University of Cincinnati

Date: 7/13/2022

I, Vonn Kee G Wong, hereby submit this original work as part of the requirements for the degree of Doctor of Philosophy in Mathematical Sciences.

It is entitled:

Post-quantum self-tallying voting protocol

Student's name: **Vonn Kee G Wong**

This work and its defense approved by:

Committee chair: Jintai Ding, Ph.D.

Committee member: Robert Buckingham, Ph.D.

Committee member: Seungki Kim, Ph.D.

UNIVERSITY OF Cincinnati

43035

Post-quantum self-tallying voting protocol

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Department of Mathematical Sciences
College of Arts and Sciences
University of Cincinnati, July 2022

Author: Vonn Kee Wong
Degrees: B.S. Mathematics, 2010,
University of the Philippines
M.S. Mathematics, 2015,
University of the Philippines

Chair: Jintai Ding, Ph.D.
Committee: Seungki Kim, Ph.D.
Robert Buckingham, Ph.D.

Abstract

We provide a self-tallying voting protocol based on the hardness of decision version of the ring learning with errors problem. We present two versions of the protocol, the first one is passively secure against semi-honest adversaries and the other version is actively secure where we additionally employ $\Sigma'-$proofs which are zero-knowledge proofs. We provide proofs of security for two adversarial models, namely a semi-honest one and a malicious one. First, we show that our protocol is secure against semi-honest adversaries by simulating the view given its local inputs and the public messages. We also show that our voting protocol is secure in the sense of privacy of the honest voters' individual votes against a malicious adversary controlling a coalition of dishonest voters. The idea of our security proof follows the real/ideal world-paradigm where we show that our protocol emulates the ideal-world execution of tallying the votes. Hence any attack in the real-world can be translated to an attack in the ideal-world. However, a successful attack in the ideal-world is impossible which means any attack in the real-world will not be successful as well.

Acknowledgments

First, I would like to thank my advisor Professor Jintai Ding for his guidance, patience and kind support throughout my PhD journey.

I would also like to thank the members of my dissertation committee Professor Robert Buckingham and Professor Seungki Kim for all the helpful comments and suggestions in improving this dissertation.

I would like to extend my gratitude to the Mathematical Sciences Department of the University of Cincinnati for being supportive to make sure that I would be finishing the program successfully.

I would also like to acknowledge my alma mater, the University of the Philippines Diliman, especially the Institute of Mathematics, for which I first experienced what real mathematics is all about and for being my first mathematical home.

Finally, I would like to thank my family, especially my wife for the support and love to which I dedicate this work.

Contents

Chapter 1

Introduction

1.1 Motivation

Suppose a company wants to elect a new set of officers and decides to conduct an electronic-based election where the voting procedure is private, meaning that everyone encrypts their votes, and is self-tallying, that is given all the encrypted votes, we can determine the result of the election without decrypting any of the individual votes and without the help of a trusted third party that is tasked to do the computation of the result. Several papers, such as [Hao and Zieliński 2006], [Groth 2004] proposed solutions to the above problem but both solutions rely on the hardness of the decisional Diffie-Hellman (DDH) problem and in addition, [Groth 2004] also uses a trusted third party as the last voter to ensure fairness of the protocol, that is, if the last voter is not honest, he can determine the partial tally of votes just before he sends his own vote. Protocols relying on the hardness of the DDH problem is susceptible to a quantum algorithm attack, known as the Shor's algorithm [Shor 1999] and the threat of having a large enough quantum computer that implements Shor's algorithm can efficiently break DDH-based protocols. Hence, a post-quantum voting protocol seems to be the natural alternative solution to the voting problem mentioned above. In fact, there are several post-quantum voting

protocols already in the literature, namely based on the worst-case hardness of lattice problems, for example [Pino et al. 2017] and [Aranha et al. 2021]. However, since the above protocols are large-scale voting schemes, a trusted third party or a trusted server is necessary to ensure security of individual votes as well as correctness of the voting result. In this paper, we propose a self-tallying lattice-based voting protocol which relies on the hardness of the decisional Ring Learning with Errors (RLWE) problem. Our protocol is the RLWE analogue of [Hao, Ryan, et al. 2010] and in addition, we employ lattice-based commitment schemes and non-interactive zero knowledge proofs inside our protocol to ensure that voters follow the protocol specification honestly. In fact, the zero-knowledge proof that we will use is a weaker one in the sense that the proving knowledge of the secret only guarantees proof of knowledge of a scaled version of the witness. Nevertheless, this zero-knowledge proof is similarly employed in [Benhamouda, Krenn, et al. 2015] and for our scheme, this weaker zero-knowledge proof is enough to prove knowledge of the secret.

1.2 Overview of Structure

This paper is organized as follows: In Chapter 2, we review some related work on voting protocols. First, we will discuss two anonymous veto protocols one is based on the hardness of the decision Diffie-Hellman (DDH) problem [Hao and Zieliński 2006] and one is a lattice-based [Ding, Emery, et al. 2020]. Then we will take a look at three voting protocols with the first two based on the DDH problem [Groth 2004], [Hao, Ryan, et al. 2010] and the other is a lattice-based one [Pino et al. 2017]. In Chapter 3, we discuss relevant background material where we start with a discussion of lattices in $\mathbb{R}^n$ and some of the hard lattice problems. Next, we mention the discrete Gaussian distribution on lattices and some relevant results. We also mention average-case problems such as the Learning with Errors (LWE) problem which was introduced by [Regev 2009] which relies on the hardness of an approximate version of the shortest vector problem. We also

mention a special variant of the LWE problem, namely, the Ring Learning with Errors problem (RLWE) which can be thought of as LWE on ideal lattices and was introduced in [Lyubashevsky, Peikert, et al. 2013]. Cryptographic protocols based on the RLWE problem are more efficient than those who rely on LWE and also guarantees shorter keys on the same security level. One of the first primitives based on RLWE is given in [Ding, Xie, et al. 2012] which is an RLWE version of the Diffie-Hellman key exchange protocol. We also take inspiration from [Ding, Xie, et al. 2012] to construct our voting protocol in addition to [Hao and Zieliński 2006]. Even though the RLWE problem assumes additional structure on the lattices considered, there are no known efficient (that is, polynomial-time) attacks, even quantum ones, yet in the literature. We also discuss how the rejection sampling technique works which is about a sampling technique that allows one to sample from an auxiliary distribution but such samples look like they are drawn from the target distribution. The rejection sampling technique is employed in our zero-knowledge protocol which allows the prover to hide information of his secret when he do knowledge proofs. Finally, we also define what a commitment scheme is, which is an important cryptographic primitive acting as a procedure to construct "electronic sealed envelopes" which hide the sender's, in our case the voter, secret keys and messages with the additional property that the envelope does not open to any other secret message, this is called the binding property of the commitment. Also to ensure that voters follow the protocol specification we add another layer to our protocol with the use of zero-knowledge proofs which provides a way for the prover to prove to a verifier that he knows some secret information given a statement without revealing this secret information.

In Chapter 4, we describe our protocol where we consider two separate cases, one is for a voting protocol for two candidates and the other is for multiple candidates. The protocol for two candidates uses bit values while for multiple candidates case, we use monomial powers X^k as our secret votes. We also show how our protocol performs using

specific parameters and we shall see the limitations of our protocol if we modify specific parameters.

In Chapter 5, we define what we mean by a secure voting protocol, in particular define the desirable properties of a secure voting protocol, namely, being a *self-tallying* one, second, having *fairness* in the sense that the voting result can only be learned after all the votes are cast and achieving *privacy* that is any individual vote of honest parties cannot be learned by any adversary. We analyze the security of our protocol, first against semi-honest adversaries, that is adversaries that follow the protocol specification but tries to learn secret information from its inputs and from the public outputs. We also check that our protocol is secure against malicious adversaries; adversaries that deviate from the protocol specification, that is, it can change its inputs, prematurely abort, or do whatever it wants in trying to learn secret information from honest voters' votes. To prove security against malicious adversaries, we will follow the "real-vs-ideal world" paradigm, where we show that our real protocol "emulates" the ideal-world protocol for our voting problem so any malicious attack in the ideal-world can be efficiently built from an attack in the real-world. However, since the ideal-world is secure by its definition, any attack on it is futile and thus, a malicious attack in the real-world is impossible.

In Chapter 6, we make a summary of our paper and provide some recommended future work.

1.3 Contributions

The original work and contributions presented in this paper are the following: First, we show a natural, simple commitment scheme based on the hardness of the RLWE hardness assumption in Section 3.9.1. In Section 3.9.3 we propose two proofs of knowledge, namely what we call $\Sigma'-$ protocols based on the hardness of the RLWE problem, with the first one provides a prover a way to prove that he knows small RLWE secrets that convinces a verifier without revealing any information about the secrets other than the fact that they

are small. Secondly, in the post-quantum veto paper [Ding, Emery, et al. 2020] where I am one of the authors, I present a contributed work primarily on the design and the proof of correctness of the passively-secure veto protocol. Thirdly, since we are proposing a voting protocol involving two candidates where voters cast a 0 vote corresponding to the first candidate and a 1 vote for the second candidate, we propose another Σ'-protocol for proving that a voter honestly casted an encryption of a 0 or 1 vote without revealing the specific vote to any verifier. Finally in Chapter 6, we provide security proofs for our voting protocol in two adversarial models, namely the first one is in the semi-honest adversarial model and the other one is in the malicious adversarial model.

Chapter 2

Review of Related Literature

In this chapter, we review some of the existing voting protocols that would serve as inspirations in constructing our own self-tallying lattice-based voting protocol. In the first section, we will review what is known as Anonymous Veto Networks (AV-net) which was introduced in [Hao and Zieliński 2006] as a solution to the Dining Cryptographers Problem presented in [Chaum 1988]. The Dining Cryptographers Problem is essentially a protocol where participants anonymously choose 0 (non-veto) or 1 (veto) as vote such that if at least one participant casts a vote of 1, the result is 1, while if the votes are unanimously 0, then the result is 0. In other words, participants want to securely compute a function

$$f(x_1, \ldots, x_m) = \begin{cases} 0 & \text{if all } x_i = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.0.1}$$

where $x_i \in \{0, 1\}$ is the secret vote of the $i-$th participant and m is the number of participants. Also, we present a lattice-based AV-net [Ding, Emery, et al. 2020] which is analogue of the AV-net in [Hao and Zieliński 2006].

Additionally, we present three voting protocols that actually count the number of votes for each candidate, namely, [Hao, Ryan, et al. 2010], [Groth 2004] and [Pino et al. 2017].

The first two relies on the hardness of the decisional Diffie-Hellman problem while the third one is a lattice-based voting protocol.

2.1 Anonymous Veto Networks (AV-net)

In this section, we present two anonymous veto networks (AV-net), one is based on the decisional Diffie-Hellman problem (DDH) [Hao and Zieliński 2006] and the other is based on the hardness of the ring learning with errors (RLWE) problem [Ding, Emery, et al. 2020]. The latter is a RLWE analogue of the DDH-based AV-net but the main difference is that in the RLWE-based AV-net, tallying the total votes results to errors and therefore, correctness of the result only holds with overwhelming probability for specific parameters.

2.1.1 A discrete logarithm-based AV-net

Now, we first present the AV-net due to Hao and Zielinski proposed in 2006 [Hao and Zieliński 2006]. The security of their protocol is based on the decisional Diffie-Hellman (DDH) assumption which we will define below:

Definition 2.1.1 (Boneh 1998)**.** *Let g be a generator of a multiplicative cyclic group G of prime order q. The decisional Diffie-Hellman (DDH) assumption states that the distribution of $\{(g^a, g^b, g^{ab})\}$ where a, b are chosen independently from the uniform distribution on $\mathbb{Z}_q$ and the distribution of $\{(g^a, g^b, g^c)\}$ where a, b, c are chosen independently from the uniform distribution on $\mathbb{Z}_q$ are computationally indistinguishable.*

The primary tool in [Hao and Zieliński 2006] is the following lemma:

Lemma 2.1.2. *Let $\mathfrak{R}$ be a commutative ring with unity with $m \geq 2$ and suppose $r_1, \ldots, r_m \in \mathfrak{R}$. Then*

$$\sum_{i=2}^{m} \sum_{j=1}^{i-1} r_i \cdot r_j = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} r_i \cdot r_j$$

Proof.

$$\begin{aligned}\sum_{i=1}^{m}\sum_{j=1}^{i-1} r_i \cdot r_j &= r_2r_1 + (r_3r_1 + r_3r_2) + \cdots + (r_mr_1 + r_mr_2 + \cdots + r_mr_{m-1}) \\ &= (r_2r_1 + r_3r_1 + \ldots + r_mr_1) + (r_3r_2 + \cdots + r_mr_2) + \cdots + r_mr_{m-1} \\ &= r_1\sum_{j=2}^{m} r_j + r_2\sum_{j=3}^{m} r_j + \cdots + r_{m-1}r_m \\ &= \sum_{i=1}^{m-1}\sum_{j=i+1}^{m} r_i \cdot r_j\end{aligned}$$

which completes the proof. ■

Their protocol for securely computing f in Equation 2.0.1 consists of two rounds. First, all participants agree on a generator g of a public multiplicative cyclic group G of prime order q for which the DDH assumption holds. In the first round, each participant P_i chooses a secret exponent x_i and publishes g^{x_i} along with a knowledge proof for x_i. After all the participants have published g^{x_i}, participant P_i computes

$$g^{y_i} = \frac{\prod_{j<i} g^{x_j}}{\prod_{j>i} g^{x_j}}.$$

In the second and final round, participant P_i publishes its vote $g^{c_i y_i}$, where

$$c_i = \begin{cases} r_i & \text{if } P_i\text{'s vote is 1 where } r_i \text{ is uniformly random chosen from } \mathbb{Z}_q \\ x_i & \text{if } P_i\text{'s vote is 0} \end{cases}.$$

To tally the result, everyone computes $\mathsf{res} = \prod_{i=1}^{n} g^{c_i y_i}$. The result is 0 or no-veto if the product is equal to 1 while the result is 1 if the product is not equal to 1. Hence, the result is correct with probability $1 - \frac{1}{|G|}$.

In [Hao and Zieliński 2006], they showed that their protocol above resists any partial collusion attack, that is, if some number of participants, not all, collude in an attempt

to attack the privacy of votes of the other participants. If the attack is passive, that is, the corrupted participants try to learn, without loss of generality, the private key of one participant, say P_i, then there exists some x_k (which is a private key of a non-colluding participant, not equal to the secret x_i of P_i) oblivious to the corrupted participants since not everyone is corrupted, and therefore g^{y_i} remains uniformly distributed since it depends on the uniform x_k and thus, the corrupted participants does not learn y_i.

Moreover, the corrupted participants that try to actively overturn the result from veto to a non-veto one must solve the discrete logarithm problem on the group G to be successful which is at least as hard as the DDH problem. Hence, the AV-net proposed in [Hao and Zieliński 2006] achieves provable security based on the DDH assumption.

The major problem with any cryptographic protocol that is based on the hardness of the discrete logarithm problem, which includes the DDH assumption is that it can be easily broken by a polynomial-time quantum algorithm due to [Shor 1999]. Hence a need for an alternative hardness assumption for cryptographic protocols is needed that is believed to resist any efficient quantum attack. Some of the known examples of post-quantum cryptography are lattice-based cryptography, multivariate public key cryptography and code-based cryptography. Our discussion and results in this paper fall under the category of lattice-based cryptography.

2.2 Voting protocols

In this section, we discuss some of the existing voting protocols in the literature analogous to the veto protocols mentioned in the previous section, in the sense that the first two voting protocols also relies on the DDH assumption while the third voting protocol relies on the hardness of the LWE/RLWE problem.

2.2.1 A self-tallying voting protocol based on the DDH assumption due to Hao, Zielinski, Ryan

Hao, Zielinski and Ryan extended the veto protocol in [Hao and Zieliński 2006] into an anonymous self-tallying voting protocol [Hao, Ryan, et al. 2010] that turned out to be the main inspiration in constructing our own voting protocol.

As in [Hao and Zieliński 2006], their voting protocol assumes a finite cyclic group G of order prime q for which the DDH assumption holds. Also, let g be a generator of G. First, we describe their protocol involving m voters and 2 candidates, where a vote of 0 corresponds to a vote for candidate 1 and a vote of 1 for candidate 2.

Round 1: Each voter P_i selects a uniformly random chosen $x_i \in \mathbb{Z}_q$ and publishes g^{x_i} and a zero-knowledge proof for the exponent x_i.

After all g^{x_i} are published and the corresponds proofs are verified, each voter P_i computes g^{y_i} as in Equation 2.1.1.

Round 2: To cast vote $v_i \in \{0, 1\}$, voter P_i computes and publishes $g^{x_i y_i} g^{v_i}$ and a corresponding zero-knowledge proof for the exponent $x_i y_i + v_i$.

To tally the votes, everyone computes $\prod_{i=1}^{m} g^{x_i y_i} g^{v_i}$ which is equal to $g^{\sum_{i=1}^{m} v_i}$ since $\sum_{i=1}^{m} x_i y_i = 0$. The exponent $\sum_{i=1}^{m} v_i$ can be determined for example, by precomputing the values of g^{ℓ}, for $\ell \in \{0, 1, \ldots, m\}$.

Moreover, they have extended their protocol to the case of multiple candidates. Suppose that there are m voters and t candidates. First, choose a smallest positive integer k such that $2^k > m$. The first round of the protocol is the same as round 1 of the two candidates case while in the second round, a vote for candidate $j \in \{1, \ldots, t\}$, corresponds to a vote $2^{(j-1)k}$, that is, voter P_i publishes $g^{x_i y_i} g^{v_i}$, where $v_i = 2^{(j-1)k}$ as an encryption of the vote for candidate j and a corresponding zero-knowledge proof.

To tally the votes, everyone still computes $\prod_{i=1}^{m} g^{x_i y_i} g^{v_i}$ which is again $g^{\sum_{i=1}^{m} v_i}$ but this time, we have

$$\sum_{i=1}^{m} v_i = 2^0 c_1 + 2^k c_2 + \cdots + 2^{(t-1)k} c_t$$

where c_j is the number of votes obtained by candidate j.

2.2.2 A self-tallying voting protocol based on the DDH assumption due to Groth

Another voting protocol we will mention is a self-tallying voting protocol due to [Groth 2004] that relies on the DDH assumption. Assume for simplicity that there are two candidates in the election, let 0 be a vote for the first candidate and 1 corresponds to a vote to the second candidate.

The design of the protocol in [Groth 2004] is as follows:

1. At the beginning of the protocol, all voters agree on a group G of prime order q with a generator g, where the DDH assumption holds. Also, we assume an authenticated bulletin board where voters publish their public keys and encryption of their votes.

2. In the key registration phase, each voter V_i selects a uniformly random element $x_i \in \mathbb{Z}_q$ and sets $h_i = g^{x_i}$. Here, they publish h_i and a proof of knowledge on x_i such that it satisfies $h_i = g^{x_i}$. After all voters are done publishing their h_i along with a corresponding proof of knowledge, they set the current state of the election to $(1, 1)$.

3. In the voting phase, to cast vote $v_i \in \{0, 1\}$, voter V_i downloads the current state of the election, say the current state is (u, v), verifies the proof of knowledge of the keys and all votes cast so far. Then to encrypt its vote, voter V_i selects a uniformly random chosen $r_i \in \mathbb{Z}_q$ and updates the current state of the election from (u, v)

into (U, V), where

$$U = ug^{r_i}$$

and

$$V = vu^{-x_i} \left(\prod_{j \in T} h_j \right)^{r_i} g^{v_i},$$

where T is the set of remaining voters.

To better see what is happening in the current state of the protocol, assume that V_i is the $i-$th voter, that is, the voter who casts its vote after $i-1$ voters. Assume also that there are m voters.

Then the state of the election updates in the following manner:

1. The initial state is (1,1).

2. After voter V_1 casts its vote, state becomes

$$\left(g^{r_1}, g^{v_1} \prod_{j=2}^{m} h_j^{r_1} \right)$$

3. After voter V_2 casts its vote, state is updated into

$$\left(g^{r_1+r_2}, g^{v_1+v_2} \prod_{j=3}^{m} h_j^{r_1+r_2} \right)$$

4. Hence, after voter V_i casts its vote, state is updated into

$$\left(g^{\sum_{k=1}^{i} r_k}, g^{\sum_{k=1}^{i} v_k} \cdot \prod_{j=i+1}^{m} h_j^{\sum_{k=1}^{i} r_k} \right)$$

5. Finally when the last voter casts its vote, the final state of the protocol becomes

$$\left(g^{\sum_{i=1}^{m} r_i}, g^{\sum_{i=1}^{m} v_i} \right)$$

since the product in terms of powers of h_j eventually become 1.

Hence, we can easily determine the sum $\sum_{i=1}^{m} v_i$ by precomputing the powers g^a, where $a \in \{0, 1, \ldots, m\}$. The total number of votes for the first candidate is $m - \sum_{i=1}^{m} v_i$ while the total number of votes for the second candidate is $\sum_{i=1}^{m} v_i$.

Groth has shown the security of the protocol in the sense of perfect secrecy/privacy of the votes against malicious adversaries by using a hybrid argument to show that any adversary of the real execution of his protocol can be simulated such that the simulator only has knowledge of the sum of the honest voters' votes.

We will use the same idea later when we prove the security of our own protocol against malicious adversaries.

2.2.3 A lattice-based voting protocol

Here, we briefly mention the lattice-based voting protocol by [Pino et al. 2017]. The design of their protocol uses homomorphic commitments and homomorphic zero-knowledge proofs. The commitment scheme employed in the voting protocol in [Pino et al. 2017] relies on the hardness of the ring version of short integer solution (SIS) problem (see [Ajtai 1996]).

Definition 2.2.1. *Informally, the ring-*SIS *problem states that given a uniformly random matrix* $A \in R_q^{n \times m}$*, where* $m \geq n$*, find a short, nonzero vector* $s \in R_q^m$ *such that* $As = 0 \bmod q$.

The ring-SIS hardness result states that an efficient oracle that solves ring-SIS implies an efficient solver for certain worst-case approximate ideal lattice problems. We describe their commitment scheme below: First, fix a public commitment key $\begin{bmatrix} A \\ B \end{bmatrix}$ where A and B are uniformly random chosen matrices over R_q. To commit to $x \in \mathbb{Z}_q$, with x being viewed as a polynomial whose constant term is x and the other terms are 0 in

$R_q := \mathbb{Z}_q[X]/(X^n + 1)$, n a power of two, $q \geq 2$ a prime, integer, we pick a random vector r of polynomials where each polynomial r_i of r has small coefficients and the commitment to x is given by

$$\begin{bmatrix} \mathrm{a} \\ b \end{bmatrix} := \begin{bmatrix} A \\ B \end{bmatrix} \mathrm{r} + \begin{bmatrix} 0 \\ x \end{bmatrix}.$$

To open the commitment, simply compute

$$\begin{bmatrix} \mathrm{a} \\ b \end{bmatrix} - \begin{bmatrix} A \\ B \end{bmatrix} \mathrm{r}$$

The binding property of the above commitment scheme follows from the hardness of ring-SIS while the hiding property comes from the decision version of the ring-SIS problem.

The above commitment scheme is homomorphic in the sense that the commitment of the sum of a number of elements in $\mathbb{Z}_q$ is the sum of their commitments which only holds for a bounded number of messages since the coefficients of the randomness becomes larger as the number of messages increases.

Moreover, the key specification of their protocol assumes a fixed number of trusted authorities such that each voter decomposes its vote into several parts and for each part of its vote, the voter applies the above commitment scheme and sends the commitment of each part of its vote to each authority.

On the other hand, each voting authority receives the randomness used for the commitment of each voter and using all the randomness from all the voters, confirms that each randomness is short by doing a proof of knowledge to its fellow authorities.

For more details of their protocol, see [Pino et al. 2017].

Chapter 3

Relevant Background Material

3.1 Lattices and Hard Lattice Problems

We start this section by defining what a lattice is and some of the standard lattice problems that are conjectured to be hard to solve which serves as the basis of security of lattice-based cryptographic protocols. After we introduce the hard lattice problems, we also state its worst-case to average-case connection to the learning with errors and ring learning with errors problems introduced by [Regev 2009] and [Lyubashevsky, Peikert, et al. 2010]. All the discussion in this section can be referenced from the following lecture notes: [Micciancio 2021].

Definition 3.1.1. *A lattice Λ is an additive discrete subgroup of $\mathbb{R}^n$. Equivalently, a lattice Λ is a subset of $\mathbb{R}^n$ consisting of integral combinations of k linearly independent vectors in $\mathbb{R}^n$. If $k = n$, we say that Λ is a full-rank lattice in $\mathbb{R}^n$.*

Definition 3.1.2. *We define the minimum distance, denoted by $\lambda_1(\Lambda)$, of a lattice Λ to be the length of a shortest nonzero vector of Λ, that is,*

$$\lambda_1(\Lambda) = \min\{\|v\| : 0 \neq v \in \Lambda\}$$

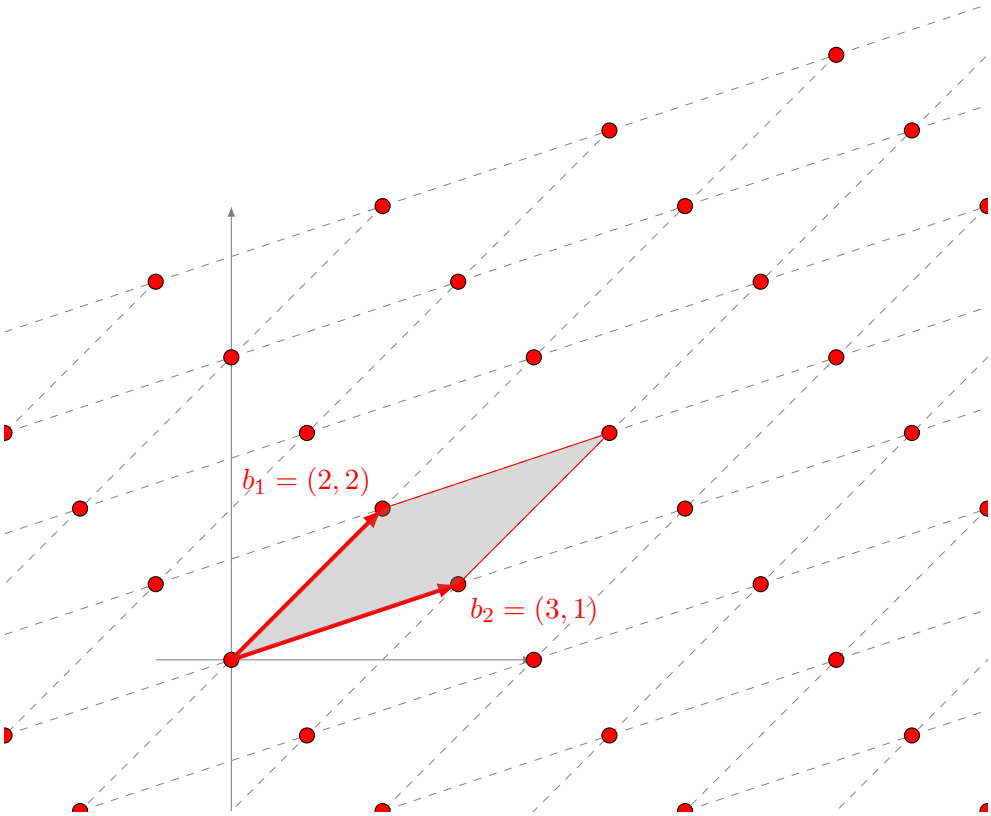


Figure 3.1: A two-dimensional lattice with basis vectors $b_1 = (2, 2)$ and $b_2 = (3, 1)$. Points on the lattice are denoted by the red dots. The grey shaded area denotes the fundamental parallelepiped of the lattice.

Calculating $\lambda_1(\Lambda)$ is computationally hard as the dimension of the lattice goes to infinity but there are several known bounds to it. A lower bound to $\lambda_1(\Lambda)$ can be obtained by considering any basis B of lattice Λ, then calculate the Gram-Schmidt orthogonalization $\tilde{B}$ of B and it can be shown that $\min_i \|\tilde{b}_i\| \leq \lambda_1(\Lambda)$, where $\tilde{b}_1, \ldots, \tilde{b}_n$ are the Gram-Schmidt orthogonalized vectors associated to basis B, which gives a lower bound for $\lambda_1(\Lambda)$. Notice that the lower bound above is dependent on the choice of basis B. On the other hand, an upper bound to $\lambda_1(\Lambda)$ can be obtained by applying Minkowski's theorem [Minkowski 1910] which implies that $\lambda_1(\Lambda) \leq \sqrt{n}(\det \Lambda)^{1/n}$. Unfortunately, these bounds

are not good enough to estimate for $\lambda_1(\Lambda)$ since we can construct lattices where its $\lambda_1(\Lambda)$ are too far from the bounds mentioned above.

Hard Lattice Problems

First, we present three versions of the approximate shortest vector problem SVP_γ, namely the search, decision and estimate versions of the problem.

Definition 3.1.3. *Let Λ be a full-rank lattice in $\mathbb{R}^n$ and let $\gamma = \gamma(n) \geq 1$. We define the three variants of the shortest vector problem SVP_γ.*

1. *The search version of SVP_γ takes a basis B of Λ and γ as inputs and outputs a nonzero vector $v \in \Lambda$ such that $\|v\| \leq \gamma \cdot \lambda_1(\Lambda)$.*

2. *The decision version of SVP_γ, also denoted as GapSVP_γ takes as input a real number $d > 0$ and outputs YES if $\lambda_1(\Lambda) \leq d$ while it outputs NO if $\lambda_1(\Lambda) > \gamma \cdot d$.*

3. *The estimate version of SVP_γ takes a basis B of Λ and outputs a real number $r \in [\lambda_1(\Lambda), \gamma \cdot \lambda_1(\Lambda))$.*

Remark 3.1.4. It can be shown that the decision and estimate versions of SVP_γ are equivalent by showing the an oracle solving one of them implies an algorithm that solves the other. However, the only proven fact here is that the search version of SVP_γ is at least as hard as GapSVP_γ. In other words, showing that an oracle solving GapSVP_γ implies an algorithm solving the search version still remains an open problem for $\gamma > 1$ (the case where $\gamma = 1$ can be shown to be true).

Another hard lattice problem is the shortest independent vectors problem or SIVP but before we define the problem, we need to introduce the k-th successive minimum of a lattice.

Definition 3.1.5. *The k-th successive minimum of an $n-$dimensional lattice Λ, denoted by $\lambda_k(\Lambda)$ is the smallest $r > 0$ such that the $n-$dimensional ball centered at 0 of radius r, $B(0, r)$ contains at least k linearly independent lattice vectors in Λ.*

As a consequence of the definition above, we have that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Definition 3.1.6. *Let $\gamma = \gamma(n) \geq 1$ and Λ be an $n-$dimensional lattice. The search version of $\gamma-$shortest independent vectors problem, denoted by SIVP_γ asks to find n linearly independent vectors $v_i \in \Lambda$ such that $\max \|v_i\| \leq \gamma \cdot \lambda_n(\Lambda)$.*

Remark 3.1.7. *The hardness of SVP_γ and SIVP_γ stems from the following setup: if we are given an arbitrary basis of a lattice Λ both problems are computationally hard to solve for polynomial (in the dimension n) approximation factors. In fact, the best-known algorithms such as the LLL algorithm [Lenstra et al. 1982] take exponential time and space to solve the above problems with polynomial approximation factor $\gamma(n)$. However, if the approximation factor $\gamma(n)$ is exponential in n, then the above problems are "easy" in the sense that it will take polynomial time to solve them. Hence, in cryptography, it is very important to base the protocols to lattice problems with polynomial approximation factors to guarantee provable security.*

Finally we define the approximate closest vector problem, denoted CVP_γ and its "promise" variant called the bounded distance decoding problem denoted by BDD. The learning with errors problem that we will define later turns out to be an average-case reformulation of the BDD problem and we will see its connection to the worst-case lattice problems we defined earlier, specifically GapSVP and SIVP.

Definition 3.1.8. *Let $\gamma = \gamma(n) \geq 1$. The $\gamma-$closest vector problem, denoted by CVP_γ, takes as input a lattice Λ and a target vector $t \in \mathbb{R}^n$ and asks to find a vector $v \in \Lambda$ such that $\|v - t\| \leq \gamma \cdot \mathsf{dist}(t, \Lambda)$, where $\mathsf{dist}(t, \Lambda) = \inf\{\|t - x\| : x \in \Lambda\}$.*

Definition 3.1.9. *Let $\gamma = \gamma(n) \geq 1$. The bounded distance decoding (BDD) problem, takes as input a lattice Λ, a real number $d > 0$ and a vector $t \in \mathbb{R}^n$ such that $\mathsf{dist}(t, \Lambda) \leq d$, and asks to find a lattice vector $v \in \Lambda$ such that $\|v - t\| \leq d$.*

Notice that the BDD problem does not always guarantee that a vector $t \in \mathbb{R}^n$ exists such that $\mathsf{dist}(t, \Lambda) \leq d$ but in the setup of the learning with errors problem, such vector

t always exists. Moreover, the real number $d > 0$ is typically taken to be less than $\frac{\lambda_1(\Lambda)}{2}$ to guarantee that the lattice vector v is unique such that $\|t - v\| \leq d$.

Dual lattices

Now, we will define the dual of a lattice which is similar to the construction of a dual of a vector space. The dual of a lattice is itself a lattice so we can simply call it the dual lattice.

Definition 3.1.10. *Let Λ be a full-rank lattice in $\mathbb{R}^n$. The dual lattice Λ^* to Λ is given by*

$$\Lambda^* := \{y \in \mathbb{R}^n : \langle x, y\rangle \in \mathbb{Z}, \forall x \in \Lambda\}.$$

Remark 3.1.11. *Λ^* is itself a lattice since if B is the matrix consisting of basis vectors of the full-rank lattice Λ, then B^{-T} is the matrix consisting of basis vectors for Λ^*. As a consequence, $\det(\Lambda) = \frac{1}{\det(\Lambda^*)}$. Moreover, the dual of the dual lattice of Λ is Λ, that is, $(\Lambda^*)^* = \Lambda$ and if we scale the lattice, $(s\Lambda)^* = \frac{1}{s}\Lambda^*$.*

3.2 Notions of indistinguishability and some statistical background

In this section, we define the three notions of indistinguishability. First let's define what we mean by a *negligible function* and state its properties.

Definition 3.2.1. *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if for any polynomial function $p : \mathbb{N} \to \mathbb{R}^+$, there exists $n_0 \in \mathbb{N}$ such that for any $n \geq n_0$, we have $\epsilon(n) \leq \frac{1}{p(n)}$.*

Most of the time, we simply write $\epsilon(n) = \text{negl}(n)$ to denote that ϵ is a negligible function without explicitly specifying its function formula.

Theorem 3.2.2. *The following are the properties of negligible functions:*

1. *Sum of two negligible functions is negligible.*

2. *Product of a negligible function and a polynomial function is negligible.*

3. *Subtracting a nonnegligible function by a negligible function is nonnegligible.*

As a consequence, we have:

Corollary 3.2.3. *Let ε_i be a sequence of negligible functions and q_i be a sequence of polynomial functions. Then for any constant $c \in \mathbb{N}$, we have that $\sum_{i=1}^{c} q_i \varepsilon_i$ is a negligible function.*

Now, let $X = (X_n)_{n\in\mathbb{N}}$ be a sequence of random variables, where we can think of n as a security parameter. We write $x \xleftarrow{\$} X_n$ to denote sampling a value from distribution X_n.

Definition 3.2.4 (Statistical distance)**.** *The statistical distance of two sequences of random variables $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (Y_n)_{n\in\mathbb{N}}$ is defined as the function*

$$SD_{X,Y}(n) := \frac{1}{2} \sum_{z\in\{0,1\}^*} |\Pr[X(1^n) = z] - \Pr[Y(1^n) = z]|$$

Now we are ready to define the three different notions of *indistinguishability* between two sequences of random variables.

Definition 3.2.5 (Indistinguishability)**.** *Let $X = (X_n)_{n\in\mathbb{N}}$ and $Y = (Y_n)_{n\in\mathbb{N}}$ be two sequences of random variables.*

1. *We say X and Y are perfectly indistinguishable, denoted $X \overset{p}{=} Y$, if for any $n \in \mathbb{N}$, $SD_{X,Y}(n) = 0$.*

2. *We say X and Y are statistically indistinguishable, denoted $X \overset{s}{\approx} Y$, if for any $n \in \mathbb{N}$, $SD_{X,Y}(n) = negl(n)$.*

3. *We say X and Y are computationally indistinguishable, denoted $X \overset{c}{\approx} Y$, if for all efficient (polynomial-time) algorithms $\mathcal{D}$, the advantage*

$$\mathrm{Adv}^{indist}_{X,Y,\mathcal{D}}(n) := |\Pr[\mathcal{D}(1^n, X(1^n)) = 1] - \Pr[\mathcal{D}(1^n, Y(1^n)) = 1]|$$

is a negligible function.

3.3 Discrete Gaussian Distribution on Lattices

In the learning with errors and ring learning with errors problems, the error distribution is typically taken to be the discrete Gaussian distribution on a lattice Λ. In fact the worst-case to average-case hardness result of the above problems assumes a discrete Gaussian error distribution. In this subsection, we define the discrete Gaussian distribution on a lattice and state some facts on the total measure of the discrete Gaussian distribution, in particular if the lattice is perturbed either by scaling or by a lattice coset. Also, we briefly mention the connection between the total discrete Gaussian measures between a lattice and its dual via the Poisson summation formula which serves as a vital tool in proving the facts mentioned above on the effect on the total Gaussian measure by perturbing the lattice. We continue our discussion with reference to [Micciancio 2021].

Definition 3.3.1. *Let Λ be a full-rank lattice in $\mathbb{R}^n$, $\sigma > 0$ and $v \in \mathbb{R}^n$. Define the Gaussian function on $\mathbb{R}^n$ with parameter σ as the function $\rho_\sigma(x) = e^{-\pi\|x\|^2/\sigma^2}$. Given any lattice coset $v + \Lambda$, where $v \in \mathbb{R}^n$, define $\rho_\sigma(v + \Lambda) := \sum_{x \in v+\Lambda} \rho_\sigma(x)$. The discrete Gaussian distribution $D_{v+\Lambda,\sigma}$ on the lattice coset $v + \Lambda$ with parameter σ is given by the distribution on $v + \Lambda$ whose density is given by $\rho_\sigma(x)/\rho_\sigma(v + \Lambda)$, where $x \in v + \Lambda$. Moreover, if the lattice Λ is already understood, the discrete Gaussian distribution $D_{\Lambda,\sigma}$ on Λ is simply denoted by D_σ.*

Before we state the Poisson summation formula, we need to briefly recall the Fourier transform of a function f on $\mathbb{R}^n$.

Definition 3.3.2. *Let $f : \mathbb{R}^n \to \mathbb{C}$ be a function such that $\int_{\mathbb{R}^n} |f(x)|\ dx < \infty$. The Fourier transform of f, denoted $\hat{f}$, is defined as a function $\hat{f} : \mathbb{R}^n \to \mathbb{C}$ such that*

$$\hat{f}(y) = \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, y \rangle}\ dy$$

Now, we state the Poisson summation formula on Gaussian measure for lattices below.

Theorem 3.3.3. *Let ρ be the Gaussian function on $\mathbb{R}^n$ with parameter 1, that is, $\rho = \rho_1$. Then $\rho = \hat{\rho}$. Now, let Λ be a $n-$dimensional lattice and define $\rho(\Lambda)$ to be $\sum_{x \in \Lambda} \rho(x)$. Then $\rho(\Lambda) = \det(\Lambda^*)\rho(\Lambda^*)$.*

Notice that for any $s > 0$ and any lattice Λ, $\rho(s\Lambda) = \rho_{1/s}(\Lambda)$. As consequences of the Poisson summation formula stated above, we have the following bounds on the Gaussian measure when the lattice is scaled by a positive real factor greater than 1 and when the lattice is perturbed as a lattice coset.

Theorem 3.3.4. *Let $s \geq 1$ and Λ be an $n-$dimensional lattice. Then $\rho\left(\frac{1}{s}\Lambda\right) \leq s^n \rho(\Lambda)$.*

Theorem 3.3.5. *Let $v \in \mathbb{R}^n$ and Λ be any full-rank lattice in $\mathbb{R}^n$. Then $\rho(v + \Lambda) \leq \rho(\Lambda)$.*

Lemma 3.3.6. *Let $v + \Lambda$ be a lattice coset, let $h \in \mathbb{R}^n$ be a fixed vector and let $H = \{x \in \mathbb{R}^n : \langle x, h \rangle \geq \|h\|^2\}$. Then $\rho\left((v + \Lambda) \cap H\right) \leq \rho(h)\rho(v - h + \Lambda)$.*

A useful corollary to the above lemma says that if we choose $x \xleftarrow{\$} D_\Lambda := D_{\Lambda,1}$, the probability that the max norm $\|x\|_\infty := \max_i\{|x_i|\}$ is greater than some sufficiently large $t > 0$ is negligible in t. The proof of the corollary is left as an exercise in [Micciancio 2021] but in this paper, we provide a proof of the corollary. Before we give the proof, we briefly mention the union bound.

Lemma 3.3.7. *Let $A_1, A_2, \ldots$ be a countable set of events in a sample space S. Then* $\Pr\left[\bigcup_{i=1}^{\infty} A_i\right] \leq \sum_{i=1}^{\infty} \Pr[A_i]$.

Corollary 3.3.8. *For any $n-$dimensional lattice Λ and real number $t > 0$, we have*

$$\Pr[\|x\|_\infty \geq t | x \xleftarrow{\$} D_\Lambda] \leq 2ne^{-\pi t^2}.$$

Proof. Let $B := \{x \in \mathbb{R}^n : \|x\|_\infty \geq t\} = \bigcup_{i=1}^{n} \{x \in \mathbb{R}^n : |x_i| \geq t\}$. For each $i = 1, \ldots, n$, let $e_i \in \mathbb{R}^n$ be the vector whose $i-$th coordinate is 1 and 0 elsewhere. Let

$$A_i := \{x \in \mathbb{R}^n : \langle x, e_i \rangle \geq t\} = \{x \in \mathbb{R}^n : \langle x, te_i \rangle \geq t^2\}$$

and let

$$B_i := \{x \in \mathbb{R}^n : \langle x, e_i \rangle \leq -t\} = \{x \in \mathbb{R}^n : \langle x, -te_i \rangle \geq t^2\}.$$

Then $B = \bigcup_{i=1}^{n} (A_i \cup B_i)$. Thus by Theorem 3.3.5, Lemma 3.3.6 and Lemma 3.3.7, we have

$$\begin{aligned}
\Pr[\|x\|_\infty \geq t | x \xleftarrow{\$} D_\Lambda] &= \frac{\rho(\Lambda \cap B)}{\rho(\Lambda)} \\
&\leq \frac{1}{\rho(\Lambda)} \sum_{i=1}^{n} (\rho(\Lambda \cap A_i) + \rho(\Lambda \cap B_i)) \\
&\leq \sum_{i=1}^{n} 2e^{-\pi t^2} = 2ne^{-\pi t^2}.
\end{aligned}$$

since $\rho(\Lambda \cap A_i) \leq \rho(te_i)\rho(-te_i + \Lambda) \leq e^{-\pi t^2}\rho(\Lambda)$ and similarly, $\rho(\Lambda \cap B_i) \leq e^{-\pi t^2}\rho(\Lambda)$. ■

A direct consequence of the corollary above is that if we choose $x \xleftarrow{\$} D_{\Lambda,s}$ for some parameter $s > 0$, then the probability that $\|x\|_\infty \geq t$, where $t > 0$, is less than or equal to $2ne^{-\pi(t/s)^2}$.

Now, if we replace the max norm by the Euclidean norm, we have the following:

Theorem 3.3.9. *For any $\alpha \geq 1$ and any $n-$dimensional lattice Λ, we have*

$$\Pr\left[\|x\| \geq \alpha\sqrt{\frac{n}{2\pi}} : x \xleftarrow{\$} D_\Lambda\right] \leq \left(\frac{\alpha^2}{e^{\alpha^2-1}}\right)^{n/2}$$

The smoothing parameter of a lattice

In [Micciancio and Regev 2007], another lattice invariant called the smoothing parameter was introduced. The smoothing parameter tells us the least value of the parameter $\sigma > 0$ of the discrete Gaussian distribution $D_{\Lambda,\sigma}$ such that $D_{\Lambda,\sigma}$ becomes statistically close to the "uniform" distribution on Λ. To be precise, Theorem 3.3.11 says that samples from the discrete Gaussian distribution on a lattice with parameter σ at least as large as the smoothing parameter of a sublattice Λ' is indistinguishable from the uniform distribution on the points $\Lambda \bmod \Lambda'$. The importance of the smoothing parameter lies to the result in the learning with errors problem that if we pick the parameter σ of the discrete Gaussian error distribution for our learning with errors samples to be at least as large as the smoothing parameter, then the samples become indistinguishable from uniform ones.

Definition 3.3.10. *Let Λ be an $n-$dimensional lattice and let $\varepsilon > 0$. The smoothing parameter of Λ is the number $\eta_\varepsilon(\Lambda) := \min\{s > 0 : \rho(s\Lambda^*) \leq 1 + \varepsilon\}$.*

The term smoothing parameter comes from the proceeding result [Micciancio and Regev 2007]:

Theorem 3.3.11. *Let Λ, Λ' be $n-$dimensional lattices such that $\Lambda' \subseteq \Lambda$. Then for any $\varepsilon \in (0, 1/2)$, any $\sigma \geq \eta_\varepsilon(\Lambda')$, the distribution of $D_{\Lambda,\sigma}$ is within statistical distance at most 2ε from the uniform distribution on $\Lambda \bmod \Lambda'$.*

We also have the following lemma that relates the smoothing parameter and the $\lambda_1(\Lambda^*)$ and $\lambda_n(\Lambda)$ of any $n-$dimensional lattice Λ, due to [Micciancio and Regev 2007]:

Theorem 3.3.12. *For any $n-$dimensional lattice Λ, we have:*

1. $\eta_{2^{-2n}}(\Lambda) \leq \sqrt{n}/\lambda_1(\Lambda^*)$

2. $\eta_\varepsilon(\Lambda) \leq \sqrt{\ln(n/\varepsilon)}\lambda_n(\Lambda)$ *for all* $\varepsilon \in (0,1)$.

Finally, we state a useful lemma from [Banaszczyk 1993].

Lemma 3.3.13. *For any $n-$dimensional lattice Λ and $r > 0$, a point sampled from $D_{\Lambda,r}$ has Euclidean norm at most $r\sqrt{n}$, with probability at least $1 - 2^{-2n}$.*

Finally we mention the complexity of an algorithm for sampling from a discrete Gaussian distribution on an $n-$dimensional lattice according to [Gentry et al. 2008] which states that the running time takes $\mathcal{O}(n^2)$ plus $\omega(n \log^2 n)$ operations to generate a sample.

3.4 Learning with Errors

In [Regev 2009], Regev introduced the learning with errors (LWE) problem which asks about finding a secret vector $s \in \mathbb{Z}_q^n$, $q \geq 2$ an integer, in a system of noisy linear equations. To be precise we define the learning with errors problem below.

Definition 3.4.1. *Let $q \geq 2$ be an integer and $n \in \mathbb{N}$. Let χ be some distribution on $\mathbb{Z}$, typically taken to be a discrete Gaussian distribution. Let $s \in \mathbb{Z}_q^n$ be some secret vector chosen from the uniform distribution on $\mathbb{Z}_q^n$. Consider the system of noisy linear equations modulo q:*

$$\begin{aligned} b_1 &= \langle a_1, s\rangle + e_1 \\ b_2 &= \langle a_2, s\rangle + e_2 \\ &\vdots \\ b_m &= \langle a_m, s\rangle + e_m \end{aligned}$$

where $a_1, \ldots, a_m$ are drawn independently from the uniform distribution on $\mathbb{Z}_q^n$ and the error vectors $e_1, \ldots, e_m$ are drawn independently from χ. Denote the LWE *distribution by $A_{s,\chi}$ whose samples are of the form (a_i, b_i) generated as above.*

The learning with errors (LWE) *problem asks the following: Given arbitrarily number of pairs $(a_1, b_1), \ldots, (a_m, b_m)$ from the* LWE *distribution $A_{s,\chi}$, find the secret s.*

Remark 3.4.2. *Choosing specific parameters for the error distribution χ, χ has standard deviation αq, where $\alpha > 0$ is chosen to be $1/poly(n)$. Also, the modulus q is also typically taken to be polynomial in n.*

The above definition of the LWE problem is the also called search-LWE since the problem asks to find the secret s. The decision variant of the LWE problem is given below and Regev showed in [Regev 2009] that the search and decision variants of the LWE problem are equivalent.

Definition 3.4.3. *In the search version of the* LWE *problem defined above, denote by $A_{s,\chi}$ the distribution of the* LWE *pairs $(a, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The decision-*LWE *problem takes as input pairs $(a, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and asks to distinguish with non-negligible advantage whether the pairs are generated according to $A_{s,\chi}$ or according to the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Another variant of the LWE problem takes a secret s from the error distribution χ called the Hermite Normal Form of the LWE problem, denoted $\mathsf{HNF} - \mathsf{LWE}$. It was shown that $\mathsf{HNF} - \mathsf{LWE}$ is at least as hard as the standard LWE [Applebaum et al. 2009]. In other words, selecting the secret from χ does not change the hardness of LWE.

3.4.1 Hardness of the LWE problem

In [Regev 2009], Regev showed that under certain parameters, an oracle solving the LWE problem implies an efficient quantum algorithm that solves GapSVP and SIVP with polynomial approximation factors on any lattice. His proof involves an iterative process

broken down into two components. The first component says that an LWE oracle implies we can efficiently solve the BDD problem on any dual lattice Λ^* to within some specified distance. More precisely, we state the first component from [Regev 2009]:

Theorem 3.4.4. *Let $q \geq 2$ be an integer and $\alpha \in (0,1)$. Suppose we have an oracle that solves the* LWE *problem with modulus q and error distribution parameter α. Then, given as input any $n-$dimensional lattice Λ, a large enough polynomial number of samples from $D_{\Lambda,r}$, for some $r \geq \sqrt{2}q \cdot \eta_\varepsilon(\Lambda)$ (or we can use the stronger inequality $r \geq q\sqrt{2n}/\lambda_1(\Lambda^*)$), and a point $x \in \mathbb{R}^n$ such that $\mathsf{dist}(x, \Lambda^*) \leq \alpha q/(\sqrt{2}r)$, then we can output the unique closest lattice point to x in polynomial time.*

The second component involves an efficient quantum algorithm that assumes an oracle to the BDD problem to within distance $d > 0$ on the dual lattice Λ^* and outputs a sample from $D_{\Lambda,\sqrt{n}/d}$. The proof idea for the second component is using the BDD oracle, we can efficiently create a quantum state equivalent to a periodic discrete Gaussian distribution on the dual lattice Λ^*, take its quantum Fourier transform, and the resulting quantum state corresponds to $D_{\Lambda,\sqrt{n}/d}$. Thus, to sample from $D_{\Lambda,\sqrt{n}/d}$, we simply measure the quantum state we obtained.

When we combine the two components above, assuming an LWE oracle, we start by selecting polynomially many samples from $D_{\Lambda,r}$ for large r (in fact, $r \geq 2^{2n}\lambda_n(\Lambda)$) and this can be done efficiently by the Bootstrapping lemma in [Regev 2009]. Apply the first component to solve BDD to within distance $d := \sqrt{2n}/r$ on Λ^*. Since we have a solver for BDD on Λ^* to within distance d, by the second component, using a quantum algorithm, we obtain polynomially many samples from $D_{\Lambda,r/\sqrt{2}}$. Now, we iterate the process by applying the first component with $D_{r/(\sqrt{2})^i}$, $i = 1, 2, \ldots$ until we get samples from $D_{\Lambda,r'}$, where $r' = \mathrm{poly}(n)/\lambda_1(\Lambda^*)$. If we apply the first component one more time with $D_{\Lambda,r'}$, Regev showed that a solution to BDD allows us to solve GapSVP on Λ to within a polynomial approximation factor. On the other hand, if we take n samples from $D_{\Lambda,r'}$, we get a solution to SIVP on Λ to within a polynomial approximation factor.

Before we state the hardness result of LWE, we define the discrete Gaussian sampling problem DGS below.

Definition 3.4.5. *The discrete Gaussian sampling problem* DGS *takes a lattice* Λ *and an* $r \geq \phi(\Lambda)$ *for some function* ϕ *on* Λ *as inputs, and asks to sample from* $D_{\Lambda,r}$.

More precisely, the hardness result of the LWE problem can be stated below:

Theorem 3.4.6. *Let* $\varepsilon = \varepsilon(n)$ *be some negligible function of* n. *Also, let* $q = q(n)$ *be some integer and* $\alpha = \alpha(n) \in (0,1)$ *such that* $\alpha q > 2\sqrt{n}$. *Assume that we have access to an oracle that solves* LWE *with modulus* q *and error parameter* α *given polynomially many samples. Then there exists an efficient quantum algorithm that solves* DGS *on lattice* Λ *with* $r \geq \sqrt{2n} \cdot \eta_\varepsilon(\Lambda)/\alpha$.

As we mentioned earlier, it suffices to have an efficient solver for the DGS problem in the above theorem to solve GapSVP and SIVP on any lattice Λ to within polynomial approximation factors.

3.4.2 Attacks on the LWE problem

Here, we briefly mention direct attacks on the LWE problem. We start with an algebraic attack from [Arora and Ge 2011] where the idea is converting LWE samples (a, b) to polynomial equations mod q,

$$f_{a,b}(s) = \prod_{e \in S} (b - \langle a, s \rangle - e) \bmod q = 0.$$

where $S \subseteq \mathbb{Z}_q$ (think of S as the subset of $\mathbb{Z}_q$ consisting of all error values with overwhelming probability). Notice that for each sample (a, b), $f_{a,b}(s) = 0$ is a polynomial equation of degree $|S|$ and the attack actually solves a system of polynomial equations (induced by the LWE samples) using a linearization technique (a technique that transforms monomial terms to variables so that the system of polynomial equations becomes a system of linear

equations). However, Arora-Ge would need exponential time and space in the dimension n to recover the secret s.

The second attack is due to Blum, Kalai and Wasserman (BKW) [Blum et al. 2003] and an improvement to Arora-Ge. The BKW attack is so far the best known attack on LWE and is a combinatorial algorithm. BKW requires $2^{\mathcal{O}(n)}$ samples and time. Their algorithm finds a small subset S among the $2^{\mathcal{O}(n)}$ samples (a_i, b_i) such that $\sum_{i \in S} a_i = (1, 0, \ldots, 0)$ and since S is small, they can find the first coordinate of the secret s from $\sum_{i \in S} b_i = s_1 + \sum_{i \in S} e_i$, where s_1 is the first coordinate of s. Hence, they can recover s by applying the same idea to obtain the remaining coordinates.

Another attack to the LWE problem is presented in [Ding 2010], but the assumption is that the errors come from a fixed bounded subset D of $\mathbb{Z}_q$, that is, the set of errors do not span the entire set $\mathbb{Z}_q$. Ding's attack is polynomial-time in the dimension n, thus if we are to generate LWE samples, it is very important that the error distribution is supported on the entire set $\mathbb{Z}_q$.

3.5 Algebraic Number Theory

In this section, we will establish some algebraic number theory background to setup an improvement of the LWE problem, efficiency-wise for cryptographic applications, called the ring learning with errors RLWE problem. In particular, we will recall the notion of cyclotomic number fields, describe how algebraic number fields can be embedded as inner product space isomorphic to the Euclidean space $\mathbb{R}^n$ for some n which in consequence allows us to view ideals in a number field as lattices. We start with the following definition.

Definition 3.5.1. *An algebraic number field K, or simply a number field, is an extension field of $\mathbb{Q}$ of finite dimension when K is viewed as a vector space over $\mathbb{Q}$. The elements of K are the algebraic numbers, that is, the elements $\alpha \in \mathbb{C}$ whose minimal polynomial*

are in $\mathbb{Q}[X]$. The ring of integers $\mathcal{O}_K$ of a number field K consists of $\alpha \in K$ that is a root of an irreducible, monic polynomial in $\mathbb{Z}[X]$. The elements in $\mathcal{O}_K$ are called the algebraic integers in K.

Definition 3.5.2. *Let K be a number field and $\mathcal{O}_K$ be its ring of algebraic integers. An integral ideal of K is a subset $I \subseteq \mathcal{O}_K$ that is closed under addition and that for all $r \in \mathcal{O}_K$ and $x \in I$, we have that $rx \in I$.*

Definition 3.5.3. *A fractional ideal of a number field K is a subset J of K such that there exists nonzero $d \in \mathcal{O}_K$ such that $dJ \subseteq \mathcal{O}_K$.*

It is a standard fact in algebraic number theory that the set of fractional ideals of a number field form a multiplicative group, that is, it is a group under multiplication of ideals and the ring of integers $\mathcal{O}_K$ serves as the identity element.

Definition 3.5.4. *Given a positive integer m, an element $\zeta \in \mathbb{C}$ is called a primitive mth root of unity if ζ is a root of $X^m - 1 = 0$ and for all $1 \leq m' < m$, $\zeta^{m'} \neq 1$. The mth cyclotomic field is the number field $\mathbb{Q}(\zeta_m)$, that is the smallest field containing $\mathbb{Q}$ and ζ_m, where ζ_m is a primitive mth root of unity. The dimension of $\mathbb{Q}(\zeta_m)$ over $\mathbb{Q}$ can be shown to be $n = \varphi(m)$, where φ is the Euler totient function.*

From now on, we only consider cyclotomic number fields $\mathbb{Q}(\zeta_{2n})$, where n is a power of two. Hence, the degree of the algebraic extension $\mathbb{Q}(\zeta_{2n})$ over $\mathbb{Q}$ is given by $n = \varphi(2n)$. Moreover, the polynomial $X^n + 1$ is irreducible over $\mathbb{Q}$ since it is the minimal polynomial of the $2n$th primitive root of unity ζ_{2n} and so the $2n$th cyclotomic field $K = \mathbb{Q}(\zeta_{2n})$ is isomorphic to $\mathbb{Q}[X]/(X^n + 1)$ as fields, and thus as vector spaces over $\mathbb{Q}$. Moreover, the ring of integers $\mathcal{O}_K$ is $\mathbb{Z}[\zeta_{2n}]$ and thus, $\mathcal{O}_K$ is isomorphic to $R := \mathbb{Z}[X]/(X^n + 1)$ as rings, as well as $\mathbb{Z}-$modules. Notice that R is isomorphic to $\mathbb{Z}^n$ via the coefficient embedding, that is, given a polynomial $a = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1} \in R$, a is mapped to $(a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}^n$. Another way to embed K into $\mathbb{C}^n$ is via the canonical embedding. We define the canonical embedding below.

Definition 3.5.5. *Let $K = \mathbb{Q}(\zeta)$, where $\zeta = \zeta_m$ is a primitive $m-$th root of unity, be a cyclotomic field of dimension $n = \varphi(m)$ over $\mathbb{Q}$. Let $\{\sigma_k\}_{k\in\mathbb{Z}_m^\times}$ be the set of embeddings $\sigma_k : K \to \mathbb{C}$, where $\sigma_k(\zeta) = \zeta^k$ and $\mathbb{Z}_m^\times := \{a \in \mathbb{Z}_m : \gcd(a, m) = 1\}$. Define the canonical embedding σ of K into $\mathbb{C}^n$ to be the map that takes ζ into an $n-$tuple $(\sigma_k(\zeta))_{k\in\mathbb{Z}_m^\times}$.*

If n is a power of two, notice that $K = \mathbb{Q}(\zeta_{2n})$ is isomorphic to $\mathbb{Q}[X]/(X^n + 1)$ as fields via the map $\zeta := \zeta_{2n} \mapsto X$. Hence, for any element $\alpha \in K$, with $\alpha = \sum_{i=0}^{n-1} a_i\zeta^i$, we can identify α as the polynomial $f_\alpha(X) = \sum_{i=0}^{n-1} a_iX^i$ in $\mathbb{Q}[X]/(X^n + 1)$. As a consequence, $\sigma(\alpha) = (f_\alpha(\zeta^k))_{k\in\mathbb{Z}_{2n}^\times}$.

Furthermore, notice that given $\alpha \in K$, identified as the polynomial $\sum_{i=0}^{n-1} a_iX^i$ we can relate the coefficient vector and embedding representation of α by the following equation:

$$(\alpha(\zeta)), \ldots, \alpha(\zeta_{2n-1}))^T = \mathrm{V}_n \cdot (a_0, a_1, \ldots, a_{n-1})^T$$

where V_n is the $n \times n$ Vandermonde matrix

$$\begin{pmatrix} 1 & \zeta & \zeta^2 & \cdots & \zeta^{n-1} \\ 1 & \zeta^3 & \zeta^{3\cdot 2} & \cdots & \zeta^{3(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^{2n-1} & \zeta^{2\cdot(2n-1)} & \cdots & \zeta^{(2n-1)(n-1)} \end{pmatrix} \tag{3.5.1}$$

Definition 3.5.6. *The trace of an element $\alpha \in K$, denoted $\mathrm{Tr}(a)$ is defined as*

$$\mathrm{Tr}(\alpha) = \sum_{k\in\mathbb{Z}_{2n}^\times} \sigma_k(a)$$

and the (algebraic) norm of α is defined as

$$\mathrm{N}(\alpha) = \prod_{k\in\mathbb{Z}_{2n}^\times} \sigma_k(\alpha)$$

where both sum and product is over all the n embeddings of K into $\mathbb{C}^n$.

Definition 3.5.7. *For $x \in \mathbb{C}^n$, we define the ℓ_2-norm as $\|x\| = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}$, where $x = (x_1, \ldots, x_n)$ and the $\ell_\infty-$norm as $\|x\|_\infty = \max_i |x_i|$.*

Definition 3.5.8. *Let $s_1, s_2 \in \mathbb{N} \cup \{0\}$ such that $s_1 + 2s_2 = n$. Define the space $H \subset \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2}$ by*

$$H = \{(x_1, \ldots, x_n) \in \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} : x_{s_1+s_2+j} = \overline{x_{s_1+j}}, \forall j \in \{1, \ldots, s_2\}\}.$$

It is a standard fact that H is isomorphic to $\mathbb{R}^n$ as an inner product space over $\mathbb{R}$, where the inner product on H is induced from $\mathbb{C}^n$. Also, since the embeddings σ_k on K satisfy $\overline{\sigma_k} = \sigma_{2n-k}$, we may identify the cyclotomic field K with $\sigma(K) \subset H$. In other words, we endow K with a geometry via the canonical embedding. Also, the canonical embedding σ is a ring homomorphism between K and H, where the addition and multiplication operations on H are coordinate-wise.

Definition 3.5.9. *The (geometric) ℓ_2-norm of an element $\alpha \in K$, denoted $\|\alpha\|$ is defined by $\|\alpha\| = \|\sigma(\alpha)\|$, where the right hand side is the ℓ_2-norm on H. Replace $\|\cdot\|$ by $\|\cdot\|_\infty$, we have a similar definition of the $\ell_\infty-$norm of an element $\alpha \in K$.*

Remark 3.5.10. It was shown in [Lyubashevsky and Seiler 2018] that the matrix norm of V_n defined in Equation 3.5.1, is equal to $\sqrt{n}$, thus, $\|\sigma(\alpha)\| \leq \sqrt{n}\|(a_0, \ldots, a_{n-1})\|$., where $\sigma(\alpha)$ is the canonical embedding representation of $\alpha \in K$. In other words, the canonical embedding norm is just $\sqrt{n}$ times the coefficient embedding norm of an element in the $2n-$th cyclotomic field K, where n is a power of two.

Let $e_j \in \mathbb{C}^n$, $j = 1, \ldots, n$ be the $n-$dimensional vector whose $j-$th coordinate is 1 and 0 elsewhere. From [Lyubashevsky, Peikert, et al. 2013], we note that H has basis given by $h_j \in \mathbb{C}^n$, where $h_j = e_j$ for $j = 1, \ldots, s_1$, and $h_j = \frac{1}{\sqrt{2}}(e_j + e_{j+s_2})$ and $h_{j+s_2} = \frac{i}{\sqrt{2}}(e_j - e_{j+s_2})$ for $j = s_1 + 1, \ldots, s_1 + s_2$, where $n = s_1 + 2s_2$.

Now, let$r > 0$. We can define the Gaussian function on H to be $\rho_r(x) = e^{-\pi\|x\|^2/r^2}$ and define the continuous Gaussian distribution D_r on H whose density function is $r^{-n}\rho_r(x)$.

Definition 3.5.11. *Let* $\mathrm{r} = (r_1, \ldots, r_n) \in (\mathbb{R}^+)^n$ *such that* $r_{s_1+s_2+j} = r_{s_1+j}$ *for* $j \in \{1, \ldots, s_2\}$. *The elliptical Gaussian distribution* D_{r} *on* H *is a distribution which samples a vector* $\sum_{i=1}^{n} a_i h_i$, *where each* a_i *is sampled independently from the Gaussian distribution* D_{r_i} *on* $\mathbb{R}$ *and* $\{h_i\}$ *is the basis above for* H.

Remark 3.5.12. *It is easy to check that for any* $a, b \in K$, *we have* $\|a \cdot b\| \leq \|a\|_\infty \|b\|$. *Also, observe that*

$$\mathrm{Tr}(a \cdot b) = \sum_k \sigma(a)\sigma(b) = \langle \sigma(a), \overline{\sigma(b)} \rangle.$$

In other words, the trace map can be viewed as an inner product on K.

For any fractional ideal I of K, I has a $\mathbb{Z}-$ basis, that is, there exists, $u_1, \ldots, u_n \in K$ such that $I = \mathbb{Z}u_1 + \cdots + \mathbb{Z}u_n$. Under the canonical embedding σ, I embeds as a lattice $\sigma(I)$ in H.

Definition 3.5.13. *In the above discussion, for any fractional ideal* I, *we define* $\sigma(I)$ *to be an ideal lattice.*

With some abuse of notation, we use the notation I to also denote the ideal lattice $\sigma(I)$ so that as a geometric object, we can also talk about the length of the shortest nonzero vector in I, determinant of I, and other geometric properties of a lattice.

Definition 3.5.14. *The (ideal) norm of an integral ideal* I *of* K *is defined to be* $\mathrm{N}(I) = |R/I|$, *the size of the quotient ring* R/I. *We can extend the definition of the norm to fractional ideals, that is, if* J *is a fractional ideal of* K *with* $d \in \mathcal{O}_K$ *such that* $dJ \subseteq \mathcal{O}_K$, *then we define* $\mathrm{N}(J) = N(dJ)/|N(d)|$.

Hence, we may view the ideal norm map as a group homomorphism between the multiplicative group of fractional ideals in K and the nonzero elements of the rationals $\mathbb{Q}$ since it is known that the ideal norm and the algebraic norm are multiplicative.

Definition 3.5.15. *We define the discriminant of K, denoted by Δ_K to be* $(\det(\mathcal{O}_K))^2$.

In the case of the cyclotomic field $K = \mathbb{Q}(\zeta)$, $\Delta_K = \det(\sigma_j(\zeta^i))^2$, where $\zeta = \zeta_{2n}$, $i \in \{0, \ldots, n-1\}$ and $j \in \mathbb{Z}_{2n}^{\times}$.

In fact, for any $m-$th cyclotomic field K of dimension $n = \varphi(m)$, it is known that

$$\Delta_K = \frac{m^n}{\prod\limits_{\text{prime } p|m} p^{n/(p-1)}} \leq n^n$$

and if m is a power-of-two, we actually have equality, that is $\Delta_K = n^n$.

Remark 3.5.16. *Let I be a fractional ideal in K. It can be shown that* $\det(I) = \mathrm{N}(I)\sqrt{\Delta_K}$.

As a consequence of the above remark, we have the following approximation of the length of the shortest nonzero vector on any ideal lattice I of a number field K of degree n:

$$\sqrt{n}\mathrm{N}(I)^{1/n} \leq \lambda_1(I) \leq \sqrt{n}(\mathrm{N}(I)\sqrt{\Delta_K})^{1/n}.$$

The lower bound can be shown using the AM-GM (arithmetic mean-geometric mean) inequality while the upper bound is a direct application of Minkowski's theorem.

This also tells us that GapSVP on ideal lattices is easy since we can approximate $\lambda_1(I)$ to within a factor $\sqrt{\Delta_K}^{1/n}$ using the approximation $\sqrt{n}(\mathrm{N}(I)\sqrt{\Delta_K})^{1/n}$.

Now, for brevity, when we refer to ideals in K, we mean fractional ideals. Since ideals in K can be viewed as a lattice in H, we may speak of the dual of the ideal lattice I.

Definition 3.5.17. *For any fractional ideal I in K, we define the dual ideal $I^\vee$ of I to be*

$$I^\vee := \{a \in K : \mathrm{Tr}(aI) \subseteq \mathbb{Z}\}.$$

As a special case when $I = \mathcal{O}_K$, we call $\mathcal{O}_K^\vee$ to be the codifferent ideal and $(\mathcal{O}_K^\vee)^{-1}$ to be the different ideal. It can be shown that $I^\vee = I^{-1}\mathcal{O}_K^\vee$, where I^{-1} is the multiplicative

inverse of I in the multiplicative group of fractional ideals in K. More precisely, $I^{-1} = \{a \in K : aI \subseteq \mathcal{O}_K\}$. Since $\sqrt{\Delta_K} = \det(\mathcal{O}_K)$, then $1/\sqrt{\Delta_K} = \det(\mathcal{O}_K^\vee)$. As a consequence, $\mathrm{N}(\mathcal{O}_K^\vee) = \Delta_K^{-1}$.

Now, if we let m to be a power-of-two, $n = m/2$ and $\mathcal{O}_K = \mathbb{Z}[\zeta_m]$ to be the ring of integers of the $m-$th cyclotomic field, then by referring to [Lyubashevsky, Peikert, et al. 2013] we have, $\mathcal{O}_K^\vee = \frac{1}{n}\mathcal{O}_K$.

Let's state the Chinese Remainder Theorem for rings which will be useful later on.

Lemma 3.5.18. *Let $I_1, \ldots, I_r$ be pairwise relatively prime ideals of a ring R, that is, $I_i + I_j = R$ for $i \neq j$. Let $I = \prod_{i=1}^{r} I_i$. Then the natural ring homomorphism $R \to \prod_{i=1}^{r} R/I_i$ induces a ring isomorphism $R/I \cong \prod_{i=1}^{r} R/I_i$.*

Now, let's consider a prime $q \equiv 1 \bmod 2n$, n is a power-of-two. Also, let $R = \mathcal{O}_K$, K is the $2n-$th cyclotomic field. We may also view R to be $\mathbb{Z}[X]/(X^n + 1)$. Consider the principal ideal

$$qR := \{q\alpha : \alpha \in R\}.$$

A standard result in algebraic number theory about the splitting of the ideal qR into prime ideals in R, says that

$$qR = \prod_{i \in \mathbb{Z}_{2n}^\times} \mathfrak{q}_i,$$

where $\mathfrak{q}_i = qR + (\zeta_{2n} - \omega_{2n}^i)R$ are distinct prime ideals in R of norm q, where ω_{2n} is a primitive $2n-$th root of unity modulo q.

By the Chinese Remainder Theorem, we have the isomorphism of rings

$$R/qR \cong \prod_{i \in \mathbb{Z}_{2n}^\times} R/\mathfrak{q}_i \cong \mathbb{Z}_q^n.$$

Denote $R_q := R/qR$. The above isomorphism allows us to view polynomials $a \in R_q$ as $n-$tuples $\left(a(\omega_{2n}^i)\right)_{i \in \mathbb{Z}_{2n}^\times}$.

3.6 Ideal Lattice Problems

Now, we state the analogous hard lattice problems on ideal lattices. Here, we will skip the analog of GapSVP since it is easy on ideal lattices. Hence, we will only define the analogues of SVP, SIVP and BDD on ideal lattices.

Definition 3.6.1. *Let K be a number field endowed with some geometric norm (e.g. either the ℓ_2 or ℓ_∞ norm), and let $\gamma \geq 1$. The $K - \mathsf{SVP}_\gamma$ problem in the given norm is: given a fractional ideal I in K, find some nonzero $x \in I$ such that $\|x\| \leq \gamma\lambda_1(I)$. The $K - \mathsf{SIVP}_\gamma$ problem asks to find n linearly independent elements in I such that all their norms are at most $\gamma\lambda_n(I)$.*

Definition 3.6.2. *Let K be a number field endowed with some geometric norm (e.g. either the ℓ_2 or ℓ_∞ norm). Let I be a fractional ideal in K, and let $d < \lambda_1(I)/2$. The $K - \mathsf{BDD}_\gamma$ problem in the given norm is: given I and $y \in K$ of the form $y = x + e$ for some $x \in I$ and $e \in K$ with $\|e\| \leq d$, find x.*

3.7 Ring learning with errors

We define the ring learning with errors, denoted RLWE problem as in [Lyubashevsky, Peikert, et al. 2013]. First, let K be a number field with $R = \mathcal{O}_K$ be its ring of algebraic integers. Let $q \geq 2$ be an integer. Let $\mathbb{T} = K_\mathbb{R}/R^\vee$, where $K_\mathbb{R} = K \otimes_\mathbb{Q} \mathbb{R}$ which can be thought of as changing the scalar of the $\mathbb{Q}-$vector space K into $\mathbb{R}$, and we also note that $K_\mathbb{R}$ is isomorphic to H as $\mathbb{R}-$ inner product spaces. Let $R_q := R/qR$ and $R_q := R^\vee/qR^\vee$.

Definition 3.7.1. *For $s \in R_q^\vee$ and an error distribution ψ over H (or $K_\mathbb{R}$), a sample from the* RLWE *distribution denoted $A_{s,\psi}$ over $R_q \times \mathbb{T}$ is generated by choosing $a \in R_q$ uniformly at random, choosing e from ψ, and outputting $(a, b = (a \cdot s)/q + e \bmod R^\vee)$.*

Notice that since $s \in R_q^\vee$ and $R^\vee$ is a fractional ideal, then $(a \cdot s)/q \in \frac{1}{q}R^\vee/R^\vee$ so adding to it $e \in \mathbb{R}$ modulo $R^\vee$ is well-defined.

Definition 3.7.2. *Let ψ be a distribution over H. The search version of* RLWE *problem asks to find the secret $s \in R_q^\vee$, given an access to arbitrarily many independent samples from $A_{s,\psi}$.*

Definition 3.7.3. *Let Ψ be a family of distributions over H. The (average-case) decision-*RLWE *problem asks the following: distinguish with non-negligible advantage between arbitrarily many samples from $A_{s,\psi}$ and the same number of uniform, independent samples from $R_q \times \mathbb{T}$, where $(s, \psi) \xleftarrow{\$} U(R_q) \times \Psi$.*

It was shown in [Lyubashevsky, Peikert, et al. 2013] that the search and decision versions of the RLWE problem are equivalent for cyclotomic number fields so that we will just assume that K is a cyclotomic field for the rest of the paper so that equivalence holds. Also, we will argue security using the hardness of the decision variant of the RLWE problem when we construct our voting protocol.

Moreover, if we additionally assume that K is a power-of-two cyclotomic, we have that $R^\vee = \frac{1}{n}R$, so that we may choose the secret s to be in R_q by transforming samples $(a, b = (a \cdot s)/q + e)$ into samples $(a, b' = (a \cdot s')/q + e')$, where $s' = ns \in R_q$ and $e' = ne$.

Hence, the RLWE problem can be restated in the power-of-two cyclotomic case as distinguishing between samples from $A_{s,\psi}$ and from uniform ones, where $s \in R_q$ and ψ is a distribution on H.

Now, before we state the hardness result of the RLWE problem, we first define a specific family of elliptical distributions over H for which the hardness result holds.

Definition 3.7.4. *Let K be an $m-$th cyclotomic field with degree $n = \varphi(m)$. Let $\alpha > 0$. Define the distribution Ψ_α to be the distribution on the family of elliptical distributions D_{r} over H where* r *is chosen such that $r_i^2 = r_{i+n/2}^2 = \alpha^2(1 + \sqrt{n}x_i)$, where each x_i, $i = 1, \ldots, n/2$, are chosen independently from the gamma distribution, $\Gamma(2, 1)$ whose shape parameter is 2 and scale parameter 1.*

Hence, on average, if we sample an elliptical Gaussian distribution from Ψ_α, the parameters r_i are roughly $O(\alpha n^{1/4})$.

Before we give the hardness result, we denote a function $f(n)$ to be $\omega(g(n))$ if $f(n)$ grows asymptotically faster than $g(n)$. Hence, we may think of $\omega(g(n))$ as a function that is at least $g(n)$ for sufficiently large n.

Now, we state the hardness result for the RLWE problem for specific parameters:

Theorem 3.7.5 (Lyubashevsky, Peikert, et al. 2013)**.** *Let K be the mth cyclotomic number field with dimension $n = \varphi(m)$ and $R = \mathcal{O}_K$ be its ring of integers. Let $0 < \alpha < \sqrt{\log n/n}$, and let $q = q(n) \geq 2, q = 1 \mod m$ be a poly(n)-bounded prime such that $\alpha q \geq \omega(\sqrt{\log n})$. Then there is a polynomial-time quantum reduction from $\tilde{\mathcal{O}}(\sqrt{n}/\alpha)$-approximate* SIVP *(or* SVP*) on ideal lattices in K to the problem of solving decision-*RLWE *given only ℓ samples from $A_{s,\psi}$ where ψ is the Gaussian distribution $D_{\xi q}$ and s uniformly random chosen from $R_q^\vee$, for $\xi = \alpha \cdot (n\ell/\log(n\ell))^{1/4}$.*

Notice that assumption on $\alpha < \sqrt{\log n/n}$ which is required so that the RLWE problem is not impossible to solve. If we set $\alpha \geq \sqrt{\log n/n}$, then RLWE samples are indistinguishable from uniform since the error parameter would be greater than the smoothing parameter of $R^\vee$. Also, the choice for q so that $\alpha q \geq \omega(\sqrt{\log n})$ is important to resist polynomial time attacks to the LWE (hence, RLWE as well) problem and finally, q assumed to be a polynomially bounded integer in n makes the approximation factor in the worst-case SIVP problem in ideal lattices to be polynomial in n as well. We note that SIVP within polynomial approximation factors are believed to be hard as well since the best available algorithms for these approximation factors require exponential time such as the LLL algorithm. Finally, observe that the RLWE problem only relies on the hardness of SIVP and not on GapSVP since as we mentioned earlier, GapSVP is easy on ideal lattices.

Moreover, since we will pick $q = 3 \mod 8$, q a polynomially-bounded prime, we need an additional hardness result which states that there is a reduction from the RLWE problem with modulus q to the RLWE problem with modulus p. Combining the reduction theorem

below from [Langlois and Stehlé 2012], which the authors call the "modulus-switching self-reduction" for RLWE and Theorem 3.7.5, we see that solving the RLWE problem for any modulus q is at least as hard as solving any (worst-case) approximate SIVP or SVP problem on ideal lattices, with appropriate choices of parameters. In particular,

Theorem 3.7.6 (Langlois and Stehlé 2012)**.** *If* $\beta \geq \alpha \cdot \max(1, q/p) \cdot \tilde{\Omega}(n^{3/4})$ *and* $\alpha q = \tilde{\Omega}(\sqrt{n})$*, then there is a (classical) reduction from* $\mathsf{RLWE}_{q,D_{\alpha q}}$ *to* $\mathsf{RLWE}_{p,D_{\beta p}}$*.*

Finally we mention an important lemma from [Ding, Xie, et al. 2012]:

Lemma 3.7.7 (Ding, Xie, et al. 2012, lemma 2)**.** *For* $a, b \in R_q$*,* $\|a \cdot b\|_\infty \leq \|a\| \cdot \|b\|$*.*

3.8 Rejection Sampling

In this section, we will describe the rejection sampling technique which will be a vital tool when we employ our zero-knowledge proofs in our voting protocol. The rejection sampling technique that we will follow comes from [Lyubashevsky 2012] where the goal is to construct an algorithm whose output follows the distribution of a discrete Gaussian distribution on $\mathbb{Z}^n$ centered at 0. The basic idea is that we will sample from a discrete Gaussian distribution on some coset $v + \mathbb{Z}^n$ and only accept this sample according to a certain acceptance criteria.

In our zero-knowledge proof later in the following section, the discrete Gaussian distribution on a coset $v + \mathbb{Z}^n$ corresponds to some secret of the prover and the rejection sampling method allows the prover to hide the information about the secret since by rejection sampling, the information about the secret is lost since the output follows the discrete Gaussian distribution on $\mathbb{Z}^n$. This gives the prover confidence that no information about the secret is leaked so that no verifier learns the prover's secret information. We follow the rejection sampling technique in [Lyubashevsky 2012].

For simplicity, we denote D_σ to be the discrete Gaussian distribution on $\mathbb{Z}^n$ with parameter σ and denote $D_{v,\sigma}$ to be the discrete Gaussian distribution on the coset $v+\mathbb{Z}^n$, where $v \in \mathbb{R}^n$.

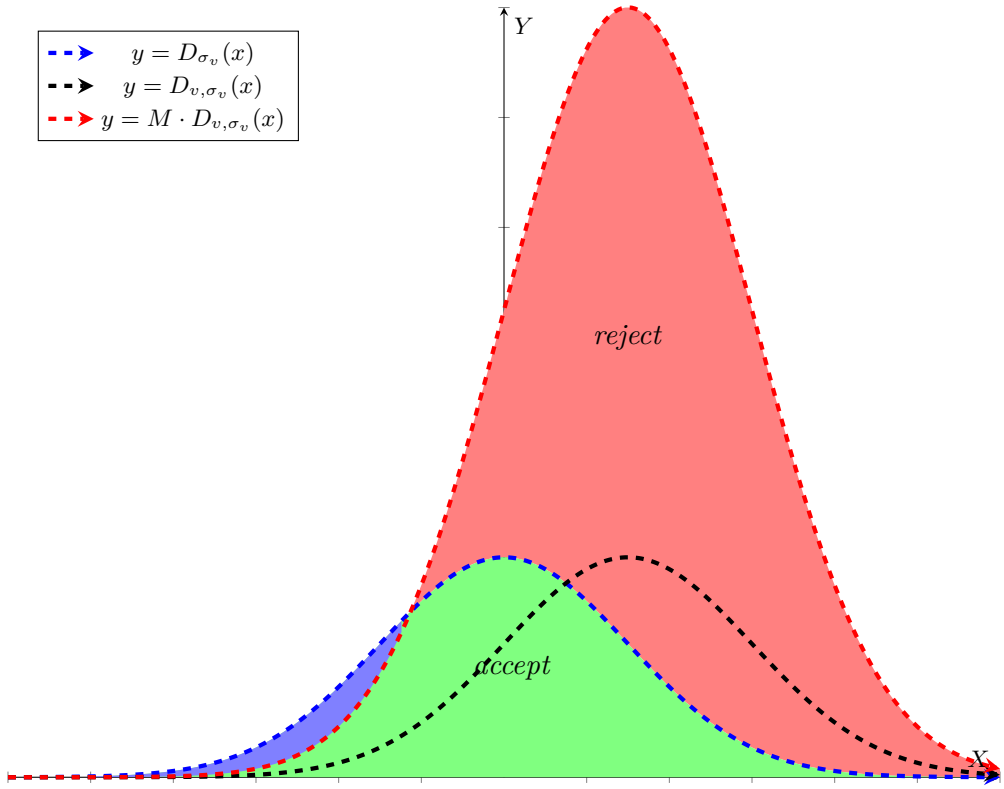


Figure 3.2: The green shaded area represents the region where a sample from D_{v,σ_v} is accepted while the red shaded area represents the rejection region. The blue shaded area is also part of the acceptance region but the samples from D_{v,σ_v} gets accepted with probability 1 since its image on the red dotted curve is always less than its image on the blue dotted curve.

First, the following lemma states that with high probability, namely with probability greater than $1-2^{-100}$, there exists $M > 0$ such that if $z \xleftarrow{\$} D_{\sigma_v}$ then $D_{\sigma_v}(z) < MD_{v,\sigma_v}(z)$, where $v \in \mathbb{Z}^n$ and σ_v is a parameter that depends on v.

Lemma 3.8.1 (Lyubashevsky 2012)**.** *For any $v \in \mathbb{Z}^n$, if $\sigma_v = \alpha\|v\|$, for any $\alpha > 0$ then*

$$\Pr[D_{\sigma_v}(z)/D_{v,\sigma_v}(z) < M : z \xleftarrow{\$} D_{\sigma_v}] > 1 - 2^{-100}, \tag{3.8.1}$$

where $M = e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}}$.

The existence of such M in the preceding lemma allows us to employ the rejection sampling technique and the following theorem states the distribution of samples from rejection sampling algorithm is statistically close to the distribution of samples from algorithm which are drawn from the standard discrete Gaussian distribution with probability $\frac{1}{M}$.

Theorem 3.8.2 (Lyubashevsky 2012)**.** *Let* $V \subset \mathbb{Z}^n$ *in which all elements of* V *have norm less than* $T > 0$, $\tilde{\sigma} \in \mathbb{R}$ *such that* $\tilde{\sigma} = \omega(T\sqrt{\log n})$, *and* $h : V \to \mathbb{R}$ *be a probability distribution. Then there exists* $M > 0$ *such that the following algorithm* $\mathcal{A}$*:*

1. $v \xleftarrow{\$} h$
2. $z \xleftarrow{\$} D_{v,\tilde{\sigma}}$
3. *output* (z, v) *with probability* $\min\left(\frac{D_{\tilde{\sigma}}(z)}{MD_{v,\tilde{\sigma}}(z)}, 1\right)$.

is within statistical distance $\frac{2^{-\omega(\log n)}}{M}$ *of the distribution of the following algorithm* $\mathcal{F}$*:*

1. $v \xleftarrow{\$} h$
2. $z \xleftarrow{\$} D_{\tilde{\sigma}}$
3. *output* (z, v) *with probability* $\frac{1}{M}$.

Moreover, the probability that $\mathcal{A}$ *outputs something is at least* $\frac{1-2^{-\omega(\log n)}}{M}$. *More concretely, if* $\tilde{\sigma} = \alpha T$ *for any positive* α, *then* $M = e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}}$, *the output of algorithm* $\mathcal{A}$ *is within statistical distance* $\frac{2^{-100}}{M}$ *of the output of* $\mathcal{F}$, *and the probability that* $\mathcal{A}$ *outputs something is* $\frac{1-2^{-100}}{M}$.

How algorithms $\mathcal{A}$ and $\mathcal{F}$ precisely work:

The idea behind the rejection sampling algorithm $\mathcal{A}$ is to sample from a so-called proposal distribution $D_{v,\tilde{\sigma}}$ but this sample looks like it was drawn from $D_{\tilde{\sigma}}$. So step 1 of $\mathcal{A}$ samples $v \xleftarrow{\$} h$ which serves as the center of $D_{v,\tilde{\sigma}}$. By the previous lemma, we can find such $M > 0$ such that with high probability, if $z \xleftarrow{\$} D_{\tilde{\sigma}}$, then $D_{\tilde{\sigma}}(z) < MD_{v,\tilde{\sigma}}(z)$. In step 2, it samples $z \xleftarrow{\$} D_{v,\tilde{\sigma}}$ and we can split the acceptance criteria into two cases:

- if $D_{\tilde{\sigma}}(z) \geq MD_{v,\tilde{\sigma}}(z)$, output z;
- otherwise if $D_{\tilde{\sigma}}(z) < MD_{v,\tilde{\sigma}}(z)$, sample $u \xleftarrow{\$} \mathcal{U}$, where $\mathcal{U}$ is the uniform distribution on $[0,1]$ and if $u < \dfrac{D_{\tilde{\sigma}}(z)}{MD_{v,\tilde{\sigma}}(z)}$, output z, otherwise reject and repeat step 2. In this case, the probability that $\mathcal{A}$ outputs z is $\dfrac{D_{\tilde{\sigma}}(z)}{MD_{v,\tilde{\sigma}}(z)}$.

Hence, in both cases, the probability that $\mathcal{A}$ outputs z is $\min\left(\dfrac{D_{\tilde{\sigma}}(z)}{MD_{v,\tilde{\sigma}}(z)}, 1\right)$.

For step 2 in algorithm $\mathcal{F}$, sample $z \xleftarrow{\$} D_{\tilde{\sigma}}$ and draw $u \xleftarrow{\$} \mathcal{U}$, where $\mathcal{U}$ is the uniform distribution on $[0,1]$. If $u < \dfrac{1}{M}$, output (z, v), otherwise reject and repeat step 2.

3.9 Cryptographic Primitives

3.9.1 Commitment Schemes

In our protocol, the encryption of secret keys and votes are actually commitments and we show that these commitments enjoy perfectly binding and computationally hiding properties. These properties are important which allows an honest verifier to trust that the voter/prover follows our voting protocol. Below we recall the definition of a commitment scheme and also include some lemmas which are essential in proving the hiding and binding properties that we claim above.

Definition 3.9.1. *A commitment scheme consists of three algorithms* $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Ver})$ *such that:*

- *On input* 1^n*, the key generation algorithm* KGen *outputs a public commitment key* pk*.*
- *The commitment algorithm* Com *takes as input a message* m *from a message space* $\mathcal{M}$ *and a commitment key* pk*, and outputs a commitment/opening pair* (c, d)
- *The verification algorithm* Ver *takes a key* pk*, message* m*, a commitment* c *and an opening* d*, and outputs accept or* $reject$*.*

In addition, a commitment scheme has to satisfy the following security requirements:

1. *Correctness:* Ver *outputs accept whenever the inputs were computed by an honest party, that is,*

$$\Pr[\mathsf{Ver}(pk, m, c, d) = accept : pk \leftarrow \mathsf{KGen}(1^n), m \in \mathcal{M}, (c, d) \xleftarrow{\$} \mathsf{Com}(m, pk)] = 1.$$

2. *Binding: A commitment cannot be opened to different messages. Below are two types of binding properties:*
   - *Perfect: With overwhelming probability over the choice of public key* $pk \leftarrow \mathsf{KGen}(1^n)$*, we have that*

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

   - *Computational: If no PPT (probabilistic, polynomial-time) adversary can come up with a commitment and two different openings, that is, for every PPT adversary* A*, there is a negligible function* $negl$ *on* n *such that*
     $\Pr[\mathsf{Ver}(pk, m, c, d) = \mathsf{Ver}(pk, m', c, d') : pk \leftarrow \mathsf{KGen}(1^n), (c, m, m', d, d') \leftarrow \mathsf{A}(pk)]$
     $\leq negl(n)$.

3. *Computationally Hiding: A commitment computationally hides the committed message, that is, any adversary is restricted to polynomial-time and has negligible advantage in identifying the message in the commitment.*

Before we give our simple commitment scheme whose hiding property is based on the hardness of the decision RLWE problem, we mention the following simple lemma which states that a uniformly random chosen element in R_q has small coefficients with small probability.

Lemma 3.9.2. *The probability that a uniformly chosen random element $r \in R_q$ has max norm less than or equal to $\beta \geq 1$ is given by*

$$\Pr[\|x\|_\infty \leq \beta \colon x \leftarrow R_q] \leq \frac{(2\beta+1)^n}{q^n}.$$

Proof. Since the coefficients of the polynomials in $R_q = R/qR$, where $R = \mathbb{Z}[X]/(X^n + 1)$ are chosen to be in the interval $[-\frac{q-1}{2}, \frac{q-1}{2}]$, then the number of elements in R_q whose infinity norm is less than β is given by $(2\beta+1)^n$. The result follows. ■

Now, another important lemma from [Stehlé et al. 2009] states that $X^n + 1$ for n a power of two splits into two irreducible polynomials modulo q where $q \equiv 3 \bmod 8$.

Lemma 3.9.3 (Stehlé et al. 2009)**.** *Let $k \geq 0$ and $n = 2^k$. Then $f(X) = X^n + 1$ is irreducible in $\mathbb{Q}[X]$. If $k \geq 2$ and q is a prime with $q \equiv 3 \pmod 8$, then $f = f_1 f_2 \pmod q$, where for any $i \in \{1, 2\}$, we have that f_i is irreducible in $\mathbb{Z}_q[X]$ and can be written as $f_i = X^{n/2} + t_i X^{n/4} - 1$ for some $t_i \in \mathbb{Z}_q$.*

The next lemma comes from [Lyubashevsky and Seiler 2018], which gives a sufficient condition on the Euclidean norm of an element in R_q so that it is invertible in R_q. Here, denote by $\|y\|$ to be the coefficient embedding norm and $\|y\|_e$ to be the canonical embedding norm of an element $y \in R_q$.

Lemma 3.9.4 (Lemma 3.1, Lyubashevsky and Seiler 2018)**.** *Let n be a power of two, q any prime and d be any integer such that*

$$X^n + 1 \equiv \prod_{i=1}^{d} f_i(X) \pmod{q} \tag{3.9.1}$$

for distinct polynomials $f_i(X)$ of degree n/d that are irreducible in $\mathbb{Z}_q[X]$, and let y be any element in R_q. If $0 < \|y\| < q^{1/d}$, then y is invertible in R_q.

Proof. Proceed by contradiction. Suppose y is not invertible. Then by the Chinese Remainder Theorem, there exists an i such that y has a zero coordinate in $\mathbb{Z}_q[X]/(f_i(X))$, that is, $f_i(X)$ divides y in $\mathbb{Z}_q[X]$.

Define $I := \{z \in R : z \mod (f_i(X), q) = 0\}$. We claim that I is an nonzero ideal of R. Notice that $y \in I$ and $y \neq 0$ by assumption. Also, note that $\sqrt{n}\|y\| \geq \|y\|_e \geq \sqrt{n} \cdot |N(y)|^{1/n}$. Thus, $\|y\| \geq |N(y)|^{1/n} \geq N(I)^{1/n}$. Now, we compute $N(I) = |R/I|$.

Notice that we can write I as $(f_i(X), q)$ an ideal of R. Moreover, we set $J = qR = (q)$, the principal ideal in R generated by q so that $J \subseteq I$. Hence,

$$\begin{aligned} R/I &\cong \frac{R/J}{I/J} \text{ (by third isomorphism theorem for rings)} \\ &= \frac{R_q}{f_i(X)R_q} \\ &\cong \frac{\mathbb{Z}_q[X]}{f_i(X)\mathbb{Z}_q[X]} \text{ (by third isomorphism theorem for rings).} \end{aligned}$$

But, $\dfrac{\mathbb{Z}_q[X]}{f_i(X) \cdot \mathbb{Z}_q[X]}$ is just the finite field $\mathbb{F}_{q^{n/d}}$, since deg $f_i = n/d$ and $f_i(X)$ is irreducible in $\mathbb{Z}_q[X]$. Thus, $N(I) = q^{n/d}$. Hence, $\|y\| \geq N(I)^{1/n} = q^{1/d}$ which contradicts the assumption on the norm of y. Therefore, y is invertible in R_q. ■

We note that for any $y \in R_q$, $\|y\| \leq \sqrt{n}\|y\|_\infty$, thus as a consequence of Lemma 3.9.3 above and Lemma 3.9.4, we have:

Lemma 3.9.5. *Let n be a power of two and $q \equiv 3 \pmod 8$ be a prime. Let $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$. If $y \in R_q$ such that $0 < \|y\|_\infty < \sqrt{\frac{q}{n}}$, then y is invertible in R_q.*

The above lemma about the invertibility of elements in R_q for n a power of two and $q \equiv 3 \bmod 8$ serves as an important ingredient in proving the binding property of our simple commitment scheme which we will now describe:

Commitment Scheme

Let n be a power of two integer, $q \equiv 3 \pmod 8$ a prime , $m \in \mathbb{N}$ and $D_{\mathbb{Z}^n,\sigma}$ be the discrete Gaussian distribution on $\mathbb{Z}^n$. Let the message space be the set $\{s \in R_q : \|s\|_\infty \leq \sigma\sqrt{n}\}$. The procedure of our commitment scheme is described below:

- KGen : The public commitment key $pk = a$ where $a \xleftarrow{\$} R_q$.
- Com : To commit to a message $s \in R_q$ with $\|s\|_\infty \leq \sigma\sqrt{n}$, the commitment algorithm draws $e \leftarrow D_{\mathbb{Z}^n,\sigma}$ conditioned on $\|e\|_\infty \leq \sigma\sqrt{n}$ (note that such an e is drawn with overwhelming probability by Lemma 3.3.13) and output $b = as + (m+1)e$ as the commitment for s. The opening information for b is given by the pair (s, e).
- Ver : Given commitment b and opening pair (s', e'), the verifier accepts if and only if $b = as' + (m+1)e'$, $\|s'\|_\infty, \|e'\|_\infty \leq \sigma\sqrt{n}$.

Proposition 3.9.6. *Let n be a power of two, $n > 16$, and let $D_{\mathbb{Z}^n,\sigma}$ be the discrete Gaussian distribution on $\mathbb{Z}^n$, $m \in \mathbb{N}$ and $q \equiv 3 \pmod 8$, q a poly(n)-prime such that $q > 4n\beta^2$, where $\beta = \sigma\sqrt{n}$. Then under the decision* RLWE *hardness assumption, the above commitment scheme is a computationally hiding and perfectly binding commitment scheme with overwhelming probability over the choices of the public commitment key.*

Proof. *Correctness.* This is trivial.

Computationally hiding. Under the decision-RLWE hardness assumption, then the commitment $b = as + (m+1)e$ is pseudorandom, that is, it is indistinguishable from uniform ones over R_q.

Binding. We need to show that given $b = as + (m+1)e$ with $\|s\|_\infty, \|e\|_\infty \leq \beta$, where $\beta = \sigma\sqrt{n}$ and if there exists $\|s'\|_\infty, \|e'\|_\infty \leq \beta$ such that

$$b = as + (m+1)e = as' + (m+1)e' \tag{3.9.2}$$

then $s = s'$, or alternatively, if

$$a(s - s') + (m+1)(e - e') = 0 \pmod{q} \tag{3.9.3}$$

then $s = s'$ with overwhelming probability over the choices of a in R_q. We will show that the probability that there exists $(\tilde{s}, \tilde{e}) \in R_q^2$ such that $0 < \|\tilde{s}\|_\infty, \|\tilde{e}\|_\infty \leq 2\beta$ and $a\tilde{s} + (m+1)\tilde{e} = 0 \pmod{q}$ over the choices of $a \in R_q$ is negligible in n. First, fix $\tilde{s}, \tilde{e} \in R_q$ such that $0 < \|\tilde{s}\|_\infty, \|\tilde{e}\|_\infty \leq 2\beta$. Since $q > 4n\beta^2$, then $2\beta < \sqrt{\frac{q}{n}}$. Then by Lemma 3.9.5, $\tilde{s}$ is an invertible element of R_q. Define the map $F_{\tilde{s},\tilde{e}} : R_q \to R_q$ given by $a \mapsto a\tilde{s} + (m+1)\tilde{e}$. Since $\tilde{s}$ is invertible, it is not hard to show that $F_{\tilde{s},\tilde{e}}$ is an injective map on R_q. Thus,

$$\Pr[a\tilde{s} + (m+1)\tilde{e} = 0 \pmod{q} : a \xleftarrow{\$} R_q] = q^{-n} \tag{3.9.4}$$

Taking the union bound over all pairs $(\tilde{s}, \tilde{e}) \in R_q^2$ such that $0 < \|\tilde{s}\|_\infty, \|\tilde{e}\|_\infty \leq 2\beta$, the probability that there exists $(\tilde{s}, \tilde{e}) \in R_q^2$ such that $0 < \|\tilde{s}\|_\infty, \|\tilde{e}\|_\infty \leq 2\beta$ and $a\tilde{s} + (m+1)\tilde{e} = 0 \pmod{q}$ over the choices of $a \in R_q$ is given by $\frac{(4\beta+1)^{2n}}{q^n} < 2^{-n}$ since $q > 2(4\beta+1)^2$ (Indeed, $q > 4n\beta^2 > 50\beta^2 > 2(4\beta+1)^2$). This proves the binding property of the commitment scheme. ■

3.9.2 Σ'- Protocols and Proofs of Knowledge

In [Damgård 2010], Damgard introduced the notion of a $\Sigma-$protocol between a prover and verifier where a prover convinces the verifier that he knows some secret information without revealing any information about the secret other than the claim that the prover knows the secret. In [Benhamouda, Camenisch, et al. 2014], they have introduced the notion of $\Sigma'-$protocol which is a weaker version of Damgard's $\Sigma-$protocol. For our voting protocol, we only need this weaker version since we employ rejection sampling to prove knowledge of votes without revealing what the vote really is. Before we give the definition of a $\Sigma'-$protocol, we need some preliminary background..

First, the notation $\{0,1\}^*$ means the set of bit strings of arbitrary length.

Definition 3.9.7. *A (binary) language is a subset $\mathcal{L} \subseteq \{0,1\}^*$.*

Definition 3.9.8. *A (binary) relation is a subset $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ such that if $(x,w) \in \mathcal{R}$, then $|w| \leq p(|x|)$, where $|\cdot|$ denotes the bit string length and $p(\cdot)$ is any polynomial function.*

For some $(x,w) \in \mathcal{R}$, x can be thought of as an instance of a computational problem and w as the solution of that instance. We call w a witness of x in this case.

Σ-protocols

Now, we will define what a $\Sigma-$protocol is which is a protocol that involves a prover P and a verifier V with the goal that P wishes to convince V that he knows a witness w for some $(x,w) \in \mathcal{R}$, $\mathcal{R}$ a relation without revealing any information about w. Also, denote by $A(z)$ to be the output of an algorithm A on input z.

Consider the following protocol, let's denote it by $\mathcal{P}$: Assume that P and V are polynomial-time algorithms where P denotes the prover and V denotes the verifier. Let x be a common input to both P and V, and w be a private input to P such that $(x,w) \in \mathcal{R}$. Let t be some positive integer be the size of the challenge set $\mathcal{C}$, that is $\mathcal{C} = \{0,1\}^t$.

The protocol $\mathcal{P}$ works in the following steps:

1. P sends a message $a := P(x, w)$ to V.

2. V sends a uniformly random bit string e of length t to P as challenge.

3. P sends a reply $z := P(x, w, a, e)$ and V decides to accept or reject, that is either $V(x, a, e, z) = 1$ or $V(x, a, e, z) = 0$.

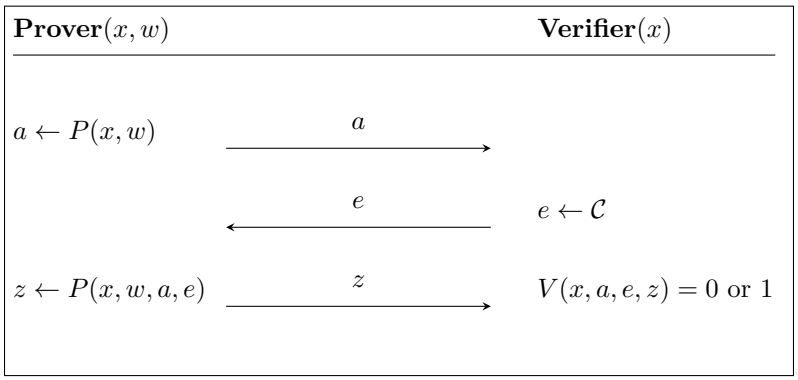


Figure 3.3: By convention, the output 0 of verifier V denotes V rejects the proof while output 1 denotes V accepts it.

Definition 3.9.9. *A protocol $\mathcal{P}$ is said to be a $\Sigma-$protocol for a relation $\mathcal{R}$ if $\mathcal{P}$ is of the form in Figure 3.3 such that $\mathcal{P}$ satisfies the following three properties:*

1. *Completeness: Whenever $(x, w) \in \mathcal{R}$, the verifier V accepts.*

2. *Special soundness: There exists a probabilistic, polynomial-time algorithm (PPT) E (knowledge extractor) which takes two accepting transcripts (a, e, z), (a, e', z') such that $e \neq e'$ as inputs, and outputs witness w such that $(x, w) \in \mathcal{R}$.*

3. *Special honest-verifier zero knowledge: There exists a PPT algorithm M (the simulator) which takes x and e as inputs, and outputs (a, z) so that the triple (a, e, z) is indistinguishable from an accepting protocol transcript generated by a real protocol run.*

Example 3.9.10. A standard example of a $\Sigma-$protocol is a protocol that allows the prover to convince a verifier that he knows a secret discrete logarithm without revealing it. This protocol is called Schnorr's protocol [Schnorr 1989].

Now, let G be a finite group of order prime q. Think of G as a large prime order subgroup of the multiplicative group $\mathbb{Z}_p^*$, in this case G is a cyclic subgroup. Suppose that g is a generator for G. Now, Peggy wants to prove to Victor that she knows $x \in \mathbb{Z}_q^*$ such that $y = g^x$ without revealing x. Both of them proceed to conduct a ZKP.

1. First, Peggy chooses a uniformly random $r \in \mathbb{Z}_q^*$, computes $R = g^r$ and sends R to Victor.

2. Victor responds by sending a challenge $c \in \mathbb{Z}_q^*$ to Peggy.

3. Peggy computes $z = xc + r$ and sends z to Victor.

4. Victor checks if $g^z = Ay^c$. If true, he accepts, otherwise reject.

First, completeness is immediate. For soundness, suppose that Peggy is able to answer with two distinct challenges with the same initial response R, that is, assume we have two accepting transcripts (R, c_1, z_1) and (R, c_2, z_2). Then we have that $x := (z_2 - z_1)/(c_2 - c_1)$ satisfies $y = g^x$. Hence, we extracted the secret x which shows soundness. Finally, to prove zero-knowledge, we need to construct a simulator S that takes y, c as inputs and outputs a triple indistinguishable from an accepting real transcript. S does the following:

1. Choose a uniformly random $z \in \mathbb{Z}_q^*$.

2. Define $A = g^z y^{-c}$

3. Output (A, c, z)

The output of S is clearly indistinguishable from a real accepting transcript.

Proofs of Knowledge

Now, according to [Damgård 2010], a $\Sigma-$protocol for a relation $\mathcal{R}$ is a proof of knowledge for $\mathcal{R}$. But, what do we mean by a "proof of knowledge"? A proof of knowledge essentially means that a certain prover P can prove that it knows something without revealing the secret information about it.

Going further, what do we mean by the statement that "a prover P claims to know something"? In [Damgård 2010], the statement "machine P knows something" means that P can be used to compute relevant information about that "something" efficiently.

Below we give a precise definition of a proof of knowledge, but before we do that, given any prover P^*, denote by $\epsilon(x)$ to be the probability that a verifier V accepts on input x.

Definition 3.9.11. *Let $\kappa : \{0,1\}^* \to [0,1]$, a function from bit strings to the interval $[0,1]$. The protocol (P,V) is said to be a proof of knowledge for relation $\mathcal{R}$ with knowledge error κ, if the following are satisfied:*

1. *Completeness: On common input x to P and V and private input w of P such that $(x,w) \in \mathcal{R}$, then V always accepts.*

2. *Knowledge soundness: There exists a PPT algorithm E, called the knowledge extractor, which gets x as input and has a rewindable black-box access to P, and attempts to compute w such that $(x,w) \in \mathcal{R}$. Additionally, we require the following holds: There exists a constant c such that whenever $\kappa(x) < \epsilon(x)$, E will output a correct w in expected time at most*

$$\frac{|x|^c}{\epsilon(x) - \kappa(x)}$$

where access to P counts as one step only.

Here, we think of $\kappa(x)$ as the probability that a prover, without knowing w, convinces a verifier.

Notice that as long as $\kappa(x)$ is way more smaller than $\epsilon(x)$ which is the situation one gets better at convincing V with knowledge about w, then the efficiency of computing w gets better as well.

The theorem below states the every $\Sigma-$protocol is a proof of knowledge.

Theorem 3.9.12 (Damgård 2010)**.** *Let $\mathcal{P}$ be a $\Sigma-$protocol for a relation $\mathcal{R}$ with challenge length t. Then $\mathcal{P}$ is a proof of knowledge with knowledge error 2^{-t}.*

A classical example of a $\Sigma-$protocol can be found in [Schnorr 1989] which is about a protocol for proving that some prover knows a secret discrete logarithm.

Note that $\Sigma-$protocols only guarantee privacy against honest-but-curious verifiers, that is, verifiers that do not deviate from the protocol specification. Thus, the natural question is how do we mitigate any attack from a malicious verifier, one that deviates from the protocol? The standard technique is given by [Fiat and Shamir 1986], called the Fiat-Shamir heuristic where we make the protocol non-interactive by replacing the challenge of the verifier by a hash value of the inputs of a prover. We illustrate the modification of a $\Sigma-$protocol via a Fiat-Shamir heuristic below:

Let H be a cryptographic hash function whose range is the challenge set $\mathcal{C}$ and that both prover P and verifier V have access to. Then a non-interactive $\Sigma-$protocol $\mathcal{P}$ for relation $\mathcal{R}$ works as follows: Let x be a common input to prover P and verifier V and w be a private input to P such that $(x, w) \in \mathcal{R}$.

1. First, P computes a on input x, w and computes challenge $e = H(x, a)$.
2. Then P computes response z in $\mathcal{P}$ on input a, e, w.
3. P sends (a, e, z) to verifier V and V checks whether $e = H(x, a)$ and outputs $V(x, a, e, z) = 0$ or 1.

It was shown in [Pointcheval and Stern 1996] that non-interactive $\Sigma-$protocols, in fact, non-interactive zero-knowledge protocols in general are secure against chosen message attacks in the random oracle model, if we assume that random oracles exist.

Moreover, it is more advantageous to deploy non-interactive $\Sigma-$protocols over the interactive ones in self-tallying voting protocols to reduce the communication cost. Hence, we will resort to non-interactive proofs and thus, we assume security in the random oracle model as well.

The OR-proof

Suppose a prover P wants to convince a verifier V that on inputs x_0, x_1, P knows a witness w such that either $(x_0, w) \in \mathcal{R}$ or $(x_1, w) \in \mathcal{R}$ without revealing which is the case. To solve the problem above, we assume first that we have a $\Sigma-$protocol $\mathcal{P}$ for a relation $\mathcal{R}$ and use $\mathcal{P}$ as a subroutine to construct another $\Sigma-$protocol for the relation

$$\mathcal{R}_{OR} = \{((x_0, x_1), w) : (x_0, w) \in \mathcal{R} \text{ or } (x_1, w) \in \mathcal{R}\}.$$

First, let (x_0, x_1) be a common input to both prover P and verifier V and a private input w to P such that $(x_b, w) \in \mathcal{R}$, where either $b = 0$ or 1. Assume also that we have a $\Sigma-$protocol $\mathcal{P}$ for relation $\mathcal{R}$.

We construct a protocol denoted by $\mathcal{P}_{OR}$ for relation $\mathcal{R}_{OR}$ as follows:

1. P computes $a_b = P(x_b, w)$ in $\mathcal{P}$ and P chooses e_{1-b} at random and runs the simulator M in $\mathcal{P}$ on input (x, e_{1-b}) and let $(a_{1-b}, e_{1-b}, z_{1-b})$ as its output. P sends a_0, a_1 to V.

2. V chooses a random bit string s of length t and sends it to P.

3. P sets $e_b := s \oplus e_{1-b}$ ($\oplus$ denotes the XOR operation on bit strings) and computes an answer $z_b := P(x_b, w, a_b, e_b)$ to e_b in $\mathcal{P}$. P sends e_0, z_0, e_1, z_1 to V.

4. V checks that $s = e_0 \oplus e_1$ and that both (a_0, e_0, z_0) and (a_1, e_1, z_1) are accepting conversations in $\mathcal{P}$ on inputs x_0, respectively x_1.

We summarize the protocol $\mathcal{P}_{OR}$ in the figure below.

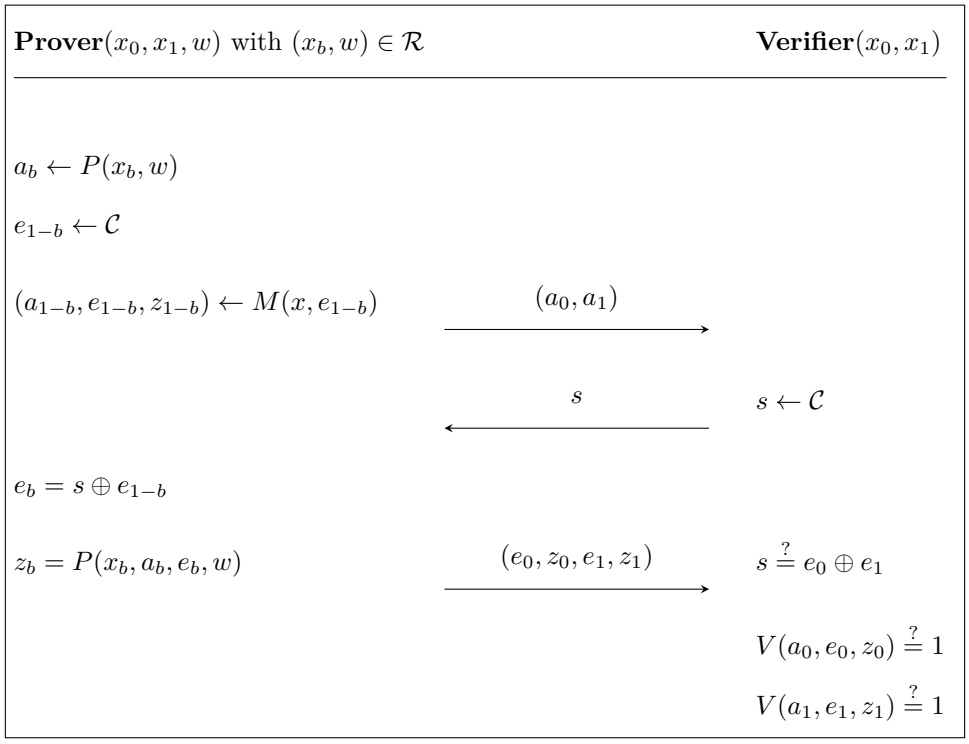


Figure 3.4: Summary of protocol $\mathcal{P}_{OR}$, a $\Sigma-$protocol for $\mathcal{R}_{OR}$.

Theorem 3.9.13 (Damgård 2010)**.** *$\mathcal{P}_{OR}$ above is a $\Sigma-$protocol for $\mathcal{R}_{OR}$.*

Proof. *Completeness*: Completeness is straightforward since (a_b, e_b, z_b) is a real protocol transcript generated by an honest prover while $(a_{1-b}, e_{1-b}, z_{1-b})$ is indistinguishable from a real protocol transcript. Thus, any verifier will accept both as valid transcripts.
Special soundness: Given two accepting transcripts

$$((a_0, a_1), s, (e_0, z_0, e_1, z_1))$$

and

$$((a_0, a_1), s', (e'_0, z'_0, e'_1, z'_1)),$$

then both (a_b, e_b, z_b) and (a_b, e'_b, z'_b) are accepting transcripts for relation $\mathcal{R}$. Using the special soundness property of the Σ-protocol for relation $\mathcal{R}$, there is a knowledge extractor

E that outputs witness w such that $(x_b, w) \in \mathcal{R}$. Hence w is a witness for (x_0, x_1) for relation $\mathcal{R}_{\text{OR}}$.

Special honest-verifier zero knowledge: On input $s \in \mathcal{C}$, we construct simulator S_{OR} as follows: first, S chooses $e_0 \xleftarrow{\$} \mathcal{C}$ and let $e_1 = s \oplus e_0$. Now, run simulator S in the $\Sigma-$protocol for relation $\mathcal{R}$ twice, the first time on input (x_0, e_0) and on the second time on input (x_1, e_1). Hence, S produces two accepting transcripts for $\mathcal{R}$ indistinguishable from real protocol runs, which are (a_0, e_0, z_0) and (a_1, e_1, z_1). Finally, S_{OR} outputs

$$((a_0, a_1), s, (e_0, z_0, e_1, z_1))$$

which is indistinguishable from a real accepting transcript when $\mathcal{P}_{OR}$ is run by an honest prover and honest verifier, ■

$\Sigma'-$protocols

Now, we introduce a somewhat weaker version of $\Sigma-$protocols called $\Sigma'-$protocols and in fact, we will employ a $\Sigma'-$protocol to our voting protocol to ensure that voters honestly follow the voting protocol specification that we will illustrate in Section 5.

Definition 3.9.14. *Let $\mathcal{P} = (P, V)$ be a two-party protocol, where V is a PPT algorithm, and let $\mathcal{R}, \mathcal{R}'$ be two relations such that $\mathcal{R} \subseteq \mathcal{R}'$. Then $\mathcal{P}$ is called a $\Sigma'-$protocol for relations $\mathcal{R}, \mathcal{R}'$ with completeness error α, challenge set $\mathcal{C}$, public input x and private input w, if $\mathcal{P}$ is of the same form as in Figure 3.3 but $\mathcal{P}$ satisfies the rather weaker conditions:*

- *Completeness: Whenever $(x, w) \in \mathcal{R}$, the verifier V accepts with probability at least $1 - \alpha$.*
- *Special soundness: There exists a PPT algorithm E (knowledge extractor) which takes two accepting transcripts (a, e, z), (a, e', z') such that $e \neq e'$ as inputs, and outputs witness w' such that $(x, w') \in \mathcal{R}'$.*

- *Special honest-verifier zero knowledge: There exists a PPT algorithm M (the simulator) which takes x and e as inputs, and outputs (a, z) so that the triple (a, e, z) is indistinguishable from an accepting protocol transcript generated by a real protocol run.*

Notice the difference between the completeness and special soundness properties of $\Sigma'-$protocols and $\Sigma-$protocols, since in a $\Sigma'-$protocol, the verifier may reject the proof of an honest prover with probability at most α due to a weaker completeness property and also the extracted witness w' in the special soundness property in Σ' has (x, w') lying in the larger relation $\mathcal{R}'$ which means that the verifier is only assured that the prover knows a witness in $\mathcal{R}'$, not necessarily in the original relation $\mathcal{R}$.

3.9.3 RLWE-based $\Sigma'-$ protocols

Now, we present two $\Sigma'-$protocols that will be an essential cryptographic primitive in constructing our self-tallying voting scheme. The first protocol is about proving knowledge of a secret in an RLWE sample and the other is OR-proof that either a certain polynomial is an RLWE sample or an RLWE sample added by 1. The $\Sigma'-$protocols that we will present are very similar to the one in [Benhamouda, Camenisch, et al. 2014] with just a slight modification.

Before we present the two $\Sigma'-$protocols, we state the following technical lemma from [Benhamouda, Camenisch, et al. 2014] which says that certain binomials in $\mathbb{Z}[X]/(X^n+1)$ can be inverted, and their inverses have small coefficients. This lemma is essential for the proof of the special soundness property of the construction of the Σ'-protocol in [Benhamouda, Camenisch, et al. 2014] as well as our own protocol.

Lemma 3.9.15 (Benhamouda, Camenisch, et al. 2014)**.** *Let n be a power of 2 and let $0 < i, j < n-1$, $i \neq j$. Then $2(X^i - X^j) \pmod{X^n+1}$ is invertible and $2(X^i - X^j)^{-1} \pmod{X^n+1}$ only has coefficients in $\{-1, 0, 1\}$.*

Now, we present our first $\Sigma'-$protocol which is a protocol for the following relations:

$$\mathcal{R}_{b,m,\sigma,n} = \left\{((a,y),(s,e)) \in R_q^2 \times R_q^2 : y = as + (m+1)e + b, \|s\|_\infty, \|e\|_\infty \leq \sigma\sqrt{n}\right\} \tag{3.9.5}$$

$$\mathcal{R}'_{b,m,\sigma,n} = \left\{((a,y),(s,e)) \in R_q^2 \times R_q^2 : 2y = 2(as + (m+1)e + b), \|2s\|_\infty, \|2e\|_\infty \leq 2\sigma n(\sqrt{n}\log^k 2n + 1)\right\} \tag{3.9.6}$$

where $b \in \{0,1\}$, $m, n \in \mathbb{N}$, $q \equiv 3 \bmod 8$, $\sigma > 0$, for some constant $k > \dfrac{1}{2}$, and $2s$ and $2e$ are reduced modulo q. Moreover, it is trivial to check that $\mathcal{R}_{b,m,\sigma,n} \subseteq \mathcal{R}'_{b,m,\sigma,n}$.

Description of the $\Sigma'-$protocol for relations $\mathcal{R}_{b,m,\sigma,n} \subseteq \mathcal{R}'_{b,m,\sigma,n}$.

First, both prover and verifier agree to $\sigma > 0$, quotient ring $R_q := \mathbb{Z}_q[X]/(X^n + 1)$, where n is a power of two, $q \equiv 3 \bmod 8$ a prime and $m \in \mathbb{N}$. Set $\tilde{\sigma} = \alpha\sigma\sqrt{2n}$ with $\alpha = (\log 2n)^k$, where $k > \dfrac{1}{2}$ is fixed. Set $M = e^{\frac{12}{(\log 2n)^k} + \frac{1}{2(\log 2n)^{2k}}}$. Fix $b \in \{0,1\}$. All the above parameters are public.

Let (a,y) be common inputs to prover and verifier and let (s,e) be a private input to prover such that $((a,y),(s,e)) \in \mathcal{R}_{b,m,\sigma,n}$.

1. First, the prover draws $r_s, r_e \xleftarrow{\$} D_{\mathbb{Z}^n, \frac{\tilde{\sigma}}{\sqrt{2}}}$ and computes $t = ar_s + (m+1)r_e$ in R_q, then sends t to the verifier.

2. The verifier draws a uniformly random chosen $c \in \mathcal{C} := \{0, 1, \ldots, n-1\}$.

3. After receiving c from the verifier, the prover sends $(s_s, s_e) := (r_s, r_e) + (X^c s, X^c e)$ to the verifier if either:

   - $D_{\mathbb{Z}^{2n},\tilde{\sigma}}(s_s, s_e) \geq M \cdot D_{(X^c s, X^c e)+\mathbb{Z}^{2n},\tilde{\sigma}}(s_s, s_e)$ or;
   - if $D_{\mathbb{Z}^{2n},\tilde{\sigma}}(s_s, s_e) < M \cdot D_{(X^c s, X^c e)+\mathbb{Z}^{2n},\tilde{\sigma}}(s_s, s_e)$, prover draws u from uniform distribution on $[0, M \cdot D_{(X^c s, X^c e)+\mathbb{Z}^{2n},\tilde{\sigma}}(s_s, s_e)]$ and output (s_s, s_e) if $u < D_{\mathbb{Z}^{2n},\tilde{\sigma}}(s_s, s_e)$.

If none of the two cases holds, the prover sends an abort message $\perp$ to the verifier and returns to step 1. In this step, by Theorem 3.8.2, the probability that (s_s, s_e) is output and thus, the prover sends a response to the verifier is at least $\dfrac{1-2^{-100}}{M}$ and so, the probability that the prover aborts is at most $1 - \left(\dfrac{1-2^{-100}}{M}\right)$.

Notice that our protocol employs a rejection sampling algorithm for the response of the prover to the challenge of the verifier. We set the following parameters. First, given the standard deviation σ of the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$, we put $T := \sigma\sqrt{2n}$ and $\tilde{\sigma} = \omega(T\sqrt{\log 2n})$, for instance, take $\tilde{\sigma} = \sigma\sqrt{2n}(\log 2n)^k$ for any fixed $k > \dfrac{1}{2}$. Then according to Theorem 3.8.2, with $M = e^{\frac{12}{(\log 2n)^k} + \frac{1}{2(\log 2n)^{2k}}}$, the algorithm $\mathcal{A}$ outputs (s_s, s_e), viewed as an element of $\mathbb{Z}^{2n}$ in the protocol with probability at least $\dfrac{1-2^{-100}}{M}$. Also, choosing k as close to $1/2$ means that the acceptance probability in a single run is as small as possible but it has also the advantage of having a smaller gap factor between the two relations $\mathcal{R}_{b,m,\sigma,n}$ and $\mathcal{R}'_{b,m,\sigma,n}$, which is the ratio $\dfrac{2\sigma n(\sqrt{n}\log^k 2n + 1)}{\sigma\sqrt{n}} = 2\sqrt{n}(\sqrt{n}(\log 2n)^k + 1)$. Notice that the smaller the gap factor is, the more the verifier is convinced that the prover knows witnesses in the original relation $\mathcal{R}_{b,m,\sigma,n}$. Hence, we see a trade off between the running time of the $\Sigma'-$protocol and the gap factor since both depend on n and k.

Our $\Sigma'-$protocol for relations $\mathcal{R}_{b,m,\sigma,n} \subseteq \mathcal{R}'_{b,m,\sigma,n}$ can be summarized by Figure 3.5:

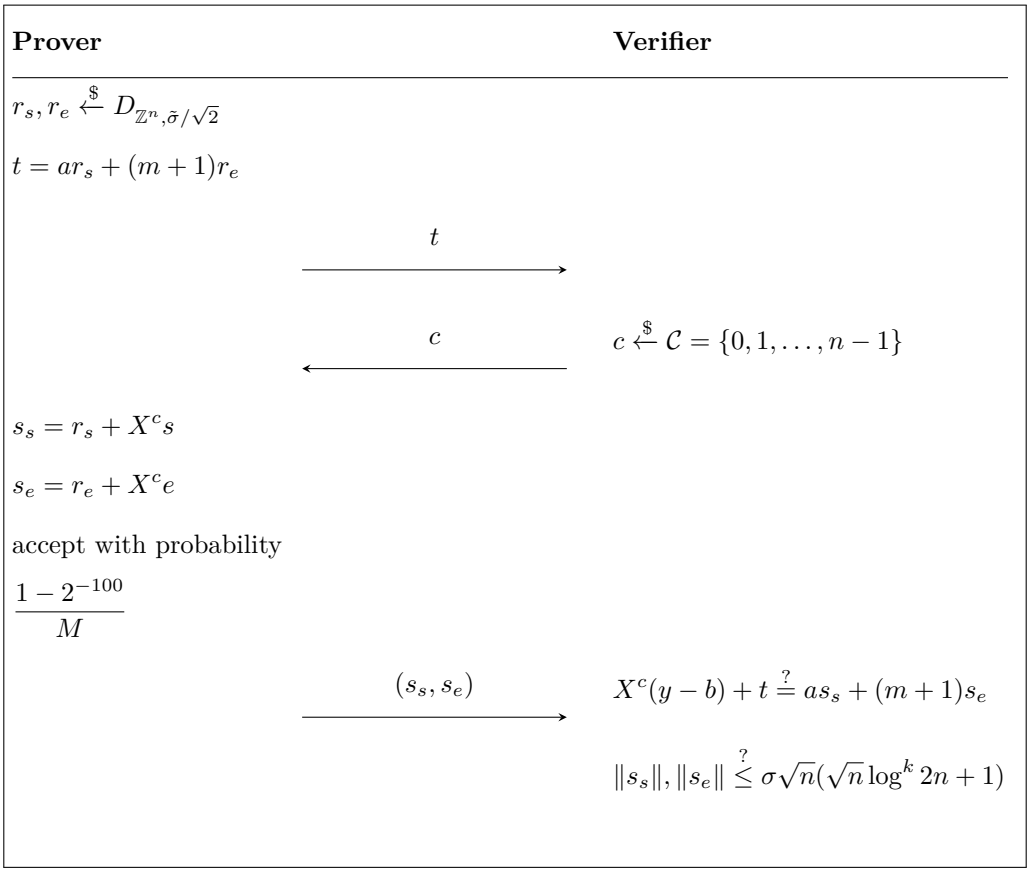


Figure 3.5: Σ' - protocol for relations $\mathcal{R}_{b,m,\sigma,n} \subseteq \mathcal{R}'_{b,m,\sigma,n}$. Here the prover wants to prove knowledge of secret (s, e) such that $((a, y), (s, e)) \in \mathcal{R}_{b,m,\sigma,n}$.

Despite the fact that $\Sigma'-$protocols are weaker in the sense that verifiers are only convinced that honest provers know a witness in the larger relation $\mathcal{R}'_{b,n,\sigma,n}$, such weaker protocols are already used for example in [Xie et al. 2013], [Benhamouda, Camenisch, et al. 2014] and [Benhamouda, Krenn, et al. 2015].

Theorem 3.9.16. *The Σ' - protocol in Figure 3.5 is an honest-verifier zero knowledge Σ'-protocol for the relations $\mathcal{R}_{b,m,\sigma,n}$ and $\mathcal{R}'_{b,m,\sigma,n}$. The protocol has a knowledge error of $1/n$ and completeness error of $1 - 1/M$.*

Proof. *Completeness*: In a single run, the probability that an honest verifier accepts the proof of an honest prover is $1/M$, hence the completeness error is $1 - \frac{1}{M}$. Also, in

the case that an honest prover successfully sends a challenge response, then

$$\begin{aligned} X^c(y-b)+t &= X^c(as+(m+1)e+b-b)+ar_s+(m+1)r_e \\ &= a(X^cs+r_s)+(m+1)(X^ce+r_e) \\ &= as_s+(m+1)s_e \end{aligned}$$

and also,

$$\begin{aligned} \|s_s\| &\leq \|r_s\|+\|X^cs\| \\ &\leq \frac{\tilde{\sigma}\sqrt{n}}{\sqrt{2}}+\sigma\sqrt{n} \\ &= \sigma n(\log^k 2n)+\sigma\sqrt{n} \\ &= \sigma\sqrt{n}(\sqrt{n}\log^k 2n+1) \end{aligned}$$

with overwhelming probability by Lemma 3.3.13 and also since X^cs is just an anti-cyclic shift of s in R_q, hence $\|X^cs\| = \|s\|$. By similar arguments, $\|s_e\| \leq \sigma\sqrt{n}(\sqrt{n}\log^k 2n+1)$. This shows completeness.

Special soundness: Let c, c' be two distinct challenges from $\mathcal{C}$ and let (s_s, s_e), (s'_s, s'_e) be two distinct final responses from prover P such that both $(t, c, (s_s, s_e))$ and $(t, c', (s'_s, s'_e))$ are accepting transcripts. The knowledge extractor E proceeds as follows to extract witness in the relation $\mathcal{R}'_{b,m,\sigma,n}$:

Taking the difference between the equations

$$X^c(y-b)+t = as_s+(m+1)s_e$$

and

$$X^{c'}(y-b)+t = as'_s+(m+1)s'_e$$

we get

$$(y-b)(X^c-X^{c'}) = a(s_s-s'_s)+(m+1)(s_e-s'_e) \tag{3.9.7}$$

Then multiplying both sides of (3.9.7) by $2(X^c - X^{c'})^{-1}$, we have

$$2y = 2a\hat{s} + 2(m+1)\hat{e} + 2b$$

where $\hat{s} = (s_s - s'_s)(X^c - X^{c'})^{-1}$ and $\hat{e} = (s_e - s'_e)(X^c - X^{c'})^{-1}$. Moreover,

$$\begin{aligned}
\|2\hat{s}\|_\infty &\leq \|s_s - s'_s\| \cdot \|2(X^c - X^{c'})^{-1}\| \ \text{ (by Lemma 3.7.7)} \\
&\leq (\|s_s\| + \|s'_s\|)\sqrt{n} \ \text{ (by Lemma 3.9.15)} \\
&\leq 2\sigma n(\sqrt{n}\log^k 2n + 1)
\end{aligned}$$

and similarly, $\|2\hat{e}\|_\infty \leq 2\sigma n(\sqrt{n}\log^k 2n + 1)$. Hence, $(\hat{s}.\hat{e})$ is a witness for (a, y) under the relation $\mathcal{R}'_{b,m,\sigma,n}$.

Special honest-verifier zero knowledge: Fix challenge $c \xleftarrow{\$} \mathcal{C}$ as an input of the simulator S. The goal of the simulator S is to produce $(t, (s_s, s_e))$ such that $(t, c, (s_s, s_e))$ is indistinguishable from a real accepting transcript from a real interaction between an honest prover and honest verifier. Note that the simulator S also knows the values of the public key a, the bit b and y. We now describe the execution of S. First S chooses $s_s, s_e \xleftarrow{\$} D_{\mathbb{Z}^n,\sigma(\log^k n+1)}$ with probability $\dfrac{1-2^{-100}}{M}$. Otherwise, with probability $1 - \left(\dfrac{1-2^{-100}}{M}\right)$, take $(s_s, s_e) = \perp$, where $\perp$ is the abort message. In the case where s_s and s_e are chosen from the discrete Gaussian distribution, we have that $\|s_s\|, \|s_e\| \leq \sigma\sqrt{n}(\log^k 2n+1)$ with overwhelming probability by Lemma 3.3.13. Now, S takes $t = as_s + (m+1)s_e - X^c(y-b)$ in the case where s_s, s_e are taken from the discrete Gaussian distribution while take t as a uniformly random chosen element in R_q if $(s_s, s_e) = \perp$. Hence, it can be easily seen that $(t, c, (s_s, s_e))$ is indistinguishable from a real accepting transcript since in addition, the construction of t is indistinguishable from uniformly random ones by the decisional RLWE assumption whenever s_s, s_e are chosen from the discrete Gaussian distribution which happens with probability $\dfrac{1-2^{-100}}{M}$ while otherwise, t is uniformly random in R_q. Thus, in both cases, the distribution of t generated by an honest prover is indistinguishable

from the distribution of t generated by S. This proves the special honest-verifier zero knowledge property. ■

Remark 3.9.17. *As in $\Sigma-$protocols, we can make our $\Sigma'-$protocols non-interactive by using a cryptographic hash function H and replace the challenge of an interactive verifier by the value $c = H((a, y), t)$, where (a, y) is the first pair of an element in $\mathcal{R}_{b,m,\sigma,n}$ and t is the first message/commitment of the prover in the protocol. Hence, we eliminate any attack of a dishonest verifier through the above Fiat-Shamir heuristic but again, we have privacy in the random oracle model.*

OR-proof using $\Sigma'-$protocols

For simplicity, if the parameters σ, m and n are already understood, we simply denote the relation $\mathcal{R}_b := \mathcal{R}_{b,m,\sigma,n}$ for $b \in \{0, 1\}$. Similarly, denote by $\mathcal{R}'_b := \mathcal{R}'_{b,m,\sigma,n}$ Now, define $\mathcal{R}_{OR} := \mathcal{R}_0 \cup \mathcal{R}_1$ and $\mathcal{R}'_{OR} := \mathcal{R}'_0 \cup \mathcal{R}'_1$. Denote by $\mathcal{P}_b$ be the $\Sigma'-$protocol for proving relations $\mathcal{R}_b \subseteq \mathcal{R}'_b$, $b \in \{0, 1\}$.

We consider the following problem: Suppose a prover and a verifier has common input (a, y) and the prover has private input (s, e) such that $((a, y), (s, e)) \in \mathcal{R}_{OR}$. Furthermore, assume that the private input $((a, y), (s, e))$ of the prover satisfies only $((a, y), (s, e)) \in \mathcal{R}_b$ for some $b \in \{0, 1\}$ but not in $\mathcal{R}_{1-b}$. The prover wants to prove to the verifier that he knows some secret pair (s, e) such that $((a, y), (s, e)) \in \mathcal{R}_{OR}$ without revealing which among the relations $\mathcal{R}_0$ and $\mathcal{R}_1$ does $((a, y), (s, e))$ belong to.

We propose a solution to the above problem using an OR-proof for the relation $\mathcal{R}_{OR}$ similar to the OR-proof using $\Sigma-$protocols but in our case, the verifier is only guaranteed that the prover knows the witness in the larger relation $\mathcal{R}'_{OR}$.

Given two relations $\mathcal{R}_0, \mathcal{R}_1$ with corresponding $\Sigma'-$protocols, say $\mathcal{P}_i$ for $\mathcal{R}_i$, $i = 0, 1$, the prover wishes to prove knowledge of witness (s, e) such that given (a, y) as common inputs to both the prover and the verifier, either $((a, y), (s, e)) \in \mathcal{R}_0$ or $((a, y), (s, e)) \in \mathcal{R}_1$ without revealing which one.

In a similar fashion as in the OR-proof using $\Sigma-$protocols, we construct a $\Sigma'-$protocol for the relations $\mathcal{R}_{OR} \subseteq \mathcal{R}'_{OR}$, with

$$\mathcal{R}_{\text{OR}} = \{((a, y), (s, e)) : ((a, y), (s, e)) \in \mathcal{R}_0 \cup \mathcal{R}_1\}$$

and

$$\mathcal{R}'_{\text{OR}} = \{((a, y), (s, e)) : ((a, y), (s, e)) \in \mathcal{R}'_0 \cup \mathcal{R}'_1\}$$

where

$$\mathcal{R}_b = \left\{((a, y), (s, e)) \in R_q^2 \times R_q^2 : y = as + (m + 1)e + b, \|s\|_\infty, \|e\|_\infty \leq \sigma\sqrt{n}\right\}$$

and

$$\mathcal{R}'_b = \left\{((a, y), (s, e)) \in R_q^2 \times R_q^2 : 2y = 2(as + (m + 1)e + b), \|2s\|, \|2e\|_{\leq} 2\sigma n(\sqrt{n}\log^k 2n + 1)\right\}$$

for $b \in \{0, 1\}$.

Now we describe our $\Sigma'-$protocol for relations $\mathcal{R}_{OR} \subseteq \mathcal{R}'_{OR}$. This can be realized by the following steps: First, let (a, y) be a common input to both prover P and verifier V and let (s, e) be a private input to the prover such that $((a, y), (s, e)) \in \mathcal{R}_{OR}$. Without loss of generality, assume that $((a, y), (s, e)) \in \mathcal{R}_b$ but not in $\mathcal{R}_{1-b}$ for a fixed $b \in \{0, 1\}$. Furthermore, denote by $\mathcal{P}_0$ (respectively, $\mathcal{P}_1$) to be the $\Sigma'-$protocol for relations $\mathcal{R}_0 \subseteq \mathcal{R}'_0$ (resp. $\mathcal{R}_1 \subseteq \mathcal{R}'_1$).

1. P chooses $c_{1-b} \xleftarrow{\$} \mathcal{C} = \{0, 1, \ldots, n - 1\}$ and runs the simulator S_{1-b} of $\mathcal{P}_{1-b}$ to obtain the tuple $(t_{1-b}, c_{1-b}, (s_{s,1-b}, s_{e,1-b}))$ which is indistinguishable from real protocol runs with probability $\dfrac{1}{M}$, so this entire step may take approximately M times before a tuple is obtained.

2. Now, P chooses $r_{s,b}, r_{e,b} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n,\tilde{\sigma}/\sqrt{2}}$ and computes for $t_b = ar_{s,b} + (m+1)r_{e,b}$ in R_q.

3. Now, P sends the tuple (t_0, t_1) to the verifier.

4. V sends a challenge $c \xleftarrow{\$} \mathcal{C} = \{0, 1, \ldots, n-1\}$.

5. P computes for $c_b = c \oplus c_{1-b}$. Also, he computes $s_{s,b} = r_{s,b} + X^{c_b}s$ and $s_{e,b} = r_{e,b} + X^{c_b}e$ with probability $\dfrac{1}{M}$, (otherwise, the prover aborts and restarts the protocol) and sends the following two responses

$$(c_0, s_{s,0}, s_{e,0}), (c_1, s_{s,1}, s_{e,1})$$

   to the verifier.

6. Finally, the verifier will check if the tuple $(t_0, c_0, (s_{s,0}, s_{e,0}))$ (also for $(t_1, c_1, (s_{s,1}, s_{e,1}))$) is an accepting transcript for $\mathcal{P}_0$ (respectively for $\mathcal{P}_1$) and $c = c_0 \oplus c_1$.

We can summarize the above $\Sigma'-$ protocol for $\mathcal{R}_{OR} \subseteq \mathcal{R}'_{OR}$ in the figure below:

Prover		**Verifier**
$c_{1-b} \leftarrow \mathcal{C} = \{0, 1, \ldots, n-1\}$		
$(t_{1-b}, (s_{s,1-b}, s_{e,1-b})) \xleftarrow{\$} S_{1-b}(c_{1-b})$		
$r_{s,b}, r_{e,b} \xleftarrow{\$} \mathcal{D}_{\mathbb{Z}^n, \tilde{\sigma}/\sqrt{2}}$		
$t_b = ar_{s,b} + (m+1)r_{e,b}$		
	(t_0, t_1) $\longrightarrow$	
	$\longleftarrow$ c	$c \xleftarrow{\$} \mathcal{C} = \{0, 1, \ldots, n-1\}$
$c_b = c \oplus c_{1-b}$		
$s_{s,b} = r_{s,b} + X^{c_b} s$		
$s_{e,b} = r_{e,b} + X^{c_b} e$		
accept with probability $\dfrac{1 - 2^{-100}}{M}$		
from rejection sampling theorem		
	$(c_0, (s_{s,0}, s_{e,0}))$	
	$(c_1, (s_{s,1}, s_{e,1}))$ $\longrightarrow$	
		accept if for each $b \in \{0, 1\}$,
		$(t_b, \mathsf{c}_b, (s_{s,b}, s_{e,b}))$
		is an accepting
		transcipt for $\mathcal{P}_b$ and $\mathsf{c}_0 \oplus \mathsf{c}_1 = c$.

Figure 3.6: $\Sigma'-$protocol for proving knowledge of secret for relations $\mathcal{R}_{OR} \subseteq \mathcal{R}'_{OR}$. Here, assume that the prover knows secret (s, e) such that $((a, y), (s, e)) \in \mathcal{R}_b$ but does not want the verifier to learn exactly which among the relations $\mathcal{R}_0$ and $\mathcal{R}_1$ does $((a, y), (s, e))$ belong to.

Theorem 3.9.18. *Figure 3.6 is a $\Sigma'-$ protocol for relations $\mathcal{R}_{OR}$ (3.9.3), $\mathcal{R}'_{OR}$ (3.9.3) with completeness error $1 - \frac{1}{M}$ and knowledge error $\dfrac{1}{n}$.*

Proof. *Completeness*: If $(s_{s,b}, s_{e,b})$ is accepted by the rejected sampling algorithm which occurs with probability $\frac{1-2^{-100}}{M}$ then both $(t_0, \mathsf{c}_0, (s_{s,0}, s_{e,0}))$ and $(t_1, \mathsf{c}_1, (s_{s,1}, s_{e,1}))$ are accepting transcripts for both $\mathcal{P}_0$ and $\mathcal{P}_1$ protocols, respectively. If $(s_{s_b}, s_{e,b})$ is rejected, that is, the challenge response of the prover is an abort message then the proof halts with probability $1 - \frac{1-2^{-100}}{M}$.

Special soundness: Let $((t_0, t_1), c, (c_0, c_1, (s_{s,0}, s_{e,0})))$ and $((t_0, t_1), c', (c'_0.c'_1, (s'_{s,0}, s'_{e,0})))$ be two accepting transcripts such that $c \neq c'$. Thus, there exists a $b \in \{0, 1\}$ such that $c_b \neq c'_b$ and that both $(t_b, c_b, (s_{s,b}, s_{e,b}))$ and $(t_b, c'_b, (s'_{s,b}, s'_{e,b}))$ are accepting transcripts for $\mathcal{P}_b$. Using the special soundness property of $\mathcal{P}_b$, there is a knowledge extractor E_b that outputs $(\hat{s}_b, \hat{e}_b)$ such that $((a, y), (\hat{s}_b, \hat{e}_b)) \in \mathcal{R}'_b$. Hence, $((a, y), (\hat{s}_b, \hat{e}_b)) \in \mathcal{R}'_{\text{OR}}$ which shows the special soundness property.

Special honest-verifier zero-knowledge: We construct the simulator S_{OR} which takes $c \in \mathcal{C}$ as input and outputs $((t_0, t_1), (c_0, c_1), (s_{s,0}, s_{e,0}), (s_{s,1}, s_{e,1}))$ such that

$$((t_0, t_1), c, ((c_0, (s_{s,0}, s_{e,0})), (c_1, (s_{s,1}, s_{e,1}))))$$

is indistinguishable from a real protocol transcript. First, S_{OR} generates $c, c_0 \xleftarrow{\$} \mathcal{C}$ and computes $c_1 := c \oplus c_0$. Then S_{OR} runs the simulators S_0 and S_1 from the $\mathcal{P}_0$ and $\mathcal{P}_1$ protocols, respectively to produce two accepting transcripts $(t_0, c_0, (s_{s,0}, s_{e,0}))$ for $\mathcal{P}_0$ and $(t_1, c_1, (s_{s,1}, s_{e,1}))$ for $\mathcal{P}_1$ which are both indistinguishable from real accepting transcripts. Finally, S_{OR} outputs $((t_0, t_1), (c_0, c_1), (s_{s,0}, s_{e,0}), (s_{s,1}, s_{e,1}))$ such that $((t_0, t_1), c, ((c_0, (s_{s,0}, s_{e,0})), (c_1, (s_{s,1}, s_{e,1}))))$ is indistinguishable from a real protocol transcript. This proves the special honest-verifier zero-knowledge property. ■

Non-interactive $\Sigma'-$ protocols

So far, all the protocols we mentioned require an interaction between a prover and a verifier but as mentioned in [Damgård 2010], we can replace the challenge response from

the verifier using a Fiat-Shamir heuristic [Fiat and Shamir 1986], that is, given that a prover and verifier initially agrees with a common public hash function H, then the prover takes (or first response) (t_0, t_1) and the public pair (a, y) as inputs of H and he receives $c = \mathsf{H}(t, (a, y))$ as a random challenge which is indistinguishable from a challenge of an interactive verifier. Finally, the prover proceeds to obtain final response $(c_0, (s_{s,0}, s_{e,0})), (c_1, (s_{s,1}, s_{e,1}))$. Now, the verifier accepts if

$$((t_0, t_1), c, (c_0, (s_{s,0}, s_{e,0})), (c_1, (s_{s,1}, s_{e,1})))$$

is a valid transcript and also checks if $c = \mathsf{H}(t, (a, y))$.

Interactive proofs seem to be not too costly efficiency-wise in the case where one proves that the statement is true without revealing any secret information to only a single verifier. Now, the question is, what if the prover needs to prove a certain statement to multiple, possibly a large number of verifiers? If the prover insists on conducting an interactive zero knowledge proof, the verification process involving a large number of verifiers may seem to be a very impractical or inefficient task. Hence, a solution to this problem is by "simulating" the challenge response part of the verifier in the interactive $\Sigma'-$protocol described above.

Simulating the challenge response of an interactive verifier by the output of a *random oracle* on input the given statement, public parameters and the initial response of the prover makes the protocol non-interactive. A random oracle can be viewed as a black box that takes certain inputs and outputs something that is indistinguishable from a uniformly random chosen element in its range.

The question whether random oracles exist is still an open problem, but in practice, in place of random oracles, we apply a *cryptographic hash function* that somehow mimics the ideal behavior of a random oracle. A hash function is a function that maps arbitrary sized bit strings to fixed-sized bit strings and is collision-resistant and inversion-resistant.

Hash functions are deterministic but its outputs follow a distribution indistinguishable from uniform. An example of a hash function that is currently deployed is SHA-256.

This noninteractive setup provides security against chosen message attacks in the random oracle model [Pointcheval and Stern 1996] .

Chapter 4

A lattice-based veto protocol

In 2020, [Ding, Emery, et al. 2020] constructed an AV-net based on the hardness of the decision RLWE problem defined in Section 3.7 which is at least as hard (quantumly) as worst-case ideal lattice problems such as SIVP. The protocol is a lattice-based analogue of [Hao and Zieliński 2006], that is, the specifications are very similar. In the passively secure version of [Ding, Emery, et al. 2020], the protocol has two rounds while the actively secure version has four rounds where the additional layer of a commitment scheme is employed to resist any attack of an active, malicious adversary.

4.1 Passively Secure Lattice-Based AV-Net

In the passively secure version of the protocol, the adversarial model is the one controlling corrupted parties in the protocol and tries to learn the secret keys and votes of the honest participants just by reading the published messages on the authenticated bulletin board.

Now, we describe the protocol with the following parameters:

- Let n be a power of 2.
- Let R be the cyclotomic ring $\mathbb{Z}[X]/f(X)$ where $f(X) = X^n + 1$.
- Let χ to be the discrete Gaussian distribution on $\mathbb{Z}^n$ with parameter σ.

- Let $\beta = \sigma\sqrt{n}$.
- Let m be an integer. (This will be the number of voters.)
- Let q be a prime such that $q \equiv 1 \bmod 2n$ such that $m \cdot (m-1) \cdot \beta^2 + m \cdot \beta \leq \frac{q}{4} - 2$.
- Let R_q be the quotient ring R/qR.
- Let the coefficients of a polynomial in R_q be in the interval $[-\frac{q-1}{2}, \frac{q-1}{2}]$.
- Let $\|\cdot\|$ be the ℓ_2-norm on R_q and $\|\cdot\|_\infty$ be the $\ell_\infty-$norm on R_q.

At the beginning of the protocol, all participants select a uniformly random element $a \in R_q$. Now, in the first round, each participant P_i selects s_i, e_i from the χ. Then P_i computes $b_i = a \cdot s_i + e_i$, where all the computations are done in the ring R_q.

After all the b_i's are published, each participant P_i computes

$$y_i := \sum_{j<i} b_j - \sum_{j>i} b_j. \tag{4.1.1}$$

In the second round, that is, the voting phase, each participant P_i publishes their vote c_i, where

$$c_i = \begin{cases} s_i y_i + e_i' & \text{if vote is 0 (no-veto)} \\ r_i & \text{if vote is 1,} \end{cases}$$

where e_i' is a fresh error chosen from the discrete Gaussian distribution on $\mathbb{Z}^n$. The result is computed by taking the sum $\sum_{i=1}^{m} c_i \in R_q$, where m is the number of participants.

After all voters have published their c_i's, each voter (locally) computes the final result as follows:

$$\mathsf{res} \leftarrow \begin{cases} \text{no veto} & \text{if } \left\| \sum_{i=1}^{m} c_i \right\|_\infty \leq \frac{q}{4} - 2 \\ \text{veto} & \text{otherwise} \end{cases}.$$

We show that the passively secure veto protocol is correct, i.e., it outputs the correct result (with overwhelming probability) if all participants follow the protocol specification correctly (Theorem 4.1.2). To this end, we use the following result which ensures that the error terms introduced (for privacy reasons) do not undermine correctness of the veto protocol except with negligible probability.

Lemma 4.1.1. *The probability that a uniformly chosen random element $r \in R_q$ has max norm less than or equal to $N \geq 1$ is given by*

$$\Pr[\|x\|_\infty \leq N : x \leftarrow R_q] = \frac{(2N+1)^n}{q^n}.$$

Theorem 4.1.2 (Correctness)**.** *Let Π_{veto} be the veto protocol defined in Section 4.1. Assume that all voters $\mathsf{V}_1, \ldots, \mathsf{V}_m$ (and the bulletin board B) are honest, i.e., run their programs as specified by the protocol. Then, we have that for all runs (of this instance) of Π_{veto}, the following equivalence holds true with overwhelming probability: The final result res is "veto" if and only if there exists (at least) one voter V_i who chooses "veto".*

Proof. First, let's consider the case where all voters choose "no veto". Then we have that

$$\begin{aligned}
\sum_{i=1}^{m} c_i &= \sum_{i=1}^{m} (s_i \cdot y_i + e'_i) \\
&= \sum_{i=1}^{m} s_i \cdot \left(\left(\sum_{j=1}^{i-1} b_j \right) - \left(\sum_{j=i+1}^{m} b_j \right) \right) + \sum_{i=1}^{m} e'_i \\
&= \sum_{i=1}^{m} s_i \cdot \left(\left(\sum_{j=1}^{i-1} a \cdot s_j + e_j \right) - \left(\sum_{j=i+1}^{m} a \cdot s_j + e_j \right) \right) + \sum_{i=1}^{m} e'_i \\
&= \left(\sum_{i=1}^{m} \sum_{j=1}^{i-1} s_i \cdot e_j - \sum_{i=1}^{m} \sum_{j=i+1}^{m} s_i \cdot e_j \right) + \sum_{i=1}^{m} e'_i
\end{aligned}$$

holds true, where the last equality follows from Lemma 2.1.2.

Recall that all s_i, e_i and e'_i are chosen according to χ, hence their norm is bounded by $\beta = \sigma\sqrt{n}$ (with overwhelming probability in the security parameter n) by Lemma 3.3.13. Hence, by triangle inequality and by Lemma 3.7.7, with overwhelming probability we have that

$$\begin{aligned}
\left\|\sum_{i=1}^{m} c_i\right\|_\infty &= \left\|\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i \cdot e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i \cdot e_j\right) + \sum_{i=1}^{m} e'_i\right\|_\infty \\
&\leq \sum_{i=1}^{m}\sum_{j\neq i} \|s_i \cdot e_j\|_\infty + \sum_{i=1}^{m} \|e'_i\|_\infty \\
&\leq \sum_{i=1}^{m}\sum_{j\neq i} \|s_i\| \cdot \|e_j\| + \sum_{i=1}^{m} \|e'_i\| \\
&\leq m \cdot (m-1) \cdot \beta^2 + m \cdot \beta \\
&\leq \frac{q}{4} - 2.
\end{aligned}$$

Conversely, assume that one of the voters vetoes, hence chooses c_i uniformly at random from R_q. From Lemma 4.1.1, it follows that the probability that $\sum_{i=1}^{m} c_i$ has max norm $\leq \frac{q}{4} - 2$ is negligible:

$$\Pr[\|r\|_\infty \leq \frac{q}{4} - 2 : r \leftarrow R_q] = 2^{-n}\left(\frac{q-6}{q}\right)^n < 2^{-n}.$$

Hence, altogether, we can conclude that (with overwhelming probability) the final result res equals "veto" if and only if at least one voter vetoes. This proves the correctness of the passively secure veto protocol. ■

Notice that the public b_i's are RLWE samples from RLWE distribution whose secret is s_i and by the RLWE hardness assumption (see Section 3.7), it is indistinguishable from uniform. The main issue in the protocol is that if one of the parties actively deviate from the protocol.

Specifically the issue is at the point of the protocol where the last voter casts its vote. Suppose there are m participants and $m-1$ participants have already published their votes c_i. Then the last voter can check the result of the protocol just by adding $s_m y_m$ to $\sum_{i=1}^{m-1} c_i$ and see if the infinity norm is small, hence the result is no-veto, and if the infinity norm is more than the acceptance bound of a no-veto result, then the final result is veto. If the last voter wants to overturn a veto result into a no-veto one, the last voter must choose c_m such that $c_m + \sum_{i=1}^{m-1} c_i$ is small which he can easily do by choosing the coefficients of c_m so that the sum falls within the acceptance bound of a no-veto result.

Hence, to resist such an attack of an active adversary controlling all but at least two parties, [Ding, Emery, et al. 2020] proposed an actively secure version of the protocol by employing a commitment scheme before publishing the b_i's and c_i's.

4.2 Actively Secure Lattice-Based AV-Net

We now describe how the passively secure veto protocol from Section 4.1 can be extended in order to defend against active adversaries that want to break privacy as well as the correctness of the voting result. Now, we employ an arbitrary lattice-based commitment scheme $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Open})$ which is (at least) computationally hiding and (at least) computationally binding under standard lattice hardness assumptions. The actively secure protocol specifies that voters open their commitments exactly in the reverse order according to which they published them to resist such active attacks we mentioned at the end of Section 4.1.

More precisely, we extend the veto protocol from Section 4.1 as follows. Denote a generic commitment scheme by $(\mathsf{KGen}, \mathsf{Com}, \mathsf{Open})$.

Parameters (extended). We denote by prm the joint public parameters of the commitment scheme (computed by running KGen).

Offline phase (extended). Each voter V_i, after having computed b_i, executes the following steps:

3. Compute $(\gamma_i, \rho_i) \leftarrow \mathsf{Com}(\mathsf{prm}, b_i)$.*
4. Publish γ_i.
5. Wait until all γ_j were published ($j \in \{1, \ldots, m\}$).
6. Set $\sigma \leftarrow$ order of published γ_j's (according to their time stamps).
7. Wait until all (b_j, ρ_j) were published for $\sigma(j) > \sigma(i)$.
8. Publish (b_i, ρ_i).
9. Wait until all (b_j, ρ_j) were published for $\sigma(j) < \sigma(i)$.
10. If $\mathsf{Open}(\mathsf{prm}, b_j, \gamma_j, \rho_j) = 0$ for some $j \neq i$, then abort.

Online phase (extended). Each voter V_i, after having computed c_i, executes the following steps:

3. Compute $(\gamma'_i, \rho'_i) \leftarrow \mathsf{Com}(\mathsf{prm}, c_i)$.
4. Publish γ'_i.
5. Wait until all γ'_j were published ($j \in \{1, \ldots, m\}$).
6. Set $\sigma' \leftarrow$ order of published γ'_j's (according to their time stamps).
7. Wait until all (c_j, ρ'_j) were published for $\sigma'(j) > \sigma'(i)$.
8. Publish (c_i, ρ'_i).
9. Wait until all (c_j, ρ'_j) were published for $\sigma'(j) < \sigma'(i)$.
10. If $\mathsf{Open}(\mathsf{prm}, c_j, \gamma'_j, \rho'_j) = 0$ for some $j \neq i$, then abort.

*In other words, γ_i is the commitment to b_i using randomness ρ_i.

Verifiable correctness

In this section, we show that the veto protocol defined in Section **??** is verifiably correct even if an arbitrary adversary actively corrupts (a subset of) voters.

We note that we can restrict our attention to the case that an adversary aims to swap an honest "veto" into "no veto". In fact, if an adversary (controlling at least one voter) wants the final result to be "veto", then he can simply let the corrupted voter run her "veto" program.

The theorem below shows verifiable correctness of the actively secure version of the veto protocol. Also, we include the proof from [Ding, Emery, et al. 2020] to illustrate how this modified version resists the attack of an adversary who may adaptively change its input, specifically in the case where it tries to overturn a veto result into a no-veto one.

Theorem 4.2.1 (Verifiable correctness)**.** *[Ding, Emery, et al. 2020, Theorem 4] Let* Π_{veto} *be the veto protocol defined in Section 4.2. Assume that the bulletin board* B *is honest. Assume that the commitment scheme is computationally binding and hiding. Then, we have that for all runs (of these instances) of* Π_{veto}*, the following implication holds true with overwhelming probability: If there exists an honest voter who chooses "veto", then the final result is "veto" (or the protocol aborts prematurely).*

Proof. We follow the proof from the paper [Ding, Emery, et al. 2020].

We assume without loss of generality that there exists one honest voter, namely, V_1. This voter always chooses "veto". Furthermore, we first restrict our attention to the case that there exists one more voter, V_2, which is controlled by an arbitrary PPT adversary A. Now, we distinguish between the following two sets of protocol runs:

1. Voter V_1 publishes γ_1' before voter V_2 has published γ_2'.

2. Voter V_2 publishes γ_2' before voter V_1 has published γ_1'.

In the first set of protocol runs, the probability that there is a final result and that this result is "no veto" equals to the probability that an arbitrary adversary A' can win the following game (run with challenger C):

1. C: choose $c_1 \xleftarrow{r} R_q$
2. C: compute $(\gamma_1, \rho_1) \leftarrow \mathsf{Com}(\mathsf{prm}, c_1)$
3. C: return γ_1
4. A': return c_2
5. A' wins if and only if $\|c_1 + c_2\|_\infty < \frac{q}{4} - 2$

Since we assume that the commitment scheme is *computationally hiding* and since Lemma 4.1.1 holds true, any PPT A' can win this game only with at most negligible probability.

In the second set of protocol runs, the probability that there is a final result and that this result is "no veto" equals to the probability that an arbitrary adversary A' can win the following game (run with challenger C):

1. A': return γ_2
2. C: choose $c_1 \xleftarrow{r} R_q$
3. C: return c_1
4. A': return (c_2, ρ_2)
5. A' wins if and only if $\|c_1 + c_2\|_\infty < \frac{q}{4} - 2$ and $\mathsf{Open}(\mathsf{prm}, c_2, \gamma_2, \rho_2) = 1$.

Since we assume that the commitment scheme is *computationally binding* and since Lemma 4.1.1 holds true, any PPT A' can win this game only with at most negligible probability.

This proves that Theorem 4.2.1 holds true for one honest plus one dishonest voter. It is easy to see that the more general result, i.e., Theorem 4.2.1 with an arbitrary number of dishonest voters, effectively reduces to the two cases with one dishonest voter discussed above. ■

Privacy

Finally privacy of individual votes from honest voters in the presence of malicious adversaries hold by the theorem below:

Theorem 4.2.2 (Privacy)**.** *[Ding, Emery, et al. 2020, Theorem 5] Assume that the* RLWE *hardness result holds true. Assume that the commitment scheme is computationally binding and hiding. Let A be an arbitrary malicious ppt adversary which controls (at most) all but two voters* $(\mathsf{V}_i)_{i\in\mathcal{I}_{dis}}$*. Let* $(\mathsf{V}_i)_{i\in\mathcal{I}_{hon}}$ *denote the remaining (uncorrupted) voters. Let* $(v_i)_{i\in\mathcal{I}_{hon}}$ *and* $(v_i')_{i\in\mathcal{I}_{hon}}$ *be two arbitrary vectors of choices that yield the same result* res*. Then, the probability that the adversary A can distinguish between the set of runs in which the honest voters* $(\mathsf{V}_i)_{i\in\mathcal{I}_{hon}}$ *vote according to* $(v_i)_{i\in\mathcal{I}_{hon}}$ *or to* $(v_i')_{i\in\mathcal{I}_{hon}}$ *is negligible.*

Chapter 5

Self-tallying voting protocol

In this chapter, we will present a self-tallying voting protocol which involves m voters, let's call the voters V_i for $i \in \{1, \ldots, m\}$ and two candidates, say, C_1 and C_2. We will present two variants of the protocol: one is secure against semi-honest adversaries and the other is secure against malicious adversaries.

Furthermore, we will mention an extension of our protocol which is the case involving multiple candidates, possibly more than 2. In this chapter, we will only describe the protocol specifications and the proof of correctness and leave out the security proofs in Chapter 6.

5.1 Self-tallying voting protocol specification

Suppose that there are m voters and 2 candidates and an authenticated broadcast channel which all the voters and candidates trust as secure. The result of the voting protocol is very simple: we will just tally all the votes for C_1 and C_2 and whoever gets the higher number of votes wins. In our protocol description, the voters will cast a vote of 0 for C_1 and a vote of 1 for C_2. The final tally of votes will be the sum of all the 0 votes and the sum of all the 1 votes. However, the voters do not want anyone to know how they

voted, that is, for some reason they want to keep the privacy of their votes. To achieve privacy of the votes, we encrypt the votes and we claim privacy using the RLWE hardness assumption. But at the same time, when they cast their votes, other participants of the protocol must be convinced that the encrypted votes are honestly computed according to the protocol specification without revealing any information how the voters have voted other than the fact that the vote is either a 0 vote or a 1 vote. To convince other participants that votes are generated according to the protocol specification, we propose the use of $\Sigma'-$protocols which we discussed in Section 3.9.3.

Now, let's describe our self-tallying voting protocol involving m voters V_i and two candidates C_1 and C_2. First, we set the following parameters:

- Let n be the main security parameter, where n is a power of two.
- Let $\sigma > 0$ be the standard deviation of the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$.
- Let m be the number of voters.
- Let $q \equiv 3 \bmod 8$ be a prime such that

$$m + m\sigma\sqrt{n}[(m^2-1)\sigma\sqrt{n} + m + 1] < \frac{q}{4} - 2 \tag{5.1.1}$$

- Let $R_q = R/qR = \mathbb{Z}_q[X]/(X^n + 1)$.

Choosing q in Equation 5.2.1 ensures the correctness of our protocol since as we tally all the votes, we have to make sure that the coefficients of the total error does not wrap around modulo q.

Initially, all voters agree on a public, uniformly random chosen $a \in R_q$ and an authenticated broadcast channel, for example, an authenticated message board where all participants can post and view all the public messages. Also, for simplicity let χ be the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ and we let χ to be public knowledge among all the participants of the protocol.

Round 1: Each voter P_i randomly selects a secret-error pair $(s_i, e_i) \leftarrow \chi^2$. Voter P_i computes $b_i := a \cdot s_i + (m+1)e_i$ and publishes b_i and a $\Sigma'-$proof described in Figure 3.5 for $((a, b_i), (s_i, e_i)) \in \mathcal{R}_0 \subseteq \mathcal{R}'_0$.

Once all the b_i are published and all the $\Sigma'-$proofs are verified, each participant computes

$$y_i = \sum_{j=1}^{i-1} b_j - \sum_{j=i+1}^{m} b_j. \tag{5.1.2}$$

Round 2: Each voter P_i chooses a random $e'_i \leftarrow \chi$ and casts the vote c_i where

$$c_i = \begin{cases} s_i y_i + (m+1)e'_i & \text{if } P_i \text{ votes for } C_1 \\ s_i y_i + (m+1)e'_i + 1 & \text{if } P_i \text{ votes for } C_2 \end{cases},$$

along with a $\Sigma'-$proof for $((y_i, c_i), (s_i, e'_i)) \in \mathcal{R}_{OR} \subseteq \mathcal{R}'_{OR}$ described in Figure 3.6. Finally, once all the c_i are published and all its corresponding $\Sigma'-$proofs are verified, everyone computes for $k \equiv \left(\sum_{i=1}^{m} c_i \mod q \right) \mod (m+1)$, where $k \in \{0, \ldots, m\}$. This k gives the total number of votes for Candidate 2 and consequently, gives $m - k$ votes for Candidate 1. The result is accepted as long as

$$\left\| \sum_{i=1}^{m} c_i \right\|_\infty \leq \frac{q}{4} - 2.$$

Protocol 4.1 : Voting Protocol Summary

1. Each voter computes $b_i = as_i + (m+1)e_i$ where $s_i, e_i \in \chi$ and publishes b_i and a $\Sigma'-$proof corresponding to (a, b_i).

2. After all the b_i and its corresponding proofs are published and verified, the $i-$th voter computes y_i according to Equation 5.1.2.

3. To cast votes, voters compute c_i and publishes it and a corresponding OR-proof.

4. After all the votes are published and its proofs verified, everyone tallies the result by computing $k = \sum_{i=1}^{m} c_i \bmod q \bmod (m+1)$ and its value corresponds to the number of votes for candidate C_2 while $m - k$ is the number of votes received by candidate C_1. Everyone checks if $\left\| \sum_{i=1}^{m} c_i \right\|_\infty \leq \frac{q}{4} - 2$ and if that is the case, accepts the voting result.

Notice that using the $\Sigma'-$proof for $((y_i, c_i), (s_i, e_i)) \in \mathcal{R}_{OR} \subseteq \mathcal{R}'_{OR}$ ensures that only an encryption of either a 0 vote or a 1 vote is being casted by a voter in the protocol. Also, we may assume that the $\Sigma'-$proofs in both rounds are non-interactive by applying the Fiat-Shamir technique so in addition, all the participants agree on a cryptographic hash function whose range serve as the challenge set. To prove the correctness of the protocol with overwhelming probability, we need the following Lemma 2.1.2 from [Hao and Zieliński 2006] which serves as the trick to properly decrypt the aggregated votes. For completeness, we provide a proof.

Now, we prove the correctness of our protocol. The proposition below tells us that the final tally of votes decrypts properly with overwhelming probability in the security parameter n as long as all the voters follow the voting protocol specification.

Proposition 5.1.1. *Let n be a power of two, $\sigma > 0$ and let m be the number of voters. Let $q \equiv 3 \bmod 8$ be a prime such that*

$$m + m\sigma\sqrt{n}[(m^2-1)\sigma\sqrt{n} + m + 1] < \frac{q}{4} - 2.$$

Let s_i and y_i be defined as in (5.1.2), respectively. Then we have

$$k \equiv \left(\sum_{i=1}^{m} c_i \mod q \right) \mod (m+1)$$

where k is the total number of votes for Candidate 2, with overwhelming probability.

Proof. Notice that

$$\sum_{i=1}^{m} c_i = \sum_{i=1}^{m} s_i y_i + (m+1) \sum_{i=1}^{m} e_i' + k \mod q$$

where k is the total number of votes for Candidate 2. Moreover, we see that

$$\begin{aligned} \sum_{i=1}^{m} s_i y_i &= a \left(\sum_{i=1}^{m} \sum_{j=1}^{i-1} s_i s_j - \sum_{i=1}^{m} \sum_{j=i+1}^{m} s_i s_j \right) + (m+1) \left(\sum_{i=1}^{m} \sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m} \sum_{j=i+1}^{m} s_i e_j \right) \\ &= (m+1) \left(\sum_{i=1}^{m} \sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m} \sum_{j=i+1}^{m} s_i e_j \right). \end{aligned}$$

Thus, we get

$$\sum_{i=1}^{m} c_i \equiv k + (m+1) \left(\sum_{i=1}^{m} \sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m} \sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i' \right) \mod q.$$

Observe that

$$\begin{aligned}
\left\|\sum_{i=1}^{m} c_i\right\|_\infty &= \left\|k + (m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i'\right)\right\|_\infty \\
&\leq m + (m+1)\left(\sum_{i=1}^{m}\sum_{j\neq i} \|s_i e_j\|_\infty + \sum_{i=1}^{m} \|e_i'\|_\infty\right) \\
&\leq m + (m+1)\left(\sum_{i=1}^{m}\sum_{j\neq i} \|s_i\| \cdot \|e_j\| + \sum_{i=1}^{m} \|e_i'\|\right) \\
&\leq m + (m+1)(m(m-1)\sigma^2 n + m\sigma\sqrt{n}) \\
&= m + m\sigma\sqrt{n}[(m^2-1)\sigma\sqrt{n} + m + 1] \\
&\leq \frac{q}{4} - 2
\end{aligned}$$

where the first line of inequality is by triangle inequality, the next line of inequality holds via Lemma 3.7.7 and the third line of inequality comes from Lemma 3.3.13.

We claim that

$$(m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i'\right) \mod q$$

has infinity norm less than or equal to $\frac{q-1}{2}$. Suppose otherwise, that is, it has infinity norm greater than $\frac{q-1}{2}$. Then

$$\begin{aligned}
\left\|\sum_{i=1}^{m} c_i\right\|_\infty &= \left\|k + (m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i'\right)\right\|_\infty \\
&\geq \left\|(m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i'\right)\right\|_\infty - m \\
&> \frac{q-1}{2} - m.
\end{aligned}$$

Thus, $\frac{q}{4} - 2 > \frac{q-1}{2} - m$ which is equivalent to $q < 4m - 6$ which contradicts our choice for q.

Hence, all coefficients of

$$(m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e'_i\right) \mod q$$

are divisible by $m+1$. Hence, we have

$$k \equiv \left(\sum_{i=1}^{m} c_i \mod q\right) \mod (m+1).$$

■

Remark 5.1.2. Notice that the construction of b_i for each i corresponds to an RLWE sample (a, b_i) whose secret s_i. Also, notice that each y_i is in fact equal to

$$y_i = a\tilde{s}_i + (m+1)\tilde{e}_i,$$

where $\tilde{s}_i = \sum_{j<i} s_j - \sum_{j>i} s_j$ and $\tilde{e}_i = \sum_{j<i} e_j - \sum_{j>i} e_j$ so that $(y_i, y_i s_i + (m+1)\tilde{e}_i)$ is also an RLWE sample and by the RLWE hardness assumption, y_i and c_i, whether $c_i = y_i s_i + (m+1)\tilde{e}_i$ or $y_i s_i + (m+1)\tilde{e}_i + 1$ is indistinguishable from a uniformly random element in R_q. Hence, the encrypted votes are pseudorandom and hence, any passive attacker cannot distinguish the individual votes of the honest voters. We discuss more on privacy of votes in Chapter 6.

Remark 5.1.3. The reason we require that $\left\|\sum_{i=1}^{m} c_i\right\|_\infty \leq \frac{q}{4} - 2$ must hold true for the voting result to be accepted is to make sure that all the voters use their secret s_i in constructing their encrypted vote c_i. If there is a malicious attack where a voter instead use a different secret, say s'_i , other than the s_i used in constructing the b_i, to construct c_i, then Lemma 2.1.2 will not apply in our protocol which would mean that no cancellations

occur. If that is the case, the sum $\sum_{i=1}^{m} c_i$ can regarded as an RLWE sample as well, hence pseudorandom in R_q. Hence, the probability that $\sum_{i=1}^{m} c_i$ has norm $\frac{q}{4} - 2$ is negligible in n, and therefore the voting result will be rejected.

5.2 A modified self-tallying voting protocol: multiple candidates case

Now, we present a version of the voting protocol involving multiple candidates, in particular, more than two candidates. The difference between the protocol involving two candidates and the protocol we present in this section is that instead of encrypting 0 and 1 as votes, we will instead encrypt a monomial X^{k-1}, where k is a positive integer corresponding to candidate C_k, for $k \in \{0, 1, \ldots, t\}$, where t is the total number of candidates.

In this version, we will not provide $\Sigma'-$proofs but one can devise an analogue of the $\Sigma'-$proofs we discussed in Section 3.9.3 for this modified version.

Now, let's describe our self-tallying voting protocol involving m voters $V_1, V_2, \ldots, V_m$ and a number of t candidates, say $C_1, C_2, \ldots, C_t$ First, we set the following parameters similar as in the previous protocol:

- Let n be the main security parameter, where n is a power of two.
- Let $\sigma > 0$ be the standard deviation of the discrete Gaussian distribution $\chi := D_{\mathbb{Z}^n,\sigma}$.
- Let m be the number of voters.
- Let $q \equiv 3 \bmod 8$ be a prime such that

$$m + m\sigma\sqrt{n}[(m^2 - 1)\sigma\sqrt{n} + m + 1] < \frac{q}{4} - 2 \tag{5.2.1}$$

- Let $R_q = R/qR = \mathbb{Z}_q[X]/(X^n + 1)$.

As in the previous protocol, all voters agree on a public, uniformly random chosen $a \in R_q$.

Round 1: Each voter P_i randomly selects a secret-error pair $(s_i, e_i) \leftarrow \chi^2$. Voter P_i computes $b_i := a \cdot s_i + (m+1)e_i$ and publishes b_i.

Once all the b_i are published, each participant computes

$$y_i = \sum_{j=1}^{i-1} b_j - \sum_{j=i+1}^{m} b_j.$$

Round 2: Each voter P_i chooses a random $e_i' \leftarrow \chi$ and casts the vote for candidate C_k where $k \in \{1, \ldots, t\}$ as the "encryption" of the monomial X^{k-1}, that is, P_i's vote for candidate k is encrypted as $c_i = s_i y_i + (m+1)e_i' + X^{k-1}$. Finally, once all the c_i are published, all voters compute $\sum_{i=1}^{m} c_i \mod q \mod (m+1)$. Again, the voting result is only accepted if $\left\| \sum_{i=1}^{m} c_i \right\|_\infty \leq \frac{q}{4} - 2$.

Proposition 5.2.1. *Let s_i and y_i be defined as above. Let $\beta = \sigma\sqrt{n}$. Assume that $m + m\beta[(m^2-1)\beta + m + 1] < \frac{q}{4} - 2$. Then we have*

$$p(X) \equiv \left(\sum_{i=1}^{m} c_i \mod q \right) \mod (m+1),$$

where $p(X) = v_0 + v_1 X + \cdots + v_{t-1} X^{t-1}$, $v_k \in \{0, \ldots, t-1\}$ for each k and $v_0 + \ldots + v_{t-1} = m$, so that v_k is the number of votes for candidate k.

Proof.

Notice that

$$\sum_{i=1}^{m} c_i = \sum_{i=1}^{m} s_i y_i + (m+1) \sum_{i=1}^{m} e_i' + p(X) \mod q,$$

where $p(X) = v_0 + v_1 X + \cdots + v_{t-1}X^{t-1}$, $v_{k-1} \in \{0, \ldots, m\}$ for each $k \in \{1, \ldots, t\}$ such that $v_0 + \ldots + v_{t-1} = m$, where v_{k-1} is the number of votes for candidate C_k.

Moreover, we see that

$$\begin{aligned}\sum_{i=1}^{m} s_i y_i &= a\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i s_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i s_j\right) + (m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j\right) \\ &= (m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j\right).\end{aligned}$$

Thus, we get

$$\sum_{i=1}^{m} c_i \equiv p(X) + (m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i'\right) \mod q.$$

Note that $p(X)$ has infinity norm at most m and therefore we have,

$$\begin{aligned}\left\|\sum_{i=1}^{m} c_i\right\|_\infty &= \left\|p(X) + (m+1)\left(\sum_{i=1}^{m}\sum_{j=1}^{i-1} s_i e_j - \sum_{i=1}^{m}\sum_{j=i+1}^{m} s_i e_j + \sum_{i=1}^{m} e_i'\right)\right\|_\infty \\ &\leq m + (m+1)\left(\sum_{i=1}^{m}\sum_{j\neq i} \|s_i e_j\|_\infty + \sum_{i=1}^{m} \|e_i'\|_\infty\right) \\ &\leq m + (m+1)\left(\sum_{i=1}^{m}\sum_{j\neq i} \|s_i\| \cdot \|e_j\| + \sum_{i=1}^{m} \|e_i'\|\right) \\ &\leq m + (m+1)(m(m-1)\sigma^2 n + m\sigma\sqrt{n}) \\ &= m + m\sigma\sqrt{n}[(m^2-1)\sigma\sqrt{n} + m + 1] \\ &\leq \frac{q}{4} - 2\end{aligned}$$

Hence, by similar arguments as in the proof of Proposition 5.1.1 we have

$$p(X) \equiv \left(\sum_{i=1}^{m} c_i \mod q\right) \mod (m+1)$$

which proves our proposition. ■

5.3 Choice of parameters to ensure correctness

Recall that given m voters and parameters n a power of two (usually taken to be 512 or 1024 to ensure security against best-known attacks on the RLWE problem) and $\sigma > 0$, the parameter for the error distribution of our voting protocol, we need to choose our modulus q for R_q to satisfy the inequality

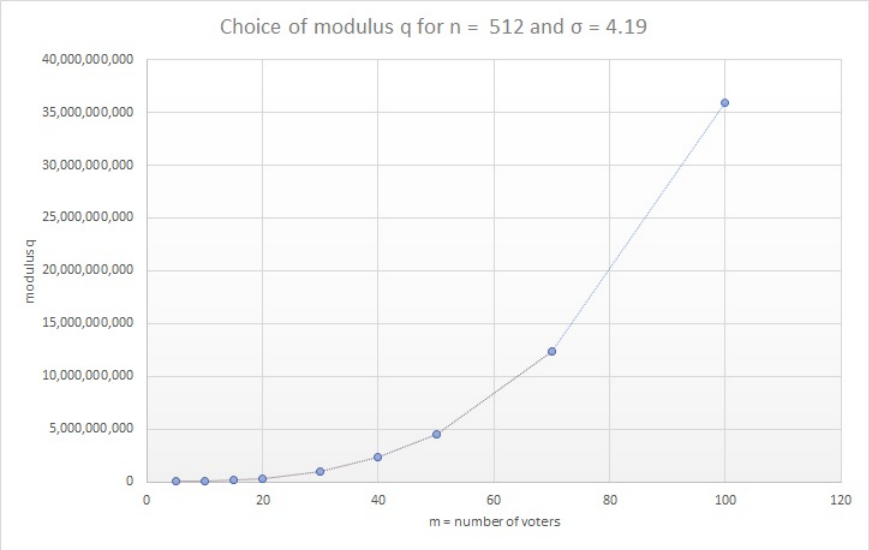
$$m + m\sigma\sqrt{n}[(m^2 - 1)\sigma\sqrt{n} + m + 1] < \frac{q}{4} - 2.$$

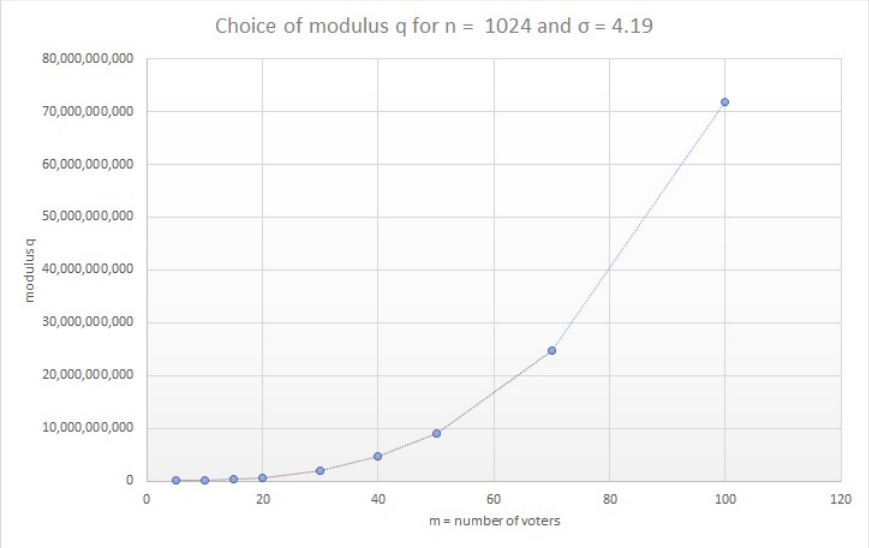
For simplicity, a sufficient condition for modulus q must be at least as large as $4m^3\sigma^2 n$ to guarantee correctness with overwhelming probability in the security parameter n. The drawback of course of our protocol is when the number of voters m becomes at least as large as n which means q will need to be quadratic in n and hence might undermine efficiency. Thus, our voting protocol will not be a viable option for large-scale elections. For example, in the case where $n = 512$ and $\sigma = 4.19$, we have the following table to guarantee correctness with overwhelming probability:

Table 5.1: Choice of modulus q for $n = 512$ and $\sigma = 4.19$

m	$q >$
5	4,325,992
10	35,595,391
15	120,808,507
20	286,920,132
30	969,703,586
40	2,299,675,111
50	4,492,564,063
70	12,330,011,675
100	35,951,297,718

Moreover, we can see in the figures below how we must choose q as we increase the number of voters to ensure correctness for fixed security parameter n and error distribution parameter σ.





5.4 Performance

Now, we analyze the performance of our voting protocol. First, we note that the choice of the security parameter n to be a power of two allows us to use the Fast Fourier Transform [Cooley and Tukey 1965] to perform fast ring operations in R_q, in particular, a ring operation (addition or multiplication) between two polynomials in R_q takes $O(n \log n)$ operations. Also, getting a sample from the discrete Gaussian distribution on $\mathbb{Z}^n$ takes at least $O(n^2)$ time according to the sampling procedure in [Peikert 2010]. The rejection sampling technique makes the $\Sigma'-$proofs presented in Section 3.9.3 take on average $O(Mn^2)$ time, where $M = e^{\frac{12}{\omega(\log 2n)} + \frac{1}{\omega(2 \log 2n)}}$. Hence, our voting protocol has the following time complexity: in the key registration phase, where each voter has to compute b_i, it requires two calls on the discrete Gaussian sampler, one polynomial multiplication and addition which takes a total time for each voter $O(n^2) + O(n \log n)$ bit operations. In addition, producing a $\Sigma'-$proof each requires $O(Mn^2)$ time. Computing y_i, requires $m - 1$ polynomial additions which take $O(n)$ for each addition and finally in the voting phase, where each voter casts c_i, requires $O(n^2) + O(n \log n)$ time. Hence, the total complexity of our voting protocol takes $m \cdot (O(Mn^2) + O(n^2) + O(n \log n))$ time on average where $M = e^{\frac{12}{\omega(\log 2n)} + \frac{1}{\omega(2 \log 2n)}}$ such that all the other constant factors are absorbed by the big-O notation.

5.4.1 Experimental results

We have implemented the voting protocol described in Section 5.1. Hence, we have only implemented the voting protocol for two candidates case and the multiple candidates case.

The implementation of the protocol uses the C++ language and NTL library, which is a library for number theoretic calculations. We ran the experiment 10,000 times using the parameters $n = 512, \sigma = 4.19$ with varying q values on a computer with AMD Ryzen 3 3350U CPU with Radeon Vega Mobile Gfx, @ 2100 Mhz, 4 Core(s), 4 Logical Processor(s),

running Cygwin version 3.3.5, g++ compiler version 11.3.0-1. We evaluate the average runtime for discrete Gaussian sampling based on [Peikert 2010] (DGS Time), then the average runtime for ring operations (RO Time), namely addition and multiplication of polynomials in R_q, and vote tallying (Vote Time). Below are the experimental results where we vary the number of voters with two decimal precision shown in table below.

Table 5.2: Runtime (millisecond) of our implementation.

m	q	DGS Time	RO Time	Vote Time
5	120851	2.15	2.88	4.33
10	120851	3.89	7.59	7.79
20	250027	7.38	21.17	14.58
50	1500019	15.64	88.34	32.31

Even though we ran our experiment for a rather smaller value of the modulus q which is way smaller than what would require for the provable correctness of the protocol prescribed by the inequality

$$m + m\sigma\sqrt{n}[(m^2 - 1)\sigma\sqrt{n} + m + 1] < \frac{q}{4} - 2,$$

no errors showed up, in the sense that the voting results are correctly computed in all the executions. But we note that the choice of the modulus q in Table 5.3 is increased as we increase the number of voters m so we always get the correctness of the result for each execution. Hence, experimentally we could choose smaller values of the modulus q to enhance the efficiency of the protocol.

We ran our experiment using the -O2 optimizer where this option increases both compilation time and the performance of the generated code.

In addition, we also ran the experiment for the modified voting protocol but only for the case where there are 4 candidates with similar parameters as in Table 4.1. There is not much noticeable difference in the run times as we see in the table below.

Table 5.3: Runtime (millisecond) of our implementation.

m	q	DGS Time	RO Time	Vote Time
5	120851	2.25	3.06	3.73
10	120851	3.98	7.55	7.21
20	250027	7.57	21.31	14.54
50	1500019	17.19	99.60	37.34

In a similar fashion, no errors occurred in all the executions due to the choice of the modulus q. If we increase the number of voters to say 70 with the same parameters used for the case of 50 voters, we only get correctness 80% of the time.

Chapter 6

Security of the voting protocol

In this chapter, all the definitions of security for multi-party computation follows from Chapter 7 of [Goldreich 2004]. Our goal in this chapter is to prove that our protocol in Chapter 5 is secure in the sense of privacy of the individual votes against two adversarial models: the first one is security against semi-honest adversaries which are adversaries the follow the protocol specification honestly but tries to learn any secret information through the public information available to the adversary during the execution of the protocol. The second adversarial model is a malicious one, that is, an adversary that carries out an active attack on the protocol to learn the secret inputs of the honest participants of the protocol. In the end, any attacker only learns the sum of the votes of the honest voters but not their individual votes. Hence, our protocol is secure in the sense of privacy of the honest individual votes against both adversarial models as we shall see in the next few sections.

6.1 Security against Semi-honest adversaries

Our goal in this section is to define security against static, semi-honest adversaries of a protocol for computing a deterministic $m-$ary functionality $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$

and to prove that our voting protocol is secure in this security model. Here, a static, semi-honest adversary means an adversary $\mathcal{A}$ than controls a fixed number of the parties but these parties still follow the protocol specification. The goal of attack of this type of adversary is to learn information about the secret choices of the honest parties from $\mathcal{A}'$s point of view during the execution of the protocol. Informally, to show security against semi-honest adversaries, we need to show the existence of a simulator $\mathcal{S}$ such that when $\mathcal{S}$ takes as input the input-output pair of the adversary, then $\mathcal{S}$ can generate the view of the adversary on the execution of the protocol, where this view consists of all the inputs, internal random choices, and messages received by all the controlled parties during the execution of the protocol. The above condition on the existence of a simulator captures the idea that anyone can simulate the protocol just by the input and output information of the adversary and hence, any information that the adversary learns can only be extracted from its input and output alone, and nothing else. An additional feature of a secure deterministic protocol also must satisfy that it gives the correct outputs to all the parties (both the honest and semi-honest ones) with overwhelming probability.

Now we give the formal definition of security against semi-honest adversaries.

Definition 6.1.1 (Security against semi-honest adversaries)**.** *Let* $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$ *be an* $m-$*ary deterministic functionality and let* Π *denote a protocol for computing* f*. Let* $I \subseteq [m] := \{1,\ldots,m\}$ *denote the index set of semi-honest parties controlled by adversary* $\mathcal{A}$*. For each* $i \in [m]$*, define the following:*

- *The view of party* P_i *on the execution of* Π *on input* $\bar{x} := \{x_1,\ldots,x_m\}$*, denoted by* $\mathbf{view}_i^{\Pi}(\bar{x})$*, is the triple*

$$(x_i, r^i, m_j^i),$$

  where $x_i, r^i, \{m_j^i\}$ *are the input, internal random coin tosses, and messages received by* P_i*, respectively. Moreover the view of* $\mathcal{A}$ *is just the tuple* $\mathbf{view}_I(\bar{x}) := (\mathbf{view}_i(\bar{x}))_{i\in I}$

- *The output of party P_i after the execution of Π is denoted by $\mathbf{output}_i^{\Pi}(\bar{x})$.*

Notice the difference between $f(\bar{x})$ and $\mathbf{output}^{\Pi}(\bar{x})$, the former does not depend on the protocol, in fact it is the intended output of the functionality while the latter depends on the protocol specification of Π as it tries to securely compute f and it does not guarantee that it outputs the same value as $f(\bar{x})$ all the time.

We say that Π securely computes f if the following conditions hold:

- *Correctness:* $\Pr[\mathbf{output}^{\Pi}(\bar{x}) \neq f(\bar{x})]$ *is a negligible function on the security parameter.*

- *Privacy: There exists an efficient simulator $\mathcal{S}$ such that*

$$\{\mathcal{S}((x_i)_{i\in I}, f_I(\bar{x}))\}_{x_i\in(\{0,1\})^I} \stackrel{\mathsf{c}}{\equiv} \{\mathbf{view}_I(\bar{x})\}$$

These are the steps on how to prove security against semi-honest adversaries:

1. First, summarize your protocol Π which illustrates the interaction between the parties as they try to securely compute the functionality f.

2. After making the illustration in step 1, identify the output of the protocol in presence of semi-honest adversaries and also identify the view of each party i.

3. Now, knowing the view of each party, we prove into cases where we assume if the adversary controls party i. Given the adversary A controlling parties in I, we construct a simulator S, where we feed S the local inputs and local outputs of the controlled parties, and the simulator's task is to output something indistinguishable to the view of the adversary. The view of the adversary is just the tuple consisting of the views of its controlled parties.

Now, we give the proof of security of the protocol for computing the functionality $f : \{0,1\}^m \to (\{0,1\}^*)^m$, where $f(v_1, \ldots, v_m) = \left(\sum_{i=1}^{m} v_i, \ldots, \sum_{i=1}^{m} v_i\right)$.

Proposition 6.1.2. *Suppose that the RLWE hardness assumption holds. Then the protocol* Π *described in Protocol 5.1 securely computes* f *in the presence of static, semi-honest adversaries.*

Proof. Correctness: The proof of correctness is provided in Proposition 5.1.1.

Privacy: Let $I \subseteq [m] := \{1, \ldots, m\}$ be the index set of corrupted voters controlled by an adversary $\mathcal{A}$. For each $i \in I$,

$$\mathbf{view}_i^{\Pi}(v_1, \ldots, v_m) = (v_i, r^i; \{b_j\}_{j \neq i} \cup \{c_j\}_{j \neq i}) \tag{6.1.1}$$

Thus, the view of $\mathcal{A}$ is just $\mathbf{view}_I^{\Pi}(v_1, \ldots, v_m) = \left(\mathbf{view}_i^{\Pi}(v_1, \ldots, v_m)\right)_{i \in I}$. Moreover the input-output pair of $\mathcal{A}$ is given by $\left(\{v_i\}_{i \in I}, \sum_{i=1}^{m} v_i\right)$. Now, our goal is to construct a simulator $\mathcal{S}$ such that given the input-output pair of $\mathcal{A}$ as input to $\mathcal{S}$, $\mathcal{S}$ outputs something that is indistinguishable to the view of $\mathcal{A}$. Before we proceed, notice that the view of $\mathcal{A}$ has all the messages including those of the honest voters. Thus, to construct $\mathcal{S}$, we also need to simulate the behavior of the honest parties. Now we describe the simulation process of the simulator $\mathcal{S}$.

1. For all $i \in [m]$ (hence we include the honest voters' indices as well), $\mathcal{S}$ chooses uniformly random coins as input to the discrete Gaussian sampling algorithm and outputs $s_i, e_i \xleftarrow{\$} \chi$ and computes $b_i = as_i + (m+1)e_i$. Notice that the random coins used by the simulator is indistinguishable from the random coins used in the real execution of the protocol.

2. After computing all the b_i, $\mathcal{S}$ just computes

$$y_i = \sum_{j<i} b_j - \sum_{j>i} b_j. \tag{6.1.2}$$

To simulate the votes and to make sure that the output of $\mathcal{S}$ has the same distribution as the real output of the execution, we need to make sure that the constructed c_j's of $\mathcal{S}$ 'sum up' to the real output $\sum_{i=1}^{m} v_i$.

3. Since $\mathcal{S}$ has both $\{v_i\}_{i\in I}$ and $\sum_{i=1}^{m} v_i$, then $\mathcal{S}$ can compute $\sum_{i\in[m]\backslash I} v_i$, which is the sum of the honest voters' votes. Now, to simulate the encrypted votes of the corrupted parties, $\mathcal{S}$ computes $c_i = s_i y_i + (m+1)e'_i + v_i$ for each $i \in I$, while to simulate the honest voters' individual votes, for each $j \in [m]\backslash I$, $\mathcal{S}$ just chooses a random sequence $\{v'_j\}_{j\in[m]\backslash I}$ such that $v'_j \in \{0,1\}$ with the property that $\sum_{j\in[m]\backslash I} v'_j = \sum_{i\in[m]\backslash I} v_i$. Then for each $j \in [m]\backslash I$, $\mathcal{S}$ computes $c_j = s_j y_j + (m+1)e'_j + v'_j$, where $e'_j \xleftarrow{\$} \chi$.

4. Finally $\mathcal{S}$ outputs $\left((v_i, r^i; \{b_j\}_{j\neq i} \cup \{c_j\}_{j\neq i})\right)_{i\in I}$.

Notice that the output of $\mathcal{S}$ is indistinguishable from $\mathbf{view}_I^{\Pi}(v_1, \ldots, v_m)$ since the b_j's are indistinguishable from uniform by the RLWE assumption. Moreover, the individual, simulated c_j's as well as the real ones are indistinguishable from uniform by the RLWE assumption as well and by construction, the simulated c_j's has the property that $\sum_{j=1}^{m} c_j = \sum_{i=1}^{m} v_i \pmod{m+1}$ with overwhelming probability. Hence, any distinguisher of the real view of the adversary $\mathcal{A}$ and the simulated view has only negligible advantage of distinguishing. Hence, this proves security of the protocol Π against semi-honest adversaries. ■

6.2 Security against Malicious Adversaries with abort

In this subsection, we present the malicious model with abort, that is, the adversary has the ability to abort the execution of the protocol and can control an arbitrary number of parties.

To prove security in the above model, the approach is completely different compared in the semi-honest case, that is, we prove security via the "real-vs-ideal" model paradigm. What we mean by this is that given the real execution of the protocol Π for computing f in the presence of a malicious adversary $\mathcal{A}$, we can construct an "ideal" adversarial model for computing f such that the outputs of the parties in the real model and the ideal model are indistinguishable. The idea is that the adversary in the ideal model makes subroutine calls to the adversary in the real model and at the end of the ideal execution, the outputs of the parties in the ideal model are indistinguishable to the real outputs.

6.2.1 The ideal model - malicious model with abort

Assume that there is a malicious adversary controlling a fixed number of parties and we assume that this malicious adversary can instruct one of its controlled parties to abort at any moment during the execution of the protocol.

We define how the ideal model execution works for computing the functionality f. The ideal model assumes that there is a trusted third party $\mathcal{T}$ where all the parties send their inputs to. Upon receiving all the inputs from the parties, $\mathcal{T}$ honestly computes each output of the parties, that is, given the inputs x_i of party i, $\mathcal{T}$ computes each $f_i(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_m)$, m is the number of parties. Now, we consider two cases: the first case is when Party 1 is honest and the other is when Party 1 is corrupted. Suppose that Party 1 is honest. In this case, we postulate that no aborts occur during the ideal execution for computing the functionality. Thus, all the parties receive their corresponding outputs from $\mathcal{T}$. At the end of the execution, the honest parties output what they received from $\mathcal{T}$, while the corrupted parties output whatever the ideal adversary $\mathcal{B}$ tells them to output. On the other hand, if Party 1 is corrupted, upon receiving its output from $\mathcal{T}$ and before $\mathcal{T}$ sends the output $f_2(\bar{x})$ to Party 2, Party 1 may send an abort message to $\mathcal{T}$ and in this case, the execution for computing f halts. Otherwise, if Party 1 decides to continue, then the execution proceeds as in the previous case. We formally define

this ideal execution below. Note that in the ideal execution, the corrupted parties may change their inputs upon the instruction of the adversary that controls them.

Definition 6.2.1 (malicious ideal model with abort)**.** *Let $f : (\{0,1\}^*)^m \to (\{0,1\}^*)^m$ be an m-ary functionality. For $I \subseteq [m] := \{1,\ldots,m\}$, let $\bar{I} = [m]\backslash I$ and let $\bar{x} = \{x_1,\ldots,x_m\}$. A pair $(I,\mathcal{B})$, where $\mathcal{B}$ is a probabilistic, polynomial-time algorithm represents an adversary in the ideal model. The joint execution of f under $(I,\mathcal{B})$ in the ideal model, denoted $\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{x}) := \Psi(\bar{x},I,z,r)$, where z is the auxiliary input of $\mathcal{B}$ and r is a uniformly chosen random tape for $\mathcal{B}$ is defined as follows:*

- *If Party 1 is honest,*

$$\Psi(\bar{x},I,z,r) := (f_{\bar{I}}(\bar{x}'),\mathcal{B}(\bar{x}_I,I,z,r,f_I(\bar{x}'))), \tag{6.2.1}$$

  where $\bar{x}' := (x'_1,\ldots,x'_m)$ such that $x'_i = \mathcal{B}(\bar{x}_I,I,z,r)_i$ for $i \in I$, and $x'_i = x_i$ otherwise and $f_{\bar{I}}(\bar{x}') := (f_i(\bar{x}'))_{i\in\bar{I}}$

- *If Party 1 is not honest, $\Psi(\bar{x},I,z,r)$ equals*

$$(\perp^{|\bar{I}|},\mathcal{B}(\bar{x}_I,I,z,r,f_I(\bar{x}'),\perp)) \text{ if } \mathcal{B}(\bar{x}_I,I,z,r,f_I(\bar{x}')) = \perp \tag{6.2.2}$$

$$(f_{\bar{I}}(\bar{x}'),\mathcal{B}(\bar{x}_I,I,z,r,f_I(\bar{x}'))) \text{ otherwise} \tag{6.2.3}$$

  where $\bar{x}' := (x'_1,\ldots,x'_m)$ such that $x'_I = \mathcal{B}(\bar{x}_I,I,z,r)_i$ for $i \in I$, and $x'_i = x_i$ otherwise.

Remark 6.2.2. *In the above definition, the notation $\mathcal{B}(\cdot)$ means the output of the algorithm $\mathcal{B}$ given an input and this output represents the behavior of the adversary controlling the corrupted parties, which captures the malicious behavior that the inputs of the corrupted parties sent to the trusted party may be substituted by $\mathcal{B}$.*

Now, we formally define the real execution of a protocol Π for computing f in the presence of a real, malicious adversary $\mathcal{A}$.

Definition 6.2.3 (real malicious model)**.** *Given a functionality f and a protocol Π for computing f, the joint execution of Π under $(I, \mathcal{A})$ in the real model, where $(I, \mathcal{A})$ represents a real adversary, denoted by $\mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{x})$, is defined as the output sequence resulting from the interaction between all the parties, where the messages of the corrupted parties are computed according to $\mathcal{A}(\bar{x}_I, I, z)$ and the messages of the honest parties are computed according to Π. Here, the messages of the corrupted parties are determined by the adversary $\mathcal{A}$ based on the initial inputs of the corrupted parties, the auxiliary input z of $\mathcal{A}$, and all the public messages sent by all parties, including the honest ones.*

6.2.2 Security in the malicious model with abort

Now, we define security of a protocol Π in the malicious model for computing a functionality f. Informally, the proceeding definition says that given any real adversary $\mathcal{A}$ of Π, there exists an adversary $\mathcal{B}$ in the ideal model such that for any fixed $I \subseteq [m]$, the joint execution in the real model under $(I, \mathcal{A})$ and the joint execution in the ideal model under $(I, \mathcal{B})$ are indistinguishable. In other words, since the executions are indistinguishable, if $\mathcal{A}$ successfully carries out an attack in Π, there is an adversary $\mathcal{B}$ in the ideal model that also carries out a successful attack with the same effect as the attack of $\mathcal{A}$.

We now give the formal definition.

Definition 6.2.4 (security in the malicious model with abort)**.** *Protocol Π securely computes f if for every probabilistic, polynomial-time algorithm $\mathcal{A}$ (representing a real model adversary), there exists a probabilistic, polynomial-time algorithm $\mathcal{B}$ (representing an ideal model adversary), such that for every $I \subseteq [m]$,*

$$\left\{\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{x})\right\}_{\bar{x},z} \stackrel{\mathsf{c}}{\equiv} \left\{\mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{x})\right\}_{\bar{x},z} \tag{6.2.4}$$

Intuitively, the ideal model is assumed to be “secure" against any adversary so if the definition of security in the first malicious model holds for a protocol Π for computing f, then Π also resists any adversarial behavior.

6.2.3 Security proof of the voting protocol against malicious adversaries with abort

Before we give the security proof of our protocol against malicious adversaries, let's briefly recall the two relations $\mathcal{R}$ and $\mathcal{R}_{\text{OR}}$. Given a fixed integer m and $b \in \{0,1\}$, the relation $\mathcal{R}_b$ is given by

$$\mathcal{R}_b = \{((a,y),(s,e)) \in R_q^2 \times R_q^2 : y = as + (m+1)e + b, \text{ for some } \|s\|_\infty, \|e\|_\infty \leq \beta\} \tag{6.2.5}$$

while $\mathcal{R}_{\text{OR}}$ is given by

$$\mathcal{R}_{\text{OR}} = \{((a,y),(s,e)) \in R_q^2 \times R_q^2 : ((a,y),(s,e)) \in \mathcal{R}_0 \cup \mathcal{R}_1\}. \tag{6.2.6}$$

We have shown in the previous chapter the existence of Σ'-proofs for the above relations. Let's first recall our protocol Π for computing f, where $f(v_1,\ldots,v_m) = \left(\sum_{i=1}^{m} v_i, \ldots, \sum_{i=1}^{m} v_i\right)$, $v_i \in \{0,1\}$, that is, f is an example of a deterministic functionality that computes the result of an election consisting of two candidates with m voters. Π is described as follows:

- First, all voters agree on a uniformly, random chosen $a \xleftarrow{\$} R_q$ and on an authenticated message board that displays their public keys and encrypted votes.
- **Key registration phase.** Each voter V_i runs the discrete Gaussian sampler twice and receives $s_i, e_i \xleftarrow{\$} \chi$. Then V_i computes

$$b_i = as_i + (m+1)e_i \tag{6.2.7}$$

and runs a Σ'-proof for $((a, b_i), (s_i, e_i)) \in \mathcal{R}_0$ as described in Section 3.9.3. Then V_i posts b_i as well as its Σ'-proof to the message board. After all the b_i's and their proofs are published. Once all the proofs are checked as valid, V_i computes its voting key y_i as

$$y_i = \sum_{j<i} b_j - \sum_{j>i} b_j. \tag{6.2.8}$$

Note that every voter can compute the voting keys of all the other voters.

- **Voting phase.** To cast vote $v_i \in \{0, 1\}$ of voter V_i, V_i chooses $e_i' \xleftarrow{\$} \chi$ and computes

$$c_i = y_i s_i + (m + 1)e_i' + v_i \tag{6.2.9}$$

and runs a Σ'-proof for $((y_i, c_i), (s_i, e_i')) \in \mathcal{R}_{\text{OR}}$ as described in Section 3.9.3.

- **Result phase.** After all c_i are published and all Σ'-proofs are checked as valid, every voter computes the result of the election as

$$\sum_{i=1}^{m} c_i \bmod (m + 1). \tag{6.2.10}$$

Before we prove security of the above protocol in the malicious model, let's take a look at the ideal model execution of computing the functionality f and the real model execution Π of computing f in the presence of malicious adversaries. First, let's describe the ideal model execution.

A malicious ideal model for voting.

Let $I \subseteq [m]$ and suppose $\mathcal{B}$ is a malicious adversary in the ideal model that controls the voters V_i, $i \in I$. Let $\bar{I} = [m] \backslash I$. Let z be the auxiliary input and r be the internal random coins of $\mathcal{B}$. Denote by $\mathcal{T}$ the trusted third party in this ideal model. The inputs of all the voters are their votes $v_i \in \{0, 1\}$.

- Every honest voter sends its vote v_i to $\mathcal{T}$ while a corrupted voter sends $v'_j \xleftarrow{\$} \mathcal{B}(\bar{v}_I, z, r)_j$ to $\mathcal{T}$.

- Upon receiving all the v_i, $i \in \bar{I}$ and v'_j, $j \in I$, $\mathcal{T}$ computes

$$\mathsf{res} := \sum_{i \in \bar{I}} v_i + \sum_{j \in I} v'_j \tag{6.2.11}$$

- Then $\mathcal{T}$ sends res first to the first voter who sent its vote, say this voter is V_1. If V_1 is honest, then $\mathcal{T}$ proceeds by sending res to all the other voters. If V_1 is not honest, it may stop $\mathcal{T}$ from sending res to all the other voters by sending an abort message $\perp$.

- In the case that no abort has occurred, the honest voters output res while the corrupt voter V_j outputs $\mathcal{B}(\bar{v}_I, z, r, \mathsf{res})_j$, where $j \in I$.

- Thus, $\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{v}) = (\mathsf{res}, \mathcal{B}(\bar{v}_I, z, r, \mathsf{res}))$ which denotes the sequence of outputs of all the voters. In the case that V_1 aborts, $\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{v}) = (\perp, \mathcal{B}(\bar{v}_I, z, r, \mathsf{res}, \perp))$.

The malicious real model for voting.

In the real execution of Π, the adversary $\mathcal{A}$ controlling a fixed number of corrupted parties determines the inputs of the corrupted voters on the message board. Let $\bar{v} := (v_1, \ldots, v_m)$ denote the sequence of votes, where $v_i \in \{0, 1\}$ is the vote of voter V_i. Suppose I is the index set for the corrupted voters and let $i \in I$. Then in the key registration phase, V_i sends b'_i and a Σ'-proof for $(a, b'_i) \in \mathcal{R}_0$ where $b'_i \leftarrow \mathcal{A}(z, r)_i$ to the message board. Now, for each $k \in [m]$, $y_k := \sum_{j<k} \tilde{b}_j - \sum_{j>k} \tilde{b}_j$, where $\tilde{b}_j = b'_j$ if $j \in I$, and $\tilde{b}_j = b_j$ otherwise. In the voting phase, $\mathcal{A}$ computes $c'_i \leftarrow \mathcal{A}(z, r, \bar{v}_I, b_{\bar{I}})_i$ and a Σ'-proof for $(y_i, c'_i) \in \mathcal{R}_{\mathrm{OR}}$ and sends c'_i and the corresponding Σ'-proof to the message board. Finally, $\mathcal{A}$ outputs $\mathcal{A}(z, r, \bar{v}_i, b_{\bar{I}}, c_{\bar{I}})$ so that the corrupted voter V_i outputs the i-th coordinate of $\mathcal{A}$'s output

while the honest voters output $\mathsf{res} := \sum_{k\in\bar{I}} c_k + \sum_{i\in I} c_i'$ since the honest ones follow the protocol specification. Hence, in this case, the real output of i-th corrupted voter is given by $\mathcal{A}(z, r, \bar{v}_I, b_{\bar{I}}, c_{\bar{I}})_i$. and $\mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{v}) = (\mathsf{res}, \mathcal{A}(z, r, \bar{v}_I, b_{\bar{I}}, c_{\bar{I}}))$ if no abort occurred while if $\mathcal{A}$ aborts at some point in the execution of Π, $\mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{v}) = (\perp, \mathcal{A}(z, r, \bar{v}_I, b_{\bar{I}}, c_{\bar{I}}))$.

Proposition 6.2.5. *Suppose that the RLWE assumption holds. Then protocol* Π *securely computes* f *in the presence of a malicious adversary controlling a fixed number of corrupted voters with abort.*

Proof. Suppose that $\mathcal{A}$ is a real adversary on the execution of Π and let $I \subseteq [m]$ be the index set of corrupted voters controlled by $\mathcal{A}$. We want to show the existence of an ideal adversary $\mathcal{B}$ such that $\mathcal{B}$ makes subroutine calls to $\mathcal{A}$ and at the end of the ideal execution, we have that $\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{v}) \overset{\mathsf{c}}{\equiv} \mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{v})$. Now, we describe how the ideal adversary $\mathcal{B}$ works into several steps:

1. $\mathcal{B}$ sends the votes v_i of each V_i, $i \in I$ to $\mathcal{T}$. Note that the honest voters also send their votes to $\mathcal{T}$.

2. After receiving all the votes v_i from all the voters, $\mathcal{T}$ computes

$$\mathsf{res} := \sum_{i=1}^{m} v_i$$

3. Just before $\mathcal{T}$ sends res to the first honest party, $\mathcal{B}$ sends either an abort message $\perp$ or a continue message to $\mathcal{T}$. If $\mathcal{T}$ receives $\perp$, then $\mathcal{T}$ sends $\perp$ to all the remaining voters. In this case, all the honest voters output $\perp$ while the corrupted voters will output according to $\mathcal{B}(z, r, \mathsf{res}, \perp)$. If $\mathcal{T}$ receives a continue message from $\mathcal{B}$, then $\mathcal{T}$ sends res to all the voters. In this case, the honest voters output res while the corrupted voters output according to $\mathcal{B}(z, r, \mathsf{res})$.

Hence, our task is how $\mathcal{B}$ computes $\mathcal{B}(z, r, \mathsf{res})$ such that $\mathcal{B}(z, r, \mathsf{res})$ is indistinguishable as the output of the real model adversary $\mathcal{A}$, $\mathcal{A}(z, r, b_{\bar{I}}, c_{\bar{I}}, \bar{v}_I)$, where $b_{\bar{I}} := (b_i)_{i\in\bar{I}}$,

$b_i = as_i + (m+1)e_i$, $c_{\bar{I}} := (c_i)_{i \in I}$ in the real execution of Π and also we want res to be indistinguishable to the output of the honest voters in the real execution. So after $\mathcal{B}$ receives res from $\mathcal{T}$, $\mathcal{B}$ computes $\mathsf{res} - \sum_{i \in I} v_i$ which is equal to the sum of the votes of the honest voters $\sum_{i \in \bar{I}} v_i$. Now, for each $i \in \bar{I}$, $\mathcal{B}$ selects $s_i, e_i \xleftarrow{\$} \chi$ and computes $b_i := as_i + (m+1)e_i$ and for $i \in I$, $\mathcal{B}$ runs $\mathcal{A}$ to obtain $b_i \leftarrow \mathcal{A}(z, r)_i$ and the corresponding Σ'-proof for $(a, b_i) \in \mathcal{R}_0$. For each $i \in [m]$, define

$$y_i := \sum_{j<i} b_j - \sum_{j>i} b_j. \tag{6.2.12}$$

Now, $\mathcal{B}$ feeds $\mathcal{A}$ with $b_{\bar{I}}$ and for each $i \in I$, c_i is computed as $c_i \leftarrow \mathcal{A}(z, r, b_{\bar{I}})$ along with a Σ'-proof for $(y_i, c_i) \in \mathcal{R}_{\mathrm{OR}}$. Since $\mathcal{A}$ is required to provide a Σ'-proof for $(y_i, c_i) \in \mathcal{R}_{\mathrm{OR}}$, this ensures that with overwhelming probability that c_i is computed as encryption of either the 0-vote or the 1-vote for $i \in I$. In other words, the Σ'-proof forces $\mathcal{A}$ to behave honestly. Moreover, $\mathcal{A}$ cannot use a different secret s_i other than the one used for choosing b_i, in constructing c_i since as we told in Remark 5.1.3, the voting result will be rejected. Finally, $\mathcal{B}$ must simulate the encryption of the honest voters' votes so that the output of the honest voters in the real and ideal execution are the same with overwhelming probability. For each $i \in \bar{I}$, $\mathcal{B}$ computes $c_i := y_i s_i + (m+1)e'_i + \tilde{v}_i$, where $e'_i \xleftarrow{\$} \chi, \tilde{v}_i \in \{0, 1\}$ such that $\sum_{i \in I} \tilde{v}_i = \sum_{i=1}^{m} v_i - \sum_{i \in \bar{I}} v_i$. Thus, we feed $\mathcal{A}$ with $b_{\bar{I}}$ and $c_{\bar{I}}$, where $c_{\bar{I}} := (c_i)_{i \in I}$. Hence, the output of $\mathcal{B}$ is $\mathcal{A}(z, r, b_{\bar{I}}, c_{\bar{I}})$. The outputs of $\mathcal{A}$ and $\mathcal{B}$ are indistinguishable by the RLWE assumption since we note that $\tilde{c}_i := y_i s_i + (m+1)e'_i$ is indistinguishable from uniform by RLWE and so is c_i since the distribution of the shifted uniform distribution is still uniform. All we need to check is if $\mathsf{res} = \sum_{i=1}^{m} v_i$ and $\sum_{i=1}^{m} c_i \bmod (m+1)$ are the same. Now,

$$\begin{aligned}\sum_{i=1}^{m} c_i \bmod (m+1) &= \sum_{i\in I} c_i + \sum_{i\in \bar{I}} c_i \bmod (m+1)\\ &= \sum_{i\in \bar{I}} v_i + \sum_{i\in I} \tilde{v}_i\\ &= \sum_{i=1}^{m} v_i.\end{aligned}$$

Hence, we have

$$\begin{aligned}\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{v}) &= (\mathsf{res}, \mathcal{B}(z, r, \mathsf{res}))\\ &\stackrel{\mathsf{c}}{\equiv} (\mathsf{res}, \mathcal{A}(z, r, b_{\bar{I}}, c_{\bar{I}}, \bar{v}_I))\\ &= \left(\sum_{i=1}^{m} c_i \bmod (m+1), \mathcal{A}(z, r, b_{\bar{I}}, c_{\bar{I}}, \bar{v}_I)\right)\\ &= \mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{v})\end{aligned}$$

with overwhelming probability. Thus, the two distributions $\{\mathsf{IDEAL}_{f,I,\mathcal{B}(z)}(\bar{v})\}_{\bar{v},z}$ and $\{\mathsf{REAL}_{\Pi,I,\mathcal{A}(z)}(\bar{v})\}_{\bar{v},z}$ are computationally indistinguishable.

The case when $\mathcal{B}$ aborts trivially implies indistinguishability between the real and ideal outputs. Hence, in all cases, we have shown security of Π for computing f against malicious adversaries with abort. ■

To summarize this chapter, we have shown that our voting protocol ensures privacy of the individual votes against both semi-honest and malicious adversaries and that adversaries can only learn the sum of the votes of the honest voters and nothing else.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

To summarize what we have done in this paper, we first presented a simple commitment scheme based on the hardness of the RLWE problem where we commit small elements in the ring R_q. Then, we presented two $\Sigma'-$proofs, the first one allows a prover that he knows a small secret element in R_q and convinces any verifier that the committed element is an RLWE sample on the small secret. This proof does not reveal exactly what the small secret element is. The second $\Sigma'-$protocol is an OR-proof where any prover convinces a verifier that his committed value is either an RLWE sample or an RLWE sample plus one.

The voting protocol we have presented uses the commitment scheme and the $\Sigma'-$proofs mentioned above. The commitment scheme allows any voter to commit to his public keys and encrypted votes while the $\Sigma'-$proofs convinces any verifier that voters follow the protocol specifications honestly. As a result of the specification of our protocol, voters can tally the votes themselves without the need of having a central authority that is usually used in standard voting procedures.

Also, we have provided security proofs so anyone who wishes to apply our protocol in a real setting are guaranteed privacy of the individual votes from the honest voters.

However, there are some issues regarding our protocol. The first one is that it can only support a small number of voters hence it seems our protocol are only useful for small elections. If one insists of using the protocol with a large number of voters, then to guarantee correctness, it will require a choice of larger parameters, hence efficiency is compromised. Another issue that we have is in the use of $\Sigma'-$proofs since this only convinces verifiers that the prover knows secret in a larger relation. Also, the proofs require rejection sampling since if we don't employ this technique, any verifier may learn information about the secret. Thus, this makes our proofs somehow expensive since it might take several takes before a prover successfully generates a proof.

Finally, what we have done is a simple voting protocol that guarantees privacy and also our protocol is post-quantum secure for which we believe is one of the few post-quantum voting protocols in the existing literature as we speak.

7.2 Future Work

One research direction that could improve our protocol is the use of $\Sigma-$protocols instead of $\Sigma'-$protocols so that any verifier is convinced that prover knows the secret in the original relation. Also, another line of direction is constructing proofs without the use of rejection sampling to gain more efficiency.

Of course, constructing new post-quantum voting protocols would be an important research direction especially ones that would effectively reduce the size of the parameters and protocols that are more scalable, that is, that would allow a large number of voters without compromising efficiency, privacy and correctness.

Bibliography

Ajtai, Miklós (1996). "Generating hard instances of lattice problems". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 99–108.

Applebaum, Benny, David Cash, Chris Peikert, and Amit Sahai (2009). "Fast cryptographic primitives and circular-secure encryption based on hard learning problems". In: *Advances in Cryptology - CRYPTO 2009*, pp. 595–618.

Aranha, Diego F., Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge (2021). "Lattice-Based Proof of Shuffle and Applications to Electronic Voting". In: *Topics in Cryptology – CT-RSA 2021*. Ed. by Kenneth G. Paterson. Cham: Springer International Publishing, pp. 227–251. ISBN: 978-3-030-75539-3.

Arora, Sanjeev and Rong Ge (2011). "New algorithms for learning in presence of errors". In: *International Colloquium on Automata, Languages, and Programming*. Springer, pp. 403–415.

Banaszczyk, Wojciech (1993). "New bounds in some transference theorems in the geometry of numbers". In: *Mathematische Annalen* 296.1, pp. 625–635.

Benhamouda, Fabrice, Jan Camenisch, Stephan Krenn, Vadim Lyubashevksy, and Gregory Neven (2014). "Better Zero-Knowledge Proofs for Lattice Encryption and Their Applications to Group Signatures". In: *ASIACRYPT 2014*.

Benhamouda, Fabrice, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak (2015). "Efficient zero-knowledge proofs for commitments from learning with errors over rings". In: *European symposium on research in computer security*, pp. 305–325.

Blum, Avrim, Adam Kalai, and Hal Wasserman (2003). "Noise-tolerant learning, the parity problem, and the statistical query model". In: *Journal of the ACM (JACM)* 50.4, pp. 506–519.

Boneh, Dan (1998). "The decision Diffie-Hellman problem". In: *International Algorithmic Number Theory Symposium*. Springer, pp. 48–63.

Chaum, David (1988). "The dining cryptographers problem: Unconditional sender and recipient untraceability". In: *Journal of cryptology* 1.1, pp. 65–75.

Cooley, James W and John W Tukey (1965). "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of computation* 19.90, pp. 297–301.

Damgård, Ivan (2010). "On Σ-protocols". In: *Lecture Notes, University of Aarhus, Department for Computer Science*, p. 84.

Ding, Jintai (2010). "Solving LWE problem with bounded errors in polynomial time". In: *Cryptology ePrint Archive*.

Ding, Jintai, Doug Emery, Johannes Müller, Peter YA Ryan, and Vonn Kee Wong (2020). "Post-Quantum Anonymous Veto Networks". In: *Cryptology ePrint Archive*.

Ding, Jintai, Xiang Xie, and Xiaodong Lin (2012). "A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem". In: *Cryptology ePrint Archive*.

Fiat, Amos and Adi Shamir (1986). "How to prove yourself: Practical solutions to identification and signature problems". In: *Conference on the theory and application of cryptographic techniques*. Springer, pp. 186–194.

Gentry, Craig, Chris Peikert, and Vinod Vaikuntanathan (2008). "Trapdoors for hard lattices and new cryptographic constructions". In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 197–206.

Goldreich, Oded (2004). *Foundations of Cryptography, Volume II Basic Applications*. Cambridge University Press. ISBN: 0521830842.

Groth, Jens (2004). "Efficient maximal privacy in boardroom voting and anonymous broadcast". In: *International Conference on Financial Cryptography*, pp. 90–104.

Hao, Feng, Peter YA Ryan, and Piotr Zieliński (2010). "Anonymous voting by two-round public discussion". In: *IET Information Security* 4.2, pp. 62–67.

Hao, Feng and Piotr Zieliński (2006). "A 2-round anonymous veto protocol". In: *International Workshop on Security Protocols*. Springer, pp. 202–211.

Langlois, Adeline and Damien Stehlé (2012). "Hardness of decision (R) LWE for any modulus". In: *Cryptology ePrint Archive*.

Lenstra, A. K., H. W. Lenstra, and L. Lovasz (1982). "Factoring polynomials with rational coefficients". In: *MATH. ANN* 261, pp. 515–534.

Lyubashevsky, Vadim (2012). "Lattice signatures without trapdoors". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, pp. 738–755.

Lyubashevsky, Vadim, Chris Peikert, and Oded Regev (2010). "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–23. ISBN: 978-3-642-13190-5.

— (2013). "A toolkit for ring-LWE cryptography". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer, pp. 35–54.

Lyubashevsky, Vadim and Gregor Seiler (2018). "Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 204–224.

Micciancio, Daniele (2021). *CSE206A: Lattices Algorithms and Applications*. `https://cseweb.ucsd.edu/classes/fa21/cse206A-a/`.

Micciancio, Daniele and Oded Regev (2007). "Worst-case to average-case reductions based on Gaussian measures". In: *SIAM Journal on Computing* 37.1, pp. 267–302.

Minkowski, Hermann (1910). *Geometrie der zahlen*. Vol. 40. Teubner.

Peikert, Chris (2010). "An efficient and parallel Gaussian sampler for lattices". In: *Annual Cryptology Conference*. Springer, pp. 80–97.

Pino, Rafaël Del, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler (2017). "Practical quantum-safe voting from lattices". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1565–1581.

Pointcheval, David and Jacques Stern (1996). "Security proofs for signature schemes". In: *International conference on the theory and applications of cryptographic techniques*. Springer, pp. 387–398.

Regev, Oded (2009). "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *STOC*, pp. 84–93.

Schnorr, Claus-Peter (1989). "Efficient identification and signatures for smart cards". In: *Conference on the Theory and Application of Cryptology*. Springer, pp. 239–252.

Shor, Peter W. (Jan. 1999). "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Review* 41.2, pp. 303–332. DOI: 10.1137/S0036144598347011.

Stehlé, Damien, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa (2009). "Efficient Public Key Encryption Based on Ideal Lattices". In: *ASIACRYPT 2009*.

Xie, Xiang, Rui Xue, and Minqian Wang (2013). "Zero knowledge proofs from Ring-LWE". In: *International Conference on Cryptology and Network Security*. Springer, pp. 57–73.