &\xleftarrow{\$} \mathbb{Z}_q \\
s_\delta &\xleftarrow{\$} \mathbb{Z}_q \\
\hat{n} &= n_v + n_r + n_c \\
s' &\xleftarrow{\$} \left[0, 2^{\hat{n}} - 1\right] \\
B_0 &= g_1 \\
B_i &= g^{r_i} B_{i-1}^{\lambda'_i} \\
\alpha &= g^{s_\alpha} \prod_{i=1}^{N} g_i^{s'_i} \\
\beta_i &= g^{s_i} B_{i-1}^{s'_i} \\
\gamma &= g^{s_\gamma} \\
\delta &= g^{s_\delta}
\end{aligned}
$$

$$\mathcal{P} \xrightarrow{B_0, B_i, \alpha, \beta_i, \gamma, \delta} \mathcal{V}$$

$$c \xleftarrow{\$} [0, 2^{n_c} - 1]$$

$$\mathcal{P} \xleftarrow{c} \mathcal{V}$$

$$
\begin{aligned}
\lambda''_1 &= s_1 \\
\lambda''_i &= \lambda''_{i-1} \lambda'_i + s_i \\
d_\alpha &= ck + s_\alpha \\
d'_i &= c\lambda'_i + s'_i \\
d_i &= cr_i + s_i \\
d_\gamma &= c \left\langle \mathbf{s}, \mathbf{1} \right\rangle + s_\gamma \\
d_\delta &= c\lambda''_N + s_\delta
\end{aligned}
$$

$$\mathcal{P} \xrightarrow{d_\alpha, d'_i, d_i, d_\gamma, d_\delta} \mathcal{V}$$

$$\left(\prod_{j=1}^{N} c_{\mathbf{m_j}}^{\lambda_j}\right)^c \alpha \stackrel{?}{=} g^{d_\alpha} \prod_{i=1}^{N} g_i^{d'_i}$$

$$B_i^c \beta_i \stackrel{?}{=} g^{d_i} B_{i-1}^{d'_i}$$

$$\left(\prod_{j=1}^{N} c_{\mathbf{m_j}}^{1_j} \Big/ \prod_{i=1}^{N} g_i\right)^c \gamma \stackrel{?}{=} g^{d_\gamma}$$

$$\left(B^N \Big/ g^{\prod_{i=1}^{N} \lambda_i}\right)^c \delta \stackrel{?}{=} g^{d_\delta}$$

3.5 Proof of Knowledge of small exponents

The second step of the offline phase will be to prove that the random values used to re-encrypt are small. Remember that in order to re-encrypt a message, the following randomness is used: $\boldsymbol{r'^{(i)}_{\mathsf{E}}} = \left(r'^{(i)}_{\mathsf{E},1}, \ldots, r'^{(i)}_{\mathsf{E},n}\right)$, $\boldsymbol{e'^{(i)}_{\mathsf{E},u}} = \left(e'^{(i)}_{\mathsf{E},u,1}, \ldots, e'^{(i)}_{\mathsf{E},u,n}\right)$ and $\boldsymbol{e'^{(i)}_{\mathsf{E},v}} = \left(e'^{(i)}_{\mathsf{E},v,1}, \ldots, e'^{(i)}_{\mathsf{E},v,n}\right)$ for $i \in \{1, \ldots, N\}$. In our case, we would require that the coefficients of these vectors belong to $[-\beta + 1, \beta - 1]$ where $\beta = 2^k$. In order to prove this we are going to use the strategy proposed in [100] by Ling *et al.* As it is explained in [29] the probability of obtaining an element from the error distribution with norm larger than β is negligible (notice that β will depend on the parameters of the encryption). Even when this restriction on the re-encryption elements norm is applied, the RLWE samples remain pseudorandom. This prevents a corrupted node from modifiying the plaintext of the ciphertexts, while an honest node can still use the pseudorandomness to hide the relation between its input an output.

In order to prove that the re-encryption parameters are *small*, each of their coefficients are represented using their bit decomposition:

$$
\begin{aligned}
r'^{(i)}_{\mathsf{E},j} &= \sum_{l=0}^{k-1} r'^{(i)}_{\mathsf{E},j,l} 2^l \\
e'^{(i)}_{\mathsf{E},u,j} &= \sum_{l=0}^{k-1} e'^{(i)}_{\mathsf{E},u,j,l} 2^l \\
e'^{(i)}_{\mathsf{E},v,j} &= \sum_{l=0}^{k-1} e'^{(i)}_{\mathsf{E},v,j,l} 2^l
\end{aligned}
$$

with $r'^{(i)}_{\mathsf{E},j,l}, e'^{(i)}_{\mathsf{E},u,j,l}, e'^{(i)}_{\mathsf{E},v,j,l} \in \{-1, 0, 1\}$. Then, we can also express the bit decomposition of all the re-encryption parameters using matrix and vector notation:

$$
\underset{nN\times 1}{\begin{pmatrix} r'^{(1)}_{\mathsf{E},1} \\ r'^{(1)}_{\mathsf{E},2} \\ \vdots \\ r'^{(1)}_{\mathsf{E},n} \\ r'^{(2)}_{\mathsf{E},1} \\ \vdots \\ r'^{(N)}_{\mathsf{E},n} \end{pmatrix}} = \underset{nN\times k}{\begin{pmatrix} r'^{(1)}_{\mathsf{E},1,0} & r'^{(1)}_{\mathsf{E},1,1} & \cdots & r'^{(1)}_{\mathsf{E},1,k-1} \\ r'^{(1)}_{\mathsf{E},2,0} & r'^{(1)}_{\mathsf{E},2,1} & \cdots & r'^{(1)}_{\mathsf{E}2,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ r'^{(1)}_{\mathsf{E},n,0} & r'^{(1)}_{\mathsf{E},n,1} & \cdots & r'^{(1)}_{\mathsf{E},n,k-1} \\ r'^{(2)}_{\mathsf{E},1,0} & r'^{(2)}_{\mathsf{E},1,1} & \cdots & r'^{(2)}_{\mathsf{E},1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ r'^{(N)}_{\mathsf{E},n,0} & r'^{(N)}_{\mathsf{E},n,1} & \cdots & r'^{(N)}_{\mathsf{E},n,k-1} \end{pmatrix}} \underset{k\times 1}{\begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{k-1} \end{pmatrix}}
$$

$$\underset{2nN\times 1}{\begin{pmatrix} e'^{(1)}_{\mathsf{E},u,1} \\ \vdots \\ e'^{(1)}_{\mathsf{E},u,n} \\ e'^{(1)}_{\mathsf{E},v,1} \\ \vdots \\ e'^{(1)}_{\mathsf{E},v,n} \\ e'^{(2)}_{\mathsf{E},u,1} \\ \vdots \\ e'^{(N)}_{\mathsf{E},v,n} \end{pmatrix}} = \underset{2nN\times k}{\begin{pmatrix} e'^{(1)}_{\mathsf{E},u,1,0} & e'^{(1)}_{\mathsf{E},u,1,1} & \cdots & e'^{(1)}_{\mathsf{E},u,1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ e'^{(1)}_{\mathsf{E},u,n,0} & e'^{(1)}_{\mathsf{E},u,n,1} & \cdots & e'^{(1)}_{\mathsf{E},u,n,k-1} \\ e'^{(1)}_{\mathsf{E},v,1,0} & e'^{(1)}_{\mathsf{E},v,1,1} & \cdots & e'^{(1)}_{\mathsf{E},v,1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ e'^{(1)}_{\mathsf{E},v,n,0} & e'^{(1)}_{\mathsf{E},v,n,1} & \cdots & e'^{(1)}_{\mathsf{E},v,n,k-1} \\ e'^{(2)}_{\mathsf{E},u,1,0} & e'^{(2)}_{\mathsf{E},u,1,1} & \cdots & e'^{(2)}_{\mathsf{E},u,1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ e'^{(N)}_{\mathsf{E},v,n,0} & e'^{(N)}_{\mathsf{E},v,n,1} & \cdots & e'^{(N)}_{\mathsf{E},v,n,k-1} \end{pmatrix}} \underset{k\times 1}{\begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{k-1} \end{pmatrix}}$$

The commitment to each element of the decomposition can be expressed as:

$$c_{r'^{(i)}_{\mathsf{E},j,l}} = g^{\alpha_{r'^{(i)}_{\mathsf{E},j,l}}} g_i^{r'^{(i)}_{\mathsf{E},j,l}}$$
$$c_{e'^{(i)}_{\mathsf{E},u,j,l}} = g^{\alpha_{e'^{(i)}_{\mathsf{E},uj,l}}} g_i^{e'^{(i)}_{\mathsf{E},u,j,l}}$$
$$c_{e'^{(i)}_{\mathsf{E},v,j,l}} = g^{\alpha_{e'^{(i)}_{\mathsf{E},uj,l}}} g_i^{e'^{(i)}_{\mathsf{E},v,j,l}}$$

From these commitments we can easily compute the commitments to $r'^{(i)}_{\mathsf{E},j}$, $e'^{(i)}_{\mathsf{E},u,j}$ and $e'^{(i)}_{\mathsf{E},v,j}$: $c_{r'^{(i)}_{\mathsf{E},j}}$, $c_{e'^{(i)}_{\mathsf{E},u,j}}$, $c_{e'^{(i)}_{\mathsf{E},v,j}}$. For example, using the commitments to the elements $r'^{(i)}_{\mathsf{E},j,l}$, we can compute the commitment $c_{r'^{(i)}_{\mathsf{E},j}}$ in the following way:

$$c_{r'^{(i)}_{\mathsf{E},j}} = \prod_{l=0}^{k-1} (c_{r'^{(i)}_{\mathsf{E},j,l}})^{2^l} = \prod_{l=0}^{k-1} (g^{\alpha_{r'^{(i)}_{\mathsf{E},j,l}}} g_i^{r'^{(i)}_{\mathsf{E},j,l}})^{2^l} = g^{\sum_{l=0}^{k-1} 2^l \alpha_{r'^{(i)}_{\mathsf{E},j,l}}} g_i^{\sum_{l=0}^{k-1} r'^{(i)}_{\mathsf{E},j,l} 2^l} = g^{\alpha_{r'^{(i)}_{\mathsf{E},j}}} g_i^{r'^{(i)}_{\mathsf{E},j}}$$

And finally, using the commitments $c_{r'^{(i)}_{\mathsf{E},j}}$ we can compute the commitment to the vector $\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}$ which corresponds to the column j of matrix $\mathbf{R'}_{\mathsf{E}}$:

$$c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}} = \prod_{i=1}^{N} c_{r'^{(i)}_{\mathsf{E},j}} = \prod_{i=1}^{N} g^{\alpha_{r'^{(i)}_{\mathsf{E},j}}} g_i^{r'^{(i)}_{\mathsf{E},j}} = g^{\sum_{i=1}^{N} \alpha_{r'^{(i)}_{\mathsf{E},j}}} \prod_{i=1}^{N} g_i^{r'^{(i)}_{\mathsf{E},j}} = g^{\alpha_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}} \prod_{i=1}^{N} g_i^{r'^{(i)}_{\mathsf{E},j}}$$

We prove in zero knowledge that the elements $r'^{(i)}_{\mathsf{E},j,l}, e'^{(i)}_{\mathsf{E},u,j,l}, e'^{(i)}_{\mathsf{E},v,j,l}$ have one of the possible values in the set $\{-1, 0, 1\}$ using an OR-proof (see Protocol 2.2 in Section 2.3.2.3).

The protocol used to demonstrate that a value belongs to a specific set, $x \in \{-1, 0, 1\}$, is based on a zero knowledge proof that proves that the element x has one of the values in the set without revealing which one it is.

$$\Sigma\text{-proof}\left[\, x \,\middle|\, x \in \{-1, 0, 1\}, c = g^r h^x \,\right]$$

Informally, the proof consists of computing three proofs simultaneously, for $x = -1, x = 0$ and $x = 1$, where two of them will be simulated and only that which

corresponds to the real value of x will be the *real* proof. This is a standard proof [52] and we give the details hereunder.

$$s, t_{x+1}, t_{x-1}, e_{x+1}, e_{x-1} \xleftarrow{\$} \mathbb{Z}_q$$

$$d_y = \begin{cases} g^s & \text{if } y = x \\ g^{t_y} \left(ch^{-y}\right)^{-e_y} & \text{if } y \neq x \end{cases}$$

$$\mathcal{P} \xrightarrow{d_0, d_1, d_{-1}} \mathcal{V}$$

$$k \xleftarrow{\$} \mathbb{Z}_q$$

$$\mathcal{P} \xleftarrow{k} \mathcal{V}$$

$$e_x = k - e_{x+1} - e_{x-1}$$
$$t_x = s + re_x$$

$$\mathcal{P} \xrightarrow[t_0, t_1, t_{-1}]{e_0, e_1, e_{-1}} \mathcal{V}$$

$$k \stackrel{?}{=} e_0 + e_1 + e_{-1}$$
$$\forall y \in \{-1, 0, 1\}$$
$$g^{t_y} \stackrel{?}{=} (ch^{-y})^{e_y} d_y$$

Notice that given that the values of x could be -1, 0 or 1, variables t_{x-1}, t_x, t_{x+1} correspond to t_{-1}, t_0, t_1.

The *completeness* of the protocol is easy to demonstrate considering that if both the prover and the verifier follows the protocol, the equation $k = e_0 + e_1 + e_{-1}$ holds since $e_x = k - e_{x+1} - e_{x-1}$. Regarding the second verification equation, we will distinguish between the situation where $y = x$:

$$\begin{aligned} g^{s+re_x} &= g^{s+re_x} \\ g^{s+re_x} &= (g^r h^x h^{-x})^{e_x} g^s \\ g^{t_x} &= (ch^{-x})^{e_x} d_x \end{aligned}$$

and where $y \in \{x-1, x+1\}$:

$$\begin{aligned} g^{t_y} &= g^{t_y} \\ g^{t_y} &= (ch^{-y})^{e_y} g^{t_y} \left(ch^{-y}\right)^{-e_y} \\ g^{t_y} &= (ch^{-y})^{e_y} d_y \end{aligned}$$

In order to prove the *consistency*, we define two accepted transcriptions of the pro-

tocol:

$$\begin{aligned}(d_0, d_1, d_{-1}, k, t_0, t_1, t_{-1}, e_0, e_1, e_{-1})\\ \big(d_0, d_1, d_{-1}, k', t'_0, t'_1, t'_{-1}, e'_0, e'_1, e'_{-1}\big)\\ k \neq k'\end{aligned}$$

Since $k \neq k'$, one of the values e_y must be different from e'_y.

$$\begin{aligned}&e_{-1} + e_0 + e_1 = k \neq k' = e'_{-1} + e'_0 + e'_1\\ &\implies \exists y \in \{-1, 0, 1\} \text{ such that } e_y \neq e'_y\\ &\implies (e_y - e'_y) \neq 0 \in \mathbb{Z}_q\end{aligned}$$

On the other hand, given that both transcriptions are accepted:

$$\begin{aligned}g^{t_y} &= (ch^{-y})^{e_y} d_y\\ g^{t'_y} &= (ch^{-y})^{e'_y} d_y\\ g^{t_y - t'_y} &= (ch^{-y})^{e_y - e'_y}\\ g^{(t_y - t'_y)/(e_y - e'_y)} h^y &= c\end{aligned}$$

We can conclude that $\big((t_y - t'_y)/(e_y - e'_y), y\big)$ would be an opening for the commitment c to a value $y \in \{-1, 0, 1\}$.

Finally, the protocol is *zero-knowledge* since it is possible to construct a simulator that generates accepted transcriptions indistinguishable from real transcriptions between an honest prover and verifier.

$$\begin{aligned}t_{-1}, t_0, t_1, e_{-1}, e_0, e_1 &\xleftarrow{\$} \mathbb{Z}_q\\ k &= e_{-1} + e_0 + e_1\\ d_{-1} &= g^{t_{-1}} (ch)^{-e_{-1}}\\ d_0 &= g^{t_0} c^{-e_0}\\ d_1 &= g^{t_1} (c/h)^{-e_1}\\ &(d_0, d_1, d_{-1}, k, t_0, t_1, t_{-1}, e_0, e_1, e_{-1}) \text{ is a valid transcription.}\end{aligned}$$

3.6 Opening the commitments

Given the commitments to the permutation matrix and to the re-encryption matrices, the only thing that is left to prove is that these matrices have been used during the mixing process. This is an operation that should be done online since we need the list of encrypted messages to compute the proof. In order to do that we propose a methodology that differs from what Wikström proposes.

Given the commitments to the columns of matrices $\mathbf{M}$, $\mathbf{R}'_{\mathsf{E}}$, $\mathbf{E}'_{\mathsf{E},\mathbf{u}}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v}}$:

$$\begin{aligned} c_{\boldsymbol{m_j}} &= \mathsf{Com}\left(\boldsymbol{m_j}, \alpha_{\boldsymbol{m_j}}\right) \\ c_{\boldsymbol{r'_{\mathsf{E},j}}} &= \mathsf{Com}\left(\boldsymbol{r'_{\mathsf{E},j}}, \alpha_{\boldsymbol{r'_{\mathsf{E},j}}}\right) \\ c_{\boldsymbol{e'_{\mathsf{E},u,j}}} &= \mathsf{Com}\left(\boldsymbol{e'_{\mathsf{E},u,j}}, \alpha_{\boldsymbol{e'_{\mathsf{E},u,j}}}\right) \\ c_{\boldsymbol{e'_{\mathsf{E},v,j}}} &= \mathsf{Com}\left(\boldsymbol{e'_{\mathsf{E},v,j}}, \alpha_{\boldsymbol{e'_{\mathsf{E},v,j}}}\right) \end{aligned}$$

and Equation 3.6:

$$\begin{pmatrix}\mathbf{U}'' & \mathbf{V}''\end{pmatrix} = \mathbf{M}\begin{pmatrix}\mathbf{U} & \mathbf{V}\end{pmatrix} + \mathbf{R}'_{\mathsf{E}}\begin{pmatrix}\mathbf{A}^T & \mathbf{B}^T\end{pmatrix} + \begin{pmatrix}\mathbf{E}'_{\mathsf{E},\mathbf{u}} & \mathbf{E}'_{\mathsf{E},\mathbf{v}}\end{pmatrix}$$

we can compute the commitment to the output of the mix-node, i.e., commitments to $\mathbf{U}''$ and $\mathbf{V}''$. Note that each column k of these matrices can be expressed in the following way:

$$\begin{aligned} \boldsymbol{u''_k} &= \mathbf{M}\cdot\boldsymbol{u_k} + \mathbf{R}'_{\mathsf{E}}\cdot\boldsymbol{a_k} + \boldsymbol{e'_{\mathsf{E},u,k}} = \left(u''^{(1)}_k, \ldots, u''^{(N)}_k\right) \\ \boldsymbol{v''_k} &= \mathbf{M}\cdot\boldsymbol{v_k} + \mathbf{R}'_{\mathsf{E}}\cdot\boldsymbol{b_k} + \boldsymbol{e'_{\mathsf{E},v,k}} = \left(v''^{(1)}_k, \ldots, v''^{(N)}_k\right) \end{aligned}$$

In addition, knowing that $\boldsymbol{u_k} = \left(u^{(1)}_k, \ldots, u^{(N)}_k\right)$, $\boldsymbol{v_k} = \left(v^{(1)}_k, \ldots, v^{(N)}_k\right)$, $\boldsymbol{a_k} = (a_{1k}, \ldots, a_{nk})$, $\boldsymbol{b_k} = (b_{1k}, \ldots, b_{nk})$, $\boldsymbol{e'_{\mathsf{E},u,k}} = \left(e'^{(1)}_{\mathsf{E},u,k}, \ldots, e'^{(N)}_{\mathsf{E},u,k}\right)$ and $\boldsymbol{e'_{\mathsf{E},v,k}} = \left(e'^{(1)}_{\mathsf{E},v,k}, \ldots, e'^{(N)}_{\mathsf{E},v,k}\right)$, each element of vectors $\boldsymbol{u''_k}$ and $\boldsymbol{v''_k}$ is computed as:

$$\begin{aligned} u''^{(i)}_k &= \sum_{j=1}^{N} m_{ij}\cdot u^{(j)}_k + \sum_{j=1}^{n} r^{(i)}_{\mathsf{E},j}\cdot a_{jk} + e'^{(i)}_{\mathsf{E},u,k} \\ v''^{(i)}_k &= \sum_{j=1}^{N} m_{ij}\cdot v^{(j)}_k + \sum_{j=1}^{n} r^{(i)}_{\mathsf{E},j}\cdot b_{jk} + e'^{(i)}_{\mathsf{E},v,k} \end{aligned}$$

Finally, using the property of Pedersen commitments which allow us to compute a commitment to the product of a matrix by a vector from the commitment to the matrix (see Equation 3.2), we can calculate the commitment to each column of $\mathbf{U}''$ and $\mathbf{V}''$ in the following way:

$$\begin{aligned} \mathsf{Com}(\boldsymbol{u''_k}, \alpha_{\boldsymbol{u''_k}}) &= c_{\boldsymbol{e'_{\mathsf{E},u,k}}}\left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{u^{(j)}_k}\right)\left(\prod_{j=1}^{n} c_{\boldsymbol{r'_{\mathsf{E},j}}}^{a_{jk}}\right) \\ \mathsf{Com}(\boldsymbol{v''_k}, \alpha_{\boldsymbol{v''_k}}) &= c_{\boldsymbol{e'_{\mathsf{E},v,k}}}\left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{v^{(j)}_k}\right)\left(\prod_{j=1}^{n} c_{\boldsymbol{r'_{\mathsf{E},j}}}^{b_{jk}}\right) \end{aligned}$$

where the randomness used to compute the commitments is $\alpha_{\boldsymbol{u''_k}} = \alpha_{\boldsymbol{e'_{\mathsf{E},u,k}}} + \langle\boldsymbol{\alpha}_m, \boldsymbol{u_k}\rangle + \langle\boldsymbol{\alpha}_{r'}, \boldsymbol{a_k}\rangle$ and $\alpha_{\boldsymbol{v''_k}} = \alpha_{\boldsymbol{e'_{\mathsf{E},v,k}}} + \langle\boldsymbol{\alpha}_m, \boldsymbol{v_k}\rangle + \langle\boldsymbol{\alpha}_{r'}, \boldsymbol{b_k}\rangle$.

The only thing that the mix-node should do in order to prove that it has used the appropriate values during the shuffling, is to open the commitments above revealing the openings:

$$\left(\alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{u},\boldsymbol{k}}} + \langle \boldsymbol{\alpha}_m, \left(u_k^{(1)}, \ldots, u_k^{(N)}\right)\rangle + \langle \boldsymbol{\alpha}_{r'}, (a_{1k}, \ldots, a_{nk})\rangle\right) \quad \forall k \in [1, \ldots, n]$$
$$\left(\alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{v},\boldsymbol{k}}} + \langle \boldsymbol{\alpha}_m, \left(v_k^{(1)}, \ldots, v_k^{(N)}\right)\rangle + \langle \boldsymbol{\alpha}_{r'}, (b_{1k}, \ldots, b_{nk})\rangle\right) \quad \forall k \in [1, \ldots, n]$$

The verifier has to check that these values are appropriate openings of the commitments in order to verify the node has used the committed matrices $\mathbf{M}, \mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v}}$ to shuffle the encrypted messages (at its input).

As we have seen above, given the commitments to $\mathbf{M}, \mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v}}$ we can compute the commitment to the matrix of permuted votes $\mathbf{M}\begin{pmatrix}\mathbf{U} & \mathbf{V}\end{pmatrix}$ and the re-encryption matrix $\left(\mathbf{R}'_{\mathsf{E}}\begin{pmatrix}\mathbf{A}^T & \mathbf{B}^T\end{pmatrix} + \begin{pmatrix}\mathbf{E}'_{\mathsf{E},\mathbf{u}} & \mathbf{E}'_{\mathsf{E},\mathbf{v}}\end{pmatrix}\right)$. Notice that the $2n$ linear combinations of the values $\alpha_{\boldsymbol{m_j}}, \alpha_{\boldsymbol{r}'_{\mathsf{E},\boldsymbol{j}}}, \alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}, \alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}$ that the mix-node reveals, allow us to open the commitments to the sum of these matrices, but not to each matrix separately. Given that $\boldsymbol{\alpha}_m$, and $\boldsymbol{\alpha}_{r'}$ appear on all the openings that we reveal we have to double check if they could leak any information about any relations between the α's that (in a post-quantum scenario) may reveal information about the permutation and the re-encryption elements. This is not the case because all the $\alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}$ and $\alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}$ are uniformly and independently chosen from $\mathbb{Z}_q$. All the linear combinations that we reveal have a different $\alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}$ or $\alpha_{\boldsymbol{e}'_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}$, and this implies that the combinations are also uniformly and independently distributed, and thereby it is impossible to isolate any of the α.

3.7 Full mixing protocol and its properties

In previous sections we have explained which are the main components of the shuffle proof and how they work. In this section we want to show the whole protocol and also to discuss the properties of the proof: completeness, soundness and zero-knowledge.

Offline phase

The mix-node:

Picks the re-encryption parameters $\boldsymbol{r}'^{(\boldsymbol{i})}_{\mathsf{E}}, \boldsymbol{e}'^{(\boldsymbol{i})}_{\mathsf{E},\boldsymbol{u}}, \boldsymbol{e}'^{(\boldsymbol{i})}_{\mathsf{E},\boldsymbol{v}} \xleftarrow{\$} \mathbb{Z}_q^n$; $\forall i \in \{1, \ldots, N\}$ and the permutation π.

Picks $\alpha_{r'^{(i)}_{\mathsf{E},j,l}}, \alpha_{e'^{(i)}_{\mathsf{E},u,j,l}}, \alpha_{e'^{(i)}_{\mathsf{E},v,j,l}} \xleftarrow{\$} \mathbb{Z}_q$; $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, n\}$ and $\forall l \in \{0, \ldots, k-1\}$.

Computes the bit-decomposition of each element in $\boldsymbol{r}'^{(\boldsymbol{i})}_{\mathsf{E}}$, $\boldsymbol{e}'^{(\boldsymbol{i})}_{\mathsf{E},\boldsymbol{u}}$ and $\boldsymbol{e}'^{(\boldsymbol{i})}_{\mathsf{E},\boldsymbol{v}}$ as explained in Section 3.5 and obtains:

$$r'^{(i)}_{\mathsf{E},j} = \sum_{l=0}^{k-1} r'^{(i)}_{\mathsf{E},j,l} 2^l$$

$$e'^{(i)}_{\mathsf{E},u,j} = \sum_{l=0}^{k-1} e'^{(i)}_{\mathsf{E},u,j,l} 2^l$$

$$e'^{(i)}_{\mathsf{E},v,j} = \sum_{l=0}^{k-1} e'^{(i)}_{\mathsf{E},v,j,l} 2^l$$

Commits to each component of the bit decomposition using the randomness generated in a previous step:

$$c_{r'^{(i)}_{\mathsf{E},j,l}} = \mathsf{Com}(r'^{(i)}_{\mathsf{E},j,l}, \alpha_{r'^{(i)}_{\mathsf{E},j,l}})$$

$$c_{e'^{(i)}_{\mathsf{E},u,j,l}} = \mathsf{Com}(e'^{(i)}_{\mathsf{E},u,j,l}, \alpha_{e'^{(i)}_{\mathsf{E},u,j,l}})$$

$$c_{e'^{(i)}_{\mathsf{E},v,j,l}} = \mathsf{Com}(e'^{(i)}_{\mathsf{E},v,j,l}, \alpha_{e'^{(i)}_{\mathsf{E},v,j,l}})$$

Publishes the commitments computed in the previous step $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, n\}$ and $\forall l \in \{1, \ldots, k-1\}$.

Demonstrates that each element of the bit decomposition is *small*, i.e., is either a -1, 0 or -1.

$$\Sigma\text{-proof}\left[r'^{(i)}_{\mathsf{E},j,l} \,\middle|\, \left(c_{r'^{(i)}_{\mathsf{E},j,l}} = g^{\alpha_{r'^{(i)}_{\mathsf{E},j,l}}} g_i^{r'^{(i)}_{\mathsf{E},j,l}} \right) \wedge \left(r'^{(i)}_{\mathsf{E},j,l} = -1 \vee r'^{(i)}_{\mathsf{E},j,l} = 0 \vee r'^{(i)}_{\mathsf{E},j,l} = 1 \right) \right]$$

$$\Sigma\text{-proof}\left[e'^{(i)}_{\mathsf{E},u,j,l} \,\middle|\, \left(c_{e'^{(i)}_{\mathsf{E},u,j,l}} = g^{\alpha_{e'^{(i)}_{\mathsf{E},u,j,l}}} g_i^{e'^{(i)}_{\mathsf{E},u,j,l}} \right) \wedge \left(e'^{(i)}_{\mathsf{E},u,j,l} = -1 \vee e'^{(i)}_{\mathsf{E},u,j,l} = 0 \vee e'^{(i)}_{\mathsf{E},u,j,l} = 1 \right) \right]$$

$$\Sigma\text{-proof}\left[e'^{(i)}_{\mathsf{E},v,j,l} \,\middle|\, \left(c_{e'^{(i)}_{\mathsf{E},v,j,l}} = g^{\alpha_{e'^{(i)}_{\mathsf{E},v,j,l}}} g_i^{e'^{(i)}_{\mathsf{E},v,j,l}} \right) \wedge \left(e'^{(i)}_{\mathsf{E},v,j,l} = -1 \vee e'^{(i)}_{\mathsf{E},v,j,l} = 0 \vee e'^{(i)}_{\mathsf{E},v,j,l} = 1 \right) \right]$$

The details of the zero-knowledge proof are given in Section 3.5.

Picks $\boldsymbol{\alpha}_m \xleftarrow{\$} \mathbb{Z}_q^N$ and generates the matrix $\mathbf{M}$ which characterizes the permutation π selected in the first step of the protocol.

Commits to the permutation matrix by committing to each of its columns as shown in Section 3.2.

$$\mathbf{c_M} = \mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}_m)$$

Publish the commitment to the permutation matrix.

Then, the verifier selects $\lambda \xleftarrow{\$} \mathbb{Z}_q^N$ and sends it to the mix-node.

And finally, the mix-node using the vector λ sent by V and the commitment $\mathbf{c_M}$ demonstrates that the committed matrix is a permutation matrix using the zero-knowledge proof explained in Section 3.4.

$$\Sigma\text{-proof}\left[\mathbf{M} \,\middle|\, (\mathbf{c_M} = \mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}_m)) \wedge (\mathbf{M1} = \mathbf{1}) \wedge \left(\prod_{i=1}^{N} \lambda_i = \prod_{i=1}^{N} \lambda'_i \,\middle|\, \lambda' = \mathbf{M}\lambda \right) \right]$$

Online phase

Given a list of ciphertexts $(\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}) \in \mathbb{Z}_q^{2n}$ for all $i \in \{1, \ldots, N\}$ the mix-node uses the permutation and the re-encryption parameters selected during the offline phase to shuffle them following the equation:

$$\begin{pmatrix}\mathbf{U}'' & \mathbf{V}''\end{pmatrix} = \mathbf{M}\begin{pmatrix}\mathbf{U} & \mathbf{V}\end{pmatrix} + \mathbf{R}'_{\mathsf{E}}\begin{pmatrix}\mathbf{A}^T & \mathbf{B}^T\end{pmatrix} + \begin{pmatrix}\mathbf{E}'_{\mathsf{E},\mathbf{u}} & \mathbf{E}'_{\mathsf{E},\mathbf{v}}\end{pmatrix}$$

The output is the list of shuffled ciphertexts $(\boldsymbol{u''}^{(i)}, \boldsymbol{v''}^{(i)})$ for all $i \in \{1, \ldots, N\}$.

From the randomness used to commit to each element of the bit-decomposition during the offline phase, the mix-node computes the randomness needed to commit to each $r'^{(i)}_{\mathsf{E},j}, e'^{(i)}_{\mathsf{E},u,j}$ and $e'^{(i)}_{\mathsf{E},v,j}$.

$$\alpha_{r'^{(i)}_{\mathsf{E},j}} = \sum_{l=0}^{k-1} \alpha_{r'^{(i)}_{\mathsf{E},j,l}} 2^l, \quad \alpha_{e'^{(i)}_{\mathsf{E},u,j}} = \sum_{l=0}^{k-1} \alpha_{e'^{(i)}_{\mathsf{E},u,j,l}} 2^l, \quad \alpha_{e'^{(i)}_{\mathsf{E},v,j}} = \sum_{l=0}^{k-1} \alpha_{e'^{(i)}_{\mathsf{E},v,j,l}} 2^l$$

From the previous randomness the mix-node computes the randomness needed to commit to each column of $\mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v}}$.

$$\alpha_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}} = \sum_{i=1}^{N} \alpha_{r'^{(i)}_{\mathsf{E},j}}, \quad \alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}} = \sum_{i=1}^{N} \alpha_{e'^{(i)}_{\mathsf{E},u,j}}, \quad \alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}} = \sum_{i=1}^{N} \alpha_{e'^{(i)}_{\mathsf{E},v,j}}$$

The vector $\boldsymbol{\alpha}_{r'}$ is defined as $\boldsymbol{\alpha}_{r'} = \left(\alpha_{\boldsymbol{r'}_{\mathsf{E},\mathbf{1}}}, \ldots, \alpha_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{n}}}\right)$.

Finally, the mix-node computes the randomness that will be used by the verifier to open the commitments to the output of the mixing process.

$$\alpha_{\boldsymbol{u''_k}} = \alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{k}}} + \langle \boldsymbol{\alpha}_m, \boldsymbol{u_k} \rangle + \langle \boldsymbol{\alpha}_{r'}, \boldsymbol{a_k} \rangle, \quad \alpha_{\boldsymbol{v''_k}} = \alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{k}}} + \langle \boldsymbol{\alpha}_m, \boldsymbol{v_k} \rangle + \langle \boldsymbol{\alpha}_{r'}, \boldsymbol{b_k} \rangle$$

The mix-node sends the openings $\alpha_{\boldsymbol{u''_k}}$ and $\alpha_{\boldsymbol{v''_k}}$ to the verifier.

The verifier V uses the commitments to the bit-decomposition elements published during the offline phase to compute the commitments to the elements $r'^{(i)}_{\mathsf{E},j}, e'^{(i)}_{\mathsf{E},u,j}$ and $e'^{(i)}_{\mathsf{E},v,j}$; $\forall i \in [1, \ldots, N]$ and $\forall j \in [1, \ldots, n]$.

$$c_{r'^{(i)}_{\mathsf{E},j}} = \prod_{l=0}^{k-1} \left(c_{r'^{(i)}_{\mathsf{E},j,l}}\right)^{2^l} \quad c_{e'^{(i)}_{\mathsf{E},u,j}} = \prod_{l=0}^{k-1} \left(c_{e'^{(i)}_{\mathsf{E},u,j,l}}\right)^{2^l} \quad c_{e'^{(i)}_{\mathsf{E},v,j}} = \prod_{l=0}^{k-1} \left(c_{e'^{(i)}_{\mathsf{E},v,j,l}}\right)^{2^l}$$

The verifier V uses the commitments computed in the previous step to compute the commitments to each column of $\mathbf{R}'_{\mathsf{E}}, \mathbf{E}'_{\mathsf{E},\mathbf{u}}$ and $\mathbf{E}'_{\mathsf{E},\mathbf{v}}$.

$$c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}} = \prod_{i=1}^{N} c_{r'^{(i)}_{\mathsf{E},j}} \quad c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}} = \prod_{i=1}^{N} c_{e'^{(i)}_{\mathsf{E},u,j}} \quad c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}} = \prod_{i=1}^{N} c_{e'^{(i)}_{\mathsf{E},v,j}}$$

.

The verifier V commits to each column of the matrices $\mathbf{U}''$ and $\mathbf{V}''$ which contains the input ciphertexts permuted and re-encrypted. These commitments are computed using the opening revealed by the mix-node.

$$\mathsf{Com}(\boldsymbol{u''_k}, \alpha_{\boldsymbol{u''_k}}), \quad \mathsf{Com}(\boldsymbol{v''_k}, \alpha_{\boldsymbol{v''_k}})$$

Finally, the verifier V uses the commitments computed in previous steps to check that the following equations hold (this technique is explained in detail in Section 3.6):

$$c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{k}}} \left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{u_k^{(j)}} \right) \left(\prod_{j=1}^{n} c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}^{a_{jk}} \right) \stackrel{?}{=} \mathsf{Com}(\boldsymbol{u''_k}, \alpha_{\boldsymbol{u''_k}})$$

$$c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{k}}} \left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{v_k^{(j)}} \right) \left(\prod_{j=1}^{n} c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}^{b_{jk}} \right) \stackrel{?}{=} \mathsf{Com}(\boldsymbol{v''_k}, \alpha_{\boldsymbol{v''_k}})$$

We finally discuss the properties of protocol.

Completeness. Completeness follows from the homomorphic property of the Pedersen commitment and the completeness of the Σ-protocols for the *small* elements and the permutation matrix. The prover computes $\alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}$ and $\alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}$ for all $j \in [1, \ldots, n]$ using the random elements from the initial commitments. Then, if the prover has been honest, the verifier builds the commitments to the output using the published commitments and applying Equation 3.6, and check that $\alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}}$ and $\alpha_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{v},\boldsymbol{j}}}$ are valid openings for the output commitments.

Soundness. Soundness follows from the homomorphic and binding properties of the Pedersen commitment and from the soundness of the Σ-protocols for the *small* elements and the permutation matrix. The prover has published some commitments and proved knowledge of valid openings that satisfy the required conditions. When combined into a commitment to the output he shows a valid opening. Given that the commitment scheme is binding this implies that the output of the mix node is really the desired permutation and re-randomization of the input.

This property is the only one that would not hold in a quantum scenario, as the binding property of the Pedersen commitment would be broken. Nevertheless, until the first practical quantum computer is build soundness would be achieved by our protocol.

Zero-knowledge. We can build a simulator that produces transcriptions indistinguishable from the real interactions between an honest prover and a verifier.

Given λ and the responses of the Σ-protocols we choose π and $r'^{(i)}_{\mathsf{E},j,l}, e'^{(i)}_{\mathsf{E},u,j,l}, e'^{(i)}_{\mathsf{E},v,j,l}$ $(\forall l \in \{0, \ldots, k-1\})$ uniformly at random except for $e'^{(1)}_{\mathsf{E},u,j,0}, e'^{(1)}_{\mathsf{E},v,j,0}$. We compute its commitments, publish them and answer the challenges from the Σ-protocols as usual. Then we choose $\alpha_{\boldsymbol{u''_k}}$ and $\alpha_{\boldsymbol{v''_k}}$ uniformly at random and we define $\widehat{c}_{e'^{(1)}_{\mathsf{E},u,j,0}}$ and $\widehat{c}_{e'^{(1)}_{\mathsf{E},v,j,0}}$ in the following way:

$$c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{j}}} = \prod_{i=1}^{N} c_{e'^{(i)}_{\mathsf{E},u,j}} = \prod_{i=1}^{N} \left(\prod_{l=0}^{k-1} \left(c_{e'^{(i)}_{\mathsf{E},u,j,l}} \right)^{2^l} \right) = \prod_{i=1}^{N} \left(c_{e'^{(i)}_{\mathsf{E},u,j,0}} \prod_{l=1}^{k-1} \left(c_{e'^{(i)}_{\mathsf{E},u,j,l}} \right)^{2^l} \right) =$$
$$= c_{e'^{(1)}_{\mathsf{E},u,j,0}} \prod_{l=1}^{k-1} \left(c_{e'^{(1)}_{\mathsf{E},u,j,l}} \right)^{2^l} \prod_{i=2}^{N} \left(c_{e'^{(i)}_{\mathsf{E},u,j,0}} \prod_{l=1}^{k-1} \left(c_{e'^{(i)}_{\mathsf{E},u,j,l}} \right)^{2^l} \right) =$$
$$= c_{e'^{(1)}_{\mathsf{E},u,j,0}} \prod_{l=1}^{k-1} \left(c_{e'^{(1)}_{\mathsf{E},u,j,l}} \right)^{2^l} \prod_{i=2}^{N} c_{e'^{(i)}_{\mathsf{E},u,j}}$$

$$c_{\boldsymbol{e'}_{\mathsf{E},\boldsymbol{u},\boldsymbol{k}}} \left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{u_k^{(j)}} \right) \left(\prod_{j=1}^{n} c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}^{a_{jk}} \right) = \mathsf{Com}(\boldsymbol{u''_k}, \alpha_{\boldsymbol{u''_k}})$$
$$\left(c_{e'^{(1)}_{\mathsf{E},u,j,0}} \prod_{l=1}^{k-1} \left(c_{e'^{(1)}_{\mathsf{E},u,j,l}} \right)^{2^l} \prod_{i=2}^{N} c_{e'^{(i)}_{\mathsf{E},u,j}} \right) \left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{u_k^{(j)}} \right) \left(\prod_{j=1}^{n} c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}^{a_{jk}} \right) = \mathsf{Com}(\boldsymbol{u''_k}, \alpha_{\boldsymbol{u''_k}})$$

And finally we obtain:

$$\widehat{c}_{e'^{(1)}_{\mathsf{E},u,j,0}} = \frac{\mathsf{Com}(\boldsymbol{u''_k}, \alpha_{\boldsymbol{u''_k}})}{\left(\prod_{l=1}^{k-1} \left(c_{e'^{(1)}_{\mathsf{E},u,j,l}} \right)^{2^l} \right) \left(\prod_{i=2}^{N} c_{e'^{(i)}_{\mathsf{E},u,j}} \right) \left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{u_k^{(j)}} \right) \left(\prod_{j=1}^{n} c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}^{a_{jk}} \right)}$$
$$\widehat{c}_{e'^{(1)}_{\mathsf{E},v,j,0}} = \frac{\mathsf{Com}(\boldsymbol{v''_k}, \alpha_{\boldsymbol{v''_k}})}{\left(\prod_{l=1}^{k-1} \left(c_{e'^{(1)}_{\mathsf{E},v,j,l}} \right)^{2^l} \right) \left(\prod_{i=2}^{N} c_{e'^{(i)}_{\mathsf{E},v,j}} \right) \left(\prod_{j=1}^{N} c_{\boldsymbol{m_j}}^{v_k^{(j)}} \right) \left(\prod_{j=1}^{n} c_{\boldsymbol{r'}_{\mathsf{E},\boldsymbol{j}}}^{b_{jk}} \right)}$$

The only thing that is left to prove is that $\widehat{c}_{e'^{(1)}_{\mathsf{E},u,j,0}}$ and $\widehat{c}_{e'^{(1)}_{\mathsf{E},v,j,0}}$ are commitments to -1, 0 or 1. As we have the response from the verifier we can simulate these proofs and publish its outputs. By construction this simulation will be a valid conversation, equally distributed as any honest conversation since $\alpha_{1,k}$ and $\alpha_{2,k}$ follow the same uniformly random distribution as if they were computed using linear combinations of other uniformly random elements. Fake commitments $\widehat{c}_{e'^{(1)}_{\mathsf{E},u,j,0}}$ and $\widehat{c}_{e'^{(1)}_{\mathsf{E},v,j,0}}$ follow again a uniformly random distribution as they will do if they were honestly obtained.

The same applies to the outputs of the Σ-protocols, both the one proving that an element is $-1, 0, 1$ and Wikström's protocol for the characterization of a committed permutation matrix.

The zero-knowledge property will not be compromised with quantum computers as the distribution of the simulated proof is not only computationally indistinguishable but completely identical to the honest distribution, thanks to the perfectly hiding property of the Pedersen commitments.

3.8 Conclusions

In this chapter we have proposed the first universally verifiable proof of a shuffle for a lattice-based cryptosystem. The messages at the input of the mix-net are encrypted using a post-quantum encryption scheme, i.e., the RLWE encryption system, and then they are shuffled by the mix-nodes. In order to prove the correctness of this shuffle each node must provide a proof of a shuffle, demonstrating that the protocol has been executed correctly without leaking any secret information. Our proposal follows the idea presented in [153] but introduces two significant differences: during the offline part the random elements used to re-encrypt the ciphertexts are committed using the generalized version of Pedersen commitment and it is proved that these elements belong to a certain interval using OR-proofs. On the other hand, during the online part each node computes a commitment to its output using the homomorphic properties of both the commitment scheme and the encryption scheme. Opening this commitment the mix-node proves that it has used the values committed during the offline part to compute its output. Revealing this opening does not give any information about the secret information required to do the shuffling.

Since the shuffle proof uses Pedersen commitments, which are perfectly hiding but only computationally binding under the discrete logarithm assumption, we cannot claim that the proof is fully post-quantum. Although our proposal of providing long-term privacy to the protocol is achieved since the proof will be zero-knowledge also in a future with quantum computers, the soundness property would not hold in a quantum scenario. This implies that a successful verification of the proof will not give us any guarantee on the validity of the statement proven.

On the other hand, it is worth noticing that shuffling the votes is not enough to guarantee the voters' privacy, as the system can be insecure, for instance, due to malleability attacks [150]. To avoid this kind of attack additional security proofs might be provided before the mixing process starts.

Regarding efficiency, the number of OR-proofs to be computed by each mix node is proportional to knN, where N is the number of encrypted messages received by the node, n is the dimension of the lattice and k is the number of bits of each element of the re-encryption matrices. There are some techniques that allow to reduce the computational cost of these proofs and we leave for future work to explore these improvements. We refer the reader to [153] for the details about the efficiency of the ZKP for a permutation matrix.

In the next chapter we solve some of the issues that have arisen from this construction, by proposing a new proof of a shuffle which is entirely based on post-quantum assumptions.

Chapter 4

Fully post-quantum proof of a shuffle

4.1 Introduction

In Chapter 3 we have given an overview to the state of the art of mixing protocols, starting with Chaum's mix-net [42] and ending with Boyen *et al.* proposal [32], and we have presented our protocol [47] for building a proof of a shuffle for lattice-based cryptography which is, as far as we know, the first universally verifiable mix-net for a post-quantum cryptosystem.

As we have explained, our first proposal requires Pedersen commitments, whose binding property is based on the discrete logarithm problem and for this reason we cannot consider the proof fully post-quantum. With the aim of improving our previous work, we propose in this chapter a proof of a shuffle that is fully constructed over lattice-based cryptography [48] and the first for RLWE encryption schemes, which makes it secure in a post-quantum scenario. The proof is based on Bayer and Groth's proposal [22] and uses a commitment scheme which is perfectly binding and computationally hiding under the Learning With Errors over Rings (RLWE) assumption. Finally, we also provide a formal definition for security of a mix-node and prove security of our proposal using the sequence of games approach.

This proof is used by the mixing protocol of our post-quantum online voting system (Chapter 5) in order to ensure the correctness of the shuffle but also long-term privacy.

4.1.1 Related work

After the introduction of the idea of a shuffle by Chaum in 1981 [42], several schemes have been proposed. The first universally verifiable mix-net is presented in [133] and gives a proof to check the correctness of the shuffle. Later, several solutions for an efficient universally verifiable mix-net are proposed [4, 5, 6, 109] and in [70] Furukawa and Sako suggest a paradigm based on permutation matrices in the common reference string model (CRS) for proving the correctness of a shuffle, that was improved in [69, 88]. The latest proposal for a CRS based proof of a shuffle is [35] by Bünz *et al.* Wikström also uses this idea of the permutation matrix and presents in [153] a

proof of a shuffle that can be split in an offline and online phase in order to reduce the computational complexity in the online part.

On the other hand, Neff [116] proposes another paradigm based on polynomials being identical under permutation of their roots, obtaining Honest Verifier Zero-Knowledge (HVZK) proof and improved later in [84, 117] with the drawback that these constructions are 7-move proofs. Unlike previous proposals, Groth and Ishai [86] and Bayer and Groth [22] give a practical shuffle argument with sub-linear communication complexity. We are going to use the ideas presented in [22] to build our protocol.

None of these proofs are constructed using post-quantum cryptography and, as far as we know, until the proof of a shuffle explained in this chapter was presented, only two proposals were published whose security relies on the complexity of solving lattice problems. The first is our previous construction [47], which consists on a proof of a shuffle based on lattices but that cannot be considered fully post-quantum since it uses Pedersen commitments, whose binding property relies on the discrete logarithm problem. Moreover, in [47] there is no formal definition of security, necessary to precisely know how it can be embedded in a larger construction. The second one is by Strand [145], who presents a verifiable shuffle for the GSW cryptosystem that works with any homomorphic commitment scheme. Using the lattice-based commitment scheme [20] makes the proof fully post-quantum. Additionally, there have been some proposals for a lattice-based universal re-encryption for mix-nets [141] but none of them give a proof of a shuffle.

Finally, regarding security definitions for mix-nets, in [150] Wikström provides one for a single re-encryption mix-node. It is important to note that as Wikström remarks this is not enough to completely ensure privacy since a definition of security of a complete mix-net must involve several other aspects, regarding the validity of the input messages or decryption proofs.

4.1.2 Our proposal

We propose a proof of a shuffle fully constructed over lattices. The existing published proposal for a universally verifiable proof of a shuffle for RLWE encryptions [47] based on [146], uses Generalized Pedersen commitments to hide the secret re-randomization elements. This would not be sound in a post-quantum scenario, as it is based on DL assumptions. Naively replacing the commitment scheme with the one proposed by Benhamouda *et al.* yields several difficulties since it is useful when committing to polynomials, but is quite inefficient if we only want to commit to a bit, as is the case with the entries of a permutation matrix. The fact that $\mathbb{Z}_q[x]/(x^n+1)$ is not an integral domain also has some implications for the characterization of a permutation matrix proposed in [146], that cannot be proven directly and would require additional statements different from the ones discussed in [47].

Due to this, the proposal we present in this chapter [48] is based on the technique introduced by Bayer and Groth in [22] to construct a shuffle argument; nevertheless it is not a direct adaptation of it since working with lattices requires different techniques to be applied.

The first step of the proof, which is also the first difference with [22], consists on

committing to the re-encryption parameters in order to demonstrate that they meet certain constraints. This is done using the commitment scheme and the ZKPoK proposed by Benhamouda *et al.* [29] which are perfectly binding and computationally hiding under the RLWE assumption and satisfy special soundness and special HVZK. The next step consists on proving knowledge of the permutation. The general idea here is to prove that two sets contain the same elements. This is done by computing two polynomials, each of them having as roots the elements of each set, and proving that both polynomials are equal.

The last step will prove knowledge of the re-encryption parameters, and this introduces another difference between Bayer and Groth's protocol and ours. While they demonstrate that there exists a linear combination of the parameters such that an equality holds, we have to use a different technique, since the re-encryption parameters in a RLWE re-encryption scheme are taken from an error distribution and a linear combination of them would imply the error grows uncontrollably, causing decryption errors. Indeed, what we will need to prove is that some hidden elements have small norm and also that several committed elements satisfy a polynomial relation. As these proofs are generally costly we are going to use amortized protocols to reduce the communication cost. The first amortized protocol is presented in [51] by Cramer *et al.*, it is improved first by del Pino and Lyubashevsky [55] and later by Baum and Lyubashevsky in [21].

Proofs of a shuffle commonly require universal verifiability, meaning that a proof must be generated and also published, so it can be verified by any observer. Classically, this kind of interactive protocols can be transformed into non-interactive protocols by means of the Fiat-Shamir heuristics, replacing the random responses from the verifier with a hash of the previous elements in the conversation, achieving a protocol secure in the Random Oracle Model (ROM).

However, as it is exposed in [149], this method is not secure anymore in the Quantum Random Oracle Model (QROM). As far as we know the only quantum secure general transformation from an interactive protocol to a non-interactive version is the one described by [148]. Therefore, a universally verifiable version of our protocol in the QROM requires further considerations.

Finally, we give a definition of security, based on the one proposed by Wikström in [150], and we provide a proof of security for our mix-node. His proposal implies that no adversary can properly compute two indices for the input and the output respectively such that the messages encrypted in the corresponding ciphertexts are the same, except with a probability negligibly close to the probability given by a random guess. In his definition the adversary might have some knowledge of correlations between the input messages. We provide a definition of security allowing the adversary to have full control over the input of the mix-node, and we prove that our construction meets this definition. This is a new formal definition of security, stronger than that given in [150].

The organization of the chapter is as follows: in Section 4.2 we present Bayer and Groth proof of a shuffle and in Section 4.3 the main building blocks of our proof: the encryption and commitment scheme and the ZKPoKs. In Section 4.4 we show an overview of our shuffling protocol and the details of the construction are given in Section 4.5. Finally, in Section 4.6 we prove that the mix-node is secure.

4.2 Efficient zero-knowledge argument for correctness of a shuffle

As mentioned in the introduction, our proposal is based on the shuffle argument given by Bayer and Groth in [22]. Although is not a direct adaptation of it, we want to give some intuitions here in order to better understand our construction presented in Section 4.5.

The general idea of the shuffle argument is to demonstrate knowledge of a permutation π and some re-encryption parameters $\{\rho_i\}_{i=1}^N$ such that the set of ciphertexts at the output of the shuffle $\{C_i''\}_{i=1}^N$ are those at the input $\{C_i\}_{i=1}^N$ permuted and re-encrypted using the equation $C_i'' = C_i' \mathsf{Enc}_{pk}(1, \rho_i)$ where $C_i' = C_{\pi(i)}$. In order to construct the proof, Bayer and Groth use the combination of two arguments: the multi-exponentiation ($\Sigma_{\text{multi-exp}}$) and the product argument ($\Sigma_{\text{prod-arg}}$). We are not going to enter into details about them since they are specific to ElGamal encryption and the generalized version of Pedersen commitment, but we want to give an overview of the main ideas behind the proof.

The proof can be divided in several steps (full proof is shown in Protocol 4.1):

- The prover P computes the permutation of the indexed set of elements $\{1, \ldots, N\}$: $\mathbf{a} = \{\pi(i)\}_{i=1}^N$. If we define N as $N = m \cdot n$, the vector $\mathbf{a}$ is indeed a matrix $\mathbf{A}$ with m columns and n rows, where each column is $\mathbf{a_j} = (a_{1j}, \ldots, a_{nj})$ (note that $a_{11} = \pi(1)$ and $a_{nm} = \pi(N)$).

- P picks $\mathbf{r} \in \mathbb{Z}_q^m$ and computes the commitment to $\mathbf{A}$ using as randomness the vector $\mathbf{r}$: $\mathbf{c_A} = \mathsf{Com}(\mathbf{A}, \mathbf{r}) = (c_{\mathbf{a_1}}, \ldots, c_{\mathbf{a_m}})$, where $c_{\mathbf{a_j}} = g^{r_j} \prod_{l=1}^n g_l^{a_{lj}}$.

- The verifier sends a challenge $x \in \mathbb{Z}_q^*$ and the prover computes $\mathbf{b} = \{x^{\pi(i)}\}_{i=1}^N$. Again, if the use that $N = m \cdot n$ we can express the vector $\mathbf{b}$ as a matrix $\mathbf{B}$.

- The matrix $\mathbf{B}$ is committed using as randomness the vector $\mathbf{s}$: $\mathbf{c_B} = \mathsf{Com}(\mathbf{B}, \mathbf{s}) = (c_{\mathbf{b_1}}, \ldots, c_{\mathbf{b_m}})$, where $c_{\mathbf{b_j}} = g^{s_j} \prod_{l=1}^n g_l^{b_{lj}}$.

- It is demonstrated that the permutation used to compute $\mathbf{a}$ and $\mathbf{b}$ is the same, meaning that the prover has a commitment to $\{x^1, \ldots, x^N\}$ permuted in an order that was fixed before receiving x. This demonstration is done in the following way:

  - The verifier V sends two values y and z.
  - The prover P builds the following N elements using $y, z, \mathbf{a}$ and $\mathbf{b}$:

    $$d_1 - z = y\pi(1) + x^{\pi(1)} - z, \ldots, d_N - z = y\pi(N) + x^{\pi(N)} - z$$

    Note that using the homomorphic properties of the Pedersen commitment we can compute the commitment to $\mathbf{D}$, where $\mathbf{D}$ is the matrix representation of the vector $\mathbf{d} = (d_1, \ldots, d_N)$, as $\mathbf{c_D} = \mathbf{c_A}^y \mathbf{c_B} = \mathsf{Com}(\mathbf{D}, y\mathbf{r} + \mathbf{s})$ (this is the commitment to $y\mathbf{a} + \mathbf{b}$). In addition, P computes the commitment to z in the following way: $\mathbf{c_{-z}} = \mathsf{Com}(-z, \ldots, -z, \mathbf{0})$, so it can define $\mathsf{Com}(\mathbf{d} - \mathbf{z}, \mathbf{t}) = \mathbf{c_D c_{-z}}$, where $\mathbf{t} = y\mathbf{r} + \mathbf{s}$.

– P builds two degree N polynomials, one using $y, z, \mathbf{a}$ and $\mathbf{b}$ and the second one with $y, z, \{1, \ldots, N\}$ and $\{x^i\}_{i=1}^N$, which are identical in z with the only difference that their roots are permuted:

$$\prod_{i=1}^{N}(d_i - z) = \prod_{i=1}^{N}(yi + x^i - z)$$

– The prover demonstrates using the product argument that $\mathsf{Com}(\mathbf{d} - \mathbf{z}, \mathbf{t}) = \mathbf{c_D}\mathbf{c_{-z}}$ and that both polynomials are equal, i.e., that a set of committed values has a particular product. Using the Schwartz-Zippel lemma (Lemma 3.2.1) the verifier can deduce that they are equal since the prover has negligible probability over the choice of z to generate a convincing proof unless $d_i = y\pi(i) + x^{\pi(i)}$ for $i \in \{1, \ldots, N\}$. In addition, this will not be true unless $\mathbf{c_A}$ is a commitment to $\pi(1), \ldots, \pi(N)$ and $\mathbf{c_B}$ to $x^{\pi(1)}, \ldots, x^{\pi(N)}$.

- Finally the prover demonstrates using the multi-exponentiation argument that he knows the re-encryption parameters such that

$$\prod_{i=1}^{N} C_i^{x^i} = \mathsf{Enc}_{pk}(1, \rho) \prod_{i=1}^{N} (C_i'')^{x^{\pi(i)}}$$

where $\rho = -\langle \boldsymbol{\rho}, \mathbf{b} \rangle$ and $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_N)$. Given the homomorphic properties of the encryption scheme, the verifier can deduce from the above equation

$$\prod_{i=1}^{N} M_i^{x^i} = \prod_{i=1}^{N} (M_i'')^{x^{\pi(i)}}$$

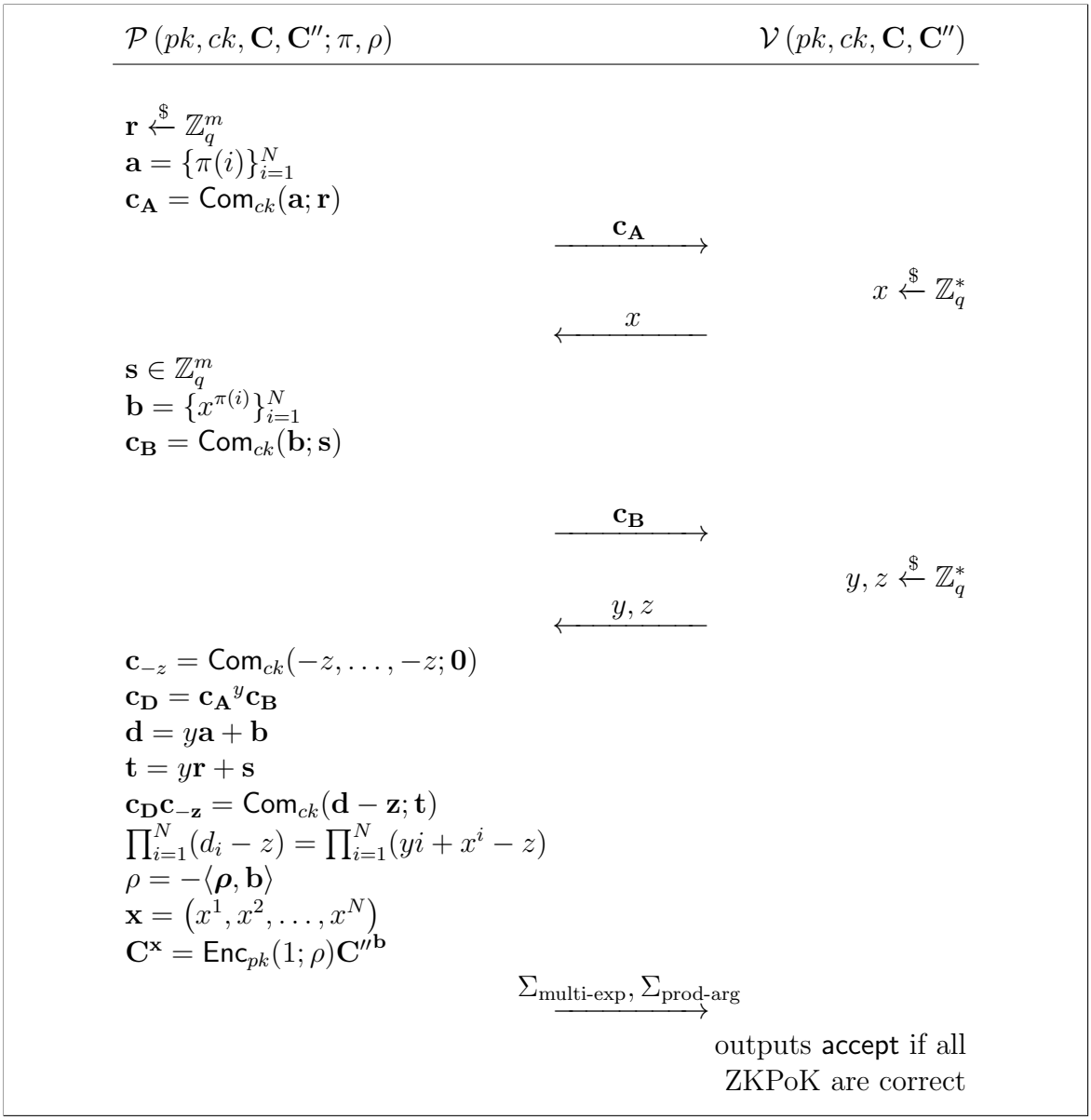
and taking discrete logarithms we have

$$\sum_{i=1}^{N} \log(M_i) x^i = \sum_{i=1}^{N} \log(M''_{\pi^{-1}(i)}) x^i.$$

As it is argued in [22], there is negligible probability over the choice of x that this equality holds true unless $M_1'' = M_{\pi(1)}, \ldots, M_N'' = M_{\pi(N)}$.

- The verifier accepts if the product and the multi-exponentiation arguments are both valid.

Protocol 4.1: Shuffle argument



4.3 Building blocks

In this section we give an overview of the building blocks used for constructing the proof of shuffle. Two of them, the encryption scheme [107] and the commitment scheme [29], have been already explained in Section 2.4.5. We recall here how a lattice-based ciphertext and commitment look like:

$$\text{Ciphertext:} \quad (u, v) = (a_{\mathsf{E}} \cdot r_{\mathsf{E}} + e_{\mathsf{E},u}, b_{\mathsf{E}} \cdot r_{\mathsf{E}} + e_{\mathsf{E},v} + \lfloor \frac{q}{2} \rceil z)$$

$$\text{Commitment:} \quad \mathbf{c} = \mathbf{a}_{\mathsf{C}} m + \mathbf{b}_{\mathsf{C}} r_{\mathsf{C}} + \mathbf{e}_{\mathsf{C}}$$

In addition to these two schemes we will need ZKPoKs to demonstrate that

the re-encryption parameters used during the mixing process are small (see Section 4.3.1) and for proving knowledge of the opening of a commitment and proving linear and multiplicative relations among committed messages (see Section 4.3.2).

4.3.1 Proving knowledge of small elements

When working with lattices there is a common hard problem which consists on recovering a vector $\mathbf{x}$ with small coefficients such that $\mathbf{Ax} = y$ with $\|\mathbf{x}\|_\infty \leq \beta$. This problem is known as Inhomogeneous Short Integer Solution (ISIS) problem [72] and it is described in Definition 42.

Definition 42 (Inhomogeneous Short Integer Solution Problem ($ISIS^p_{n,m,q,\beta}$))**.** Given an integer q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a syndrome $\mathbf{u} \in \mathbb{Z}_q^n$ and a real β, find an integer vector $\mathbf{e} \in \mathbb{Z}^m$ such that $\mathbf{Ae} = \mathbf{u} \mod q$ and $\|\mathbf{e}\|_p \leq \beta$ (in the l_p norm).

In some lattice-based constructions we would like to build a zero-knowledge proof in which the prover P convinces the verifies V that it knows $\mathbf{x}$. For constructing such ZKP there are two main techniques: Stern-like protocols [143] or *Fiat-Shamir with aborts* [104, 105, 106].

The main idea of Stern-like protocols is that the prover P generates three commitments (c_1, c_2, c_3) and sends them to the verifier V. Then, V sends a random challenge b from $\{1, 2, 3\}$ to P who, according to the challenge received reveals two of the three commitments, e.g., if $b = 1$ the prover reveals the opening of c_2 and c_3. The security of these protocols depends on the hardness of the Syndrome Decoding Problem (SDP), and due to its similarity with the ISIS problem the protocol was adapted by Kawachi *et al.* [97] in 2008 to the lattice setting. Note that one round of this protocol has soundness error 2/3 and therefore it should be repeated several times in order to achieve negligible soundness error.

The protocol presented in [97] proposed a ZKPoK for a restricted version of the $ISIS^\infty$ problem in which $\mathbf{x}$ is restricted to $\mathbf{x} \in \{0, 1\}^m$. Nevertheless, since this is not sufficient for some applications, in 2012 Ling *et al.* [100] improved the system by designing a ZKPoK whose security is based on the general ISIS problem and that achieves an optimal gap, i.e., the norm bound for the witness and the bound the prover is able to prove, are identical. We describe below this proof:

1. The prover P and the verifier V extends the matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ by appending $2m$ zero columns: $\mathbf{A}' = (\mathbf{A}|\mathbf{0}) \in \mathbb{Z}_q^{n\times 3m}$

2. The prover P represents each coordinate x_i of the vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ using its bit decomposition, i.e., $x_i = b_{i,0} \cdot 2^0 + b_{i,1} \cdot 2^1 + \ldots + b_{i,k-1} \cdot 2^{k-1}$, where $b_{i,j} \in \{-1, 0, 1\}$ for all $j = 0, \ldots, k-1$. Note that for each index j we can define a vector of the form $\tilde{\mathbf{u}}_j = (b_{1,j}, b_{2,j}, \ldots, b_{m,j}) \in \{-1, 0, 1\}^m$ and from:

$$\begin{aligned} x_1 &= b_{1,0} \cdot 2^0 + b_{1,1} \cdot 2^1 + \ldots + b_{1,k-1} \cdot 2^{k-1} \\ x_2 &= b_{2,0} \cdot 2^0 + b_{2,1} \cdot 2^1 + \ldots + b_{2,k-1} \cdot 2^{k-1} \\ &\vdots \\ x_m &= b_{m,0} \cdot 2^0 + b_{m,1} \cdot 2^1 + \ldots + b_{m,k-1} \cdot 2^{k-1} \end{aligned}$$

we can represent $\mathbf{x}$ as $\mathbf{x} = \tilde{\mathbf{u}}_0 \cdot 2^0 + \ldots + \tilde{\mathbf{u}}_{k-1} \cdot 2^{k-1} = \sum_{j=0}^{k-1} \tilde{\mathbf{u}}_j \cdot 2^j$.

3. P extends each vector $\tilde{\mathbf{u}}_j$ so all of them have the same number of coordinates $-1, 0$ and 1. In order to do so, if each vector $\tilde{\mathbf{u}}_j$ has $\lambda_j^{(-1)}$, $\lambda_j^{(0)}$, $\lambda_j^{(1)}$ coordinates $-1, 0, 1$ respectively, choose a random vector $\mathbf{t}_j \in \{-1, 0, 1\}$ that has $(m - \lambda_j^{(-1)})$ coordinates -1, $(m - \lambda_j^{(0)})$ coordinates 0 and $(m - \lambda_j^{(1)})$ coordinates 1. Append $\mathbf{t}_j$ to $\tilde{\mathbf{u}}_j$ and obtain $\mathbf{u}_j = (\tilde{\mathbf{u}}_j \| \mathbf{t}_j)$. Recall that during the first step the matrix $\mathbf{A}$ has been extended so its last $2m$ columns are 0. Due to this $\mathbf{A}'(\sum_{j=0}^{k-1} 2^j \cdot \mathbf{u}_j) = \mathbf{y} \mod q \Leftrightarrow \mathbf{A}\mathbf{x} = \mathbf{y} \mod q$.

After these preparation steps, P and V interact as shown in Protocol 4.2.

Note that if we write the RLWE encryption of a message in matrix form we observe that proving knowledge of the small random elements $r'_{\mathsf{E}}, e'_{\mathsf{E},u}$ and $e'_{\mathsf{E},v}$ is equivalent to finding a solution of the ISIS problem:

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 \\ \vdots \\ v_n \end{pmatrix} = \left\lfloor \frac{q}{2} \right\rceil \begin{pmatrix} 0 \\ \vdots \\ 0 \\ z_1 \\ \vdots \\ z_n \end{pmatrix} + \begin{pmatrix} a_1 & -a_n & \cdots & -a_2 \\ a_2 & a_1 & \cdots & -a_3 \\ \ldots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & \cdots & a_1 \\ b_1 & -b_n & \cdots & -b_2 \\ b_2 & b_1 & \cdots & -b_3 \\ \ldots & \vdots & \ddots & \vdots \\ b_n & b_{n-1} & \cdots & b_1 \end{pmatrix} \begin{pmatrix} r'_{\mathsf{E},1} \\ r'_{\mathsf{E},2} \\ \vdots \\ r'_{\mathsf{E},n} \end{pmatrix} + \begin{pmatrix} e'_{\mathsf{E},u,1} \\ \vdots \\ e'_{\mathsf{E},u,n} \\ e'_{\mathsf{E},v,1} \\ \vdots \\ e'_{\mathsf{E},v,n} \end{pmatrix}$$

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 - \lfloor \frac{q}{2} \rceil z_1 \\ \vdots \\ v_n - \lfloor \frac{q}{2} \rceil z_n \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{Id}_n & \mathbf{0}_n \\ \mathbf{B} & \mathbf{0}_n & \mathbf{Id}_n \end{pmatrix} \begin{pmatrix} r'_{\mathsf{E},1} \\ \vdots \\ r'_{\mathsf{E},n} \\ e'_{\mathsf{E},u,1} \\ \vdots \\ e'_{\mathsf{E},u,n} \\ e'_{\mathsf{E},v,1} \\ \vdots \\ e'_{\mathsf{E},v,n} \end{pmatrix}$$

We can re-write the previous equation as $\mathbf{y} = \tilde{\mathbf{A}}\mathbf{x}$ and we can see that proving knowledge of $r'_{\mathsf{E}}, e'_{\mathsf{E},u}$ and $e'_{\mathsf{E},v}$ is equivalent to proving knowledge of a vector $\mathbf{x}$ with $\|\mathbf{x}\| \leq \beta$.

Protocol 4.2: Extended Stern protocol

$\mathcal{P}\left(\mathbf{y}, \mathbf{A}'; \mathbf{x}, \{\mathbf{u}_j\}_{j=0}^{k-1}\right)$	$\mathcal{V}\left(\mathbf{y}, \mathbf{A}'\right)$
$\mathbf{r}_0, \ldots, \mathbf{r}_{k-1} \xleftarrow{\$} \mathbb{Z}_1^{3m}$	
$\pi_0, \ldots, \pi_{k-1} \xleftarrow{\$} \mathcal{S}_{3m}$	
$\mathbf{c}_1 = \mathsf{Com}(\pi_0, \ldots, \pi_{k-1}, \mathbf{A}'(\sum_{j=0}^{k-1} 2^j \cdot \mathbf{r}_j) \mod q)$	
$\mathbf{c}_2 = \mathsf{Com}(\pi_0(\mathbf{r}_0), \ldots, \pi_{k-1}(\mathbf{r}_{k-1}))$	
$\mathbf{c}_3 = \mathsf{Com}(\pi_0(\mathbf{u}_0 + \mathbf{r}_0), \ldots, \pi_{k-1}(\mathbf{u}_{k-1} + \mathbf{r}_{k-1}))$	
$\xrightarrow{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3}$	
	$c \xleftarrow{\$} \{1, 2, 3\}$
$\xleftarrow{c}$	
If c=1 :	
$\forall j, \mathbf{v}_j = \pi_j(\mathbf{u}_j), \mathbf{w}_j = \pi_j(\mathbf{r}_j)$	
RSP:=$(\mathbf{v}_0, \ldots, \mathbf{v}_{k-1}, \mathbf{w}_0, \ldots, \mathbf{w}_{k-1})$	
If c=2 :	
$\forall j, \phi_j = \pi_j, \mathbf{z}_j = \mathbf{u}_j + \mathbf{r}_j$	
RSP:=$(\phi_0, \ldots, \phi_{k-1}, \mathbf{z}_0, \ldots, \mathbf{z}_{k-1})$	
If c=3 :	
$\forall j, \psi_j = \pi_j, \mathbf{s}_j = \mathbf{r}_j$	
RSP:=$(\psi_0, \ldots, \psi_{k-1}, \mathbf{s}_0, \ldots, \mathbf{s}_{k-1})$	
$\xrightarrow{\text{RSP}}$	
	If c=1 check that:
	$\mathbf{v}_j \in B_{3m} \forall j \in \{0, \ldots, k-1\}$
	$\mathbf{c}_2 = \mathsf{Com}(\mathbf{w}_0, \ldots, \mathbf{w}_{k-1})$
	$\mathbf{c}_3 = \mathsf{Com}(\mathbf{v}_0 + \mathbf{w}_0, \ldots, \mathbf{v}_{k-1} + \mathbf{w}_{k-1})$
	If c=2 check that:
	$\mathbf{c}_1 = \mathsf{Com}(\phi_0, \ldots, \phi_{k-1}, \mathbf{A}'(\sum_{j=0}^{k-1} 2^j \cdot \mathbf{z}_j) - \mathbf{y} \mod q)$
	$\mathbf{c}_3 = \mathsf{Com}(\phi_0(\mathbf{z}_0), \ldots, \phi_{k-1}(\mathbf{z}_{k-1}))$
	If c=3 check that:
	$\mathbf{c}_1 = \mathsf{Com}(\psi_0, \ldots, \psi_{k-1}, \mathbf{A}'(\sum_{j=0}^{k-1} 2^j \cdot \mathbf{s}_j) \mod q)$
	$\mathbf{c}_2 = \mathsf{Com}(\psi_0(\mathbf{s}_0), \ldots, \psi_{k-1}(\mathbf{s}_{k-1}))$
	outputs accept if all conditions hold

As mentioned at the beginning of this section, the second technique used for

proving knowledge of the vector $\mathbf{x}$ is known as *Fiat-Shamir with aborts*. The intuition behind it is that during the interactive protocol between P and V, there is some constant fraction of time that the prover cannot respond to the challenge sent by the verifier and must abort the protocol, i.e., perform a rejection sampling step. As a consequence, the commit and challenge steps must be repeated several times. The basic protocol presented in [104] is as follows: the prover knows a short vector $\mathbf{x}$ such that $f(\mathbf{x}) = y$ (where the function f is defined by the matrix $\mathbf{A}$). In a first step P chooses a mask $\mathbf{g}$ and sends $h = f(\mathbf{g})$ to the verifier, who then selects the challenge c from the set $\{0, 1\}$ and sends it to P. The prover computes $\mathbf{z} = c \cdot \mathbf{x} + \mathbf{g}$ and aborts with a probability that depends on the value computed. This is done because if the prover always responds to the verifier, the secret key $\mathbf{x}$ could be linked from $\mathbf{z}$. Note that in number-theoretic schemes, such as the Schnorr protocol (2.1) presented in Section 2.2, the value of the mask is chosen uniformly at random and since all the operations are done in a finite ring, the mask perfectly hides the secret. Nevertheless, when working with lattices we need this mask to be *small* which implies that when added to the secret, it is sometimes leaked. Indeed, the mask $\mathbf{g}$ is chosen from a Gaussian distribution so $\mathbf{z}$ it is also a discrete Gaussian distribution centered at $c \cdot \mathbf{x}$. In order for the distribution $\mathbf{z}$ to be statistically indistinguishable from a discrete Gaussian distribution centered at the origin, the protocol uses rejection sampling and aborts with probability $\mathsf{exp}\left(\frac{-2\langle \mathbf{z}, c\mathbf{x}\rangle + \|c\mathbf{x}\|^2}{2\sigma^2}\right)$ (see [106], Lemma 4.5), where σ is the standard deviation.

If P does not abort, it sends $\mathbf{z} = c \cdot \mathbf{x} + \mathbf{g}$ to V, who finally checks if $f(\mathbf{z}) = c \cdot y + h$ and $\|\mathbf{z}\|$ is small. The domain of the challenge c is expanded from $\{0, 1\}$ to a set of sufficiently small elements, in a follow-up work by Lyubashevsky [105].

There are two main problems related to this kind of protocols which are the *overhead* and the *soundness slack*. The former refers to the number of times the protocol must be repeated and the latter to the ratio between the coefficients in the secret and those that can be extracted from the proof. As explained in [19, 51] what we want is the smallest overhead and soundness slack, but when proving knowledge of a single pre-image we do not know how to reduce both values. Using Lyubachevsky's rejection sampling technique we can reduce the soundness slack but not the overhead.

Nevertheless, if instead of proving knowledge of one pre-image we do it for a set of them, it is possible to reduce the amortized overhead. This is possible using **amortized proofs**. The prover knows $\{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ such that $f(\mathbf{x_1}) = y_1, \ldots, f(\mathbf{x_n}) = y_n$ and proves knowledge of a set of vectors $\{\mathbf{x'_i}\}_{i=1}^{n}$ with small coefficients such that $f(\mathbf{x'}_i) = y_i$. More formally, the relation to be proven in zero-knowledge is the following [19]:

$$R_{KSP} = \left\{(v, w) | v = (y_1, \ldots, y_n) \wedge w = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \wedge [y_i = f(\mathbf{x}_i) \wedge \|\mathbf{x}_i\| \leq \beta]_{i \in [n]}\right\}$$

We are mainly interested on the proposal by del Pino and Lyubashevsky [55] which is based on [51] and reduces the necessary number of equations to be proven simultaneously at the expense of a higher running time. In this protocol the amortized proof is constructed by combining **imperfect proofs** [19] and the rejection sampling mechanism explained before. Informally, in an imperfect proof of knowledge the prover only demonstrates that it knows almost all pre-images. We give below an overview of this protocol:

1. During a first step the prover chooses several masking parameters $\mathbf{g}_j$ and commits to them: $h_j = f(\mathbf{g}_j)$

2. The verifier sends a challenge string and asks the prover to reveal a fraction of the masking parameters. V checks if the values are small.

3. The prover then computes $\mathbf{x}_i + \mathbf{g}_j$ for every $1 \leq i \leq n$ and executes a rejection sampling step.

4. The prover creates several additive combinations of y_i. The pre-image of each of these combinations is the corresponding additive combination of the $\mathbf{x}_i$.

5. The prover runs an imperfect proof on each of these combinations.

For building our mixing protocol we are interested not only on proving knowledge of one witness $\mathbf{x}$ but on a set of them, i.e., each mix-node will want to prove knowledge of all re-encryption parameters used during the mixing of N ciphertexts. In order to prove this we are going to use the amortized strategy proposed in [55] which we have previously sketched.

4.3.2 Efficient zero-knowledge proofs for commitments from RLWE

For building our proof of a shuffle we are going to use the commitment schemes proposed by Benhamouda *et al.* in [29] which is perfectly binding and computationally hiding as long as the RLWE problem is hard (see Section 2.4.5 for more details about the commitment scheme). In addition, in the same paper the authors propose $\Sigma-$protocols for proving knowledge of the message contained in a commitment (Protocol 4.3) but also to prove additive (Protocol 4.4) and multiplicative relations (Protocol 4.5) among the committed messages. These protocols are those we are going to present in this section but before doing it we want to give an overview of some of the parameters used in [29]: the prime q is $q \equiv 3 \mod 8$ and $q \geq n^\gamma$, where n is the degree of polynomial, a power of 2 and γ is the integer parameter controlling the size of the modulus. σ_e is the standard deviation of the error in the commitment scheme, σ_η is the standard deviation of the randomness used for hiding $\mathbf{e}_\mathsf{C}$ in the protocols and κ is an integer, where $1/|\mathcal{C}| = 1/\binom{n/2}{\kappa}$ bounds the knowledge error of the proofs and $\mathcal{C}$ is the domain of challenges.

Note that Protocol 4.3 follows the usual structure of a $\Sigma-$protocol. In a first step the prover chooses as many masking parameters as witnesses and generates the value $\mathbf{t}$ which follows the same structure as the commitment. Then, it commits to $\mathbf{t}$ using an auxiliary string commitment scheme aCom and sends the auxiliary commitment to the verifier. V chooses the challenge d from $\mathcal{C} = \{d \in \{0,1\}^n : \|d\|_1 \leq \kappa \wedge \deg d \leq n/2\}$ and sends it to the prover. P builds $\mathbf{t} + d\mathbf{c} = \mathbf{a}_\mathsf{C}(\mu + dm) + \mathbf{b}_\mathsf{C}(\rho + dr_\mathsf{C}) + \boldsymbol{\eta} + d\mathbf{e}_\mathsf{C}$ and reveals its opening $(s_m, s_r, s_{\mathbf{e}_\mathsf{C}})$ to the verifier, who checks that the information received is correct. As explained in previous section, since $\boldsymbol{\eta}$ is chosen from the error distribution, $s_{\mathbf{e}_\mathsf{C}}$ reveals some information about $\mathbf{e}_\mathsf{C}$ and in order to correct this the protocol aborts with a probability of $\mathsf{exp}\left(\frac{-2\langle s_{\mathbf{e}_\mathsf{C}}, d\mathbf{e}_\mathsf{C}\rangle + \|d\mathbf{e}_\mathsf{C}\|^2}{2\sigma_\eta^2}\right)$. From Lemma 4.5 in

[106] it follows that the probability that P does not abort is exponentially close to $1/M$ where $M \in \mathcal{O}(\exp(\frac{\|d\mathbf{e}_\mathsf{C}\|}{\sigma_\eta}))$, so on average M iterations of the protocol until there is not any abort are required.

Protocol 4.3: Simple pre-image proof

$\mathcal{P}(\mathbf{c}; m, r_\mathsf{C}, \mathbf{e}_\mathsf{C})$		$\mathcal{V}(\mathbf{c})$
$\mu, \rho \xleftarrow{\$} \mathbb{Z}_q[x]/\langle x^n+1\rangle$ $\boldsymbol{\eta} \xleftarrow{\$} \chi^k_{\sigma_\eta}$ $\mathbf{t} = \mathbf{a}_\mathsf{C}\mu + \mathbf{b}_\mathsf{C}\rho + \boldsymbol{\eta}$ $(c_\mathsf{aux}, d_\mathsf{aux}) = \mathsf{aCom}(\mathbf{t})$		
	$\xrightarrow{c_\mathsf{aux}}$	
		$d \xleftarrow{\$} \mathcal{C}$
	$\xleftarrow{d}$	
$s_m = \mu + dm$ $s_r = \rho + dr_\mathsf{C}$ $s_{\mathbf{e}_\mathsf{C}} = \boldsymbol{\eta} + d\mathbf{e}_\mathsf{C}$ abort with probability $\exp\left(\frac{-2\langle s_{\mathbf{e}_\mathsf{C}}, d\mathbf{e}_\mathsf{C}\rangle + \|d\mathbf{e}_\mathsf{C}\|^2}{2\sigma_\eta^2}\right)$		
	$\xrightarrow{d_\mathsf{aux}, \mathbf{t}, s_m, s_r, s_{\mathbf{e}_\mathsf{C}}}$	
		$\mathsf{aVer}(c_\mathsf{aux}, d_\mathsf{aux}, \mathbf{t}) \stackrel{?}{=} \mathsf{accept}$ $\mathbf{t} + d\mathbf{c} \stackrel{?}{=} \mathbf{a}_\mathsf{C} s_m + \mathbf{b}_\mathsf{C} s_r + s_\mathbf{e}$ $\|s_\mathbf{e}\|_\infty \stackrel{?}{\leq} \lfloor n^{4/3}/4 \rfloor$

Running different instances of Protocol 4.3 in parallel it is possible to prove linear relations of different messages contained in their corresponding commitments (see Protocol 4.4). The idea is that P wants to prove knowledge of m_1, m_2 and m_3 contained in $\mathbf{c}_1, \mathbf{c}_2$ and $\mathbf{c}_3$ where m_i satisfies a linear relation of the form $m_3 = x_1 m_1 + x_2 m_2$ for $x_i \in \mathbb{Z}_q[x]/\langle x^n+1\rangle$.

Following a similar approach that in Protocol 4.4 a prover can prove knowledge of $m_i, r_{\mathsf{C},i}, \mathbf{e}_{\mathsf{C},i}$ (for $i = 1, 2, 3$) such that $\mathbf{c}_i = \mathbf{a}_\mathsf{C} m_i + \mathbf{b}_\mathsf{C} r_{\mathsf{C},i} + \mathbf{e}_{\mathsf{C},i}$ and additionally $m_3 = m_1 \cdot m_2$ (see Protocol 4.5).

As Benhamouda *et al.* explain in their article, if the auxiliary commitment scheme if perfectly binding, Protocols 4.3, 4.4 and 4.5 are honest-verifier zero-knowledge proof of knowledge with knowledge error $1/\binom{n/2}{\kappa}$, $1/\binom{n/2}{\kappa}$ and $2/\binom{n/2}{\kappa}$

correspondingly, for the following relations:

$$\begin{aligned}
\mathcal{R}'_{LWE} =& \big\{((\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}}, \mathbf{c}), (m, r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}}, f)) : \mathsf{ComVer}(\mathbf{c}, m, r_{\mathsf{C}}, \mathbf{e}_{\mathsf{C}}, f) = \mathsf{accept}\big\} \\
\mathcal{R}'_{LLWE} =& \big\{((\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}}, x_1, x_2, \{\mathbf{c}_i\}_{i=1}^3), (\{m_i\}_{i=1}^3, \{r_{\mathsf{C},i}\}_{i=1}^3, \{\mathbf{e}_{\mathsf{C},i}\}_{i=1}^3, \{f_i\}_{i=1}^3)) : \\
& \bigwedge_{i=1}^{3} \mathsf{ComVer}(\mathbf{c}_i, m_i, r_{\mathsf{C},i}, \mathbf{e}_{\mathsf{C},i}, f_i) = \mathsf{accept} \wedge m_3 = x_1 m_1 + x_2 m_2\big\} \\
\mathcal{R}'_{MLWE} =& \big\{((\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}}, \{\mathbf{c}_i\}_{i=1}^3), (\{m_i\}_{i=1}^3, \{r_{\mathsf{C},i}\}_{i=1}^3, \{\mathbf{e}_{\mathsf{C},i}\}_{i=1}^3, \{f_i\}_{i=1}^3)) : \\
& \bigwedge_{i=1}^{3} \mathsf{ComVer}(\mathbf{c}_i, m_i, r_{\mathsf{C},i}, \mathbf{e}_{\mathsf{C},i}, f_i) = \mathsf{accept} \wedge m_3 = m_1 m_2\big\}
\end{aligned}$$

Protocol 4.4: Linear relation proof

$\mathcal{P}(\mathbf{c}_i; m_i, r_{\mathsf{C},i}, \mathbf{e}_{\mathsf{C},i})$		$\mathcal{V}(\mathbf{c}_i)$
$\mu_1, \mu_2, \rho_1, \rho_2, \rho_3 \xleftarrow{\$} \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ $\mu_3 = x_1\mu_1 + x_2\mu_2$ $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_3 \xleftarrow{\$} \chi_{\sigma_\eta}^k$ $\mathbf{t}_i = \mathbf{a}_{\mathsf{C}}\mu_i + \mathbf{b}_{\mathsf{C}}\rho_i + \boldsymbol{\eta}_i$ for $i = 1, 2, 3$ $(c_{\mathsf{aux}}, d_{\mathsf{aux}}) = \mathsf{aCom}(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)$	$\xrightarrow{c_{\mathsf{aux}}}$	
		$d \xleftarrow{\$} \mathcal{C}$
	$\xleftarrow{d}$	
$s_{m_i} = \mu_i + d m_i$ for $i = 1, 2, 3$ $s_{r_i} = \rho_i + d r_{\mathsf{C},i}$ for $i = 1, 2, 3$ $s_{\mathbf{e}_{\mathsf{C},i}} = \boldsymbol{\eta}_i + d\mathbf{e}_{\mathsf{C},i}$ for $i = 1, 2, 3$ abort with probability $\exp\left(\frac{-2\langle s_{\mathbf{e}_{\mathsf{C},i}}, d\mathbf{e}_{\mathsf{C},i}\rangle + \|d\mathbf{e}_{\mathsf{C},i}\|^2}{2\sigma_{\eta_i}^2}\right)$	$\xrightarrow{d_{\mathsf{aux}}, \mathbf{t}_i, s_{m_i}, s_{r_i}, s_{\mathbf{e}_{\mathsf{C},i}}}$	
		$\mathsf{aVer}(c_{\mathsf{aux}}, d_{\mathsf{aux}}, (\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3)) \stackrel{?}{=} \mathsf{accept}$
		$s_{m_3} = x_1 s_{m_1} + x_2 s_{m_2}$
		$\mathbf{t}_i + d\mathbf{c}_i \stackrel{?}{=} \mathbf{a}_{\mathsf{C}} s_{m_i} + \mathbf{b}_{\mathsf{C}} s_{r_i} + s_{\mathbf{e}_{\mathsf{C},i}}$ for $i = 1, 2, 3$
		$\|s_{\mathbf{e}_{\mathsf{C},i}}\|_\infty \stackrel{?}{\leq} \lfloor n^{4/3}/4 \rfloor$ for $i = 1, 2, 3$

Protocol 4.5: Multiplicative relation proof

$\mathcal{P}\left(\mathbf{c}_i; m_i, r_{\mathsf{C},i}, \mathbf{e}_{\mathsf{C},i}\right)$	$\mathcal{V}\left(\mathbf{c}_i\right)$
$\mu_1, \mu_2, \mu_3, \rho_1, \rho_2, \rho_3 \xleftarrow{\$} \mathbb{Z}_q[x]/\langle x^n+1\rangle$	
$\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_3 \xleftarrow{\$} \chi^k_{\sigma_\eta}$	
$\mathbf{t}_i = \mathbf{a}_{\mathsf{C}}\mu_i + \mathbf{b}_{\mathsf{C}}\rho_i + \boldsymbol{\eta}_i$ for $i = 1,2,3$	
$m_+ = \mu_1 m_2 + \mu_2 m_1$	
$m_\times = \mu_1 \mu_2$	
$r_+, r_\times \xleftarrow{\$} \mathbb{Z}_q[x]/\langle x^n+1\rangle$	
$\mathbf{e}_+, \mathbf{e}_\times \xleftarrow{\$} \chi^k_{\sigma_e}$	
$\mathbf{c}_+ = \mathbf{a}_{\mathsf{C}} m_+ + \mathbf{b}_{\mathsf{C}} r_+ + \mathbf{e}_+$	
$\mathbf{c}_\times = \mathbf{a}_{\mathsf{C}} m_\times + \mathbf{b}_{\mathsf{C}} r_\times + \mathbf{e}_\times$	
$\mu_+, \mu_\times, \rho_+, \rho_\times \xleftarrow{\$} \mathbb{Z}_q[x]/\langle x^n+1\rangle$	
$\boldsymbol{\eta}_+, \boldsymbol{\eta}_\times \xleftarrow{\$} \chi^k_{\sigma_\eta}$	
$\mathbf{t}_+ = \mathbf{a}_{\mathsf{C}}\mu_+ + \mathbf{b}_{\mathsf{C}}\rho_+ + \boldsymbol{\eta}_+$	
$\mathbf{t}_\times = \mathbf{a}_{\mathsf{C}}\mu_\times + \mathbf{b}_{\mathsf{C}}\rho_\times + \boldsymbol{\eta}_\times$	
$\tilde{\rho} \xleftarrow{\$} \mathbb{Z}_q[x]/\langle x^n+1\rangle$	
$\tilde{\boldsymbol{\eta}} \xleftarrow{\$} \chi^k_{\sigma_\eta}$	
$\tilde{\mathbf{t}} = \mathbf{b}_{\mathsf{C}}\tilde{\rho} + \tilde{\boldsymbol{\eta}}$	
$(c_{\mathsf{aux}}, d_{\mathsf{aux}}) = \mathsf{aCom}(\mathbf{t}_+, \mathbf{t}_\times, \mathbf{t}_i, \tilde{\mathbf{t}}, \mathbf{c}_+, \mathbf{c}_\times)$	
$\xrightarrow{\quad c_{\mathsf{aux}} \quad}$	
	$d \xleftarrow{\$} \mathcal{C}$
$\xleftarrow{\quad d \quad}$	
$s_{m_i} = \mu_i + d m_i$ for $i = 1,2,3,+,\times$	
$s_{r_i} = \rho_i + d r_i$ for $i = 1,2,3,+,\times$	
$s_{\mathbf{e}_{\mathsf{C},i}} = \boldsymbol{\eta}_i + d\mathbf{e}_{\mathsf{C},i}$ for $i = 1,2,3,+,\times$	
$s_{\tilde{r}} = \tilde{\rho} + d\tilde{r}$	
$\tilde{\mathbf{e}} = -d^2\mathbf{e}_3 - \mathbf{e}_+ - d\mathbf{e}_\times$	
$\tilde{r} = -d^2 r_3 - r_+ - d r_\times$	
$s_{\tilde{\mathbf{e}}} = \tilde{\eta} + d\tilde{\mathbf{e}}$	
abort-checks for $s_{\tilde{\mathbf{e}}}, s_{\mathbf{e}_{\mathsf{C},j}}$	
	$c_{\mathsf{aux}}, \xrightarrow[s_{m_i}, s_{r_i}, s_{\mathbf{e}_{\mathsf{C},i}}, s_{\tilde{r}}, s_{\tilde{\mathbf{e}}}]{} \mathbf{t}_+, \mathbf{t}_\times, \mathbf{t}_i, \tilde{\mathbf{t}}, \mathbf{c}_+, \mathbf{c}_\times$
	$\mathsf{aVer}(c_{\mathsf{aux}}, d_{\mathsf{aux}}, (\mathbf{t}_+, \mathbf{t}_\times, \mathbf{t}_i, \tilde{\mathbf{t}}, \mathbf{c}_+, \mathbf{c}_\times)) \stackrel{?}{=} \mathsf{accept}$
	$\mathbf{t}_i + d\mathbf{c}_i \stackrel{?}{=} \mathbf{a}_{\mathsf{C},i} s_{m_i} + \mathbf{b}_{\mathsf{C},i} s_{r_i} + s_{\mathbf{e}_{\mathsf{C},i}}$ for $i = 1,2,3,+,\times$
	$\lVert s_{\mathbf{e}_{\mathsf{C},i}} \rVert_\infty \stackrel{?}{\leq} \lfloor n^{4/3}/4 \rfloor$ for $i = 1,2,3,+,\times$
	$\tilde{\mathbf{c}} = \mathbf{a}_{\mathsf{C}} s_{m_1} s_{m_2} - d^2\mathbf{c}_3 - \mathbf{c}_\times - d\mathbf{c}_+$
	$\tilde{\mathbf{t}} + d\tilde{\mathbf{c}} \stackrel{?}{=} \mathbf{b}_{\mathsf{C}} s_{\tilde{r}} + s_{\tilde{\mathbf{e}}}$
	$\lVert s_{\tilde{\mathbf{e}}} \rVert_\infty \stackrel{?}{\leq} \lfloor n^{4/3}/4 \rfloor$

4.4 Protocol overview

Having in mind Bayer and Groth's protocol (4.2) and all the building blocks (4.3), in this section we present an overview of the lattice-based proof of a shuffle which is explained in detail in Section 4.5.

Given a permutation π and a set of re-encryption parameters $\left\{r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right\}$ for each one of the messages, the shuffling of N RLWE encryptions is defined as

$$\begin{pmatrix} u'^{(1)}, & v'^{(1)} \\ \vdots & \vdots \\ u'^{(N)}, & v'^{(N)} \end{pmatrix} = \begin{pmatrix} u^{\pi(1)}, & v^{\pi(1)} \\ \vdots & \vdots \\ u^{\pi(N)}, & v^{\pi(N)} \end{pmatrix} + \begin{pmatrix} r'^{(1)}_{\mathsf{E}} \\ \vdots \\ r'^{(N)}_{\mathsf{E}} \end{pmatrix} \begin{pmatrix} a_{\mathsf{E}}, & b_{\mathsf{E}} \end{pmatrix} + \begin{pmatrix} e'^{(1)}_{\mathsf{E},u}, & e'^{(1)}_{\mathsf{E},v} \\ \vdots & \vdots \\ e'^{(N)}_{\mathsf{E},u}, & e'^{(N)}_{\mathsf{E},v} \end{pmatrix} \tag{4.1}$$

We will refer to a specific output in a compact manner as:

$$\left(u'^{(i)}, v'^{(i)}\right) = \mathsf{Re\text{-}enc}\left(\left(u^{\pi(i)}, v^{\pi(i)}\right), r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right)$$

.

A mix-node will perform the shuffling over the input ciphertexts and will generate a proof of a shuffle (see 4.2), to demonstrate that it knows the permutation π and the random elements $r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}$, without revealing any information about them.

$$\text{ZKPoK}\left[\begin{array}{c|c} \pi & \left(u'^{(i)}, v'^{(i)}\right) = \\ \left\{r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right\}_{i=1}^{N} & \mathsf{Re\text{-}enc}\left(\left(u^{\pi(i)}, v^{\pi(i)}\right), r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right) \\ & \left\|r'^{(i)}_{\mathsf{E}}\right\|_\infty, \left\|e'^{(i)}_{\mathsf{E},u}\right\|_\infty, \left\|e'^{(i)}_{\mathsf{E},v}\right\|_\infty \leq \delta \end{array}\right] \tag{4.2}$$

This proof will be published so everybody is convinced that the ciphertexts have been permuted and re-encrypted without modifying the encrypted plaintexts (even if some of the nodes are dishonest and leak the permutation). We summarize now which are the main steps of the protocol.

The first step will be to commit to the encryptions of 0 used to compute the RLWE re-encryptions. Then, each mix-node will demonstrate that the commitments computed in the previous step are indeed commitments to ciphertexts of the form: $(u_0, v_0) = (a_{\mathsf{E}} r'_{\mathsf{E}} + e'_{\mathsf{E},u}, b_{\mathsf{E}} r'_{\mathsf{E}} + e'_{\mathsf{E},v})$, i.e., commitments to the encryption of 0. Additionally, it will also be demonstrated that the polynomials $r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}$ used to compute the re-encryptions have an infinity norm that is bounded by some parameter $\delta \ll q/4$.

As it is explained in [29] for a suitable δ even if this additional restriction on the re-encryption parameters norm is applied, re-encryptions remain pseudorandom, as the two probability distributions are statistically close. This first part of the protocol is explained in detail in Section 4.5.1.

The last part of the protocol (detailed in Section 4.5.2) consists on proving that two sets contain the same elements:

$$\left\{\underbrace{\left(u''^{(i)}, v''^{(i)}\right)}_{\text{mix-node output}} - \underbrace{\left(a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},u}, b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},v}\right)}_{\text{encryptions of 0}}\right\}_{i=1}^{N} = \left\{\underbrace{\left(u^{(i)}, v^{(i)}\right)}_{\text{mix-node input}}\right\}_{i=1}^{N}$$

This is done following the strategy proposed by Bayer and Groth in [22], which consists of building two polynomials, each of them having as roots the elements of each of the sets and then prove that both polynomials are equal. Note that the left-hand side of the equality contains the input ciphertext in a permuted order, since we remove the randomness introduced during the re-encryption operation. So, intuitively, the polynomials constructed from these sets will be equal but with their roots permuted.

Our polynomials will be evaluated and have coefficients in R_q, i.e., we will work in $R_q[A]$ and the variable A takes values on R_q:

$$\sum_{i=1}^{N} A^i u^{(i)} = \sum_{i=1}^{N} A^{\pi(i)}(u''^{(i)} - a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} - u^{(i)})$$
$$\sum_{i=1}^{N} A^i v^{(i)} = \sum_{i=1}^{N} A^{\pi(i)}(v''^{(i)} - b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} - v^{(i)})$$

To convince a verifier that two polynomials are equal the prover evaluates them in a random point chosen by the verifier and uses the generalized version of Schwartz-Zippel lemma (Lemma 4.4.1). Bayer and Groth's shuffle proof uses the Schwartz-Zippel lemma, already presented in Chapter 3, which works in general commutative rings that are not necessarily integral domains. Unlike them, for the proof presented in this chapter we need the generalized version of the lemma since we work with polynomials whose coefficients belong to another ring of polynomials.

Lemma 4.4.1 (Generalized Schwartz-Zippel lemma.)**.** Let $p \in R[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$ over a commutative ring R. Let S be a finite subset of R such that none of the differences between two elements of S is a divisor of 0 and let $r_1, r_2, \ldots, r_n$ be selected at random independently and uniformly from S. Then: $\Pr[p(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|}$.

We will use this lemma to prove that two polynomials, p_1 and p_2, are equal with overwhelming probability if $p_1(r_1, r_2, \ldots, r_n) - p_2(r_1, r_2, \ldots, r_n) = 0$ for $r_1, r_2, \ldots, r_n \xleftarrow{\$} S$.

Now we need to define a suitable subset $S \subseteq \mathbb{Z}_q[x]/(x^n + 1)$ for which the condition holds.

We can guarantee it if all differences of elements in S are invertible. We choose:

$$S = \left\{ p(x) \in \mathbb{Z}_q[x]/(x^n + 1) \;\middle|\; \deg p(x) < n/2 \right\}$$

Observe that the proposed subset S meets the required condition for Lemma 4.4.1, as all differences of two polynomials in S are invertible. This is true as the condition $q \equiv 3 \mod 8$ implies that $x^n + 1$ splits into two irreducible polynomials of degree exactly $n/2$ (Lemma 3 in [142]). Then all polynomials of degree smaller that $n/2$ have an inverse that can be computed using the Chinese Remainder Theorem. The number of elements in S is still exponential in n, so we can use it as a set of challenges.

We define the mixing protocol using the following algorithms:

- $\mathsf{SetupMix}(1^\kappa)$: generate parameters (n, q, σ) and run the following algorithms:
  - $\mathsf{KeyGen}_\mathsf{E}(1^\kappa)$ to obtain the public and the private key of the RLWE encryption scheme: $pk_\mathsf{E} = (a_\mathsf{E}, b_\mathsf{E}) \in R_q \times R_q$ and $s \in R_q$.
  - $\mathsf{KeyGen}_\mathsf{C}(1^\kappa)$ to generate the public commitment key: $pk_\mathsf{C} = \mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C} \xleftarrow{\$} (R_q)^k$.

  Output $\{\{(a_\mathsf{E}, b_\mathsf{E}), s\}, (\mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C})\}$
- $\mathsf{MixVotes}(pk_\mathsf{E}, pk_\mathsf{C}, \{(u^{(i)}, v^{(i)})\}_{i=1}^N)$: taking as input a list of N encrypted messages $\{(u^{(i)}, v^{(i)})\}_{i=1}^N$ compute the shuffling of these RLWE encryptions. Generate commitments and ZKPoK (we denote by ZK_i its corresponding protocols and by Σ_i the proofs they output) as it is explained in Section 4.5 in order to demonstrate the correctness of the process. We can explicitly state the permutation and/or random elements to be used writing

  $$\mathsf{MixVotes}(pk_\mathsf{E}, pk_\mathsf{C}, \{(u^{(i)}, v^{(i)})\}_{i=1}^N; \pi, \{r_\mathsf{E}^{\prime(i)}, e_{\mathsf{E},u}^{\prime(i)}, e_{\mathsf{E},v}^{\prime(i)}\}_{i=1}^N)$$

  Output $\left(\{(u''^{(i)}, v''^{(i)})\}_{i=1}^N, \{\mathbf{c}_{u_0^{(i)}}, \mathbf{c}_{v_0^{(i)}}, \mathbf{c}_{\pi(i)}, \mathbf{c}_{\alpha^{\pi(i)}}\}_{i=1}^N, \Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\right)$.

  We denote $\Sigma_0 = \{\mathbf{c}_{u_0^{(i)}}, \mathbf{c}_{v_0^{(i)}}, \mathbf{c}_{\pi(i)}, \mathbf{c}_{\alpha^{\pi(i)}}\}_{i=1}^N$ to unify the notation of the output of $\mathsf{MixVotes}$.
- $\mathsf{VerifyMix}(pk_\mathsf{E}, pk_\mathsf{C}, \{(u^{(i)}, v^{(i)})\}_{i=1}^N, \{(u''^{(i)}, v''^{(i)})\}_{i=1}^N, \{\Sigma_l\}_{l=0}^4)$: given an input and an output of the mixing process and the ZKPoK generated, this algorithm outputs 1 if the proofs are valid and 0 otherwise.

4.5 Lattice-based proof of a shuffle

After the overview given in Section 4.4, in this section we describe in detail the proof of a shuffle (see Protocol 4.6) and explain how the building blocks presented in Section 4.3 can be used to construct it.

4.5.1 Proving knowledge of the re-encryption parameters

Notice that each mix-node runs the algorithm $\mathsf{MixVotes}$ and acts as a prover P. As a first step, P commits to N encryptions of zero obtaining for each ciphertext the following commitment $(\mathbf{c}_{u_0^{(i)}}, \mathbf{c}_{v_0^{(i)}})$:

$$\left(\mathbf{a}_\mathsf{C}\left(a_\mathsf{E} r_\mathsf{E}^{\prime(i)} + e_{\mathsf{E},u}^{\prime(i)}\right) + \mathbf{b}_\mathsf{C} r_{\mathsf{C},u}^{(i)} + \boldsymbol{e}_{\mathsf{C},\boldsymbol{u}}^{(\boldsymbol{i})}, \mathbf{a}_\mathsf{C}\left(b_\mathsf{E} r_\mathsf{E}^{\prime(i)} + e_{\mathsf{E},v}^{\prime(i)}\right) + \mathbf{b}_\mathsf{C} r_{\mathsf{C},v}^{(i)} + \boldsymbol{e}_{\mathsf{C},\boldsymbol{v}}^{(\boldsymbol{i})}\right)$$

That is, the commitment is a linear combination of the polynomials, with the additional condition of $r_\mathsf{E}^{\prime(i)}, e_{\mathsf{E},u}^{\prime(i)}, e_{\mathsf{E},v}^{\prime(i)}, \boldsymbol{e}_{\mathsf{C},\boldsymbol{u}}^{(\boldsymbol{i})}, \boldsymbol{e}_{\mathsf{C},\boldsymbol{v}}^{(\boldsymbol{i})}$ having small norm ($r_{\mathsf{C},u}^{(i)}$ and $r_{\mathsf{C},v}^{(i)}$ can be any polynomial in $\mathbb{Z}_q[x]/(x^n+1)$).

Then, P sends the commitments to the verifier and proves, using the amortized proof of knowledge of secret small elements by del Pino and Lyubashevsky [55] (see

Section 4.3.1), that the public commitments are indeed commitments to encryptions of zero.

$$\text{ZKPoK}\left[\begin{array}{c} r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v} \\ r^{(i)}_{\mathsf{C},u}, \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{u}}, r^{(i)}_{\mathsf{C},v}, \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{v}} \\ \\ \end{array} \middle| \begin{array}{l} \mathbf{c}_{u_0^{(i)}} = \mathbf{a}_{\mathsf{C}}\left(a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},u}\right) + \mathbf{b}_{\mathsf{C}} r^{(i)}_{\mathsf{C},u} + \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{u}} \\ \mathbf{c}_{v_0^{(i)}} = \mathbf{a}_{\mathsf{C}}\left(b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},v}\right) + \mathbf{b}_{\mathsf{C}} r^{(i)}_{\mathsf{C},v} + \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{v}} \\ \left\|r'^{(i)}_{\mathsf{E}}\right\|_\infty, \left\|e'^{(i)}_{\mathsf{E},*}\right\|_\infty \leq \tau\delta, \quad \left\|\boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},*}\right\|_\infty \leq \tau\delta' \end{array}\right]$$

For a linear function f, a small vector $\mathbf{x}$ and its image $y = f(\mathbf{x})$ we can prove knowledge of a small vector $\mathbf{x}'$ such that $f(\mathbf{x}') = y$. We can write this linear function in the following way:

$$\begin{aligned} &f(r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}, \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{u}}, \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{v}}, r^{(i)}_{\mathsf{C},u}, r^{(i)}_{\mathsf{C},v}) = \\ &\quad\left(\mathbf{a}_{\mathsf{C}}\left(a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},u}\right) + \mathbf{b}_{\mathsf{C}} r^{(i)}_{\mathsf{C},u} + \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{u}}, \mathbf{a}_{\mathsf{C}}\left(b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},v}\right) + \mathbf{b}_{\mathsf{C}} r^{(i)}_{\mathsf{C},v} + \boldsymbol{e}^{(\boldsymbol{i})}_{\mathsf{C},\boldsymbol{v}}\right) \end{aligned}$$

Since we need to prove knowledge of the preimages of this function for all $i \in \{1, \ldots, N\}$, we can amortize the cost by using del Pino and Lyubashebsky's technique.

As it is usual in this kind of proofs there is a gap τ between the upper bound of the norm we use for witness $\mathbf{x}$ and the upper bound we get for the extracted $\mathbf{x}'$. This has to be taken into account when determining specific parameters so that this possible error multiplied by the number of mix-nodes does not exceed the bounds allowed for a correct decryption.

We refer the reader to [55] for details, as we directly use their protocol as a building block for the ZKPoK of linear relations in ZK_1 (Protocol 4.6).

Using the amortization technique of [55] as a way of proving knowledge of valid openings for the commitments [29] has some benefits and some drawbacks. On the one hand this amortized technique allows us to prove the complex structure with an amortized cost. On the other hand the gap from the bound known by the prover and the bound he is able to prove is larger than the one originally established in the ZKPoK for valid commitment openings from [29].

As a result, the prover is only able to prove knowledge of some openings that would not be valid as originally defined. However, we can prove that, in our particular case, we can further relax this definition as the openings we obtain still ensure the binding property of the commitment scheme. Details of this and a rigorous parameter analysis are given below.

Del Pino and Lyubashevsky show in [55] how to prove knowledge of small secrets with an amortized cost. In order to do so their proof consists of two steps, an imperfect proof of knowledge, where the prover is able to prove knowledge of $N - \tau(\lambda)$ out of N secrets, and a compiler (adapted from [51]), used to transform an imperfect proof of knowledge into a regular proof of knowledge. The function $\tau(\lambda)$ defined for a security parameter λ is called imperfection.

Their initial imperfect proof has a soundness slack that depends on a parameter r and an imperfection $\tau(\lambda) = \frac{\lambda}{\log \alpha} + 1$. This r has to be an integer greater or equal

than 128 and α is another parameter that controls the minimal amount of samples required for amortization. They provide an example that suits our demands, for $\alpha = 2^{10}$ one can create amortized proofs for as few as 853 secrets with a security parameter $\lambda = 128$. The compilation step adds extra soundness gap, and as a result [55] claims that the final ZKPoK for a secret bounded by β has a slack of $4\sqrt{r}\lambda\beta/\log\alpha$ for a security parameter λ.

In our case we use n as a security parameter and consider the error term of the commitment scheme also bounded by n. Using this kind of amortized proofs we would be able to prove that the error is bounded by $4\sqrt{128}n\left(\frac{n}{10}\right)$. This is greater than $n^{4/3}/2$, as required by the definition of a valid opening. However no invertible f is involved, and we can just redo the original binding proof and show how, for a suitable set of parameters, with overwhelming probability over the choice of the commitment public key, if a valid commitment exists and a prover uses this particular amortized proof to prove knowledge of another opening, then the message cannot be a different one. This binding property is what is required for the soundness of our protocol.

Lemma 4.5.1 (Extended binding property)**.** Let $(m', r_{\mathsf{C}}', \boldsymbol{e}_{\mathsf{C}}', f')$, $(m'', r_{\mathsf{C}}'', \boldsymbol{e}_{\mathsf{C}}'', 1)$ be such that $\mathbf{c} = \mathbf{a}_{\mathsf{C}}m' + \mathbf{b}_{\mathsf{C}}r_{\mathsf{C}}' + f'^{-1}\boldsymbol{e}_{\mathsf{C}}' = \mathbf{a}_{\mathsf{C}}m'' + \mathbf{b}_{\mathsf{C}}r_{\mathsf{C}}'' + \boldsymbol{e}_{\mathsf{C}}''$ where $\left\|\boldsymbol{e}_{\mathsf{C}}'\right\|_\infty \leq \lfloor n^{4/3}/2 \rfloor$, $\|f'\|_\infty \leq 1$, $\deg f' < n/2$ and $\left\|\boldsymbol{e}_{\mathsf{C}}''\right\|_\infty \leq 4\sqrt{128}n\left(\frac{n}{10}\right)$. Then, provided that parameters are chosen appropriately, with overwhelming probability over the choice of $\mathbf{a}_{\mathsf{C}}$ and $\mathbf{b}_{\mathsf{C}}$, we have $m' = m''$.

Proof. Our goal is to find conditions on k and γ (defined as in [29], k is the dimension of $\mathbf{a}_{\mathsf{C}}$ and γ is such that $q \geq n^\gamma$) such that this lemma holds.

Assume by contradiction that $m' \neq m''$. Subtracting the two different expressions for $\mathbf{c}$ we get $\mathbf{a}_{\mathsf{C}}m + \mathbf{b}_{\mathsf{C}}r_{\mathsf{C}} = f'^{-1}\boldsymbol{e}_{\mathsf{C}}' - \boldsymbol{e}_{\mathsf{C}}''$, for some $m, r_{\mathsf{C}} \in R_q$ with $m \neq 0$. Lets fix these values $m, r_{\mathsf{C}}, f', \boldsymbol{e}_{\mathsf{C}}', \boldsymbol{e}_{\mathsf{C}}''$ and check that the chances of this being possible are negligible.

Here we use again the fact that, since $q \equiv 3 \mod 8$, $x^n + 1$ splits into two irreducible polynomials p_1 and p_2 of degree $n/2$. As $m \neq 0$ we have $m \neq 0 \mod p_b$ at least for one $b \in \{1, 2\}$. Considering all possible $a_i \in R_q$ we have that $a_i m$ takes all $q^{n/2}$ possible equivalence classes $\mod p_b$ with uniform probability. This is independent for every i, as a result only a fraction $\frac{1}{q^{kn/2}}$ of all possible $(\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})$ would satisfy the required equation.

Now, as we started fixing $m, r_{\mathsf{C}}, f', \boldsymbol{e}_{\mathsf{C}}', \boldsymbol{e}_{\mathsf{C}}''$ we have to apply a union bound for all their possible values. That is q^n for m, q^n for r_{C}, $3^{n/2}$ for f', $(n^{4/3})^{kn}$ for $\boldsymbol{e}_{\mathsf{C}}'$ and $\left(8\sqrt{128}n\left(\frac{n}{10}\right)\right)^{kn}$ for $\boldsymbol{e}_{\mathsf{C}}''$.

If this union bound is negligible then with overwhelming probability over the choice of $(\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})$ there are no $m, r_{\mathsf{C}}, f', \boldsymbol{e}_{\mathsf{C}}', \boldsymbol{e}_{\mathsf{C}}''$ satisfiying the equation with $m \neq 0$. It would imply that m has to be 0, and the commitment would be binding even when considering this relaxed opening verifications that come from the amortized proofs.

The only missing step is to check when the following quantity is negligible:

$$\frac{q^{2n}3^{n/2}(n^{4/3})^{kn}\left(8n\sqrt{128}\frac{n}{10}\right)^{kn}}{q^{kn/2}}$$
$$= \left(q^{2-k/2}3^{1/2}(n^{4/3})^{k}\left(\frac{2^{11/2}n^2}{5}\right)^{k}\right)^{n}$$

We know $k > 6$ from [29], then $2 - k/2 < 0$ and we can use $q \geq n^{\gamma}$ as defined in [29]:

$$\leq \left(n^{2\gamma - k\gamma/2}3^{1/2}(n^{4/3})^{k}\left(\frac{2^{11/2}n^2}{5}\right)^{k}\right)^{n}$$
$$= \left(n^{2\gamma + k(10/3-\gamma/2)}3^{1/2}\left(\frac{2^{11/2}}{5}\right)^{k}\right)^{n}$$
$$= \left(n^{2\gamma + k(10/3-\gamma/2+\log(2^{11/2}/5)/\log(n))}3^{1/2}\right)^{n}$$

And we want to impose that this quantity is negligible, that is:

$$\leq \left(\frac{1}{2}\right)^{n}$$

This is equivalent to:

$$\frac{1}{\sqrt{12}} \geq n^{2\gamma + k(10/3-\gamma/2+\log(2^{11/2}/5)/\log(n))}$$

And taking logarithms:

$$\frac{\log\left(1/\sqrt{12}\right)}{\log(n)} \geq 2\gamma + k(10/3 - \gamma/2 + \log(2^{11/2}/5)/\log(n))$$
$$0 \geq 2\gamma + k(10/3 - \gamma/2 + \log(2^{11/2}/5)/\log(n)) + \frac{\log(12)}{2\log(n)}$$

Notice how the contribution of the $\frac{1}{\log(n)}$ terms is positive. Therefore if the inequality is satisfied for some n_0 it would also be satisfied for any $n \geq n_0$. Therefore we can just plug in here the minimum value we want to consider for n, in this case $n = 2^9$ to achieve minimal security for the commitment scheme:

$$0 \geq 2\gamma + k\left(\frac{71 - 2\log(5) - 9\gamma}{18}\right) + \frac{\log(12)}{18}$$

Following the same reasoning, and using again $2 - k/2 < 0$ we notice that whenever this condition is satisfied for one γ_0 it will also be satisfied for any other $\gamma \geq \gamma_0$.

In order for the inequality to hold the coefficient of k, $\frac{71-2\log(5)-9\gamma}{18}$, has to be negative. This imposes $\gamma \geq 8$, and once we have this condition if the inequality holds for a given k_0 it will also be satisfied for any other $k \geq k_0$.

Summarizing, we just need to find the minimal pairs of $(k_0, \gamma_0) \in \mathbb{Z}^2$ satisfiying the following three conditions, and that would imply that any pair (k, γ) with $k \geq k_0$ and $\gamma \geq \gamma_0$ would be feasible too.

- $\gamma \geq 8$
- $k > \frac{18\gamma}{3\gamma-16}$
- $0 \geq 2\gamma + k\left(\frac{71-2\log(5)-9\gamma}{18}\right) + \frac{\log(12)}{18}$

The region of feasible parameters can be found in Figure 4.1. As long as we choose our parameters inside the green area the probability of the existence of non-zero solutions would be negligible and the commitment scheme will have the required extended binding property.

□

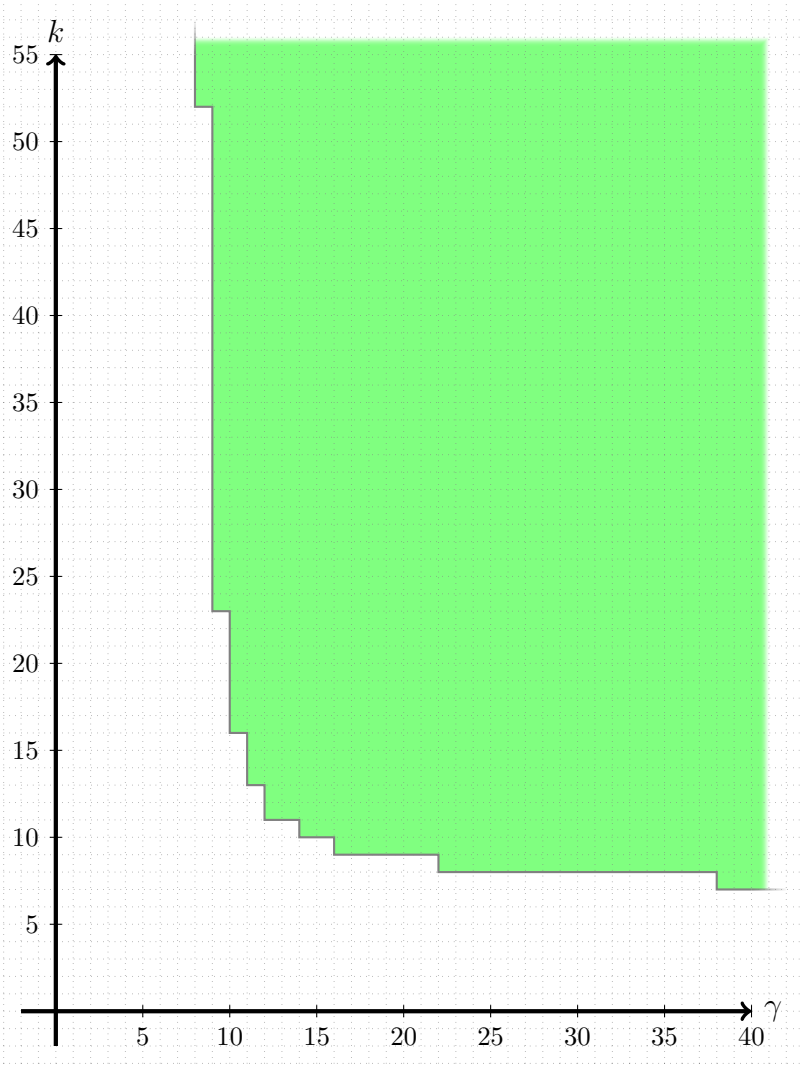


Figure 4.1: Region of feasible parameters satisfying the binding property.

4.5.2 Proving knowledge of the permutation

In order to commit to a permutation, P starts committing to $\pi(1), \ldots, \pi(N)$ using the commitment scheme presented in Section 4.3 and obtains $\mathbf{c}_{\pi(i)}$. Then, the prover

sends the commitments to V and receives a polynomial α chosen uniformly at random from the subset: $S = \{p(x) \in R_q \mid \deg p(x) < n/2\}$. As explained in Section 4.4, this subset meets the required conditions for Lemma 4.4.1.

P commits to each power $\alpha^{\pi(i)}$ in commitments $\mathbf{c}_{\alpha^{\pi(i)}}$ and publishes them. After that, P receives two more random polynomials $\beta, \gamma \xleftarrow{\$} S$.

If we denote $m_i \in \mathbb{Z}_q$ and $\widehat{m}_i \in R_q$ to the messages committed in $\mathbf{c}_{\pi(i)}$ and $\mathbf{c}_{\alpha^{\pi(i)}}$ respectively, at this point P starts proving that he knows valid integer openings m_i and $\widehat{m}_i$ to commitments $\mathbf{c}_{\pi(i)}, \mathbf{c}_{\alpha^{\pi(i)}}$ that satisfy the following relation (ZK_2 in Protocol 4.6):

$$\prod_{i=1}^{N} (\beta i + \alpha^i - \gamma) = \prod_{i=1}^{N} (\beta m_i + \widehat{m}_i - \gamma) \tag{4.3}$$

Note that ZK_2 indeed implies computing two proofs: the prover will use the amortized proposal by del Pino and Lyubashevsky [55] to demonstrate that the openings of the commitments meet certain conditions, i.e., they are integers; and it will use the $\Sigma-$protocol presented in Section 4.3.2 for proving the polynomial relation between committed messages defined by Equation 4.3.

As we will see later, since we are working in $\mathbb{Z}_q/(x^n+1)$ where x^n+1 splits into two irreducible polynomials p_1 and p_2, we need to guarantee that m_i (for $i \in \{1, \ldots, N\}$) is an integer in order to be sure that $m_i \mod p_1 = m_i \mod p_2$ and be able to prove that the Equation 4.3 holds. Note that if $m_i \mod p_1 \neq m_i \mod p_2$, by the Chinese Remainder Theorem we can obtain the polynomial m_i, thus m_i will not be an integer. In order to prove that the committed messages are integers the approach to be followed is similar to that used in Section 4.5.1. This time the linear function we need to consider maps the message, randomness and error to the commitment:.

$$f(m_i, r_{\mathsf{C}}^{(i)}, \boldsymbol{e}_{\mathsf{C}}^{(\boldsymbol{i})}) = \left(\mathbf{a}_{\mathsf{C}} m_i + \mathbf{b}_{\mathsf{C}} r_{\mathsf{C}}^{(i)} + \boldsymbol{e}_{\mathsf{C}}^{(\boldsymbol{i})}\right)$$

Originally [55] was designed for proving knowledge of small preimages, however everything works the same way if we just require part of the preimage to be small. During the generation of the proof, the small part of the preimage will be hidden with gaussian noise while the unbounded part will be hidden with uniformly random noise. The same parameter analysis that was done before applies here.

In order to verify Equation 4.3 we can use $\Sigma-$protocols presented in Section 4.3.2 that allow proving polynomial relations between committed messages.

We can consider the two sides of Equation 4.3 as polynomials in a variable Γ evaluated in a specific $\gamma \in R_q$ with coefficients in $\mathbb{Z}_q[x]/(x^n+1)$. The prover has shown that they are equal when evaluated in this specific γ chosen by the verifier, but we would like them to be equal as polynomials in $R_q[\Gamma]$. The left hand side of the equation has been determined by the choices of the verifier, and in the right hand side, by the binding property of the commitment scheme, we know that $m_i, \widehat{m}_i$ were determined before the choice for γ was made.

We have already checked that subset S satisfies the conditions of the Generalized Schwartz-Zippel lemma 4.4.1. Using this lemma the verifier is convinced that with overwhelming probability the two polynomials defined by 4.3 are indeed equal in $R_q[\Gamma]$. Nevertheless, this does not mean that the roots are also the same, so we would still have to prove that both sets of roots, $\{\beta_i + \alpha^i\}_i$, $\{\beta m_i + \widehat{m}_i\}_i$, are equal. This is not direct in general as R_q is not a unique factorization domain (in particular

it is not even a domain). However, in our particular case, both sets are going to be equal with overwhelming probability over the choice of β.

For each $j \in \{1, \ldots, N\}$, we are going to study whether $\beta j + \alpha^j$ belongs to $\{\beta m_i + \widehat{m}_i\}_i$. We know it is a root of the polynomial so $\prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - (\beta j + \alpha^j)) = 0$.

As we stated before, choosing $q \equiv 3 \mod 8$ implies that $x^n + 1$ splits into two irreducible polynomials of degree $n/2$. We are going to call these polynomials p_1 and p_2 and consider operations modulo both of them. In particular $\prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - (\beta j + \alpha^j)) \equiv 0 \mod p_1$ and $\prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - (\beta j + \alpha^j)) \equiv 0 \mod p_2$.

Given that p_1 and p_2 are irreducible, $\mathbb{Z}_q[x]/\langle p_1 \rangle$ and $\mathbb{Z}_q[x]/\langle p_2 \rangle$ are fields and it is possible to ensure that at least one of the factors has to be 0. Let i_{j1} and i_{j2} be the indexes such that $\beta m_{i_{j1}} + \widehat{m}_{i_{j1}} - (\beta j + \alpha^j) \equiv 0 \mod p_1$ and $\beta m_{i_{j2}} + \widehat{m}_{i_{j2}} - (\beta j + \alpha^j) \equiv 0 \mod p_2$.

Lets write it as affine equations on β:

$$(m_{i_{j1}} - j)\beta + (\widehat{m}_{i_{j1}} - \alpha^j) \equiv 0 \mod p_1$$
$$(m_{i_{j2}} - j)\beta + (\widehat{m}_{i_{j2}} - \alpha^j) \equiv 0 \mod p_2$$

First of all we need to see that, since m_i and $\widehat{m}_i$ were committed before β was honestly chosen uniformly from S, it is very unlikely that for any triplet $i, j \in [1, \ldots, N], b \in \{1, 2\}$ we have $(m_i - j)\beta + (\widehat{m}_i - \alpha^j) \equiv 0 \mod p_b$ unless $(m_i - j) \equiv 0 \mod p_b$. As we are now working in a field $\mathbb{Z}_q[x]/\langle p_b \rangle$ having $(m_i - j) \not\equiv 0 \mod p_b$ implies there is only one possible β satisfying the equation for each triplet (i, j, b). Notice that as elements of S have degree smaller than $n/2$ determining $\beta \mod p_b$ also determines it in R_q. There are $2N^2$ possible $\beta_{i_{jb}} \equiv (m_i - j)^{-j}(\alpha^j - \widehat{m}_i) \mod p_b$, but β is chosen uniformly at random from S, that has cardinal $q^{n/2}$ and therefore the probability of choosing one of these conflicting values is negligible.

Provided that previous proofs in ZK_2 ensure that $m_i \in \mathbb{Z}_q$ is a constant polynomial we have that $m_{i_{jb}} \equiv j \mod p_b$ implies $m_{i_{jb}} \equiv j \mod x^n + 1$. Since for each j we have $m_{i_{j1}} = m_{i_{j2}} = j$ this implies $i_{j1} = i_{j2}$ and we can directly call it i_j and write the equations $\mod x^n + 1$.

As a direct consequence we would also have $\widehat{m}_{i_j} = \alpha^j \mod x^n + 1$ via the Chinese Remainder Theorem.

Finally, we can ensure that, with overwhelming probability over the choice of β both sets commit to the same elements. Notice we have seen only one set inclusion, but since both sets contain the same number of elements and $i_j \neq i_{j'}$, if $j \neq j'$ this is everything we need.

Let $\tilde{\pi}$ be the permutation such that $j = \tilde{\pi}(i_j)$. Then , with overwhelming probability, $m_i = \tilde{\pi}(i)$ and $\widehat{m}_i = \alpha^{\tilde{\pi}(i)}$ for every $i \in [1, \ldots, N]$.

We abuse notation and call $m_{\alpha^{\pi(i)}}$ to $\widehat{m}_i$ as it has to be $\alpha^{\pi(i)}$, but understanding it is indexed by i and not the evaluation $\pi(i)$ that is unknown to the verifier.

This means that $\mathbf{c}_{\alpha^{\pi(i)}}$ are indeed commitments to α with exponents from 1 to N permuted in an order that was fixed by $\mathbf{c}_{\pi(i)}$ before α was chosen.

Protocol 4.6: Proof of a shuffle

$\mathcal{P}\left(u^{(i)}, u^{(i)}, u''^{(i)}, v''^{(i)}; \pi, r'^{(i)}_{\mathsf{E},} e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right)$ $\qquad$ $\mathcal{V}\left(u^{(i)}, u^{(i)}, u''^{(i)}, v''^{(i)}\right)$

$\forall i \in [1, \ldots, N]$
$\mathbf{c}_{u_0^{(i)}} = \mathsf{Com}\left(a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}+} e'^{(i)}_{\mathsf{E},u}\right)$
$\mathbf{c}_{v_0^{(i)}} = \mathsf{Com}\left(b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}+} e'^{(i)}_{\mathsf{E},v}\right)$

$$\xrightarrow{\mathbf{c}_{u_0^{(i)}}, \mathbf{c}_{v_0^{(i)}}}$$

$$\text{ZKPoK}\left[\begin{array}{c|l} r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v} \\ r^{(i)}_{\mathsf{C},u}, \boldsymbol{e^{(i)}_{\mathsf{C},u}}, r^{(i)}_{\mathsf{C},v}, \boldsymbol{e^{(i)}_{\mathsf{C},v}} & \begin{array}{l} \mathbf{c}_{u_0^{(i)}} = \mathbf{a}_{\mathsf{C}}\left(a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},u}\right) + \mathbf{b}_{\mathsf{C}} r^{(i)}_{\mathsf{C},u} + \boldsymbol{e^{(i)}_{\mathsf{C},u}} \\ \mathbf{c}_{v_0^{(i)}} = \mathbf{a}_{\mathsf{C}}\left(b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},v}\right) + \mathbf{b}_{\mathsf{C}} r^{(i)}_{\mathsf{C},v} + \boldsymbol{e^{(i)}_{\mathsf{C},v}} \\ \left\|r'^{(i)}_{\mathsf{E}}\right\|_\infty, \left\|e'^{(i)}_{\mathsf{E},*}\right\|_\infty \leq \tau\delta, \quad \left\|\boldsymbol{e^{(i)}_{\mathsf{C},*}}\right\|_\infty \leq \tau\delta' \end{array} \end{array}\right] \quad (\text{ZK}_1)$$

$\forall i \in [1, \ldots, N]$
$\mathbf{c}_{\pi(i)} = \mathsf{Com}(\pi(i))$

$$\xrightarrow{\mathbf{c}_{\pi(i)}}$$

$\alpha \xleftarrow{\$} S$

$$\xleftarrow{\alpha}$$

$\forall i \in [1, \ldots, N]$
$\mathbf{c}_{\alpha^{\pi(i)}} = \mathsf{Com}\left(\alpha^{\pi(i)}\right)$

$$\xrightarrow{\mathbf{c}_{\alpha^{\pi(i)}}}$$

$\beta, \gamma \xleftarrow{\$} S$

$$\xleftarrow{\beta, \gamma}$$

$$\text{ZKPoK}\left[\begin{array}{c|l} m_i, r^{(i)}_{\mathsf{C}}, \boldsymbol{e^{(i)}_{\mathsf{C}}}, f_i \\ \widehat{m}_i, \widehat{r}^{(i)}_{\mathsf{C}}, \widehat{\boldsymbol{e}}^{(i)}_{\mathsf{C}}, \widehat{f}_i & \begin{array}{l} \left(\prod_{i=1}^{N}\left(\beta i + \alpha^i - \gamma\right) = \prod_{i=1}^{N}\left(\beta m_i + \widehat{m}_i - \gamma\right)\right), \\ \bigwedge_{i=1}^{N}\left(\mathsf{ComVer}(\mathbf{c}_{\pi(i)}; m_i, r^{(i)}_{\mathsf{C}}, \boldsymbol{e^{(i)}_{\mathsf{C}}}, f_i) = \mathsf{accept}\right), \\ \bigwedge_{i=1}^{N}\left(\mathsf{ComVer}(\mathbf{c}_{\alpha^{\pi(i)}}; \widehat{m}_i, \widehat{r}^{(i)}_{\mathsf{C}}, \widehat{\boldsymbol{e}}^{(i)}_{\mathsf{C}}, \widehat{f}_i) = \mathsf{accept}\right) \\ m_i \in \mathbb{Z}_q \end{array} \end{array}\right] \quad (\text{ZK}_2)$$

$$\text{ZKPoK}\left[\begin{array}{c|l} y \in \left\{\begin{smallmatrix}\alpha^{\pi(i)} \\ u_0^{(i)}\end{smallmatrix}\right\}_i & \sum_{i=1}^{N}\alpha^i u^{(i)} = \sum_{i=1}^{N} m_{\alpha^{\pi(i)}}\left(u''^{(i)} - m_{u_0^{(i)}}\right) \\ r_{\mathsf{C},y} & \\ \boldsymbol{e_{\mathsf{C},y}} & \bigwedge_y\left(\mathsf{ComVer}(\mathbf{c}_y; m_y, r_{\mathsf{C},y}, \boldsymbol{e_{\mathsf{C},y}}, f_y) = \mathsf{accept}\right) \\ f_y & \end{array}\right] \quad (\text{ZK}_3)$$

$$\text{ZKPoK}\left[\begin{array}{c|l} y \in \left\{\begin{smallmatrix}\alpha^{\pi(i)} \\ v_0^{(i)}\end{smallmatrix}\right\}_{i,j,l} & \sum_{i=1}^{N}\alpha^i v^{(i)} = \sum_{i=1}^{N} m_{\alpha^{\pi(i)}}\left(v''^{(i)} - m_{v_0^{(i)}}\right) \\ r_{\mathsf{C},y} & \\ \boldsymbol{e_{\mathsf{C},y}} & \bigwedge_y\left(\mathsf{ComVer}(\mathbf{c}_y; m_y, r_{\mathsf{C},y}, \boldsymbol{e_{\mathsf{C},y}}, f_y) = \mathsf{accept}\right) \\ f_y & \end{array}\right] \quad (\text{ZK}_4)$$

outputs accept if all ZKPoK are correct

Until now, the prover has committed to the encryptions of 0 and has proven in zero-knowledge that they are indeed, encryptions of 0. Then, it has also commit to the permutation and has shown knowledge of it by proving that two polynomials are equal and have the same roots but in a permuted order. Combining these commitments and using the Σ-protocols from [29] presented in Section 4.3.2 we can generate a zero-knowledge proof (ZK_3 and ZK_4 in Protocol 4.6) to demonstrate that the input and the output of the mix-node hold the following relations:

$$\sum_{i=1}^{N} \alpha^i u^{(i)} = \sum_{i=1}^{N} \alpha^{\pi(i)} \left(u''^{(i)} - a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} - e'^{(i)}_{\mathsf{E},u} \right)$$

$$\sum_{i=1}^{N} \alpha^i v^{(i)} = \sum_{i=1}^{N} \alpha^{\pi(i)} \left(v''^{(i)} - a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} - e'^{(i)}_{\mathsf{E},v} \right)$$

Once again we can see them as polynomials in $R_q[\Gamma]$ with coefficients in R_q that are equal when evaluated in α.

Both polynomials were determined before α was picked up, so we can apply Lemma 4.4.1 and conclude that with overwhelming probability they are equal as polynomials, and so:

$$u''^{(i)} = u^{\pi(i)} + a_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},u} \qquad\qquad v''^{(i)} = v^{\pi(i)} + b_{\mathsf{E}} r'^{(i)}_{\mathsf{E}} + e'^{(i)}_{\mathsf{E},v}$$

The verifier $\mathcal{V}$ can conclude that the mix-net has behaved properly and the output is a permuted re-encryption of the input. Completeness, zero-knowledge and soundness follow from this reasoning and are discussed below.

Completeness, zero-knowledge and soundness

If the prover $\mathcal{P}$ chooses all re-encryption parameters from the appropriate distribution χ conditioned to have norm smaller than δ, correctly builds the commitments to the encryptions of 0 and follows the small secrets proof, the answer will be accepted. This is also the case for the proof of the committed permuted powers of α, as products $\prod_{i=1}^{N} (\beta i + \alpha^i - \gamma)$ and $\prod_{i=1}^{N} (\beta m_i + \widehat{m}_i - \gamma)$ are exactly equal, just in permuted order. Finally the two last ZKPoK are accepted as the output is exactly a permutation and re-encryption of the input, and we have built a polynomial subtracting the re-encryptions and inverting the permutation. To summarize, the protocol is complete as all the ZKPoK involved are accepted if an honest prover follows the protocols.

The special HVZK property is achieved as the only published elements are commitments (with a computationally hiding property based on the hardness of RLWE) and outputs of lattice-based ZK-protocols (that can be simulated and therefore leak no information).

Soundness follows with overwhelming probability from the soundness properties of the ZK-protocols for the commitments and the small elements, the binding property of the commitment scheme and also from the generalized Schwartz-Zippel lemma.

We start with ZK_1, if δ' is such that $\tau\delta' \leq \left\lfloor \frac{n^{4/3}}{2} \right\rceil$ the extractor of this zero-knowledge proof given by Del Pino and Lyubashevsky provides us with valid openings of $\mathbf{c}_{u_0^{(i)}}$ and $\mathbf{c}_{v_0^{(i)}}$ to a valid encryption of 0.

Then, we analyze ZK_2, using the extractor of Benhamouda *et al.* we obtain valid openings for $\mathbf{c}_{\pi(i)}$ and $\mathbf{c}_{\alpha^{\pi(i)}}$ that satisfy the equation $\prod_{i=1}^{N}(\beta i + \alpha^i - \gamma) = \prod_{i=1}^{N}(\beta m_i + \widehat{m}_i - \gamma)$. The order in which all polynomials have been determined, generalized Schwartz-Zippel and the previously discussed argument guarantees that, with overwhelming probability, those extracted messages are permuted integers from 1 to N and powers of α in the same order.

Finally we have ZK_3 and ZK_4, using the extractor of these proofs we obtain openings of $\mathbf{c}_{\pi(i)}, \mathbf{c}_{\alpha^{\pi(i)}}, \mathbf{c}_{u_0^{(i)}}, \mathbf{c}_{v_0^{(i)}}$. Given that the commitment scheme is binding we know from previous proofs that those openings are $\pi(i), \alpha^{\pi(i)}, u_0^{(i)}, v_0^{(i)}$. Then, the relations held by the messages committed that were written in terms of m_y are exactly $\sum_{i=1}^{N}\alpha^i u^{(i)} = \sum_{i=1}^{N}\alpha^{\pi(i)}(u'^{(i)} - u_0^{(i)})$ and $\sum_{i=1}^{N}\alpha^i v^{(i)} = \sum_{i=1}^{N}\alpha^{\pi(i)}(v'^{(i)} - v_0^{(i)})$. Applying the generalized Schwartz-Zippel lemma we can ensure with overwhelming probability that $u^{(i)} = u''^{\pi^{-1}(i)} - u_0^{\pi^{-1}(i)}$ and $v^{(i)} = v''^{\pi^{-1}(i)} - v_0^{\pi^{-1}(i)}$. And this implies that the mix-node has performed a correct shuffle on the input votes.

4.6 Security

Finally we propose a security definition and provide a proof of security for our proposed mix-node. Informally, a mix-node should ensure that it is not possible to link an input ciphertext with its corresponding output. However, there might be more than one ciphertext encrypting the same message (this is particularly the case in an election with many voters and only a few voting options), and we have to precisely say that it is not possible to link an input of the mix-node to an output encrypting the same message.

Some security definitions assume that the original messages are independently and uniformly distributed over the message space, but it was pointed out by Wikström in [150] that there might be known correlations between some of the input plaintexts that cannot be ignored.

We base our secure mix-node definition on that presented by Wikström in [150], but we notice that he assumes that the inputs of the mix-node are correctly computed encryptions of the messages. However the input of each mix-node comes from the (possibly malicious) previous node, and while the proof of a shuffle ensures that the input is a set of valid encryptions we do not know if the re-encryption parameters have been drawn randomly from the adequate distribution or specifically chosen by the possibly malicious previous node. Therefore we present a stronger definition where we even allow an adversary $\mathcal{A}$ to choose the messages and compute something of the form of an encryption, that is, a pair of polynomials in R_q, allowing them to completely determine the input of the mix-node. Even though, they should not be able to identify an input and output index corresponding to the same message with a probability significantly greater than a random guess. Let MixVotes be the algorithm that performs a shuffle and outputs a zero-knowledge proof Σ. Then we

can define:
$\mathbf{Exp}^{sec}_{\mathcal{A}}(\kappa)$

- $(pk, sk) \leftarrow \mathsf{SetupMix}(1^\kappa)$
- $(z^{(1)}, \ldots, z^{(N)}, aux) \xleftarrow{\$} \mathcal{A}(pk)$
- `for` $k \in \{1, \ldots, N\}$
  $(u^{(k)}, v^{(k)}) \xleftarrow{\$} \mathcal{A}(pk, z^{(k)}, aux)$
  `end for`
- $\pi \xleftarrow{\$} \mathfrak{S}_N$
- $\left(\{(u''^{(k)}, v''^{(k)})\}_{k=1}^N, \Sigma\right) \leftarrow \mathsf{MixVotes}(pk, \{(u^{(k)}, v^{(k)})\}_{k=1}^N; \pi)$
- $(i_{\mathcal{A}}, j_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}(\{(u^{(k)}, v^{(k)})\}_{k=1}^N, \{(u''^{(k)}, v''^{(k)})\}_{k=1}^N, \Sigma, aux)$
- `if` $z^{(i_A)} = z^{\pi(j_A)}$ `then` Return 1 `else` Return 0

Now we can formalize our security definition saying that no adversary can have a significant advantage over a random guess.

Definition 43 (Secure Mix-Node)**.** Let J be a uniform random variable taking values in $[1, \ldots, N]$. We say that a mix-node defined by an algorithm MixVotes is *secure* if the advantage of any PPT adversary $\mathcal{A}$ over a random guess is negligible in the security parameter. That is, for all c there exists a κ_0 such that if $\kappa \geq \kappa_0$:

$$\begin{aligned}\mathbf{Adv}^{sec}_{\mathcal{A}}(\kappa) &= \left|\Pr\left[z^{(i_A)} = z^{\pi(j_A)}\right] - \Pr\left[z^{(i_A)} = z^{\pi(J)}\right]\right| \\ &= \left|\Pr\left[\mathbf{Exp}^{sec}_{\mathcal{A}}(\kappa) = 1\right] - \Pr\left[z^{(i_A)} = z^{\pi(J)}\right]\right| < \frac{1}{\kappa^c}\end{aligned}$$

We allow the adversary to corrupt all mix-nodes except one, and the non-corrupted one is that considered in the experiment $\mathbf{Exp}^{sec}_{\mathcal{A}}$. In order to take into account any possible control of the adversary over those other corrupted nodes and possibly a subset of the voters we even allow him to fully control all the input of the mix-node. Even though, if at least one of the mix-nodes is honest, the link between the ciphertexts at the output and those at the input of the mix-net remains completely hidden.

Observe that this security definition has to be complemented with additional security proofs when this mix-node is used as a building block in a larger scheme. For instance Wikström in [150] shows how a malleable cryptosystem can be used to break anonymity. Therefore additional validity proofs are required to enforce non-malleability, as well as strict decryption policies to prevent any leakage of information during the decryption phase.

Theorem 4.6.1. The proposed mix-node given by our MixVotes algorithm is a secure mix-node according to Definition 43, under the RLWE hardness assumption.

Proof. We prove the security of a mix-node defining a sequence of games between a challenger and an adversary. Beginning from Game 0, that represents the original attack game with respect to a given efficient adversary, we use a sequence of hybrid arguments, Game 0, Game 1, Game 2 and Game 3, and we show that each game is indistinguishable from the previous one. Transitions between games are done applying very small changes to the defined experiment and we demonstrate that if an adversary can detect them, it would imply an efficient method of distinguishing between two distributions that are computationally indistinguishable under the corresponding assumptions. When Game 3 is reached, ciphertexts at the output of the mix-net are not RLWE samples any more, and are independent from the input.

Game 0 models the probability of an adversary getting output 1 from the experiment.

In Game 3 we have an output which is completely independent from the input and the original messages, and the permutation π is still chosen uniformly at random. Therefore the probability of guessing a correct pair of indices $(i_{\mathcal{A}}, j_{\mathcal{A}})$ is equivalent to choosing the second index uniformly at random from $[1, \ldots, N]$, i.e., sampling J.

This is the sequence of games:

Game (G_0)**.**

– Run $\mathsf{SetupMix}$ algorithm. $(((a_{\mathsf{E}}, b_{\mathsf{E}}), s), (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})) \xleftarrow{\$} \mathsf{SetupMix}(1^{\kappa})$.

$$pk_{\mathsf{E}} = (a_{\mathsf{E}}, b_{\mathsf{E}}) \qquad\qquad pk_{\mathsf{C}} = (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})$$

– The adversary chooses the messages. $(\{z^{(i)}\}_{i=1}^{N}, aux) \xleftarrow{\$} \mathcal{A}_1\,(pk_{\mathsf{E}}, pk_{\mathsf{C}})$.

– The adversary also computes the input of the mix-node.

$$\left(\left\{(u^{(i)}, v^{(i)})\right\}_{i=1}^{N}\right) \xleftarrow{\$} \mathcal{A}_2\left(\left\{z^{(i)}\right\}_{i=1}^{N}, aux\right)$$

– Mix the encrypted votes:

1. Choose a random permutation $\pi \xleftarrow{\$} \mathfrak{S}_N$.
2. Choose the re-encryption parameters $\{r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\}_{i=1}^{N}$ from the appropriate distribution.
3. Compute the output of the mixing process with their corresponding proofs using the $\mathsf{MixVotes}$ algorithm. $(\{(u''^{(i)}, v''^{(i)})\}_{i=1}^{N}, \{\Sigma_l\}_{l=0}^{4}) \leftarrow \mathsf{MixVotes}\left(pk_{\mathsf{E}}, pk_{\mathsf{C}}, \left\{(u^{(i)}, v^{(i)})\right\}_{i=1}^{N}; \pi, \left\{r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right\}_{i=1}^{N}\right)$

– $\mathcal{A}$ outputs $(i_{\mathcal{A}}, j_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}_3(\{(u^{(i)}, v^{(i)})\}_{i=1}^{N}, \{(u''^{(i)}, v''^{(i)})\}_{i=1}^{N}, \{\Sigma_l\}_{l=0}^{4}, aux)$.

– Check whether $z^{(i_{\mathcal{A}})} \stackrel{?}{=} z^{\pi(j_{\mathcal{A}})}$.

Game (G_1)**.**

– Run $\mathsf{SetupMix}$ algorithm. $(((a_{\mathsf{E}}, b_{\mathsf{E}}), s), (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})) \xleftarrow{\$} \mathsf{SetupMix}(1^{\kappa})$.

$$pk_{\mathsf{E}} = (a_{\mathsf{E}}, b_{\mathsf{E}}) \qquad\qquad pk_{\mathsf{C}} = (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})$$

– The adversary chooses the messages. $(\{z^{(i)}\}_{i=1}^{N}, aux) \xleftarrow{\$} \mathcal{A}_1\,(pk_{\mathsf{E}}, pk_{\mathsf{C}})$.

– The adversary also computes the input of the mix-node.

$$\left(\left\{(u^{(i)}, v^{(i)})\right\}_{i=1}^{N}\right) \xleftarrow{\$} \mathcal{A}_2\left(\left\{z^{(i)}\right\}_{i=1}^{N}, aux\right)$$

– Mix the encrypted votes:

1. Choose a random permutation $\pi \xleftarrow{\$} \mathfrak{S}_N$.
2. Choose the re-encryption parameters $\{r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\}_{i=1}^{N}$ from the appropriate distribution.

→ 3. Compute the output of the mixing process and simulate their corresponding proofs.

$$(u''^{(i)}, v''^{(i)}) \leftarrow \mathsf{Re\text{-}enc}\left(pk_{\mathsf{E}}, u^{\pi(i)}, v^{\pi(i)}; r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right)$$

$$\{\Sigma_l\}_{l=1}^{4} \xleftarrow{\$} \mathsf{Simulator}\left(pk_{\mathsf{E}}, pk_{\mathsf{C}}, \left\{(u^{(i)}, v^{(i)})\right\}_{i=1}^{N}, \left\{(u''^{(i)}, v''^{(i)})\right\}_{i=1}^{N}\right)$$

Since the zero-knowledge proofs are simulated, they are now independent from the commitments in Σ_0 and we can use their hiding property to substitute each one of them by random samples, without giving to the adversary more advantage in this game than the probability of breaking the RLWE assumption.

– $\mathcal{A}$ outputs $(i_{\mathcal{A}}, j_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}_3(\{(u^{(i)}, v^{(i)})\}_{i=1}^{N}, \{(u''^{(i)}, v''^{(i)})\}_{i=1}^{N}, \{\Sigma_l\}_{l=0}^{4}, aux)$.

– Check whether $z^{(i_{\mathcal{A}})} \stackrel{?}{=} z^{\pi(j_{\mathcal{A}})}$.

Game (G_2)**.**

→ – Run $\mathsf{SetupMix}$ algorithm. $(((a_{\mathsf{E}}, b_{\mathsf{E}}), s), (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})) \xleftarrow{\$} \mathsf{SetupMix}(1^{\kappa})$.

$$a'_{\mathsf{E}}, b'_{\mathsf{E}} \xleftarrow{\$} \mathbb{Z}_q\,[x]\,/\,(x^n+1) \qquad pk_{\mathsf{E}} = (a'_{\mathsf{E}}, b'_{\mathsf{E}}) \qquad pk_{\mathsf{C}} = (\mathbf{a}_{\mathsf{C}}, \mathbf{b}_{\mathsf{C}})$$

– The adversary chooses the messages. $(\{z^{(i)}\}_{i=1}^{N}, aux) \xleftarrow{\$} \mathcal{A}_1\,(pk_{\mathsf{E}}, pk_{\mathsf{C}})$.

– The adversary also computes the input of the mix-node.

$$\left(\left\{(u^{(i)}, v^{(i)})\right\}_{i=1}^{N}\right) \xleftarrow{\$} \mathcal{A}_2\left(\left\{z^{(i)}\right\}_{i=1}^{N}, aux\right)$$

– Mix the encrypted votes:

1. Choose a random permutation $\pi \xleftarrow{\$} \mathfrak{S}_N$.
2. Choose the re-encryption parameters $\{r'^{(i)}_{\mathsf{E}}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\}_{i=1}^{N}$ from the appropriate distribution.

3. Compute the output of the mixing process and simulate their corresponding proofs.

$$(u''^{(i)}, v''^{(i)}) \leftarrow \mathsf{Re\text{-}enc}_{pk_\mathsf{E}}\left(u^{\pi(i)}, v^{\pi(i)}; r'^{(i)}_\mathsf{E}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\right)$$
$$\{\Sigma_l\}_{l=0}^{4} \xleftarrow{\$} \mathsf{Simulator}\left(pk_\mathsf{E}, pk_\mathsf{C}, \left\{(u^{(i)}, v^{(i)})\right\}_{i=1}^{N}, \left\{(u''^{(i)}, v''^{(i)})\right\}_{i=1}^{N}\right)$$

– $\mathcal{A}$ outputs $(i_\mathcal{A}, j_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_3(\{(u^{(i)}, v^{(i)})\}_{i=1}^{N}, \{(u''^{(i)}, v''^{(i)})\}_{i=1}^{N}, \{\Sigma_l\}_{l=0}^{4}, aux)$.

– Check whether $z^{(i_\mathcal{A})} \stackrel{?}{=} z^{\pi(j_\mathcal{A})}$.

Game $(G_{2,j})$**.** We define G_3 to be $G_{2,N}$ and observe that $G_{2,0}$ is exactly G_2.

– Run SetupMix algorithm. $(((a_\mathsf{E}, b_\mathsf{E}), s), (\mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C})) \xleftarrow{\$} \mathsf{SetupMix}(1^\kappa)$.

$$a'_\mathsf{E}, b'_\mathsf{E} \xleftarrow{\$} \mathbb{Z}_q[x]/(x^n+1) \qquad pk_\mathsf{E} = (a'_\mathsf{E}, b'_\mathsf{E}) \qquad pk_\mathsf{C} = (\mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C})$$

– The adversary chooses the messages. $(\{z^{(i)}\}_{i=1}^{N}, aux) \xleftarrow{\$} \mathcal{A}_1(pk_\mathsf{E}, pk_\mathsf{C})$.

– The adversary also computes the input of the mix-node.

$$(\{(u^{(i)}, v^{(i)})\}_{i=1}^{N}) \xleftarrow{\$} \mathcal{A}_2(\{z^{(i)}\}_{i=1}^{N}, aux)$$

– Mix the encrypted votes:

1. Choose a random permutation $\pi \xleftarrow{\$} \mathfrak{S}_N$.

→ 2. Choose random polynomials and re-encryption parameters from the appropriate distribution.

$$w'^{(i)}_u, w'^{(i)}_v \xleftarrow{\$} \mathbb{Z}_q[x]/(x^n+1) \qquad \forall i \in [1, j]$$
$$\{r'^{(i)}_\mathsf{E}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}\}_{i=1}^{N} \xleftarrow{\$} \chi^n \qquad \forall i \in [j+1, N]$$

→ 3. Compute the modified output of the mixing process and simulate their corresponding proofs.

$$(u'^{(i)}, v'^{(i)}) = (u^{\pi(i)}, v^{\pi(i)}) + (w'^{(i)}_u, w'^{(i)}_v) \qquad \forall i \in [1, j]$$
$$(u''^{(i)}, v''^{(i)}) \leftarrow \mathsf{Re\text{-}enc}_{pk_\mathsf{E}}(u^{\pi(i)}, v^{\pi(i)}; r'^{(i)}_\mathsf{E}, e'^{(i)}_{\mathsf{E},u}, e'^{(i)}_{\mathsf{E},v}) \qquad \forall i \in [j+1, N]$$
$$\{\Sigma_l\}_{l=0}^{4} \xleftarrow{\$} \mathsf{Simulator}(pk_\mathsf{E}, pk_\mathsf{C}, \{(u^{(i)}, v^{(i)})\}_{i=1}^{N}, \{(u''^{(i)}, v''^{(i)})\}_{i=1}^{N})$$

– $\mathcal{A}$ outputs $(i_\mathcal{A}, j_\mathcal{A}) \xleftarrow{\$} \mathcal{A}_3(\{(u^{(i)}, v^{(i)})\}_{i=1}^{N}, \{(u''^{(i)}, v''^{(i)})\}_{i=1}^{N}, \{\Sigma_l\}_{l=0}^{4}, aux)$.

– Check whether $z^{(i_\mathcal{A})} \stackrel{?}{=} z^{\pi(j_\mathcal{A})}$.

Lemmas 4.6.2, 4.6.3 and 4.6.4 prove that, under RLWE assumptions, all four games above defined are equivalent. For any PPT adversary $\mathcal{A}$ the probability of winning in one of the games is at negligible distance to the probability of winning in any of the other games.

This proves the theorem and ensures that our mix-node is indeed secure. □

We let S_* be the event that $z^{(i_\mathcal{A})} = z^{\pi(j_\mathcal{A})}$ in game G_*.

Lemma 4.6.2. G_0 and G_1 are statistically indistinguishable.

Proof. In G_1 instead of generating the proofs $\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4$ using the witnesses, we simulate them. As simulated conversations are statistically close to real ones both games are indistinguishable in probabilistic polynomial time. Additionally, given that the commitment scheme is computationally hiding under the RLWE-assumption, we substitute each commitment in Σ_0 by random samples.

Then

$$|\Pr\{S_0\} - \Pr\{S_1\}| \leq \epsilon_{zkmix} + \epsilon_{hid}$$

where ϵ_{zkmix} is the advantage of an adversary against the zero-knowledge property of $\Sigma_1, \Sigma_2, \Sigma_3$ and Σ_4 and ϵ_{hid} is the advantage of an adversary against the RLWE problem, which are negligible. □

Lemma 4.6.3. G_1 and G_2 are computationally indistinguishable if the RLWE problem is hard.

Proof. This is immediate as we have just substituted the RLWE sample $(a_\mathsf{E}, b_\mathsf{E})$ by a uniform sample $(a'_\mathsf{E}, b'_\mathsf{E}) \xleftarrow{\$} R_q^2$.

Then

$$|\Pr\{S_1\} - \Pr\{S_2\}| \leq \epsilon_{dRLWE}$$

where ϵ_{dRLWE} is the advantage of an adversary against the decisional RLWE problem, which is negligible. □

Lemma 4.6.4. G_2 and G_3 are computationally indistinguishable if the RLWE problem is hard.

Proof. We can define N intermediate games between G_2 and G_3. $G_{2,0}$ will be G_2, $G_{2,N}$ will be G_3 and in each $G_{2,j}$ we add random $(w'^{(i)}_u, w'^{(i)}_v)$ for the first j encryptions and we use the $\mathsf{Re\text{-}enc}$ algorithm for all the others from $j+1$ to N, with correctly chosen re-encryption parameters.

Indistinguishability follows from the indistinguishability of any pair of games $G_{2,j}$ and $G_{2,j+1}$.

If they were not indistinguishable we could use them to correctly guess if two pairs of elements (g_1, h_1) and (g_2, h_2) are RLWE samples or uniformly random samples. We would just need to modify $G_{2,j+1}$ assigning $a'_\mathsf{E} = g_1, b'_\mathsf{E} = g_2, w'^{(j+1)}_u = h_1, w'^{(j+1)}_v = h_2$. If the samples came from a RLWE distribution the game would be exactly $G_{2,j}$, while if samples are uniformly random the game would be $G_{2,j+1}$.

Then

$$|\Pr\{S_{2,j-1}\} - \Pr\{S_{2,j}\}| \leq \epsilon_{dRLWE}$$

where ϵ_{dRLWE} is the advantage of an adversary against the decisional RLWE problem, which is negligible.

Finally, as in G_3 all the re-encryptions are uniformly random samples, it is clear that

$$\Pr\{S_3\} = \Pr\left[z^{(i_A)} = z^{\pi(J)}\right].$$

□

Combining all the probabilities we obtain the advantage of the adversary

$$\begin{aligned}\mathbf{Adv}^{sec}_{\mathcal{A}}(\kappa) &= \left|\Pr\left[\mathbf{Exp}^{sec}_{\mathcal{A}}(\kappa)=1\right] - \Pr\left[z^{(i_A)} = z^{\pi(J)}\right]\right| \\ &= |\Pr\{S_0\} - \Pr\{S_3\}| \leq \epsilon_{zkmix} + \epsilon_{hid} + (N+1)\epsilon_{dRLWE}\end{aligned}$$

which is negligible since ϵ_{zkmix}, ϵ_{hid} and ϵ_{dRLWE} are negligible.

4.7 Conclusions

In this chapter we have presented a proof of a shuffle fully constructed over lattice-based cryptography, which makes it secure in a post-quantum scenario. This proposal improves our previous work (see Chapter 3) but it is not a direct adaption of it to the post-quantum setting. This new construction follows the strategy proposed by Bayer and Groth in [22] but introduces some differences since working with lattices requires different techniques to be applied: while in [22] the authors demonstrate that there exists a linear combination of the re-encryption parameters such that an equality holds, we need to treat these parameters separately. We commit to them and prove that the elements committed have small norm and that satisfy a polynomial relation. Both the security of the commitment scheme and the zero-knowledge proofs used for building the fully post-quantum proof of a shuffle, is based on the hardness of solving lattice computational problems.

In addition to the description of the proof, in this chapter we also give a security definition and we prove that our shuffle satisfies it. The definition we use is based on that proposed [150] but is stronger, since we modify it in order to allow an adversary to completely determine the input of the mix-node. The proof of security is build using the game-playing technique and we demonstrate that our mix-node is secure according to the security definition under the RLWE hardness assumption.

As future work it would be worthy to have an implementation with concrete parameters in order to accurately test efficiency in a real setting. We also remark that this shuffle has to be combined with additional security requirements regarding how the input is generated as well as how the output is decrypted, in order to guarantee privacy for the overall scheme that uses this shuffle as a building block, and these requirements will depend on the specific application.

In the next chapter we show how to use this proof of a shuffle as a building block for constructing a post-quantum online voting system.

Chapter 5

A post-quantum online voting system

5.1 Introduction

In previous chapters, we have seen several lattice-based cryptographic primitives, some of them already existing in the literature, such as the RLWE encryption scheme [113] and others which are the result of the work done for this thesis, for example, the fully post-quantum proof of a shuffle presented in Chapter 4. We have also explained what is an online voting system, which are the requirements that ideally it should satisfy and which are the existing techniques that allow us to fulfill these requirements. Therefore, we already have all the *ingredients* to build a post-quantum online voting system.

In this chapter, we are going to present an overview of which was the main goal we had in mind when we started our research on lattice-based cryptography: a lattice-based online voting system secure under quantum attacks. In order to build this system, we are going to use most of the primitives we have already explained in Chapters 2 and 4 but also one extra protocol that we will explain in the current chapter.

To the best of our knowledge, there are two proposed e-voting schemes [44, 56] that are constructed using lattices. They both follow an alternative approach without shuffling, making use of the homomorphic property of their encryption schemes to compute the tally. However, mix-net based schemes are more flexible and provide better support for complex electoral processes.

In Section 5.2, we describe a protocol proposed by Guasch and Morillo [91], and we show how to implement it using lattice-based trapdoor functions and zero-knowledge proofs explained in Sections 2.4.5.2 and 4.3.1 correspondingly. This protocol allows the voters to check that their votes were cast as intended and also provides a mechanism against coercion. Then, in Section 5.3, we give an overview of the post-quantum online voting system, describing which are the algorithms involved in the protocol, how do they use the cryptographic primitives explained throughout the document, and how they are organized in the different phases. In the same section, we also discuss which are the security requirements fulfilled by the voting system and why. As this is on-going research, there is still a lot of room for

improvement, so finally, some ideas for future work are given in Section 5.4.

5.2 Coercion-resistant cast-as-intended protocol

In Section 2.3 we have talked about which are the security requirements that an ideal online voting system should satisfy and how they can be fulfilled. In this section we are interested on verifiability and more concretely on cast-as-intended verifiability. If an online voting system implements a mechanism that provides cast-as-intended verifiability, voters can check that the voting options they have selected are indeed those that were encrypted by their voting device. These mechanisms can be based on the so-called return codes, on challenging the voting device, or on decrypting the vote stored in the ballot box. Here we are going to focus on the second technique, known as *challenge or cast*, first proposed by Benaloh [28]. Just as a reminder, this technique consists on verifying the encrypted vote before being cast. Once the voter has selected the voting options, the voting device encrypts them, commits to the resulting ciphertext and the voter is asked either to challenge the voting device or to cast the vote. In the first scenario the voter uses an alternative software and the information provided by the voting device to check that the encrypted vote indeed contains the voting options selected. If the verification is successful, the voting device generates a new encryption using fresh randomness and the voter is asked again whether to challenge or cast the vote. Note that this protocol is sound as long as the voting device does not know, before committing to the encrypted voting options, whether the voter will decide to challenge it or to cast the vote. Indeed, a malicious voting device which has modified the voting options selected by the voter, has 50% of probability of being caught. It is recommended to repeat the protocol multiple times, i.e., to choose to challenge the voting device multiple times, in order to increase this probability.

The main drawback of this protocol is that, in order to prevent vote-selling attacks, it does not allow for auditing the vote that is going to be cast. With this problem in mind, Guasch and Morillo proposed in 2016 [91] a new technique for providing cast-as-intended verifiability but also coercion-resistance called *challenge and cast*, which is an improvement of the challenge or cast protocol. This new technique is explained in Section 5.2.1 and the lattice version is presented in Section 5.2.2.

5.2.1 Challenge and cast protocol

The challenge and cast solution [91] consists on the following: the voting device encrypts the voting options and computes a zero-knowledge proof of the encryption randomness. Then, instead of revealing the randomness directly to the voter as it is done in the challenge or cast protocol, it shows the ciphertext together with the proof to the voter. The voter uses an audit device to verify the proof and if the verification is successful, the vote is cast and published in the bulletin board. Note that with the proof itself it is still possible to sell the vote or to coerce the voter. This is why the authors take advantage of the simulatability of the zero-knowledge

proof to allow the voter to generate *fake proofs* which will look like valid ones to anyone else.

These proofs are called Designated Verifier Proofs [95], in which only the designated verifier, in our case the voter, is convinced of the correctness of the proof and has a trapdoor that allows them to simulate the proof for false statements to other verifiers. The intuition behind this proof is the following: the prover, which is the voting device, demonstrates that either the ciphertext contains a concrete message or the prover is the voter. When the voter checks the proof, since he knows that the prover, i.e., the voting device, is not him, he is convinced that the ciphertext contains the voting options he has selected.

In order to build this proof in the non-interactive setting using the Fiat-Shamir transformation [66] (see Section 2.2.4.2), chameleon hashes [98] are used (see Definition 44). The idea is that the challenge generated during the non-interactive protocol is substituted by the output of the chameleon hash function. These functions are trapdoor collision-resistant hash functions and were first introduced by Krawczyk and Rabin in [98]. The main difference between this kind of hash functions and the standard ones (see Definition 13 in Section 2.2.3), is that a chameleon hash is collision-resistant only for those who do not know the trapdoor, i.e., the owner of the trapdoor can find two inputs for which the output is the same; while using a standard hash function is infeasible two find collisions.

Definition 44 (Chameleon hash function [98])**.** A chameleon hash function is composed by three PPT algorithms:

- $\mathsf{CGen}(1^\kappa)$: given as input the security parameter 1^κ, the algorithm outputs a pair of public and private keys (ek, tk), namely, the evaluation and the trapdoor key. In addition it also defines the message space $\mathcal{M}_{ch}$, the randomness space $\mathcal{R}_{ch}$ and the hash space $\mathcal{H}_{ch}$.

- $\mathsf{CHash}(ek, m, r)$: given as input the evaluation public key ek, a message $m \in \mathcal{M}_{ch}$ and randomness $r_{ch} \in \mathcal{R}_{ch}$, the algorithm outputs a chameleon hash $c_{ch} \in \mathcal{H}_{ch}$.

- $\mathsf{CHash}^{-1}(tk, m, m', r)$: given as input the trapdoor key tk, two messages $m, m' \in \mathcal{M}_{ch}$ and a randomness $r_{ch} \in \mathcal{R}_{ch}$, the algorithm outputs a randomness $r'_{ch} \in \mathcal{R}_{ch}$ such that $\mathsf{CHash}(ek, m, r_{ch})$=$\mathsf{CHash}(ek, m', r'_{ch})$.

and should satisfy the following requirements:

- Collision resistance: there is no efficient PPT algorithm that given the evaluation key ek can find $(m, r_{ch}) \neq (m', r'_{ch})$ such that $\mathsf{CHash}(ek, m, r_{ch})$= $\mathsf{CHash}(ek, m', r'_{ch})$, except with negligible probability.

- Trapdoor collisions: there is an efficient algorithm that given the trapdoor key tk, two different message m, m' and a randomness r_{ch}, can find r'_{ch} such that $\mathsf{CHash}(ek, m, r_{ch})$=$\mathsf{CHash}(ek, m', r'_{ch})$.

Then, the simulatable NIZKP is composed by the following algorithms:

- $\mathsf{GenCRS}(1^\kappa)$: given as input the security parameter 1^κ, the algorithm runs $\mathsf{CGen}(1^\kappa)$ and outputs the evaluation ek and the trapdoor key tk.

- $\mathsf{NIZKProve}(ek, x, w)$: given as input the evaluation key ek, the statement x and the witness w, this algorithm executes the first two movements of a $\Sigma-$protocol (see Section 2.2.4.1) in the following way: generates the commitment a and a random $r_{ch} \in \mathcal{R}_{ch}$. Then, it defines $m = H_1(x, a)$ and computes the challenge $e = H_2(\mathsf{CHash}(ek, m, r_{ch}))$, where $H_1 : \{0,1\}^* \to \mathcal{M}_{ch}$ and $H_2 : \{0,1\}^* \to \mathcal{C}$ (the challenge space) are standard collision-resistant hash functions. Finally, it obtains the answer z and outputs the proof $\pi = (a, e, r_{ch}, z)$.

- $\mathsf{NIZKVerify}(\pi, x)$: given as input the proof π and the statement x, the algorithm computes $e' = H_2(\mathsf{CHash}(ek, H_1(x, a), r_{ch}))$ and checks whether $e = e'$ and the validations of the $\Sigma-$protocol pass. If the verifications are successful the algorithm outputs 1, 0 otherwise.

- $\mathsf{NIZKSimulate}(x, tk)$: given as input a statement x and the trapdoor key tk of the chameleon hash scheme, the simulator computes the simulates proof $\pi^* = (a^*, e^*, r^*_{ch}, z^*)$ in the following way: it uses the simulator of the $\Sigma-$protocol to generate (a^*, e^*, z^*). Then, it uses the trapdoor key tk, the simulated commitment a^*, the simulated challenge e^*, the statement x and the algorithm CHash^{-1} to obtain the value r^*_{ch}.

This cast-as-intended mechanism improves the usability and the soundness of the verification process: the vote that is cast is the same that has been audited.

In order to clarify the idea of this proof we give a concrete instantiation of the simulatable NIZKPoK using the ElGamal encryption scheme and a chameleon hash based on the discrete logarithm problem [98].

Let $(c_1, c_2) = (g^r, pk^r \cdot v)$ be the ElGamal ciphertext. A chameleon hash for a given message m and a randomness $r \xleftarrow{\$} \mathbb{Z}_q$ is computed as $c_{ch} = g^m \cdot h^r$, where $h = g^x$ and x is the trapdoor sampled uniformly from $\mathbb{Z}_q$. Additionally, define the hash functions $H_2 : \{0,1\}^* \to \mathcal{C}$ and $H_1 : \{0,1\}^* \to \mathcal{M}_{ch}$ where $\mathcal{C}$ and $\mathcal{M}_{ch}$ are the challenge and the message space respectively. The objective of the proof is to demonstrate that $log_g c_1 = log_{pk}(\frac{c_2}{v})$.

The interactive protocol between the prover and the verifier is described in Protocol 5.1. In order to make it non-interactive and simulatable, the prover computes the challenge e in the following way:

$$e = H_2(\mathsf{CHash}(H_1(c_1, \frac{c_2}{v}, a_1, a_2), r)) = H_2(g^{H_1(c_1, \frac{c_2}{v}, a_1, a_2)} h^r)$$

Then proof is then defined as $\pi = (a_1, a_2, e, r, z)$.

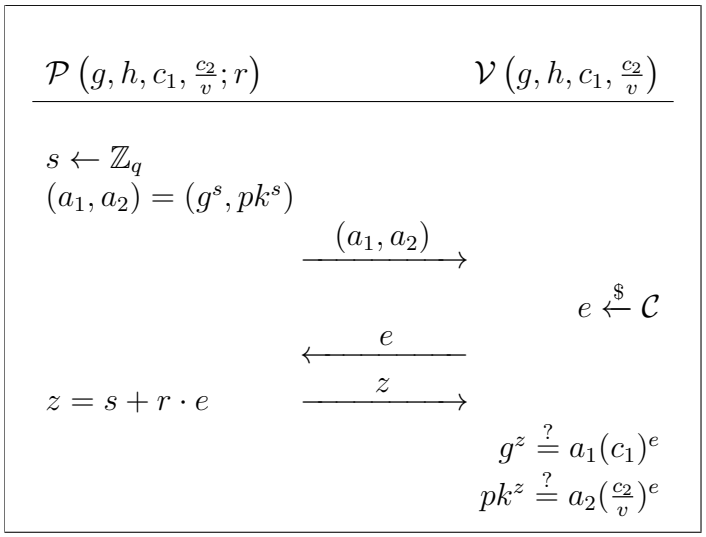
If the designated verifier wants to generate a fake proof in order to convince another verifier that the message encrypted is v^* instead of v, it uses the trapdoor key in the following way:

1. Define the new statement to be proven: $(c_1, \frac{c_2}{v^*})$.

2. Take at random the values $z^* \in \mathbb{G}$, $\alpha \in \mathbb{Z}_q$ and $\beta \in \mathbb{Z}_q$.

3. Set $e^* = H_2(g^\alpha h^\beta)$.
4. Compute ${a_1}^* = g^{z^*} {c_1}^{e^*}$ and ${a_2}^* = h^{z^*}(c_2/v^*)^{e^*}$.
5. Obtain r^* such that the following equality is fulfilled
$$H_2(g^{H_1(c_1,c_2/v^*,a_1^*,a_2^*)} h^{r^*}) = H_2(g^\alpha h^\beta)$$
Given that $h = g^x$, the randomness is $r^* = (\alpha - H_1(c_1, c_2/v^*, a_1^*, a_2^*)) \cdot x^{-1} + \beta$. This is done by running the CHash^{-1} algorithm with the following inputs: $\mathsf{CHash}^{-1}(x, \alpha, H_1(c_1, c_2/v^*, a_1^*, a_2^*), \beta)$
6. The simulated proof is $\pi = (a_1^*, a_2^*, e^*, r^*, z^*)$.

Protocol 5.1



5.2.2 Lattice-based coercion resistant cast-as-intended protocol

In this section we present our lattice version of the chameleon hash function and the simulatable NIZKP. As we have already explained in the previous section, a chameleon hash function needs a trapdoor in order to allow the owner of it to find collisions.

First of all, the lattice-based hash function we are going to use is that described in Section 2.4.5.1: $f_A(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \in \mathbb{Z}_q^n$, were $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{x}$ is a short integer vector in $\mathbb{Z}_q^m$. Then, for constructing $\mathbf{A}$ we follow the strategy for constructing lattice-based trapdoor functions presented in Section 2.4.5.2. We recall how to do it hereunder:

$$\mathbf{A} = [\mathbf{B}|\mathbf{G} - \mathbf{B}\mathbf{R}]$$

where $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ is a public matrix for which we know that the function f_G is easy to invert, $\mathbf{B} \in \mathbb{Z}_q^{n \times \overline{m}}$ is chosen uniformly at random and $\mathbf{R} \in \mathbb{Z}_q^{(m-w) \times w}$ is the trapdoor (note that $\overline{m} = m - w$).

Now, we can construct the lattice-based chameleon hash function in the following way:

- The algorithm CGen outputs the matrix $\mathbf{A} = \left(\mathbf{A}_1|\mathbf{A}_2\right) \in \mathbb{Z}_q^{n\times\tilde{m}}$ where $\mathbf{A}_1 \in \mathbb{Z}_q^{n\times\hat{m}}$ is chosen uniformly at random and $\mathbf{A}_2 \in \mathbb{Z}_q^{n\times m}$ is constructed using a trapdoor $\mathbf{R} \in \mathbb{Z}_q^{(m-w)\times w}$ as explained before (note that we define $\tilde{m} = \hat{m}+m$). The evaluation key is $\mathbf{A}$ and the trapdoor key is $\mathbf{R}$.

- The algorithm CHash receives as input the evaluation key $\mathbf{A}$, a message $\mathbf{x} \in \mathbb{Z}_q^{\hat{m}}$ and a randomness $\mathbf{r} \in \mathbb{Z}_q^m$ where both $\mathbf{x}$ and $\mathbf{r}$ are small vectors and $\mathbf{r}$ is chosen from a discrete Gaussian distribution. The chameleon hash is computed as:

$$f_{\mathbf{A}}(\mathbf{x},\mathbf{r}) = \left(\mathbf{A}_1|\mathbf{A}_2\right)\begin{pmatrix}\mathbf{x}\\ \mathbf{r}\end{pmatrix} = \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{r} = \mathbf{y}$$

  .

- The algorithm CHash^{-1} receives as input the trapdoor $\mathbf{R}$, the message $\mathbf{x}$ and the randomness $\mathbf{r}$ used to compute $f_{\mathbf{A}}(\mathbf{x},\mathbf{r})$ and the new message $\mathbf{x}'$. The randomness $\mathbf{r}'$ is computed in the following way. We can express $\mathbf{r}'$ as $\mathbf{r}' = \begin{pmatrix}\mathbf{r}'_1\\ \mathbf{r}'_2\end{pmatrix}$ and, since we know that $\mathbf{A}_2$ is constructed as $\mathbf{A}_2 = [\mathbf{B}|\mathbf{G}-\mathbf{BR}]$, we can define:

$$f_{\mathbf{A}}(\mathbf{x}',\mathbf{r}'_1,\mathbf{r}'_2) = \left(\mathbf{A}_1 \;\middle|\; \mathbf{B} \;\middle|\; \mathbf{G}-\mathbf{BR}\right)\begin{pmatrix}\mathbf{x}'\\ \mathbf{r}'_1\\ \mathbf{r}'_2\end{pmatrix} = \mathbf{A}_1\mathbf{x}' + \mathbf{B}\mathbf{r}'_1 + (\mathbf{G}-\mathbf{BR})\mathbf{r}'_2$$

  The goal is to find $\mathbf{r}'_1$ and $\mathbf{r}'_2$ such that $f_{\mathbf{A}}(\mathbf{x}',\mathbf{r}'_1,\mathbf{r}'_2) = \mathbf{y}$. Following the technique explained in Section 2.4.5.2, we choose a random $\hat{\mathbf{r}}_1$ from the discrete Gaussian distribution and compute $f_{\mathbf{B}}(\hat{\mathbf{r}}_1) = \mathbf{B}\hat{\mathbf{r}}_1$. Then, we sample a random preimage $\hat{\mathbf{r}}_2$ from $f_{\mathbf{G}}^{-1}(\mathbf{y}-\mathbf{A}_1\mathbf{x}'-\mathbf{B}\hat{\mathbf{r}}_1) = f_{\mathbf{G}}^{-1}(\mathbf{G}\hat{\mathbf{r}}_2)$ and define $\mathbf{r}'_2 = \hat{\mathbf{r}}_2$. Since we want that:

$$\begin{aligned}\mathbf{A}_1\mathbf{x}' + \mathbf{B}\mathbf{r}'_1 + (\mathbf{G}-\mathbf{BR})\mathbf{r}'_2 &= \mathbf{y}\\ \mathbf{G}\mathbf{r}'_2 &= \mathbf{y} - \mathbf{A}_1\mathbf{x}' - \mathbf{B}\mathbf{r}'_1 + \mathbf{BR}\mathbf{r}'_2\\ \mathbf{G}\mathbf{r}'_2 &= \mathbf{y} - \mathbf{A}_1\mathbf{x}' + \mathbf{B}(\mathbf{r}'_1 - \mathbf{R}\mathbf{r}'_2)\end{aligned}$$

  we define $\mathbf{r}'_1 = \hat{\mathbf{r}}_1 + \mathbf{R}\mathbf{r}'_2$.

This chameleon hash function is *collision resistance* since finding $(\mathbf{x},\mathbf{r}) \neq (\mathbf{x}',\mathbf{r}')$ such that $f_{\mathbf{A}}(\mathbf{x},\mathbf{r}) = f_{\mathbf{A}}(\mathbf{x}',\mathbf{r}')$ implies $\mathbf{A}\begin{pmatrix}\mathbf{x}-\mathbf{x}'\\ \mathbf{r}-\mathbf{r}'\end{pmatrix} = 0$, i.e., solving the SIS problem. Moreover, it also satisfies the property of *trapdoor collisions* since, as we have seen before, given $\mathbf{x},\mathbf{r}$ and $\mathbf{x}'$ it is possible to find $\mathbf{r}'$ such that $f_{\mathbf{A}}(\mathbf{x},\mathbf{r}) = f_{\mathbf{A}}(\mathbf{x}',\mathbf{r}')$ with the knowledge of the trapdoor $\mathbf{R}$.

Once we know how to construct a lattice-based chameleon hash function we can build the lattice-based simulatable NIZKP. Using this proof the prover will demonstrate that a ciphertext contains a concrete vote z which in the lattice setting translates to prove knowledge of a solution to the ISIS problem as explained in

Section 4.3.1. We recall that we can re-write the RLWE encryption of a message z as:

$$\mathbf{y} = \tilde{\mathbf{A}}\mathbf{x} \tag{5.1}$$

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 - \lfloor \frac{q}{2} \rceil z_1 \\ \vdots \\ v_n - \lfloor \frac{q}{2} \rceil z_n \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{Id}_n & \mathbf{0}_n \\ \mathbf{B} & \mathbf{0}_n & \mathbf{Id}_n \end{pmatrix} \begin{pmatrix} r'_{\mathsf{E},1} \\ \vdots \\ r'_{\mathsf{E},n} \\ e'_{\mathsf{E},u,1} \\ \vdots \\ e'_{\mathsf{E},u,n} \\ e'_{\mathsf{E},v,1} \\ \vdots \\ e'_{\mathsf{E},v,n} \end{pmatrix}$$

Then, using Ling *et al.* proposal (see Section 4.3.1, Protocol 4.2) we demonstrate knowledge of the vector of small elements $\mathbf{x}$. As one round of the protocol has soundness error 2/3 it is necessary to repeat it several times in order to achieve enough soundness. For this reason, during the execution of the $\mathsf{NIZKProve}$ the prover will compute t tuples of commitments $\{\mathbf{c}_{1,i}, \mathbf{c}_{2,i}, \mathbf{c}_{3,i}\}_{i=1}^{t}$ and will store them in a vector $\mathbf{c}$. Then, the challenge e will be computed as $e = H_2(\mathsf{CHash}(\mathbf{A}, H_1(\mathbf{y}, \mathbf{c}), \mathbf{r}))$ where $\mathbf{A}$ is the evaluation key, $\mathbf{r} \in \mathbb{Z}_q^m$ is a small vector chosen from a discrete Gaussian distribution, H_1 is a hash function that sends the statement $\mathbf{y}$ concatenated with t tuples of commitments, to small vectors in $\mathbb{Z}_q^{\hat{m}}$ and H_2 is a hash function that sends vectors from $\mathbb{Z}_q^n$ to the space of challenges $e \in \{1, 2, 3\}^t$. Finally, the prover computes the answer z and shows the proof to the verifier, i.e., the voter, which also contains the randomness $\mathbf{r}$ used for computing the chameleon hash.

5.3 Voting system overview

In Section 2.3.4 we have already defined the syntax regarding the participants of an online voting system, the phases and the algorithms executed in each one of them. In this section we are going to detail which are exactly the operations done by each of the algorithms, also specifying which cryptographic primitives of those explained throughout the document are used. In addition, we are going to introduce new algorithms which are specific to this voting system, we are going to show how the participants interact in each phase and finally we are going to explain which of the security requirements defined in Section 2.3.1 are fulfilled and how.

There are several electoral models that would be interesting to support in an online voting system, for example, *write-ins*, which allow voters to write their preferences instead of selecting them from a pre-defined list, or questions with preferential answers which allow voters to numerically order options according to their preferences. Nevertheless, for the sake of clarity, here we are going to work with the simplest electoral model which consists on one question with several answers of which the voter can only select one. It is left for future work to extend the system to support more electoral models.

We denote as $\mathcal{V} = \{v_1, \ldots, v_\psi\}$ to the set of voting options the voter can vote for. We assume that $\mathcal{V}$ is the same for all the voters.

The voting system uses the following cryptographic schemes as building blocks, all of them explained in previous sections or chapters: the RLWE encryption scheme ($\mathsf{KeyGen_E}, \mathsf{Enc}, \mathsf{Dec}$), the RLWE commitment scheme ($\mathsf{KeyGen_C}, \mathsf{Com}$, ComVer), a digital signature scheme ($\mathsf{KeyGen_S}, \mathsf{Sign}, \mathsf{SignVer}$), a simulatable NIZKPoK (GenCRS,$\mathsf{NIZKProve}$,$\mathsf{NIZKVerify}$,$\mathsf{NIZKSimulate}$) and a mixing protocol ($\mathsf{SetupMix}$,$\mathsf{MixVotes}$,$\mathsf{VerifyMix}$).

We propose to use as a lattice-based signature scheme one of those submitted to the NIST competition. The FALCON algorithm seems to be a good candidate due to its compactness and performance.

5.3.1 Configuration and registration phase

During the configuration phase the electoral authority generates the election information that is common to all the voters, such as the voting options, the election key pair or the commitment key. As we have explained in Section 2.2.2.3, in some systems it is desirable that the election private key is not owned by a single user but a group of them. In this situation the private key is split in as many shares as users using a (threshold) secret sharing scheme (or it is directly generated in a distributed way) and each user securely stores their private key share during the whole election. This key sharing procedure usually takes place during the configuration phase. For simplicity in the explanation we are going to assume that the whole private key is kept by a single entity.

The counting phase consists only on one algorithm, the Setup.

- $\mathsf{Setup}(1^\kappa)$: it receives as input the security parameter and runs the $\mathsf{SetupMix}(1^\kappa)$ algorithm of the mixing protocol (see Section 4.4). It outputs the election key pair $(pk_e, sk_e) = ((a_\mathsf{E}, b_\mathsf{E}), s)$ and the commitment key $(\mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C})$.

The steps executed during the counting phase, shown in Figure 5.1, are the following ones:

1. The electoral authority generates the list of voting options of the election $\{v_i\}_{i=1}^{\psi}$ and the empty credential list ID.

2. The electoral authority runs the $\mathsf{Setup}(1^\kappa)$ algorithm and obtains the election key pair $(pk_e, sk_e) = ((a_\mathsf{E}, b_\mathsf{E}), s)$ and the commitment key $(\mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C})$.

3. The voting options $\{v_i\}_{i=1}^{\psi}$, the list ID, the election public key pk_e and the commitment key $(\mathbf{a}_\mathsf{C}, \mathbf{b}_\mathsf{C})$ are published in the bulletin board. The election private key sk_e is kept by the electoral authority.

During the registration phase all the voter-related information is generated. Voters are provided with the keys that will allow them to authenticate their vote and to generate fake proofs of the content of their encrypted votes in case it is needed. This phase consists of the $\mathsf{Register}$ algorithm:

- $\mathsf{Register}(1^\kappa, \mathsf{id})$: it receives as input the security parameter and the identity id of a voter and runs the GenCRS algorithm of the simulatable NIZK protocol and the $\mathsf{KeyGen_S}(1^\kappa)$ algorithm of the digital signature scheme. It outputs the voter's signing and verification keys $(pk_{s,\mathsf{id}}, sk_{s,\mathsf{id}})$ and the voter's evaluation and trapdoor keys $(ek_{\mathsf{id}}, tk_{\mathsf{id}})$.

The steps executed during the registration phase, shown in Figure 5.1, are the following ones:

1. For each voter with identity id, the registration authority runs the $\mathsf{Register}$ algorithm.

2. The registration authority updates the list id with the following information for each registered voter: $(\mathsf{id}, pk_{s,\mathsf{id}}, ek_{\mathsf{id}})$. The key pairs $(pk_{s,\mathsf{id}}, sk_{s,\mathsf{id}})$ and $(ek_{\mathsf{id}}, tk_{\mathsf{id}})$ are given to the voter.

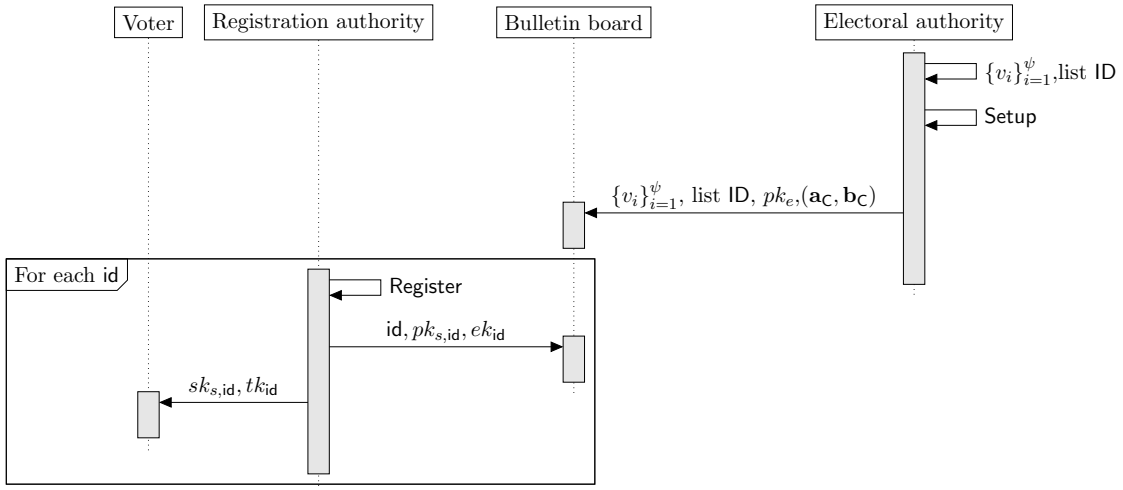


Figure 5.1: Overview of the interaction among the online voting system participants during the configuration and registration phases.

5.3.2 Voting phase

During the voting phase each voter selects their preferred voting option, which is encrypted in the voting device. Both the ciphertext and the proof of content are shown to the voter, who can use an audit device to verify that the voting device is not cheating and has encrypted the option selected by them. If the audit is successful, they introduce their signing key into the voting device that uses it to sign the vote. The signed vote is sent to the voting server that performs several validations and if all of them are successful, stores the vote both in a private ballot box and in the bulletin board and informs the voter that the vote was successfully cast. Meanwhile, the voter introduces the trapdoor key into the voting device to generate fake proofs for the voting options that were not selected. This proof can be used in case of coercion or vote buying. Finally, if the response received from the voting server is

successful, the voter can check that the audited vote has been published in the bulletin board.

The voting phase consists of the following algorithms:

- $\mathsf{CreateVote}((a_{\mathsf{E}}, b_{\mathsf{E}}), v_i, ek_{\mathsf{id}})$: this algorithm receives as input the election public key $(a_{\mathsf{E}}, b_{\mathsf{E}})$, the voting option selected v_i and the voter's evaluation key ek_{id}. It encodes the voting option v_i as z and runs the Enc algorithm from the RLWE encryption scheme. The output is the ciphertext (u, v). Then, it re-writes the ciphertext following Equation 5.1 ($\mathbf{y} = \tilde{\mathbf{A}}\mathbf{x}$) and runs $\mathsf{NIZKProve}$ algorithm from the simulatable NIZKPoK scheme using as inputs the evaluation key ek_{id}, the statement $\mathbf{y}$ and the encryption randomness $\mathbf{x}$. The algorithm outputs the ciphertext (u, v) and its hash h, and the proof π_{CH}

- $\mathsf{AuditVote}((u, v), v_i, \pi_{\mathsf{CH}}, h, ek_{\mathsf{id}})$: this algorithm receives as input the ciphertext (u, v), its hash h, the selected voting option v_i, the simulatable NIZK proof π_{CH} and the evaluation key ek_{id}. First, it checks that h corresponds to the hash of (u, v). Then, it encodes v_i as z and computes the statement $\mathbf{y}$ from (u, v) and z. Finally, it runs the $\mathsf{NIZKVerify}$ algorithm from the simulatable NIZKPoK scheme with inputs the evaluation key ek_{id}, the proof π_{CH} and the statement $\mathbf{y}$. If the verification of the hash and the proof are successful, it outputs 1, 0 otherwise.

- $\mathsf{FakeProof}((u^*, v^*), v_j^*, tk_{\mathsf{id}}, ek_{\mathsf{id}})$: given as inputs the trapdoor key tk_{id}, the evaluation key ek_{id} and the new statement for which a proof wants to be generated, this algorithm runs the $\mathsf{NIZKSimulate}$ algorithm from the simulatable NIZKPoK scheme. The output is the simulated proof π^*_{CH}.

- $\mathsf{CastVote}(\mathsf{id}, (u, v), sk_{s,\mathsf{id}})$: on inputs the voter's identity id, the ciphertext (u, v) and the voter's signing key $sk_{s,\mathsf{id}}$, this algorithm runs the Sign algorithm in order to sign the ciphertext together with the voter's identity using the signing key. The output is the authenticated ballot which consists of the signature σ, the ciphertext and the voter's identity: $b_a = (\mathsf{id}, (u, v), \sigma)$.

- $\mathsf{ProcessBallot}(\mathsf{BB}, b_a)$: this algorithm receives as input the authenticated ballot $b_a = (\mathsf{id}, (u, v), \sigma)$ and performs the following validations: it checks that the voter has not cast a vote yet, i.e., that there is no entry in BB for the identity id, or that there is no entry for the same ciphertext (u, v). Then, it checks that the voter is authorized to vote in the election, i.e., the id is in the list ID. Finally, it takes the voter's verification key $pk_{s,\mathsf{id}}$ from the bulletin board and verifies the signature by calling to the $\mathsf{SignVerify}$ algorithm of the digital signature scheme. If all validations are successful, the algorithm outputs 1, 0 otherwise.

- $\mathsf{VerifyVote}(\mathsf{BB}, b_a, \mathsf{id})$: this algorithm checks if there is an entry in the bulletin board for the identity id and, in case there is, it retrieves the stored hash h' and checks that it corresponds to the hash of the authenticated ballot $h' = H(b_a)$. It outputs 1 if the verification is successful, or 0 otherwise.

Once we have defined the algorithms to be executed during the voting phase, we need to know by whom they are run and in which order. This sequence is depicted in Figure 5.2 and detailed below:

1. The voter receives the list of voting options they are authorized to vote for, selects one of them v_i and sends it to the voting device together with their identity id.

2. The voting device retrieves the election public key $(a_{\mathsf{E}}, b_{\mathsf{E}})$ from the bulletin board and uses the voter's identity id to gather the corresponding evaluation key ek_{id}. It then runs the CreateVote algorithm which outputs the ciphertext (u, v) and the proof π_{CH}.

3. The voting device computes the hash of the ciphertext $h = H(u, v)$ and provides (u, v), h and the proof π_{CH} to the voter.

4. The voter, using an audit device, runs the AuditVote algorithm sending as input the ciphertext (u, v), its hash h and the proof π_{CH} provided by the voting device, the voting option selected v_i and the evaluation key ek_{id}. If the output of the algorithm is 1, the voter is convinced that the voting device is not cheating since it has encrypted the voting option selected. Otherwise, the voting device is corrupted and the voter is instructed to use another device to cast their vote.

5. If the execution of the AuditVote algorithm is successful, the voter introduces their signing key $sk_{s,\mathsf{id}}$ into the voting device that runs the CastVote algorithm and obtains the authenticated ballot b_a.

6. The voting device sends the authenticated ballot b_a to the voting server.

7. The voting device asks the voter to introduce their trapdoor key tk_{id} and generate a fake proofs for the options the voter has not selected by calling to the FakeProof algorithm. The simulated proofs $\{\pi^*_{\mathsf{CH}}\}^{\psi}_{j=1,j\neq i}$ are provided to the voter in case they need to show them to a possible vote buyer or coercer. It is important that the voting device does not learn the trapdoor key until the valid proof is generated since, otherwise, it could use it to generate a fake proof for the voter and convince them that the ciphertext encrypts the voting option selected when, indeed, it is encrypting another one.

8. Once the voting server receives the authenticated ballot b_a it runs the ProcessBallot algorithm. If the output is 0 the process is stopped and the voter is informed that something went wrong. Otherwise, the voting server computes a hash of the authentication ballot and posts it in the bulletin board. It also stores b_a in the private ballot box.

9. If the output of the previous step is successful, the voter can run the VerifyVote algorithm to check that their vote has been published in the bulletin board.

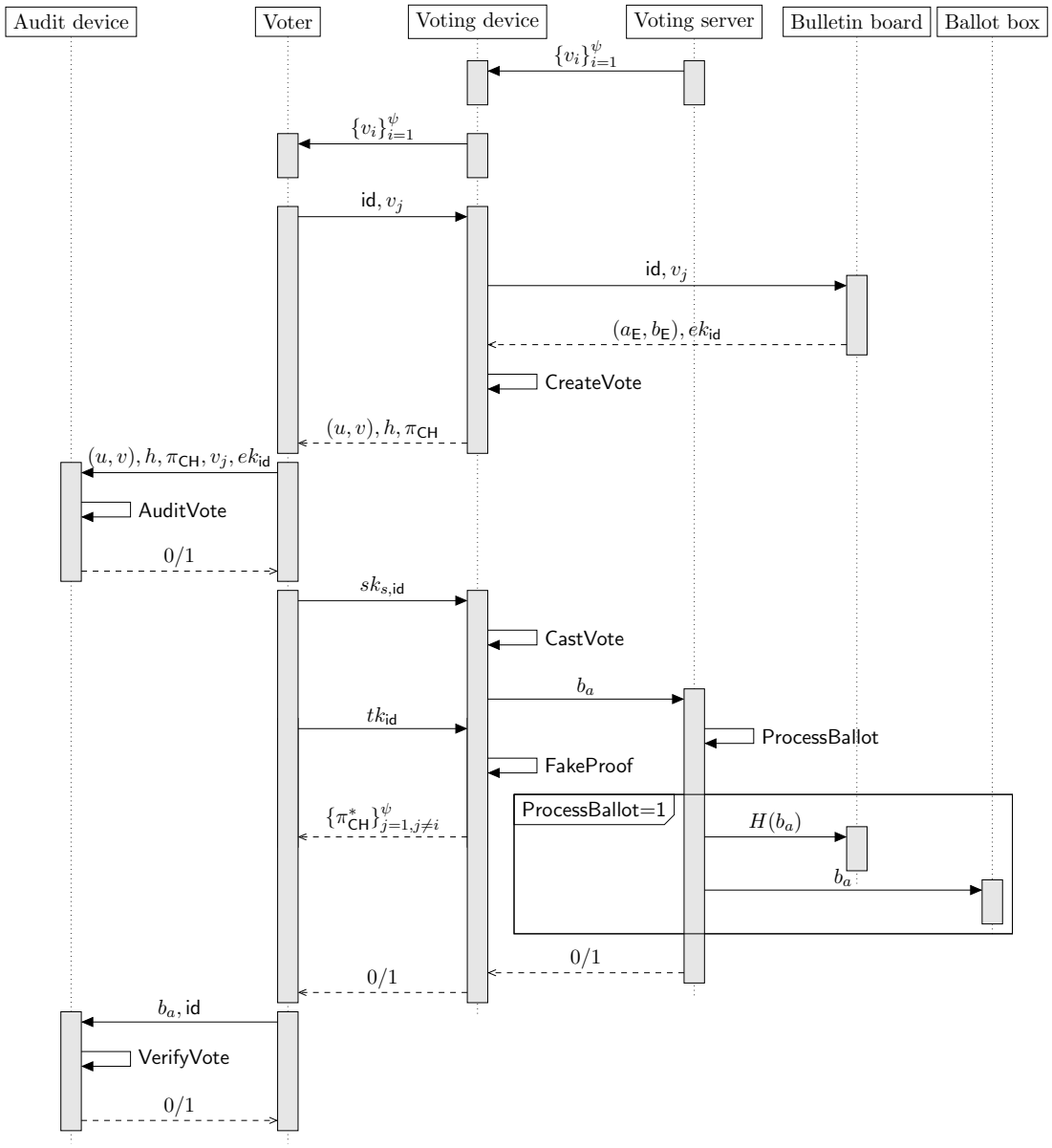


Figure 5.2: Overview of the interaction among the online voting system participants during the voting phase.

5.3.3 Counting phase

During the counting phase the votes cast by eligible voters are decrypted and the results are published. More concretely, the electoral authority obtains the encrypted votes from the ballot box, validates them and execute a mixing process in order to anonymize them. Then, if the election private key has been split during the configuration phase at this point is reconstructed and finally votes are decrypted and tallied. It is recommended that this phase is done offline, i.e., in machines not connected to the internet, due to the criticality of the operations that are going to be executed and the secrecy of the elements that are going to be used, such as the election private key. Nevertheless, in real elections, this is not always possible since it requires additional steps and infrastructure which increases the cost and decreases the usability of the system.

Both the results of the decryption and the proof of correct mixing are published in the bulletin board, so the auditors of the election can verify that the process was done correctly. The question that arises at this point is, how the auditors can check that the decryption operation has not modified any of the ciphertexts, i.e., that the decrypted votes correspond to the encrypted votes at the output of the mixing? One of the possible solutions is to generate decryption proofs as it is done in other online voting systems such as the iVote system [33] or Neuchâtel's [71]. Nevertheless, more research has to be done on this field in order to find or to come up with a decryption proof for the RLWE encryption scheme.

The counting phase consists on the following algorithms (note that we decompose the Tally algorithm mentioned in Section 2.3.4 into the first three algorithms):

- Cleansing(BB,bb): this algorithm performs several validations over each vote stored in the ballot box: it checks that the voter identity id is present in the list ID, that there is only one entry per id $(\mathsf{id}, (u, v), \sigma)$ and that the hash of (u, v) corresponds to that published in the bulletin board. Then, it picks the corresponding voter's verification key $pk_{s,\mathsf{id}}$ from the bulletin board and verifies the signature σ by calling to the SignVerify algorithm of the digital signature scheme. If all validations are successful the ciphertext (u, v) is included in the list of cleansed votes $\mathsf{bb_C}$, which is given as the output of the algorithm .

- Mixing(BB,$\mathsf{bb_C}$): this algorithm anonymizes the votes that have successfully passed all the validations done during the cleansing. It executes the MixVotes algorithm of the mixing protocol sending as input the election public key and the commitment key, both retrieved from the bulletin board, and the list of cleansed votes $\mathsf{bb_C}$. The output is the list of mixed votes $\mathsf{bb_M}$ and the shuffle proof $\Sigma_{mix} = \{\Sigma_l\}_{l=0}^{4}$.

- Decryption($\mathsf{bb_M}$,sk_e): this algorithm executes the Dec algorithm of the RLWE encryption scheme for each vote in the list of mixed votes $\mathsf{bb_M}$, sending as input the election private key sk_e. The output is the list of decrypted votes $\mathsf{bb_D}$.

- VerifyTally(BB,bb): this algorithm verifies the cleansing and mixing processes. It first repeats all the validations done by the Cleansing algorithm and checks that the list of votes that have successfully passed all the validations contains the same votes as those included in $\mathsf{bb_C}$. Then, it calls to the VerifyMix algorithms of the mixing protocol with inputs the election public key, the commitment key, the list of cleansed votes $\mathsf{bb_C}$, the list of mixed votes $\mathsf{bb_M}$ and the shuffle proof Σ_{mix}. The output is 1 if the validations are successful, 0 otherwise.

Finally, the steps executed during the counting phase, shown in Figure 5.3 are the following ones:

1. The electoral authority runs the Cleansing algorithm on the ballot box bb and publishes the list of cleansed votes $\mathsf{bb_C}$ in the bulletin board.

2. Once the cleansing is done, the electoral authority runs the Mixing algorithm sending as input the list of cleansed votes and once the process is finished, they publish the output of the mixing ($\mathsf{bb_M}$ and Σ_{mix}) in the bulletin board.

3. The electoral authority uses the election private key to run the Decryption algorithm and obtain the list of decrypted votes $\mathsf{bb_D}$, which is also published in the bulletin board.

4. Finally, the auditors obtain the ballot box bb from the voting server and run the VerifyTally algorithm to detect any problem during the execution of the cleansing and the mixing processes.

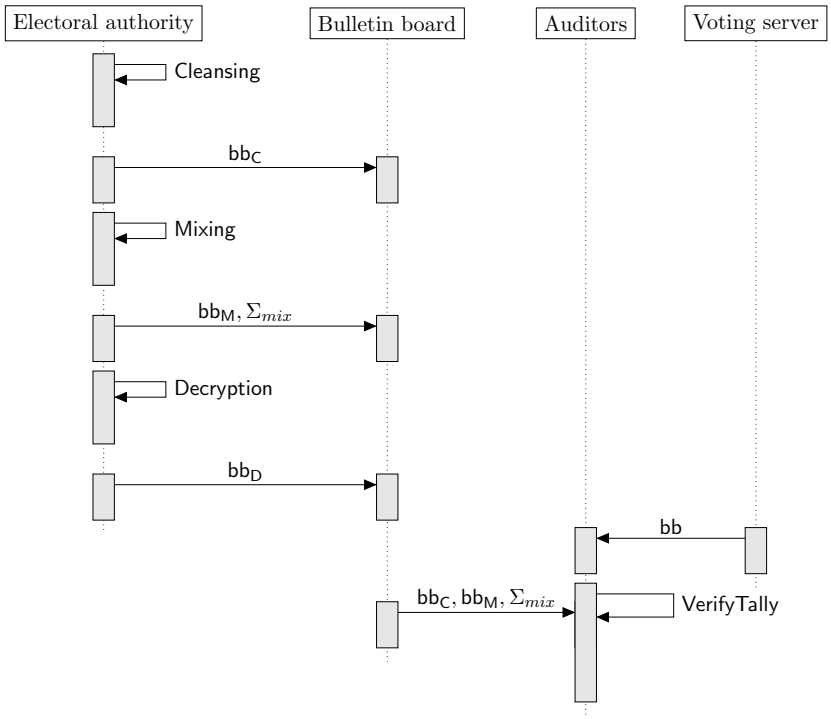


Figure 5.3: Overview of the interaction among the online voting system participants during the counting phase.

Taking as a reference the security requirements defined in Section 2.3.1 we informally analyze which of them are satisfied by the online voting system. Votes are encrypted in the voting device and they are not decrypted until the decryption phase using a private key which is protected by the electoral authority. As long as the encryption scheme is secure and the electoral authority is trusted, *vote confidentiality* and *election fairness* are ensured. In addition, before decrypting the votes they are anonymized using a mixing protocol, which ensures *vote anonymity* provided that at least one of the mix-nodes is honest and does not reveal its secret permutation or re-encryption parameters.

Vote authenticity and integrity are guaranteed by means of the signature and the validations done by both the voting server and the cleansing process. Votes are signed with a private key that is only known by the corresponding voter. If the encrypted vote is modified or an attacker tries to send a vote on behalf of a voter,

the signature verification will fail either during the execution of the ProcessBallot or the Cleansing algorithm.

The system provides the voter with a proof to check that the content of their votes is correct thus ensuring *cast-as-intended verifiablity*. This mechanism is coercion-resistant, i.e., it does not give to the voter any information that allows them to demonstrate how they voted, so *receipt-freeness* is also ensured. Voters can also check that their votes were *recorded-as-cast* by verifying that the hash published in the bulletin board corresponds to the hash of the audited vote. Then, we can conclude that the system provides *individual verifiability*. Finally, since the decryption process does not generate any proof to allow anyone to check that the operation was done successfully, we cannot say that the system is *end-to-end verifiable* since it does not provide *counted-as-recorded* verifiability.

5.4 Future work

In this chapter, we have shown that it is possible to build a post-quantum online voting system satisfying most of the security requirements presented in Section 2.3.1. Nevertheless, there is still a lot of work that can be done in order to both improve and evolve the system.

In terms of the **security** of the protocol, it is necessary to provide a security analysis in order to demonstrate that the system satisfies the following properties: ballot privacy, strong correctness, cast-as-intended , coercion-resistant cast-as-intended and recorded-as-cast. If we want also to demonstrate counted-as-recorded verifiability and consequently universal verifiability, we need first to define a lattice-based **decryption proof** or to use some technique that allows anyone to verify the correctness of the decryption process.

Thinking also on giving more functionalities to the system, it will be useful not only to demonstrate to the voter that the vote contains the selected voting options, but also to universally demonstrate the **correctness of the vote**, i.e., that the vote is well-formed and that the voting options encrypted follow the election rules. On the other hand, the security of the system would be improved if instead of requiring the electoral authority to reconstruct the private key in order to decrypt the votes, we use a **threshold decryption scheme**. The idea is that each electoral authority member owns a share of the private key and during the decryption phase they compute a decryption share, i.e., a partial decryption, for each ciphertext using the corresponding private key share. Once there is a threshold number of decryption shares, they are combined in order to decrypt the ciphertexts.

Finally, an essential step for analyzing the performance of the system and decide if it will be efficient enough for being used in a real election, is to implement it. This **implementation** is a work in progress that is being conducted as part of the European Union PROMETHEUS project and will allow us to identify which parts of the system can be improved in terms of performance.

Chapter 6

Conclusions

This work has focused on lattice-based cryptography and how to apply it to build post-quantum online voting systems. We can distinguish three main parts of the research done in the framework of this thesis.

First, after analyzing how we can contribute to the state of the art on online voting systems, we have studied lattice theory, starting with the basics and ending with the existing lattice-based cryptosystems. This has allowed us to identify how to approach our research on the field of lattice-based constructions.

Then, we have proposed three protocols that can be used as building blocks of an online voting system: a lattice-based coercion-resistant cast-as-intended protocol, a post-quantum mix-net, and a fully post-quantum proof of a shuffle. The former is the lattice version of an existing protocol and allows the voter to check that the vote cast contains the selected voting options. The second and third protocols are the result of our research on lattice-based mix-nets. Since, to the best of our knowledge, there were no proposals for a proof of a shuffle in lattice-based cryptography, we proposed two constructions. The first one allows to demonstrate that a mix-node has permuted and re-encrypted a list of RLWE ciphertexts without modifying them, but it cannot be considered fully post-quantum since the binding property of the commitment scheme relies on classical computational problems. The second one is fully post-quantum since all the cryptographic schemes used for building it, i.e., commitment scheme and zero-knowledge proofs, are based on lattices. Last but not least, for this second proposal we provide a security definition and a proof of security. The definition is based on that proposed by Wikström in [150], but we modify it in order to allow the input of the mix-node to come from a possibly malicious previous node. Consequently, the security definition we present is stronger. We demonstrate that our mix-node is secure according to that definition under the RLWE hardness assumption.

Finally, we use our previous research and also existing lattice-based constructions to build a post-quantum online voting system. We describe in detail which are the main algorithms involved in each phase, and we discuss which are the security requirements fulfilled by the system.

It has not been possible to implement the post-quantum online voting system as part of this thesis due to time and resource constraints. Nevertheless, there is an on-going implementation of a system based on ours in the context of the European

Union PROMETHEUS project, which is still in a preliminary stage but aims to finish at the end of next year. The author is participating on writing the protocol specification as well as giving support to the developers in the implementation.

This PhD thesis ends here, but we leave several doors open for future research. In our opinion, the first one would be to make the decryption process verifiable, thus providing universal verifiability to the post-quantum online voting system. Then, it would also be necessary to provide a formal analysis of the security of the system. Regarding the performance of the system, probably some of the constructions such as the proof of a shuffle can be improved in terms of efficiency by using new proposals which have emerged in the last years. Nevertheless, a better analysis can be done when the implementation finishes.

Another pending topic which we have not studied as part of this thesis is how to show the security of our protocols in the Quantum Random Oracle Model (QROM). We know that this is an open point for several post-quantum proposals in the literature which use the Fiat-Shamir framework and, although there are some articles in which it is demonstrated that under certain conditions Fiat-Shamir implies security in the QROM, further investigation should be done in order to demonstrate that our constructions are secure in the QROM.

The last conclusion we want to share, which is also a lesson learned after some years working in a company specialized in secure electronic voting solutions, is that cooperation between academia and industry is crucial for implementing real-world online voting systems while achieving strong security guarantees.

Bibliography

[1] Introducing electronic voting: Essential considerations. https://www.idea.int/sites/default/files/publications/introducing-electronic-voting.pdf (last accessed on 23/08/20)

[2] Wombat voting system (2015). https://wombat.factcenter.org/ (last accessed on 01/09/20)

[3] Council of Europe, Committee of Ministers, Recommendation CM/Rec(2017)5 of the Committee of Ministers to member States on standards for e-voting (Adopted by the Committee of Ministers on 14 june 2017 at the 1289th meeting of the Ministers' Deputies) (2017). https://rm.coe.int/0900001680726f6f (last accessed on 04/09/20)

[4] Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In: K. Nyberg (ed.) EUROCRYPT'98, *LNCS*, vol. 1403, pp. 437–447. Springer, Heidelberg (1998). doi:10.1007/BFb0054144

[5] Abe, M.: Mix-networks on permutation networks. In: K.Y. Lam, E. Okamoto, C. Xing (eds.) ASIACRYPT'99, *LNCS*, vol. 1716, pp. 258–273. Springer, Heidelberg (1999). doi:10.1007/978-3-540-48000-6_21

[6] Abe, M., Hoshino, F.: Remarks on mix-network based on permutation networks. In: K. Kim (ed.) PKC 2001, *LNCS*, vol. 1992, pp. 317–324. Springer, Heidelberg (2001). doi:10.1007/3-540-44586-2_23

[7] Adida, B.: Advances in cryptographic voting systems. Ph.D. thesis, USA (2006)

[8] Adida, B.: Helios: Web-based open-audit voting. In: P.C. van Oorschot (ed.) USENIX Security 2008, pp. 335–348. USENIX Association (2008)

[9] Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.: Electing a university president using open-audit voting: Analysis of real-world use of helios. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections, EVT/WOTE'09, p. 10. USENIX Association, USA (2009)

[10] Adida, B., Wikström, D.: How to shuffle in public. In: S.P. Vadhan (ed.) TCC 2007, *LNCS*, vol. 4392, pp. 555–574. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70936-7_30

[11] Adida, B., Wikström, D.: Offline/online mixing. In: L. Arge, C. Cachin, T. Jurdzinski, A. Tarlecki (eds.) ICALP 2007, *LNCS*, vol. 4596, pp. 484–495. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73420-8_43

[12] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: 28th ACM STOC, pp. 99–108. ACM Press (1996). doi:10.1145/237814.237838

[13] Ajtai, M.: Generating hard instances of the short basis problem. In: J. Wiedermann, P. van Emde Boas, M. Nielsen (eds.) ICALP 99, *LNCS*, vol. 1644, pp. 1–9. Springer, Heidelberg (1999). doi:10.1007/3-540-48523-6_1

[14] Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: 29th ACM STOC, pp. 284–293. ACM Press (1997). doi:10.1145/258533.258604

[15] Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Cryptology ePrint Archive, Report 2008/521 (2008). `http://eprint.iacr.org/2008/521`

[16] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: S. Halevi (ed.) CRYPTO 2009, *LNCS*, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_35

[17] Baignères, T.: Provable security in cryptography (2007)

[18] Barrat, J., Goldsmith, B., Turner, J.: International experience with e-voting. Tech. rep., International Foundation For Electoral Systems (2012)

[19] Baum, C., Damgård, I., Larsen, K., Nielsen, M.: How to prove knowledge of small secrets. Cryptology ePrint Archive, Report 2016/538 (2016). `http://eprint.iacr.org/2016/538`

[20] Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: D. Catalano, R. De Prisco (eds.) SCN 18, *LNCS*, vol. 11035, pp. 368–385. Springer, Heidelberg (2018). doi:10.1007/978-3-319-98113-0_20

[21] Baum, C., Lyubashevsky, V.: Simple amortized proofs of shortness for linear relations over polynomial rings. Cryptology ePrint Archive, Report 2017/759 (2017). `http://eprint.iacr.org/2017/759`

[22] Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: D. Pointcheval, T. Johansson (eds.) EUROCRYPT 2012, *LNCS*, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_17

[23] Bell, S., Benaloh, J., Byrne, M., Debeauvoir, D., Eakin, B., Kortum, P., McBurnett, N., Pereira, O., Stark, P., Wallach, D., Fisher, G., Montoya, J., Parker, M., Winn, M.: Star-vote: A secure, transparent, auditable, and reliable voting system. In: 2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13). USENIX Association, Washington, D.C. (2013). URL `https://www.usenix.org/conference/evtwote13/workshop-program/presentation/bell`

[24] Bellare, M., Garay, J.A., Rabin, T.: Batch verification with applications to cryptography and checking. In: C.L. Lucchesi, A.V. Moura (eds.) LATIN 1998, *LNCS*, vol. 1380, pp. 170–191. Springer, Heidelberg (1998)

[25] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: D.E. Denning, R. Pyle, R. Ganesan, R.S. Sandhu, V. Ashby (eds.) ACM CCS 93, pp. 62–73. ACM Press (1993). doi:10.1145/168588.168596

[26] Ben-Nun, J., Farhi, N., Llewellyn, M., Riva, B., Rosen, A., Ta-Shma, A., Wikström, D.: A new implementation of a dual (paper and cryptographic) voting system. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) pp. 315–329 (2012)

[27] Benaloh, J.: Verifiable secret-ballot elections. Ph.D. thesis, USA (1987)

[28] Benaloh, J.: Simple verifiable elections. In: EVT'06. Proceeding of the USENIX/accurate electronic voting technology workshop. Berkeley CA, USA: USENIX Association (2016)

[29] Benhamouda, F., Krenn, S., Lyubashevsky, V., Pietrzak, K.: Efficient zero-knowledge proofs for commitments from learning with errors over rings. In: G. Pernul, P.Y.A. Ryan, E.R. Weippl (eds.) ESORICS 2015, Part I, *LNCS*, vol. 9326, pp. 305–325. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24174-6_16

[30] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: M. Franklin (ed.) CRYPTO 2004, *LNCS*, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_3

[31] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634 (2017). `http://eprint.iacr.org/2017/634`

[32] Boyen, X., Haines, T., Müller, J.: A verifiable and practical lattice-based decryption mix net with external auditing. In: L. Chen, N. Li, K. Liang, S. Schneider (eds.) Computer Security – ESORICS 2020, pp. 336–356. Springer International Publishing (2020)

[33] Brightwell, I., Cucurull, J., Galindo, D., Guasch, S.: An overview of the ivote 2015 voting system. `https://www.elections.nsw.gov.au/about-us/reports/ivote-reports` (2015)

[34] Budurushi, J., Neumann, S., Olembo, M., Volkamer, M.: Pretty understandable democracy - a secure and understandable internet voting scheme. pp. 198–207 (2013). doi:10.1109/ARES.2013.27

[35] Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, pp. 315–334. IEEE Computer Society Press (2018). doi:10.1109/SP.2018.00020

[36] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press (2001). doi:10.1109/SFCS.2001.959888

[37] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. Cryptology ePrint Archive, Report 2010/591 (2010). `http://eprint.iacr.org/2010/591`

[38] Chancellery, S.F.: Federal Chancellery Ordinance on Electronic Voting (VEleS) (2018). `https://www.bk.admin.ch/dam/bk/de/dokumente/pore/Federal_Chancellery_Ordinance_on_Electronic_Voting_V2.0_July_2018.pdf.download.pdf/Federal_Chancellery_Ordinance_on_Electronic_Voting_V2.0_July_2018.pdf` (last accessed on 01/09/20)

[39] Chancellery, S.F.: Technical and administrative requirements for electronic vote casting (2018). `https://www.bk.admin.ch/dam/bk/de/dokumente/pore/Annex_of_the_Federal_Chancellery_Ordinance_on_Electronic_Voting_V2.0_July_2018.pdf` (last accessed on 01/09/20)

[40] Chaum, D.: Blind signatures for untraceable payments. In: D. Chaum, R.L. Rivest, A.T. Sherman (eds.) CRYPTO'82, pp. 199–203. Plenum Press, New York, USA (1982)

[41] Chaum, D.: Surevote: Technical overview. In: Proceedings of the Workshop on Trustworthy Elections (WOTE '01) (2001)

[42] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)

[43] Chen, L., Jordan, S., Liu, Y., Moody, D., Peralta, R., Perlner, R., Smith-Tone, D.: Report on post-quantum cryptography. Tech. rep., National Institute of Standards and Technology (2016)

[44] Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: A homomorphic LWE based E-voting scheme. In: T. Takagi (ed.) Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, pp. 245–265. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29360-8_16

[45] Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme (extended abstract). In: 26th FOCS, pp. 372–382. IEEE Computer Society Press (1985). doi:10.1109/SFCS.1985.2

[46] Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Distributed Elgamal à La Pedersen: Application to helios. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, WPES '13, p. 131–142. Association for Computing Machinery, New York, NY, USA (2013). URL `https://doi.org/10.1145/2517840.2517852`

[47] Costa, N., Martínez, R., Morillo, P.: Proof of a shuffle for lattice-based cryptography (full version). Cryptology ePrint Archive, Report 2017/900 (2017). `http://eprint.iacr.org/2017/900`

[48] Costa, N., Martínez, R., Morillo, P.: Lattice-based proof of a shuffle. In: A. Bracciali, J. Clark, F. Pintore, P.B. Rønne, M. Sala (eds.) FC 2019 Workshops, *LNCS*, vol. 11599, pp. 330–346. Springer, Heidelberg (2019). doi:10.1007/978-3-030-43725-1_23

[49] Council of Europe: Legal, operational and technical standards for e-voting. Recommendation Rec(2004)11 and explanatory memorandum (2004)

[50] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Y. Desmedt (ed.) CRYPTO'94, *LNCS*, vol. 839, pp. 174–187. Springer, Heidelberg (1994). doi:10.1007/3-540-48658-5_19

[51] Cramer, R., Damgård, I., Xing, C., Yuan, C.: Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. In: J. Coron, J.B. Nielsen (eds.) EUROCRYPT 2017, Part I, *LNCS*, vol. 10210, pp. 479–500. Springer, Heidelberg (2017). doi:10.1007/978-3-319-56620-7_17

[52] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: W. Fumy (ed.) EUROCRYPT'97, *LNCS*, vol. 1233, pp. 103–118. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_9

[53] Damgård, I.: On sigma protocols (2010). `https://www.cs.au.dk/~ivan/Sigma.pdf` (last accessed on 06/11/20)

[54] D'Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: SABER. Tech. rep., National Institute of Standards and Technology (2019). Available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`

[55] del Pino, R., Lyubashevsky, V.: Amortization with fewer equations for proving knowledge of small secrets. In: J. Katz, H. Shacham (eds.) CRYPTO 2017, Part III, *LNCS*, vol. 10403, pp. 365–394. Springer, Heidelberg (2017). doi:10.1007/978-3-319-63697-9_13

[56] del Pino, R., Lyubashevsky, V., Neven, G., Seiler, G.: Practical quantum-safe voting from lattices. In: B.M. Thuraisingham, D. Evans, T. Malkin, D. Xu (eds.) ACM CCS 2017, pp. 1565–1581. ACM Press (2017). doi:10.1145/3133956.3134101

[57] Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: G. Brassard (ed.) CRYPTO'89, *LNCS*, vol. 435, pp. 307–315. Springer, Heidelberg (1990). doi:10.1007/0-387-34805-0_28

[58] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976)

[59] Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC, pp. 542–552. ACM Press (1991). doi:10.1145/103418.103474

[60] Driza Maurer, A.: Updated European standards for e-voting. The Council of Europe recommendation Rec(2017)5 on standards for e-voting (2017)

[61] Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633 (2017). `http://eprint.iacr.org/2017/633`

[62] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: G.R. Blakley, D. Chaum (eds.) CRYPTO'84, *LNCS*, vol. 196, pp. 10–18. Springer, Heidelberg (1984)

[63] Fauzi, P., Lipmaa, H.: Efficient culpably sound NIZK shuffle argument without random oracles. In: K. Sako (ed.) CT-RSA 2016, *LNCS*, vol. 9610, pp. 200–216. Springer, Heidelberg (2016). doi:10.1007/978-3-319-29485-8_12

[64] Fauzi, P., Lipmaa, H., Siim, J., Zajac, M.: An efficient pairing-based shuffle argument. In: T. Takagi, T. Peyrin (eds.) ASIACRYPT 2017, Part II, *LNCS*, vol. 10625, pp. 97–127. Springer, Heidelberg (2017). doi:10.1007/978-3-319-70697-9_4

[65] Fauzi, P., Lipmaa, H., Zajac, M.: A shuffle argument secure in the generic model. In: J.H. Cheon, T. Takagi (eds.) ASIACRYPT 2016, Part II, *LNCS*, vol. 10032, pp. 841–872. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_28

[66] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: A.M. Odlyzko (ed.) CRYPTO'86, *LNCS*, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12

[67] Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-fourier lattice-based compact signatures over NTRU (2019)

[68] Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: J. Seberry, Y. Zheng (eds.) AUSCRYPT'92, *LNCS*, vol. 718, pp. 244–251. Springer, Heidelberg (1993). doi:10.1007/3-540-57220-1_66

[69] Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **88-A**, 172–188 (2005)

[70] Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: J. Kilian (ed.) CRYPTO 2001, *LNCS*, vol. 2139, pp. 368–387. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_22

[71] Galindo, D., Guasch, S., Puiggalí, J.: 2015 neuchâtel's cast-as-intended verification mechanism. In: Proceedings of the 5th International Conference on E-Voting and Identity - Volume 9269, VoteID 2015, p. 3–18. Springer-Verlag, Berlin, Heidelberg (2015). URL `https://doi.org/10.1007/978-3-319-22270-7_1`

[72] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: R.E. Ladner, C. Dwork (eds.) 40th ACM STOC, pp. 197–206. ACM Press (2008). doi:10.1145/1374376.1374407

[73] Gerlach, J., Gasser, U.: Three Case Studies from Switzerland: E-Voting. `https://cyber.harvard.edu/sites/cyber.harvard.edu/files/Gerlach-Gasser_SwissCases_Evoting.pdf` (2009)

[74] Gharadaghy, R., Volkamer, M.: Verifiability in electronic voting - explanations for non security experts. In: EVOTE'10, no. 167 in Springer, LNI, pp. 151–162 (2010)

[75] Gibson, J., Krimmer, R., Teague, V., Pomares, J.: A review of e-voting: the past, present and future. Annals of Telecommunications **71**, 279–286 (2016). URL `https://doi.org/10.1007/s12243-016-0525-8`

[76] Gjøsteen, K.: The norwegian internet voting protocol. Cryptology ePrint Archive, Report 2013/473 (2013). `http://eprint.iacr.org/2013/473`

[77] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: B.S. Kaliski Jr. (ed.) CRYPTO'97, *LNCS*, vol. 1294, pp. 112–131. Springer, Heidelberg (1997). doi:10.1007/BFb0052231

[78] Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information. In: 14th ACM STOC, pp. 365–377. ACM Press (1982). doi:10.1145/800070.802212

[79] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)

[80] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1), 186–208 (1989)

[81] Golle, P., Jakobsson, M., Juels, A., Syverson, P.F.: Universal re-encryption for mixnets. In: T. Okamoto (ed.) CT-RSA 2004, *LNCS*, vol. 2964, pp. 163–178. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24660-2_14

[82] Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: Y. Zheng (ed.) ASIACRYPT 2002, *LNCS*, vol. 2501, pp. 451–465. Springer, Heidelberg (2002). doi:10.1007/3-540-36178-2_28

[83] Grewal, G., Ryan, M., Chen, L., Clarkson, M.: Du-vote: Remote electronic voting with untrusted computers. In: 2015 IEEE 28th Computer Security Foundations Symposium, pp. 155–169 (2015). doi:10.1109/CSF.2015.18

[84] Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Y. Desmedt (ed.) PKC 2003, *LNCS*, vol. 2567, pp. 145–160. Springer, Heidelberg (2003). doi:10.1007/3-540-36288-6_11

[85] Groth, J.: Non-interactive zero-knowledge arguments for voting. In: J. Ioannidis, A. Keromytis, M. Yung (eds.) ACNS 05, *LNCS*, vol. 3531, pp. 467–482. Springer, Heidelberg (2005). doi:10.1007/11496137_32

[86] Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: N.P. Smart (ed.) EUROCRYPT 2008, *LNCS*, vol. 4965, pp. 379–396. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_22

[87] Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: K. Kurosawa (ed.) ASIACRYPT 2007, *LNCS*, vol. 4833, pp. 51–67. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_4

[88] Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: T. Okamoto, X. Wang (eds.) PKC 2007, *LNCS*, vol. 4450, pp. 377–392. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71677-8_25

[89] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: 28th ACM STOC, pp. 212–219. ACM Press (1996). doi:10.1145/237814.237866

[90] Guasch, S.: Individual verifiability in electronic voting. Ph.D. thesis (2016)

[91] Guasch, S., Morillo, P.: How to challenge and cast your e-vote. In: J. Grossklags, B. Preneel (eds.) FC 2016, *LNCS*, vol. 9603, pp. 130–145. Springer, Heidelberg (2016)

[92] Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing **28**(4), 1364–1396 (1999)

[93] Heiberg, S., Willemson, J.: Verifiable internet voting in estonia. In: 2014 6th International Conference on Electronic Voting: Verifying the Vote (EVOTE), pp. 1–8 (2014)

[94] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: J.P. Buhler (ed.) Algorithmic Number Theory, pp. 267–288. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)

[95] Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: U.M. Maurer (ed.) EUROCRYPT'96, *LNCS*, vol. 1070, pp. 143–154. Springer, Heidelberg (1996). doi:10.1007/3-540-68339-9_13

[96] Katz, J., Lindell, Y.: Introduction to Modern Cryptography (Cryptography and Network Security Series). Chapman and Hall/CRC (2007)

[97] Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: J. Pieprzyk (ed.) ASIACRYPT 2008, *LNCS*, vol. 5350, pp. 372–389. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89255-7_23

[98] Krawczyk, H., Rabin, T.: Chameleon hashing and signatures. Cryptology ePrint Archive, Report 1998/010 (1998). `http://eprint.iacr.org/1998/010`

[99] Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261** (1982). URL `https://doi.org/10.1007/BF01457454`

[100] Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: K. Kurosawa, G. Hanaoka (eds.) PKC 2013, *LNCS*, vol. 7778, pp. 107–124. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_8

[101] Lipmaa, H.: A simple cast-as-intended E-voting protocol by using secure smart cards. Cryptology ePrint Archive, Report 2014/348 (2014). `http://eprint.iacr.org/2014/348`

[102] Lipmaa, H., Zhang, B.: A more efficient computationally sound non-interactive zero-knowledge shuffle argument. In: I. Visconti, R.D. Prisco (eds.) SCN 12, *LNCS*, vol. 7485, pp. 477–502. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32928-9_27

[103] Locher, P., Haenni, R.: A lightweight implementation of a shuffle proof for electronic voting systems. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) pp. 1391–1400 (2014)

[104] Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: R. Cramer (ed.) PKC 2008, *LNCS*, vol. 4939, pp. 162–179. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78440-1_10

[105] Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: M. Matsui (ed.) ASIACRYPT 2009, *LNCS*, vol. 5912, pp. 598–616. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_35

[106] Lyubashevsky, V.: Lattice signatures without trapdoors. Cryptology ePrint Archive, Report 2011/537 (2011). http://eprint.iacr.org/2011/537

[107] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Cryptology ePrint Archive, Report 2012/230 (2012). http://eprint.iacr.org/2012/230

[108] Malkhi, D., Margo, O., Pavlov, E.: E-voting without "cryptography". In: Proceedings of the 6th International Conference on Financial Cryptography, FC'02, p. 1–15. Springer-Verlag, Berlin, Heidelberg (2002)

[109] Markus, J., Ari, J.: Millimix: Mixing in small batches. Tech. rep. (1999)

[110] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton, Florida (1996)

[111] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: D. Pointcheval, T. Johansson (eds.) EUROCRYPT 2012, *LNCS*, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_41

[112] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS, pp. 372–381. IEEE Computer Society Press (2004). doi:10.1109/FOCS.2004.72

[113] Micciancio, D., Regev, O.: Lattice-based Cryptography, pp. 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). URL https://doi.org/10.1007/978-3-540-88702-7_5

[114] Michels, M., Horster, P.: Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In: K. Kim, T. Matsumoto (eds.) ASIACRYPT'96, *LNCS*, vol. 1163, pp. 125–132. Springer, Heidelberg (1996). doi:10.1007/BFb0034841

[115] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC, pp. 427–437. ACM Press (1990). doi:10.1145/100216.100273

[116] Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: M.K. Reiter, P. Samarati (eds.) ACM CCS 2001, pp. 116–125. ACM Press (2001). doi:10.1145/501983.502000

[117] Neff, C.A.: Verifiable mixing (shuffling) of ElGamal pairs. VoteHere, Inc. (2003)

[118] Olembo, M., Schmidt, P., Volkamer, M.: Introducing verifiability in the POLYAS remote electronic voting system. In: ARES '11, IEEE Computer Society, pp. 127–134. Washington DC, USA (2011)

[119] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: J. Stern (ed.) EUROCRYPT'99, *LNCS*, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_16

[120] Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: T. Helleseth (ed.) EUROCRYPT'93, *LNCS*, vol. 765, pp. 248–259. Springer, Heidelberg (1994). doi:10.1007/3-540-48285-7_21

[121] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: J. Feigenbaum (ed.) CRYPTO'91, *LNCS*, vol. 576, pp. 129–140. Springer, Heidelberg (1992). doi:10.1007/3-540-46766-1_9

[122] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: M. Mitzenmacher (ed.) 41st ACM STOC, pp. 333–342. ACM Press (2009). doi:10.1145/1536414.1536461

[123] Peikert, C.: A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939 (2015). `http://eprint.iacr.org/2015/939`

[124] Pfitzmann, B.: Breaking an efficient anonymous channel. In: A. De Santis (ed.) Advances in Cryptology — EUROCRYPT'94, pp. 332–340. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)

[125] Pfitzmann, B., Pfitzmann, A.: How to break the direct RSA-implementation of mixes. In: J.J. Quisquater, J. Vandewalle (eds.) EUROCRYPT'89, *LNCS*, vol. 434, pp. 373–381. Springer, Heidelberg (1990). doi:10.1007/3-540-46885-4_37

[126] Puiggalí, J., Guasch, S.: Internet voting system with cast as intended verification. In: A. Kiayias, H. Lipmaa (eds.) E-Voting and Identity, pp. 36–52. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[127] Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: J. Feigenbaum (ed.) CRYPTO'91, *LNCS*, vol. 576, pp. 433–444. Springer, Heidelberg (1992). doi:10.1007/3-540-46766-1_35

[128] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: H.N. Gabow, R. Fagin (eds.) 37th ACM STOC, pp. 84–93. ACM Press (2005). doi:10.1145/1060590.1060603

[129] Regev, O.: The learning with errors problem (invited survey). In: 2010 IEEE 25th Annual Conference on Computational Complexity, pp. 191–204 (2010)

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

[131] Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: B.K. Roy, W. Meier (eds.) FSE 2004, *LNCS*, vol. 3017, pp. 371–388. Springer, Heidelberg (2004). doi:10.1007/978-3-540-25937-4_24

[132] Ryan, P.Y.A., Teague, V.: Pretty good democracy. In: B. Christianson, J.A. Malcolm, V. Matyáš, M. Roe (eds.) Security Protocols XVII, pp. 111–130. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[133] Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: L.C. Guillou, J.J. Quisquater (eds.) EUROCRYPT'95, *LNCS*, vol. 921, pp. 393–403. Springer, Heidelberg (1995). doi:10.1007/3-540-49264-X_32

[134] Sandler, D., Derr, K., Wallach, D.S.: VoteBox: A tamper-evident, verifiable electronic voting system. In: P.C. van Oorschot (ed.) USENIX Security 2008, pp. 349–364. USENIX Association (2008)

[135] Schnorr, C.P.: Efficient identification and signatures for smart cards. In: G. Brassard (ed.) CRYPTO'89, *LNCS*, vol. 435, pp. 239–252. Springer, Heidelberg (1990). doi:10.1007/0-387-34805-0_22

[136] Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (1979)

[137] Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)

[138] Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004). `http://eprint.iacr.org/2004/332`

[139] Simon, D.: Selected applications of LLL in number theory. ISC, pp. 265–282. Springer, Heidelberg (2010). doi:10.1007/978-3-642-02295-1

[140] Singh, K., Rangan, C.P., Banerjee, A.: Lattice based universal re-encryption for mixnet. J. Internet Serv. Inf. Secur. **4**, 1–11 (2014)

[141] Singh, K., Rangan, C.P., Banerjee, A.: Lattice based mix network for location privacy in mobile system. Mobile Information Systems **2015**, 1–9 (2015). doi:10.1155/2015/963628

[142] Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: M. Matsui (ed.) ASIACRYPT 2009, *LNCS*, vol. 5912, pp. 617–635. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_36

[143] Stern, J.: A new identification scheme based on syndrome decoding. In: D.R. Stinson (ed.) CRYPTO'93, *LNCS*, vol. 773, pp. 13–21. Springer, Heidelberg (1994). doi:10.1007/3-540-48329-2_2

[144] Storer, T., Duncan, I.: Two variations to the MCESG pollsterless e-voting scheme. In: Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01, COMPSAC '05, p. 425–430. IEEE Computer Society, USA (2005). URL `https://doi.org/10.1109/COMPSAC.2005.165`

[145] Strand, M.: A verifiable shuffle for the GSW cryptosystem. In: A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, M. Sala (eds.) FC 2018 Workshops, *LNCS*, vol. 10958, pp. 165–180. Springer, Heidelberg (2019). doi:10.1007/978-3-662-58820-8_12

[146] Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: D.J. Bernstein, T. Lange (eds.) AFRICACRYPT 10, *LNCS*, vol. 6055, pp. 100–113. Springer, Heidelberg (2010)

[147] Tsiounis, Y., Yung, M.: On the security of ElGamal based encryption. In: H. Imai, Y. Zheng (eds.) Public Key Cryptography, pp. 117–134. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)

[148] Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: E. Oswald, M. Fischlin (eds.) EUROCRYPT 2015, Part II, *LNCS*, vol. 9057, pp. 755–784. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_25

[149] Unruh, D.: Post-quantum security of Fiat-Shamir. In: T. Takagi, T. Peyrin (eds.) ASIACRYPT 2017, Part I, *LNCS*, vol. 10624, pp. 65–95. Springer, Heidelberg (2017). doi:10.1007/978-3-319-70694-8_3

[150] Wikström, D.: The security of a mix-center based on a semantically secure cryptosystem. In: A. Menezes, P. Sarkar (eds.) INDOCRYPT 2002, *LNCS*, vol. 2551, pp. 368–381. Springer, Heidelberg (2002)

[151] Wikström, D.: A universally composable mix-net. In: M. Naor (ed.) TCC 2004, *LNCS*, vol. 2951, pp. 317–335. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24638-1_18

[152] Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: B.K. Roy (ed.) ASIACRYPT 2005, *LNCS*, vol. 3788, pp. 273–292. Springer, Heidelberg (2005). doi:10.1007/11593447_15

[153] Wikström, D.: A commitment-consistent proof of a shuffle. Cryptology ePrint Archive, Report 2011/168 (2011). `http://eprint.iacr.org/2011/168`

[154] Xie, X., Xue, R., Wang, M.: Zero knowledge proofs from ring-LWE. In: M. Abdalla, C. Nita-Rotaru, R. Dahab (eds.) CANS 13, *LNCS*, vol. 8257, pp. 57–73. Springer, Heidelberg (2013). doi:10.1007/978-3-319-02937-5_4