UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Valeh Farzaliyev

Towards Practical Post-Quantum Voting Protocol: Shorter Exact Lattice-Based Proof of a Shuffle

Master's Thesis (30 ECTS)

Supervisor: Dominique Peer Ghislain Dr Unruh
Supervisor: Jan Willemson, PhD

Tartu 2020

Towards Practical Post-Quantum Voting Protocol: Shorter Exact Lattice-Based Proof of a Shuffle

Abstract:
Electronic voting solutions are built on complex cryptographic tools to guarantee security and fairness. Currently, those tools are based on hardness assumptions of discrete logarithm, factorization and other classical problems. While they are hard to break in classical computers, there are efficient quantum algorithms to solve using quantum computers of the near future. Thus, there is a need to develop voting protocols that are resistant to quantum attacks.

Verifiable shuffling based voting systems are a popular use-case of mix-networks first proposed by Chaum four decades ago [Cha81] as a general tool for building anonymous communication systems. A decade later the quantum threat was known and since then only a few studies searched for post-quantum secure mix-nets. Recently, Costa, Martinez and Morillo introduced new arguments of shuffle for RLWE ciphertexts and how to prove the correctness of the shuffling without leaking sensitive info [CMM17]. In this thesis, we provide exact, shorter proof of Costa *et al.*'s lattice-based shuffling arguments. As a result, we obtain a practical non-interactive zero-knowledge proof having a runtime of 1 second per voter.

Keywords:
Post-quantum cryptography, lattice based cryptography, mix net, secure voting

CERCS: P170 - Computer science, numerical analysis, systems, control

Postkvant-hääletamisprotokollide arendamine: lühem täpne võrepõhine segamistõestus

Lühikokkuvõte:
Elektroonilise hääletamise protokollid kasutavad keerukaid krüptograafilisi meetodeid, et tagada valimiste turvalisus ja ausus. Praegu kasutusel olevad meetodid tuginevad klassikalistele ülesannetele nagu diskreetne logaritm ja suurte kordarvude tegurdamine. Need ülesanded on üldjuhul rasked klassikaliste arvutite jaoks, kuid neid saab efektiivselt lahendada piisavalt võimsate kvantarvutite abil. Seega on oluline arendada välja hääletamisprotokollid, mis peaksid vastu kvantarvuti abil teostatavatele rünnetele.

Verifitseeritavad segamispõhised hääletamissüsteemid kasutavad miksimisvõrke, mille pakkus juba 40 aastat tagasi esimesena välja Chaum [Cha81] kui vahendi anonüümse kommunikatsioonivõimaluse loomiseks. Kümmekond aastat hiljem ilmnes kvantarvutite oht, kuid postkvant-miskimisvõrke on sellest ajast uuritud väga vähe. 2017. aastal pakkusid Costa, Martinez ja Morillo välja RLWE krüptogrammide miksimise põhimõtte ning näitasid, kuidas tõestada miskimise korrektsust ilma sisendite privaatsust rikkumata [CMM17]. Selles väitekirjas esitame täpse ja lühema tõestuse Costa jt võrepõhisele konstruktsioonile. Tulemusena saame praktilise mitteinteraktiivse nullteadmustõestuse ajakuluga umbes 1 sekund miksitava hääle kohta.

Võtmesõnad:

CERCS:P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Contents

1 Introduction

Adopted in 1948 by United Nations General Assembly, Article 21 point 3 of The Universal Declaration of Human Rights[1] states "The will of the people shall be the basis of the authority of government; this will shall be expressed in periodic and genuine elections which shall be by universal and equal suffrage and shall be held by secret vote or by equivalent free voting procedures". Throughout history, the means of voting have changed and are mostly affected by technological advances. In the information era, there is a digitalization trend in which services are transformed into e-services. As a matter of fact, some countries have been offering online elections at either in municipal or national level for several years. Companies like Helios and Scytl offer internet-based voting solutions for any institutions looking for internal elections.

Online voting has been a subject of numerous studies as well. A very detailed comparison of paper vs online voting is given in a paper by Jan Willemson [Wil17]. A more recent study found that by use of online voting the amount of errors made by voters when casting ballots is reduced [Ger20]. Furthermore, online voting has been formalized and several requirements are expected to be met to achieve transparency, anonymity, and verifiability. Generally, such voting protocols employ cryptographic tools and methods.

Rapid development in quantum technologies poses a serious threat to modern cryptography because of Shor's algorithm [Sho99] which can solve previously believed to be hard discrete log problems. Voting protocols are not exceptions, as well. ElGamal and Paillier's encryption scheme is the usual choice for voting protocols due to their homomorphic construction, but they are also vulnerable to quantum attacks. Experts predict that such threatening quantum computers will be a reality within a few decades[2]. Thus, there is an urgent need to design quantum-resistant practical online voting protocols.

Recently, a number of papers have been published in this area. In general, those proposed protocols make use of lattice-based post-quantum encryption schemes, again due to their homomorphic nature. Although it is theoretically possible to construct a Fully Homomorphic Encryption scheme, problems occur when it comes to efficiently implement it in practice. The goal of this work is to devise a verifiable and practical shuffling protocol using post-quantum cryptography tools based on recent academic work.

A general overview of cryptographic voting, necessary mathematical background, and quick introduction to the cryptographic primitives are given in Section 2. An adapted verifiable, fully post-quantum shuffling algorithm alongside its zero-knowledge proof is presented in Section 3. Section 4 contains the implementation overview and

[1] https://www.un.org/en/universal-declaration-human-rights/

[2] Recent study [GE19] shows that around 20 million noisy qubits are required to factor 2048 bit RSA integers in 8 hours. M.Mosca estimates that there is a 1/6 chance of building fault-tolerant quantum computer able to crack RSA-2048 by 2027. According to him, this likelihood will be a half before 2035 https://www.etsi.org/news-events/events/1173-etsi-iqc-quantum-safe-workshop-2017

results. Finally, in Section 5, we foresee a complete online voting protocol with a theoretical suggestion on zero-knowledge prover of decryption oracle. In the end, Section 6 summarizes the whole work and possible future extensions to this work.

2 Preliminaries

This section contains necessary background information on design principles of cryptographic voting, underlying mathematical notions and definitions of cryptographic primitives that are used throughout this work.

2.1 Notation

Before formulating mathematical model of cryptographic voting and all the necessary tools for construction, it is helpful to fix the notation. In this work, general sets are denoted by italic roman capital letters, e.g. S. $\mathbb{Z}$ and $\mathbb{R}$ denote set of integers and real numbers respectively. Let $\mathbb{Z}_n^\times$ denote the group of invertible elements modulo n. $\lfloor x \rceil$ represents the closest integer to x in $\mathbb{Z}_q$. An element of the set is denoted by lower-case letters, $s \in S$ and when it is sampled uniformly, we write $s \xleftarrow{\$} S$. We also use the same notation $s \xleftarrow{\$} D$ when an element is sampled according to a probability distribution D. Polynomials will be typed in bold-face to be differentiated from scalars, e.g.,$\boldsymbol{f} = \sum_i f_i X_i$. Vectors are represented by lower-case roman letters arrow atop, $\vec{\boldsymbol{u}}$(or $\vec{v}$) and their i-th components is shown with a subscript: $\boldsymbol{u}_i$ (or v_i). It should not be confused with $\vec{\boldsymbol{u}}_i$ which is a one of many vectors, possibly sharing similar properties. Preferably, all vectors are column vectors by default and concatenation of two vectors is denoted like $\vec{u}\|\vec{v}$ which is still a column vector. An inner product is denoted $\langle \cdot, \cdot \rangle$ and computed as dot product between vectors. Finally, upper-case roman letters will denote matrices, $\boldsymbol{M}$ (or A). An element y can also be the output of deterministic algorithm A on input x. In this case, we write $y \leftarrow A(x)$ and $y \xleftarrow{\$} A(x)$ when A is a probabilistic algorithm.

2.2 Lattices

2.2.1 General Introduction

In mathematics, there are two lattices which should not be confused. We will be using lattices from group theory.

Definition 2.1 (Lattice)**.** *A typical lattice $\mathcal{L}$ in $\mathbb{R}^n$ is a discrete additive subgroup of $\mathbb{R}^n$ spanned by m linearly independent vectors with coefficients called basis vectors.*

$$\mathcal{L} = \{\sum_i^m x_i \vec{b_i} \mid x_i \in \mathbb{Z}, \vec{b_i} \in \mathbb{R}^n\}$$

A matrix B whose columns are $\vec{b}_1, \ldots, \vec{b}_m$ are called basis matrix. It is equivalent to say that they form a basis of the lattice $\mathcal{L}(B)$.

The basis of a lattice is not necessarily unique. One can multiply B with a unimodular matrix U to get another basis. Although this is a simple fact, it has important consequences. Because a basis is enough to represent lattice structure, complexity of the lattice related problems differ regarding the properties of the basis. A good basis which consists of short highly orthogonal vectors is useful to solve certain problems, while a bad basis with low orthogonality is not. Shortest Vector Problem (SVP) is a perfect example for that.

Surprisingly, transforming a bad basis into a good one is also a difficult problem. In a two dimensional lattice this problem can be solved by simply using Gaussian elimination method. Lenstra-Lenstra-Lovasz (LLL) algorithm [LLL82] generalizes this method into any arbitrary dimensional lattice where Gaussian elimination is performed on the two elements of the basis at a time. Nonetheless, the length of the vectors in resulting basis can be exponentially far from the optimal value.

Schnorr's Block Korkine-Zolotarev (BKZ) reduction [SE94] algorithm is widely adopted algorithm to measure hardness of finding a good basis. Here, instead of working on two vectors at single step, the algorithm takes a block of β vectors. Internally, it searches for the shortest vector in β-dimensional sublattice which is still a hard problem.

Definition 2.2. *The i-th minimum of the lattice $\mathcal{L}$ is the radius of the smallest closed ball centered at the origin that contains at least i linearly independent points in $\mathcal{L}$ and denoted $\lambda_i(\mathcal{L})$.*

Clearly, the shortest vector in lattice has length $\lambda_1(\mathcal{L})$.

Definition 2.3 (Approximate Shortest Vector Problem (SVP$_\gamma$)**.** *Given a basis B of a lattice $\mathcal{L}(B)$, find a non-zero vector $\vec{v}$ such that $\|\vec{v}\|_2 \leq \gamma \cdot \lambda_1(\mathcal{L}(B))$.*

In 1998, Ajtai proved in his seminal work [Ajt98] that γ-SVP is an NP-hard problem in its exact version ($\gamma = 1$) and polynomial approximations ($\gamma(n)$ polynomial in the dimension of the lattice). When γ is exponentially large LLL algorithm finds the solution in polynomial time. For smaller values of γ, BKZ reduction with large block size is commonly used. The overall time complexity depends on the block size, therefore it is a direct indicator of the problem hardness.

Definition 2.4 (Approximate Closest Vector Problem (CVP$_\gamma$)**.** *Given a basis B and a vector $\vec{u}$ not necessarily in $\mathcal{L}(B)$, find the closest vector $\vec{v}$ in $\mathcal{L}(B)$ at distance at most γ, i.e $\|\vec{u} - \vec{v}\| \leq \gamma \cdot \min_{\vec{w} \in \mathcal{L}} \|\vec{u} - \vec{w}\|$.*

Definition 2.5 (Short Integer Solution)**.** *Given a matrix $A \in \mathbb{Z}^{m \times n}$ find shortest non-zero vector $\vec{x}$ such that $A\vec{x} = 0$*

All three problems are related to each other. Goldreich *et al.*'s work [GMSS99] shows that CVP$_\gamma$ and SVP$_\gamma$ have the same hardness. It easy to see the relation between SVP and SIS problems using q-ary lattices.

Definition 2.6 (q-ary lattices)**.** . *A lattice $\mathcal{L}$ is said to be q-ary if for an integer q, $q\mathbb{Z}^n \subset \mathcal{L} \subset \mathbb{Z}^n$.*

Usually, q-ary lattices are represented in two forms. First, for a given matrix $A \in \mathbb{Z}^{m\times n}$

$$\Lambda_q(A) = \{\vec{x} \in \mathbb{Z}^n \mid A\vec{z} = \vec{x} \bmod q : \vec{z} \in \mathbb{Z}^n\}.$$

Second way, called orthogonal Λ_q has form

$$\Lambda_q^{\perp}(A) = \{\vec{x} \in \mathbb{Z}^n \mid A^T\vec{x} = 0 \bmod q\}.$$

Observe that SIS solution to A is also the SVP solution in $\Lambda_q^{\perp}(A)$.

Using lattice problems, Oded Regev introduced provable post-quantum secure Learning With Errors cryptosystem [Reg05].

Definition 2.7 (Learning With Errors (LWE))**.** *Let n and q be integers, χ a discrete noise distribution in $\mathbb{Z}$, and $\vec{s}$ a secret vector in $\mathbb{Z}_q^n$. Sample m uniformly random public vectors $\vec{a}_i$ from $\mathbb{Z}_q^n$ and error term $e_i \xleftarrow{\$} \chi$ and calculate $\vec{b} = \langle \vec{a}_i, \vec{s}\rangle + e_i$. Denote this distribution $(\vec{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ as $\mathsf{LWE}_{s,\chi}$.*

The decisional LWE problem asks to distinguish $\mathsf{LWE}_{s,\chi}$ from uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Given access to m sample of $(\vec{a}_i, b_i)$, search LWE problem asks to recover secret vector $\vec{s}$.

Regev's main result is that under certain conditions (χ being discrete Gaussian distribution over integers, $m = \text{poly}(n)$) LWE problem is as hard as approximate SVP problem even for quantum computers.

Ajtai's SIS problem and Regev's LWE problem are foundations of modern lattice based cryptography.

2.2.2 Ideal lattices

In plain LWE, public vectors $\vec{a}_i$ are random and not related to each other. We can think of them being rows of a matrix $A \in \mathbb{Z}_q^{m\times n}$. There are mn independent components. Adding a particular algebraic structure to the matrix can reduce the communication cost of the matrix.

In abstract algebra, (left or right) ideal I of a ring R is a set of ring elements such that for $r \in R$ and $x, y \in I$, then $x + y \in I$ and $rx \in I$. If all elements of the ideal can be computed from an element a, then this ideal is called principal ideal and denoted as $I = (a)$.

For a rational prime q, let $\mathbb{Z}_q$ be the ring of integers modulo q, with its elements considered in the interval $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$ such that for any set element r there exists a unique positive integer less than q congruent to $r \bmod q$. Letting d be a power of two, we

consider the rings $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$. Elements of these rings are written in bold lower-case letters (e.g. $\boldsymbol{p}$), and vectors with elements from these rings will naturally be denoted as $\vec{\boldsymbol{b}}$. Matrices over $\mathcal{R}$ or $\mathcal{R}_q$ are bold upper-case letters, e.g. $\boldsymbol{B}$. Any element $\boldsymbol{a} \in \mathcal{R}_q$ can be written as column vector $\mathcal{V}_{\boldsymbol{a}} = |a_0, a_1, \ldots, a_{d-1}|^T$ where $\boldsymbol{a} = \sum_{i=0}^{d-1} a_i X^i$ and $a_i \in \mathbb{Z}_q$.

Choose a random polynomial $\boldsymbol{a}$ and construct set of all polynomials which are obtained multiplying $\boldsymbol{a}$ with other polynomials in $\mathcal{R}_q$. This set forms a principal ideal. It is not known whether additional structure of ideal lattices cause weaknesses. Respective problems are called Ideal-SVP$_\gamma$, Ideal-CVP$_\gamma$, and Ideal-SIS.

Especially for ring $\mathcal{R}_q$, the same element can be represented as a matrix in $\mathbb{Z}_q$ when it is a multiplicand:

$$\mathcal{M}_{\boldsymbol{a}} = \begin{vmatrix} a_0 & -a_{d-1} & -a_{d-2} & \cdots & -a_1 \\ a_1 & a_0 & -a_{d-1} & \cdots & -a_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{d-1} & a_{d-2} & a_{d-3} & \cdots & a_0 \end{vmatrix} .$$

Observe that principal ideal generated by $\boldsymbol{a}$ is also a lattice span by $\mathcal{M}_{\boldsymbol{a}}$. It is also called ideal lattice.

Moreover, l_2 and l_∞ norms are defined as usual:

$$\|\boldsymbol{a}\|_\infty = \max_i |a_i| \text{ and } \|\boldsymbol{a}\|_2 = \sqrt{|a_0|^2 + \ldots + |a_{d-1}|^2} .$$

These norms can naturally be extended to vectors over $\mathcal{R}_q$. For $\vec{\boldsymbol{w}} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k\} \in \mathcal{R}_q^k$, we have

$$\|\vec{\boldsymbol{w}}\|_\infty = \max_i \|\boldsymbol{w}_i\| \text{ and } \|\vec{\boldsymbol{w}}\|_2 = \sqrt{\|\boldsymbol{w}_1\|_2^2 + \ldots + \|\boldsymbol{w}_k\|_2^2} .$$

Polynomials and vectors with short norm will simply be referred to as short.

2.3 Splitting Rings, Galois authomorphisms

When $q - 1 \equiv 2l \bmod 4l$, the $2l$-th primitive root of unity is contained in $\mathbb{Z}_q$ but no higher roots. Then $X^d + 1$ splits into l irreducible polyonmials of degree d/l, i.e

$$X^d + 1 = \prod_{i \in \mathbb{Z}_{2l}^\times} (X^{d/l} - \zeta^i) \bmod q = \prod_{i=1}^{l} \boldsymbol{\varphi_i} \bmod q$$

where ζ is primitive $2l$-th root of unity in $\mathbb{Z}_q$ and $\boldsymbol{\varphi_i} = X - \zeta^{2i-1}$. Thus, the ring $\mathcal{R}_q$ is isomorphic to the product of the corresponding residue fields:

$$\mathcal{R}_q \cong \mathbb{Z}_q[X]/(\boldsymbol{\varphi}_1) \times \cdots \times \mathbb{Z}_q[X]/(\boldsymbol{\varphi}_l) .$$

We call a ring fully splitting when $l = d$.

This ring has a group of automortphisms $\mathsf{Aut}(\mathcal{R}_q)$ that is isomorpic to $\mathbb{Z}_{2l}^{\times}$,

$$i \mapsto \sigma_i : \mathbb{Z}_{2l}^{\times} \to \mathcal{R}_q,$$

where $\sigma_i(X) = X^i$ are Galois automorphisms. The group $\mathsf{Aut}(\mathcal{R}_q)$ acts transitively on prime ideals $\boldsymbol{\varphi_i}$. In other words, there exists an automorphism such that $\sigma(\boldsymbol{\varphi_i}) = \boldsymbol{\varphi_j}$ for any $i, j \in 1, \ldots, l$.

2.4 Challenge space

Elements of the ring $\mathcal{R}_q$ are not always invertible. In fact, Lyubashevsky *et al.* proved a relation between the probability of invertiblity in this ring and the number of residue fields it splits into [LS18, Corollary 1.2]. Their claim is that generally short non-zero polynomials are invertible. In lattice based zero knowledge proofs, the verifier often samples from a challenge set such that the difference of any two elements in that set is invertible. However, constructing such a set and uniformly sampling from it is not a trivial task.

Therefore, Lyubashevsky *et al.* proposed another method where they relaxed the invertiblity requirement. They defined the challenge space as the set of ternary polynomials $\mathcal{C} = \{-1, 0, 1\}^d \subset \mathcal{R}$. Coefficients of a challenge $\boldsymbol{c} \in \mathcal{C}$ are identically and independently distributed where 0 has probability $1/2$ and ± 1 both have probability $1/4$. In [ALS20, Lemma 3.3], it is shown that if $\boldsymbol{c} \leftarrow \mathcal{C}$, the distribution of coefficients of $\boldsymbol{c}$ $\mod (X^{d/l} - \zeta)$ is almost uniform and the maximum probability of coefficients over $\mathbb{Z}_q$ is bounded. Denote this bound with p. For example, authors in last reference estimated $p = 2^{-31.44}$ for $l = d = 128$, $q \approx 2^{32}$. An element $\boldsymbol{c}$ in splitting ring $\mathcal{R}_q$ is non-invertible when $\boldsymbol{c} \mod \boldsymbol{\varphi}_i = 0$ for any $i = 1, \ldots, l$. Then the difference of any two challenges $\bar{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c'}$ is non-invertible with probability at most $p^{d/l}$.

2.5 Error distribution, Discrete Gaussians and Rejection Sampling

Rejection sampling. It is a common practice to hide secret commitment randomness $\vec{\boldsymbol{r}}$ in another vector $\vec{\boldsymbol{z}}$ without leaking any information about $\vec{\boldsymbol{r}}$. In other words, these vectors should be statistically close. For this purpose, in the protocol the prover samples "masking" vector $\vec{\boldsymbol{y}}$ using discrete Gaussian distribution. Upon receiving the challenge $\boldsymbol{c} \xleftarrow{\$} \mathcal{C}$ by the verifier, the prover responds with $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\boldsymbol{r}}$. Rejection sampling lemma [Lyu12] below states that $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{z}}$ are within negligible statistical distance if masking vectors are sampled from discrete Gaussian distribution with certain standard deviation.

Definition 2.8. *The discrete Gaussian distribution on $\mathcal{R}^l$ centered around $\vec{\boldsymbol{v}} \in \mathcal{R}^l$ with*

standard deviation $\mathtt{s} > 0$ *is given by*

$$D^{ld}_{\vec{\boldsymbol{v}},\mathtt{s}}(\vec{\boldsymbol{z}}) = \frac{e^{-\|\vec{\boldsymbol{z}}-\vec{\boldsymbol{v}}\|_2^2/2\mathtt{s}^2}}{\sum_{\vec{\boldsymbol{z}}'\in\mathcal{R}^l} e^{-\|\vec{\boldsymbol{z}}'\|_2^2/2\mathtt{s}^2}} \ .$$

When it is centered around $\vec{\boldsymbol{0}} \in \mathcal{R}^l$*, we write* $D^{ld}_{\mathtt{s}}(\vec{\boldsymbol{z}}) = D^{ld}_{\vec{\boldsymbol{0}},\mathtt{s}}(\vec{\boldsymbol{z}})$.

Lemma 1 (Rejection Sampling). *Let* $V \subseteq \mathcal{R}^l$ *be a set of polynomials with norm at most* T *and* $\rho : V \to [0,1]$ *be a probability distribution. Also, write* $\mathtt{s} = 2T$ *and* $M = exp(6 + 1/16)$*. Now, sample* $\vec{\boldsymbol{v}} \xleftarrow{\$} \rho$ *and* $\vec{\boldsymbol{y}} \xleftarrow{\$} D^{ld}_{\boldsymbol{s}}$*, set* $\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \vec{\boldsymbol{v}}$*, and run* $b \leftarrow Rej(\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \mathtt{s})$ *(see Figure 1). Then the probability that* $b = 0$ *is at least* $(1-2^{-100})/M$ *and the distribution of* $(\vec{\boldsymbol{v}}, \vec{\boldsymbol{z}})$*, conditioned on* $b = 0$*, is within statistical distance of* $2^{-100}/M$ *of the product distribution* $\rho \times D^{ld}_{\mathtt{s}}$.

Rej$(\vec{\boldsymbol{z}}, \vec{\boldsymbol{v}}, \mathtt{s})$

$u \xleftarrow{\$} [0,1)$
If $u > \frac{1}{M} \cdot exp\big(\frac{-2\langle\vec{\boldsymbol{z}},\vec{\boldsymbol{v}}\rangle + \|\vec{\boldsymbol{v}}\|^2}{2\mathtt{s}^2}\big)$
return 0
Else return 1

Figure 1. Rejection sampling algorithm [Lyu12]

The following tail-bound lemma which follows from [Woj93, Lemma 1.5(i)] is useful to define the MSIS bound.

Lemma 2 (tail-bound). *Let* $\vec{\boldsymbol{z}} \leftarrow D^{ld}_{\mathtt{s}}$*. Then*

$$Pr\big[\|\vec{\boldsymbol{z}}\|_2 < \mathtt{s}\sqrt{2ld}\big] > 1 - 2^{-ld/8}.$$

2.6 Generalized Shwartz-Zippel lemma

The generalized Shwartz-Zipple lemma is stated in Lemma 3

Lemma 3. *Let* $p \in R[x_1, x_2, \ldots, x_n]$ *be a non-zero polynomial of total degree* $d \geq 0$ *over a commutative ring* R*. Let* S *be a finite subset of* R *such that none of the differences between two elements of* S *is a divisor of 0 and let* $r_1, r_2, ..., r_n$ *be selected at random independently and uniformly from* S*. Then:* $Pr[p(r_1, r_2, \ldots, r_n) = 0] \leq d/|S|$.

Proof. c.f [CMM19, Appendix A] □

In general, it is not trivial to construct set S. A polynomial in $\mathcal{R}_q$ is a zero divisor when at least one of its NTT coefficient is zero. Then, difference of two elements is not a divisor of zero when they do not have common NTT coefficient. There can be at most q pairwise different modulo degree 1 prime ideals for fully splitting rings. This strictly reduces soundness. However, for partially splitting rings this number increases to $q^{d/l}$. For any random polynomial, one can find $q^{d/l} - 1$ other polynomials which doesn't have common NTT coefficient and construct set S. Without loss of generality, we sample polynomials from $\mathcal{C}$ instead of S where we apply Lemma 3.

2.7 Cryptography overview

During the formalization of the online voting protocol, we will use several cryptographic primitives. Here, we give a brief introduction to encryption schemes, commitments, and Zero-Knowledge proofs.

Encryption & Decryption A general way to hide secret information from unwanted parties is transforming it into a specific form in which only the trusted party can get the original message. Thus, we need two parties equipped with specific functions, data, and keys who want to secretly communicate with each other. An encryption scheme is a set of **Gen, Enc, Dec** functions.

- **Gen**: generates secret and public key (sk, pk) for a given security parameter
- **Enc**: given a plaintext m and a public key pk, outputs ciphertext c. $c \leftarrow Enc(m, pk)$
- **Dec**: recovers plaintext m from ciphertext c using secret key sk. $m \leftarrow Dec(c, sk)$

If $sk = pk$ meaning that the same key is used for encryption and decryption, the encryption scheme is called symmetric, otherwise asymmetric. For an encryption scheme, the decryption failure is the probability that $m \neq Dec(Enc(m, pk), sk)$.

An encryption scheme is said to be Indistinguishable Chosen Plaintext Attack, IND-CPA if an adversary cannot distinguish ciphertexts of two chosen plaintexts.

Commitment scheme Commitment schemes are used to prove that a particular message is not changed during computation. It is very similar to encryption schemes, but differ where they are used. A commitment scheme consists of three algorithms.

- **Gen**: generates public key pk for a given security parameter.

- **Com**: given an input message m and public key pk, produces a commitment c and opening d. $(c, d) \leftarrow Com(m, pk)$.

- **Ver**: using the opening d, accepts if c is a commitment to m using public key pk or rejects.

A commitment scheme is correct if a verifier accepts honest commitments to a valid message. It is binding if it can be correctly opened to one message. Finally, hiding property claims verifier cannot recover committed message.

A commitment scheme is perfectly binding/hiding if no computationally unbounded adversary can break it. Similarly, if the advantage of PPT adversaries is negligible, then a commitment scheme is said to be computationally binding/hiding. As a fact, it is not possible to have a perfectly binding and perfectly hiding encryption scheme.

Zero-Knowledge proofs Zero-knowledge proofs are interactive protocols between two parties, a prover and a verifier when the prover wants to assure the verifier that he knows some secret value without leaking information about the value.

Definition 2.9 (Σ-protocol)**.** *An interactive three round zero-knowledge proofs between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which given an x, $\mathcal{P}$ tries to convince $\mathcal{V}$ that it knows a witness w such that $(x, w) \in R$ is called Σ-protocol.*

In a Σ-protocol, $\mathcal{P}$ sends a message y to $\mathcal{V}$ and waits for the answer c, also called challenge. Then $\mathcal{P}$ replies z. Looking at the protocol transcript (x, y, c, z), the verifier accepts or rejects the proof.

The zero-knowledge protocol is complete when an honest verifier always accepts the transcript generated with honest prover knowing valid witness w. If no cheating prover can convince an honest verifier without a witness, then the protocol is sound. Moreover, if the verifier cannot learn about the witness, it is zero-knowledge. For Σ-protocols, soundness also means that there is an extractor which can extract witness from two accepting challenges (x, y, c, z) and (x, y, c', z'). Finally, there should be a polynomial-time simulator that can simulate the behavior of an honest prover and outputs an accepting transcript.

There can be more than three rounds in zero-knowledge protocols.

Σ-protocol can be made non-interactive using Fiat-Shamir transformation. Basically, instead of a verifier choose a random challenge c, the prover uses a public one-way hash function and sets c equal to the hash of public data in the first round. Then, the protocol transcript is $(x, y, H(x||y), z)$. The non-interactive verifier accepts if the transcript is acceptable for the interactive verifier.

A zero-knowledge protocol may have statistical/perfect or computational security properties. Perfect and computational terms are defined as commitment schemes. Here,

the statistical keyword is added to allow very negligible statistical error independent of the adversary's computational capabilities.

It is desired to have perfect security properties, but having statistical or computational is also acceptable.

2.8 Lattice based Cryptography

2.8.1 Ring-LWE Encryption, Module SIS/LWE

In our constructions, we will rely on hardness of Ring-LWE (RLWE) which is proven to be as secure as Ideal-SVP$_\gamma$ [LPR13] and Module-LWE (MLWE)/ Module-SIS (MSIS) [DS15, PR05] problems.

Definition 2.10 ($RLWE_\chi$)**.** *In the decisional Ring-LWE problem with an error distribution χ over $\mathcal{R}$, the PPT adversary $\mathcal{A}$ is asked to distinguish $(\boldsymbol{a}, \boldsymbol{b}) \xleftarrow{\$} \mathcal{R}_q \times \mathcal{R}_q$ from $(\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e})$ for $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q$ and $\boldsymbol{s}, \boldsymbol{e} \leftarrow \chi$.*

The corresponding *search*-RLWE problem asks to find $\boldsymbol{s}$ from several $(\boldsymbol{a}, \boldsymbol{b})$ RLWE samples.

We implement the encryption scheme described in [LPR13]. Let χ_1 be error distribution over $\mathcal{R}$ where each coefficient is sampled from $\{-1, 0, 1\}$.

- *Gen*: Given $\boldsymbol{a}$ uniformly sampled in $\mathcal{R}_q$, a secret $\boldsymbol{s} \leftarrow \chi_1$ and an error $\boldsymbol{e} \leftarrow \chi_1$, the public key is defined as $pk = (\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}, \boldsymbol{a} \cdot \boldsymbol{s} + \boldsymbol{e})$ and private key as $\boldsymbol{s}$.

- *Enc*: To encrypt a message $\boldsymbol{z} \in \mathcal{R}_2$, sample new randomness $\boldsymbol{r}$ and error terms $\boldsymbol{e}_1, \boldsymbol{e}_2$ from error distribution χ_1. Then the ciphertext is a pair of polynomials $(\boldsymbol{u}, \boldsymbol{v})$ such that

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{a} \cdot \boldsymbol{r} + \boldsymbol{e}_1 \ , \\ \boldsymbol{v} &= \boldsymbol{b} \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \left\lfloor \frac{q}{2} \right\rceil \boldsymbol{z} \ . \end{aligned}$$

- *Dec*: Given ciphertext $(\boldsymbol{u}, \boldsymbol{v})$, compute

$$\boldsymbol{v} - \boldsymbol{u} \cdot \boldsymbol{s} = (\boldsymbol{r} \cdot \boldsymbol{e} - \boldsymbol{e}_1 \cdot \boldsymbol{s} + \boldsymbol{e}_2) + \left\lfloor \frac{q}{2} \right\rceil \boldsymbol{z} \ .$$

  If each coefficient of the resulting polynomial is close to 0, set the respective coefficient of decrypted message to 0. Otherwise, set the decrypted message as 1.

The RLWE encryption scheme defined as above is semantically secure under RLWE_{χ_1} assumption. To see this, just observe that the ciphertext consists of two RLWE samples, which by the RLWE_{χ_1} assumption indistinguishable from uniformly random elements. Thus, unless one can solve *decisional* RLWE problem, all ciphertexts look like uniform and no information can be extracted about the plaintext.

Definition 2.11 ($MLWE_{n,m,\chi}$)**.** *In the Module-LWE problem with parameters $n, m > 0$ and an error distribution χ over $\mathcal{R}$, the PPT adversary $\mathcal{A}$ is asked to distinguish $(\boldsymbol{a}, \vec{\boldsymbol{t}}) \xleftarrow{\$} \mathcal{R}_q^{m\times n} \times \mathcal{R}_q^m$ from $(\boldsymbol{A}, \boldsymbol{A}\vec{\boldsymbol{s}} + \vec{\boldsymbol{e}})$ for $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times n}$, a secret vector $\vec{\boldsymbol{s}} \leftarrow \chi^n$, and an error vector $\vec{\boldsymbol{e}} \leftarrow \chi^m$.*

Definition 2.12 ($MSIS_{m,n,\beta}$)**.** *The goal in the Module-SIS problem with parameters $n, m > 0$ and $0 < \beta < q$ is to find $\vec{\boldsymbol{x}} \xleftarrow{\$} \mathcal{R}_q^m$ for a given matrix $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$ such that $\boldsymbol{A}\vec{\boldsymbol{x}} = \vec{\boldsymbol{0}} \bmod q$ and $0 < \|\vec{\boldsymbol{x}}\|_2 < \beta$.*

In practical security estimations, the parameter m in Definitions 2.11 and 2.12 does not play a crucial role, therefore we simply omit it and use the notations $\mathsf{MLWE}_{n,\chi}$ and $\mathsf{MSIS}_{n,\beta}$. Furthermore, we let the parameters μ and λ denote the module ranks for MSIS and MLWE.

2.8.2 Commitment scheme

In this work, we will be using a variant of BDLOP commitment scheme [BDL+18]. Let, $\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu\times(\mu+\lambda+1)}$, $\vec{\boldsymbol{b}}_1 \in \mathcal{R}_q^{\mu+\lambda+1}$ and $\vec{\boldsymbol{r}} \leftarrow \chi_2^{(\mu+\lambda+1)d}$. The commitment of a single message $\boldsymbol{m} \in \mathcal{R}_q$ is a pair $(\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ where

$$\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}}\ ,$$
$$\boldsymbol{t}_1 = \langle\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}\rangle + \boldsymbol{m}\ .$$

It is easy to see that the commitment scheme is binding and hiding due to MSIS_μ and MLWE_λ assumptions respectively.

Definition 2.13. *A weak opening for the commitment $\vec{\boldsymbol{t}} = \vec{\boldsymbol{t}}_0 \| \boldsymbol{t}_1$ consists of l polynomials $\bar{\boldsymbol{c}}_i \in \mathcal{R}_q$, randomness vector $\vec{\boldsymbol{r}}^\star$ over $\mathcal{R}_q$ and a message $\boldsymbol{m}^\star \in \mathcal{R}_q$ such that*

$$\|\bar{\boldsymbol{c}}_i\|_2 \le 2\kappa \text{ and } \bar{\boldsymbol{c}}_i \bmod \varphi_i \neq 0 \text{ for all } 1 \le i \le l\ ,$$
$$\|\bar{\boldsymbol{c}}_i\vec{\boldsymbol{r}}^\star\|_2 \le 2\beta \text{ for all } 1 \le i \le l\ ,$$
$$\boldsymbol{B}_0\vec{\boldsymbol{r}}^\star = \vec{\boldsymbol{t}}_0\ ,$$
$$\langle\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{r}}^\star\rangle + \boldsymbol{m}^\star = \boldsymbol{t}_1\ .$$

The commitment scheme is proven to be binding also with respect to the weak opening in [ALS20, Lemma 4.3].

Using Galois automorphisms and fully splitting rings a zero-knowledge proof of opening for BDLOP commitments is given in [ALS20]. The protocol in Figure 2 is complete, statistical zero-knowledge and computationally sound under Module-SIS assumption. Keeping the structure same and adding some garbage terms, it is possible to

prove linear (Figure 3) and product (Figure 4) relationships between committed messages. Moreover, the soundness of the protocols is not reduced when amortized over many relations. In [ALS20] have proved security properties of zero-knowledge proof of product relation protocol.

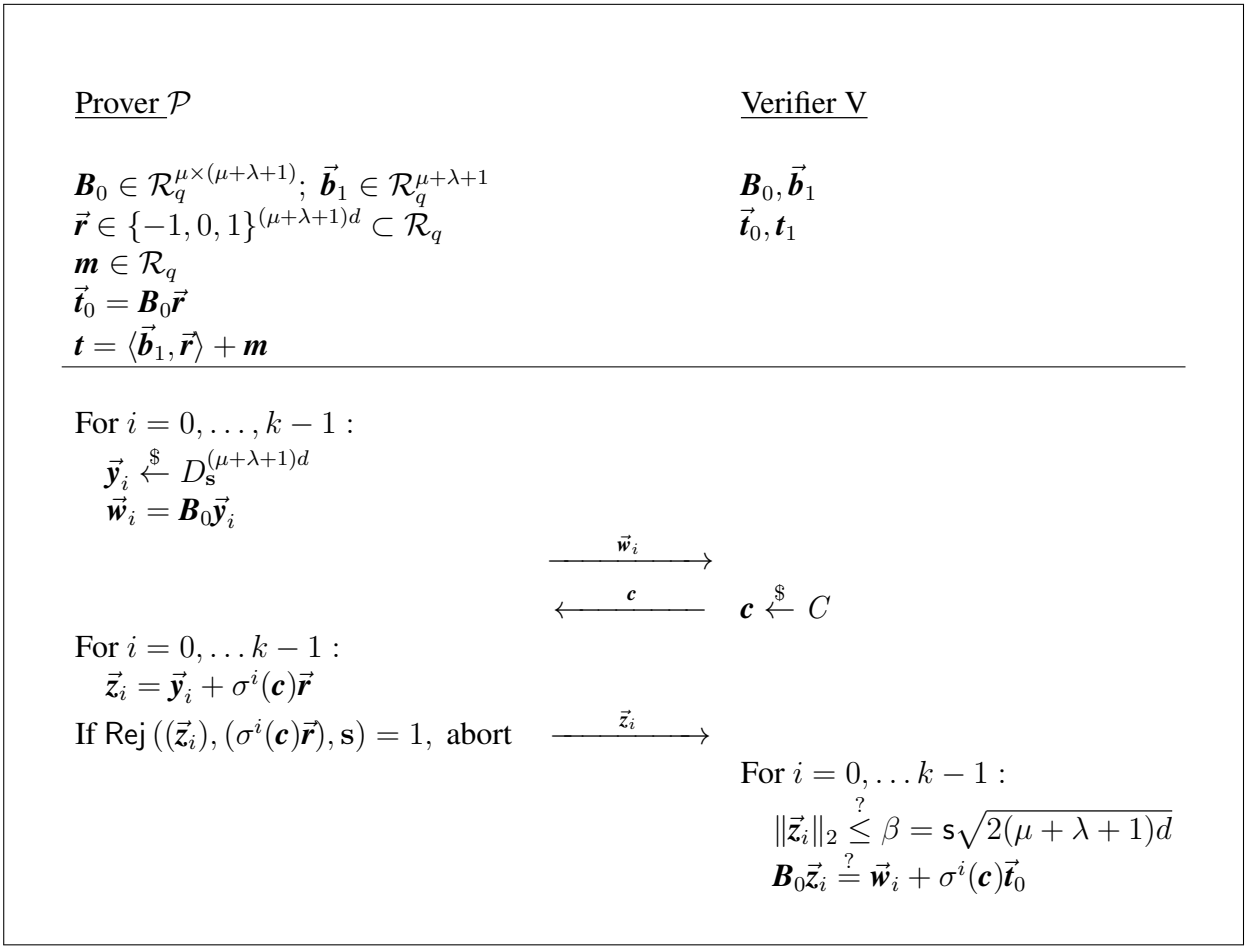


Figure 2. Automorphism opening proof. $\sigma = \sigma_{2l/k+1} \in \mathbf{Aut}(\mathcal{R}_q)$, C is the challenge distribution over $\mathcal{R}$ where each coefficient is independently identically distributed with Pr(0) = 1/2 and Pr(1)=Pr(-1)=1/4, and D_{s} is the discrete Gaussian distribution on $\mathbb{Z}$ with standard deviation $\mathrm{s} = 11kd\|\vec{\boldsymbol{r}}\|_2$ [ALS20]

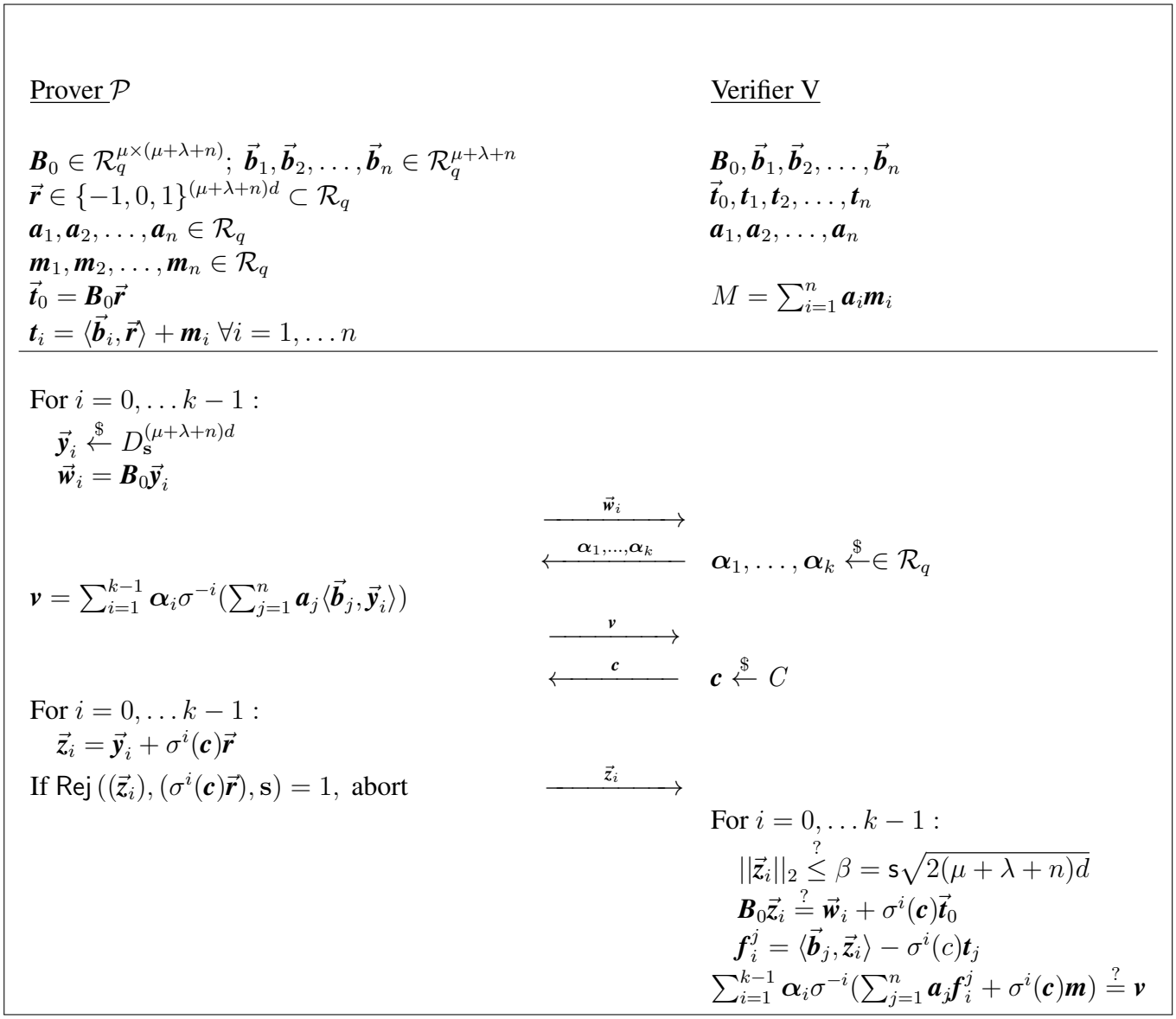


Figure 3. Amortized linear relation proof: ZK-proof $[M = \sum_{i=1}^n \boldsymbol{a}_i \boldsymbol{m}_i]$

Prover $\mathcal{P}$ | Verifier V

$= \mu + \lambda + 3n + 1$
$\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu \times l}$; $\vec{\boldsymbol{b}}_1^{(j)}, \vec{\boldsymbol{b}}_2^{(j)}, \vec{\boldsymbol{b}}_3^{(j)}, \vec{\boldsymbol{b}}_4 \in \mathcal{R}_q^l$
$\vec{\boldsymbol{r}} \in \chi^{ld}$;
$\boldsymbol{m}_1^{(j)}, \boldsymbol{m}_2^{(j)}, \boldsymbol{m}_3^{(j)} \in \mathcal{R}_q$
$\vec{\boldsymbol{t}}_{\boldsymbol{0}} = \boldsymbol{B}_0, \vec{\boldsymbol{r}}$
$\boldsymbol{t}_1^{(j)} = \langle \vec{\mathbf{b}}_1^{(j)}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{m}_1^{(j)}$
$\boldsymbol{t}_2^{(j)} = \langle \vec{\mathbf{b}}_2^{(j)}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{m}_2^{(j)}$
$\boldsymbol{t}_3^{(j)} = \langle \vec{\mathbf{b}}_3^{(j)}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{m}_3^{(j)}$

$\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1^{(j)}, \vec{\boldsymbol{b}}_2^{(j)}, \vec{\boldsymbol{b}}_3^{(j)}, \vec{\boldsymbol{b}}_4$
$\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1^{(j)}, \boldsymbol{t}_2^{(j)}, \boldsymbol{t}_3^{(j)}$

For $i = 0, \ldots, k-1$:
$\vec{\boldsymbol{y}}_i \xleftarrow{\$} D_{\mathbf{s}}^{ld}$
$\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0 \vec{\boldsymbol{y}}_i$

$\xrightarrow{\vec{\boldsymbol{w}}_i}$
$\xleftarrow{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{nk}}$ $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{nk} \xleftarrow{\$} \in \mathcal{R}_q$

$$\boldsymbol{t}_4 = \langle \vec{\boldsymbol{b}}_4, \vec{\boldsymbol{r}} \rangle + \sum_{i=0}^{k-1} \sum_{j=1}^{n} \boldsymbol{\alpha}_{in+j} \sigma^{-i} (\langle \vec{\boldsymbol{b}}_3^{(j)}, \vec{\boldsymbol{y}}_i \rangle - $$
$$- m_1^{(j)} \langle \vec{\boldsymbol{b}}_2^{(j)}, \vec{\boldsymbol{y}}_i \rangle - m_2^{(j)} \langle \vec{\boldsymbol{b}}_1^{(j)}, \vec{\boldsymbol{y}}_i \rangle)$$
$$\boldsymbol{v} = \langle \vec{\boldsymbol{b}}_4, \vec{\mathbf{y}}_0 \rangle + \sum_{i=0}^{k-1} \sum_{j=1}^{n} \boldsymbol{\alpha}_{in+j} \sigma^{-i} \left(\langle \vec{\boldsymbol{b}}_1^{(j)}, \vec{\boldsymbol{y}}_i \rangle \langle \vec{\boldsymbol{b}}_2^{(j)}, \vec{\boldsymbol{y}}_i \rangle \right)$$

$\xrightarrow{\boldsymbol{t}_4, \boldsymbol{v}}$
$\xleftarrow{\boldsymbol{c}}$ $\boldsymbol{c} \xleftarrow{\$} C$

For $i = 0, \ldots k-1$:
$\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}_i + \sigma^i(\boldsymbol{c}) \vec{\boldsymbol{r}}$
If Rej $\left((\vec{\boldsymbol{z}}_i), (\sigma^i(\boldsymbol{c}) \vec{\boldsymbol{r}}), \mathbf{s} \right) = 1$, abort

$\xrightarrow{\vec{\boldsymbol{z}}_i}$

For $i = 0, \ldots k-1$:
$||\vec{\boldsymbol{z}}_i||_2 \overset{?}{\leq} \beta = \mathsf{s}\sqrt{2ld}$
$\boldsymbol{B}_0 \vec{\boldsymbol{z}}_i \overset{?}{=} \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c}) \vec{\boldsymbol{t}}_0$
$\boldsymbol{f}_1^{(i),(j)} = \langle \vec{\boldsymbol{b}}_1^{(j)}, \vec{\boldsymbol{y}}_i \rangle - \sigma^i(\boldsymbol{c}) \boldsymbol{t}_1^{(j)}$
$\boldsymbol{f}_2^{(i),(j)} = \langle \vec{\boldsymbol{b}}_2^{(j)}, \vec{\boldsymbol{y}}_i \rangle - \sigma^i(\boldsymbol{c}) \boldsymbol{t}_2^{(j)}$
$\boldsymbol{f}_3^{(i),(j)} = \langle \vec{\boldsymbol{b}}_3^{(j)}, \vec{\boldsymbol{y}}_i \rangle - \sigma^i(\boldsymbol{c}) \boldsymbol{t}_3^{(j)}$

$\boldsymbol{f}_4 = \langle \vec{\boldsymbol{b}}_4, \vec{\boldsymbol{z}}_0 \rangle - \boldsymbol{c} \boldsymbol{t}_4$
$$\sum_{i=0}^{k-1} \sum_{j=1}^{n} \boldsymbol{\alpha}_{in+j} \sigma^{-i} \left(\boldsymbol{f}_1^{(i),(j)} \boldsymbol{f}_2^{(i),(j)} + \sigma^i(c) \boldsymbol{f}_3^{(i),(j)} \right) + \boldsymbol{f}_4 \overset{?}{=} \boldsymbol{v}$$

Figure 4. ZK-proof $\left[(\boldsymbol{m}_1^{(j)} \boldsymbol{m}_2^{(j)} = \boldsymbol{m}_3^{(j)})_{j=1}^n \right]$

2.9 Cryptographic voting

In cryptography, the secure voting protocol consists of several phases and algorithms and satisfies transparency, privacy, and end-to-end verifiability requirements. A generic voting scheme consists of the following parts.

- **Encryption scheme**: Voters encrypt their choice for privacy purposes
- **Bulletin board**: Also ballot box, where all cast ballots are stored. Usually, it is a publicly shared read-only list.
- **Digital signature scheme**: Besides hiding their choice, voters prove their eligibility to vote by digitally signing the encrypted ballot.
- **Tallying**: After all votes cast, all entries in the bulletin board counted. The responsible authority is called **tallier**.

Besides secure encryption and unforgeable signature properties, the electronic voting protocol has to be publicly verifiable for all steps. This is called end-to-end verifiability. Different approaches to electronic voting require different measures to be taken. There are two common approaches used in real elections.

2.9.1 Homomorphic e-voting

A homomorphic encryption scheme satisfies either or both of the following properties:

- Additively homomorphic: $Enc(m_1) + Enc(m_2) = Enc(m_1 + m_2)$
- Multiplicatively homomorphic: $Enc(m_1) \cdot Enc(m_2) = Enc(m_1 m_2)$

The use of homomorphic encryption schemes in cryptographic voting protocols makes tallying possible without decrypting each cast ballot. Instead, a tallying function is evaluated on ciphertexts and decrypted to get the final tally. Indeed, this is the main idea of homomorphic voting. For example, in Yes-No voting, if the answers are represented as 1 (YES) and 0 (NO), then just by adding all ciphertexts the election result can be obtained.

Homomorphic properties also make it easy to alter election results. Following the previous analogy of Yes-No voting, an adversary can cast 2, or any number, indeed any data to ruin the election. To prevent this, voters are required to submit proof that cast

votes are encoded properly. Generally, proof of plaintext knowledge protocols is used in this case.

An electoral authority may also be dishonest. Sometimes, the tallying function is not a straightforward addition. Then, the tallying authority should provide proof of evaluation. In the end, there should be another proof that evaluation on ciphertexts correctly decrypts to the published tally.

2.9.2 Verifiable shuffle-based e-voting

In verifiable shuffle-based voting protocols, cast ballots are decrypted then tallied individually. In between, a mixing network secretly shuffles ciphertexts to remove the link between voters and their choice. Shuffling is permuting the order of ciphertexts and re-encrypting afterward. Re-encryption is generally realized utilizing the homomorphic nature of the encryption scheme. In other words, if the scheme is additively homomorphic, encryption of zero is added to the original ciphertext. If it is multiplicatively homomorphic, the ciphertext is multiplied by encryption of one. As usual, the mixing authority has to provide cryptographic proof that shuffling has been done correctly and no single vote has been modified. Moreover, plaintexts of input and output ciphertexts are the same but permuted. In practice, mixing networks contain at least there independent mixing nodes to completely unlink input and output ciphertexts even if the half of mixing nodes are malicious.

2.9.3 Mix-node security

Costa *et al.* [CMM19] proposed novel security definition for a mix-node. Assume that **MixVotes** is a generic mixing algorithm such that, given input ciphertexts and a permutation vector, produces shuffled and re-encrypted ciphertexts. Moreover, let $z^{(i_{\mathcal{A}})}$ and $z^{\pi(j_{\mathcal{A}})}$ the message before and after running the algorithm.

Definition 2.14. *Let J be a uniform random variable taking values in $[1, \ldots, N]$. A mix-node given by algorithm **MixVotes** is said to be secure if the advantage of any PPT adversary $\mathcal{A}$ over random guess is negligible in the security parameter. That is, $\forall c$, $\exists \kappa_0$ s.t if $\kappa > \kappa_0$:*

$$\boldsymbol{Adv}_{\mathcal{A}}^{sec} = \left|\Pr\left[z^{(i_{\mathcal{A}})} = z^{\pi(j_{\mathcal{A}})}\right] - \Pr\left[z^{(i_{\mathcal{A}})} = z^{\pi(J)}\right]\right| < \frac{1}{\kappa^c} .$$

3 Post-Quantum mix-net

3.1 Related Works

Numerous works have been done on the idea of mixing networks (mix-nets) since its introduction by Chaum in 1981 [Cha81]. In secure e-voting context, mix nets should be verifiable as well. A very broad and detailed review of existing verification methods for mix-nets in the literature is given in a very recent paper [HM20]. However, only a small subset of those verification methods have been reconstructed using post-quantum cryptography tools. To the best of our knowledge, the only practical lattice-based mix-net [BHM20] achieves verifiability using an alternative approach to zero-knowledge proof of shuffle. Nonetheless, [CMM17] and [Str18] give proofs of a shuffle for lattice-based re-encryption mix-nets. The first one may not be considered as fully post-quantum as it uses Pedersen commitments which are based on discrete log problem. The last construction requires a Fully Homomorphic Encryption scheme and any homomorphic commitment scheme. While the commitment scheme in [BDL$^+$18] is additively homomorphic and post-quantum secure making the whole proof post-quantum safe, whereas FHE schemes are not practical yet.

3.2 Costa, Martinez and Morillo proof of shuffle

Costa *et al.'s* improved their work and proposed a proof of shuffle which is constructed over lattice-based post-quantum secure primitives only [CMM19]. The main idea is briefly given here.

Let $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d+1)$ be ring of polynomials degree less than d and $q \equiv 3 \bmod 8$. This rings splits into two residue fields of polynomials degree less than $d/2$ and every polynomial of degree smaller than $d/2$ in $\mathcal{R}_q$ are invertible. Define RLWE encryption as in Section 2.10.

Assume that there are N RLWE ciphertexts $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ encrypted with public key $(pk.\boldsymbol{a}, pk.\boldsymbol{b})$ to be shuffled. A mixing node will generate secret random zero encryption ciphertexts $(\boldsymbol{u}_i^0, \boldsymbol{v}_i^0)$ and permutation π, and output $(\boldsymbol{u}'_i, \boldsymbol{v}'_i)$ such that

$$
\begin{aligned}
(\boldsymbol{u}_i^0, \boldsymbol{v}_i^0) &= (pk.\boldsymbol{a} \cdot \boldsymbol{r}_{E,i} + \boldsymbol{e}_{u,i}, pk.\boldsymbol{b} \cdot \boldsymbol{r} + \boldsymbol{e}_{v,i} + 0) \\
(\boldsymbol{u}'_i, \boldsymbol{v}'_i) &= (\boldsymbol{u}^{\pi(i)} + \boldsymbol{u}_i^0, \boldsymbol{v}^{\pi(i)} + \boldsymbol{v}_i^0)
\end{aligned}
$$

where $\boldsymbol{r}_{E,i}, \boldsymbol{e}_{u,i}, \boldsymbol{e}_{v,i} \leftarrow \chi_1$ for all $i = 1, \ldots, N$. Then, authors prove that if π is a valid permutation, then for any $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{S}$ where S is a set of polynomials of degree less than $n/2$

$$
\prod_{i=1}^{N} (\boldsymbol{\beta} i + \boldsymbol{\alpha}^i - \boldsymbol{\gamma}) = \prod_{i=1}^{N} (\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma}) \tag{1}
$$

holds due to generalized Schwartz-Zippel lemma with small cheating probability. Similarly,

$$\sum_{i=1}^{N} \boldsymbol{\alpha}^i \boldsymbol{u}^{(i)} = \sum_{i=1}^{N} \boldsymbol{\alpha}^{\pi(i)} (\boldsymbol{u}'^{(i)} - \boldsymbol{u}_0^{(i)}) \tag{2}$$

$$\sum_{i=1}^{N} \boldsymbol{\alpha}^i \boldsymbol{v}^{(i)} = \sum_{i=1}^{N} \boldsymbol{\alpha}^{\pi(i)} (\boldsymbol{v}'^{(i)} - \boldsymbol{v}_0^{(i)}). \tag{3}$$

One can think of each formula as two polynomials with coefficients in $\mathcal{R}_q$ evaluated at the same point $\boldsymbol{\alpha}$. Again, due to generalized Schwartz-Zippel lemma, if equality holds, then both polynomials are equal to each other, thus their coefficients are the same. Moreover, the relations (1), (2) and (3) along with proof of correct encryption are shown in [CMM19] to be enough to argue correctness of shuffle.

The protocol in [CMM19] uses commitment scheme from [BKLP15] to prove aforementioned arguments mainly due to existence of zero-knowledge proofs for linear and multiplicative relations for the commitment scheme. We recall the protocol briefly below.

First, the prover $\mathcal{P}$ commits to zero encryption ciphertexts $(\boldsymbol{u}_i^0, \boldsymbol{v}_i^0)$, sends them to the verifier $\mathcal{V}$ and runs amortized zero-knowledge proof of knowledge of small secret elements that those commitments are indeed commitments to encryptions of zero with valid error parameters. Next, $\mathcal{P}$ commits to the permutation vector π and sends the commitment to the verifier again. Committing to permutation vector is committing to $\boldsymbol{\pi(1)}, \ldots, \boldsymbol{\pi(N)}$. Then, $\mathcal{V}$ samples a polynomial $\boldsymbol{\alpha}$ from the challenge set send back to the prover. Following to that, $\mathcal{P}$ calculates commitments to $\boldsymbol{\alpha}^{\pi(1)}, \ldots, \boldsymbol{\alpha}^{\pi(N)}$. To show that the permutation vector is chosen before challenges and is a valid permutation, the prover runs linear and multiplicative relation proofs several times and calculates the product in (1) using committed values. Next, again by help of those relation proofs, it proves the remaining two equalities to show shuffling is correct. During the verification phase, the verifier has to verify zero knowledge proofs of knowledge of small secret elements and relations (1), (2) and (3).

Costa *et al.* mention that it is possible to use amortization techniques described in [dPLNS17] and reduce the complexity and total cost of the protocol. Unfortunately, they have not explicitly shown how to do that, nor have they instantiated the parameters to evaluate performance and concrete security level of the protocol.

3.3 Our work

In this work, we suggest several modifications to solve both issues by replacing the commitment scheme with a variant of Module SIS/LWE based commitment scheme from [BDL$^+$18]. This replacement allows use of more efficient zero-knowledge proofs of linear and product (Figure 4) relation between committed messages [ALS20, LNS20]. Those protocols are short, efficient and have no extra cost when amortized over many

relations. Besides, there is no need to repeat the protocol several times to get desired soundness properties. Instead, Galois automorphisms are used to boost soundness. However, to achieve great results, it is required to use fully splitting rings. Nevertheless, as we change the mathematical setting, there is a need for additional careful analysis of security. Another important change we bring in is use of challenge set defined in Section 2.4 instead of S.

Next, we describe our protocol. Semantically, it follows the same structure. First, let μ and λ be rank of secure MSIS and MLWE instances, respectively, and choose $q - 1 \equiv 2l \bmod 4l$ such that $\mathcal{R}_q$ is a fully splitting ring, i.e $d = l$ and $\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu \times (\mu+\lambda+9N+1)}$, $\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_2, \dots \vec{\boldsymbol{b}}_{9N+1} \in \mathcal{R}_q^{\mu+\lambda+9N+1}$. Furthermore, choose k so that $q^{kd/l} \approx 2^{256}$.

Again, assume that there are N RLWE ciphertexts $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ which are shuffled by permutation vector π and re-encrypted adding $(\boldsymbol{u}_i^0, \boldsymbol{v}_i^0)$. The resulting ciphertexts $(\boldsymbol{u}'_i, \boldsymbol{v}'_i)$ also satisfy (1), (2) and (3) for random challanges $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{C}$.

First, the prover $\mathcal{P}$ has to prove that zero-encryptions are not ill-formed, in other words, they are correct encryptions of zero polynomial. However, ciphertexts has to be kept secret, therefore it computes and sends commitments.

$$\begin{aligned} \vec{\boldsymbol{t}}_0 &= \boldsymbol{B_0}\vec{\boldsymbol{r}} \\ \boldsymbol{t}_{u_0^{(i)}} &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \boldsymbol{u}_0^{(i)} \\ \boldsymbol{t}_{v_0^{(i)}} &= \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{v}_0^{(i)} \end{aligned}$$

where $\vec{\boldsymbol{r}} \in \mathcal{R}_q^{\mu+\lambda+9N+1}$ is the commitment randomness. Unpacking and substituting the ciphertexts we have

$$\begin{aligned} \boldsymbol{t}_{u_0^{(i)}} &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + pk.\boldsymbol{a} \cdot \boldsymbol{r}_{i,E} + \boldsymbol{e}_{i,u}, \\ \boldsymbol{t}_{v_0^{(i)}} &= \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{r}} \rangle + pk.\boldsymbol{b} \cdot \boldsymbol{r}_{i,E} + \boldsymbol{e}_{i,v} + \boldsymbol{0}. \end{aligned}$$

Rewriting as a matrix equation we get a specific structure:

$$\begin{vmatrix} \boldsymbol{t}_{u_0^{(i)}} \\ \boldsymbol{t}_{v_0^{(i)}} \end{vmatrix} = \begin{vmatrix} \vec{\boldsymbol{b}}_{i,1} & \dots & \vec{\boldsymbol{b}}_{i,n'} & pk.\boldsymbol{a} & \mathbf{1} & \mathbf{0} \\ \vec{\boldsymbol{b}}_{N+i,1} & \dots & \vec{\boldsymbol{b}}_{N+i,n'} & pk.\boldsymbol{b} & \mathbf{0} & \mathbf{1} \end{vmatrix} \begin{vmatrix} \vec{\boldsymbol{r}}_1 \\ \vdots \\ \vec{\boldsymbol{r}}_{n'} \\ \boldsymbol{r}_{i,E} \\ \boldsymbol{e}_{i,u} \\ \boldsymbol{e}_{i,v} \end{vmatrix}$$

where $n' = \mu + \lambda + 9N + 1$. Observe that the last equation has form $\mathbf{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{u}}$. Proving $\vec{\boldsymbol{s}}$ is short in this equation also proves that the committed message is a valid encryption of zero polynomial. Unfortunately, there is no practical exact proof of short solution to structured linear equation in $\mathcal{R}_q$. However, one can transfer the equation into better

understood $\mathbb{Z}_q$ domain almost at no cost. Then it is possible to use proof of knowledge of a ternary solution to an unstructured linear equation over $\mathbb{Z}_q$ described in [ENS20].

$$\begin{vmatrix} \mathcal{V}_{\boldsymbol{t}_{u_0^{(i)}}} \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(i)}}} \end{vmatrix} = \begin{vmatrix} \mathcal{M}_{\vec{\boldsymbol{b}}_{i,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{i,n'}} & \mathcal{M}_{pk.\boldsymbol{a}} & I_d & \mathbf{0}_d \\ \mathcal{M}_{\vec{\boldsymbol{b}}_{N+i,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{N+i,n'}} & \mathcal{M}_{pk.\boldsymbol{b}} & \mathbf{0}_d & I_d \end{vmatrix} \begin{vmatrix} \mathcal{V}_{\vec{\boldsymbol{r}}_1} \\ \vdots \\ \mathcal{V}_{\vec{\boldsymbol{r}}_{n'}} \\ \mathcal{V}_{\boldsymbol{r}_{i,E}} \\ \mathcal{V}_{\boldsymbol{e}_{i,u}} \\ \mathcal{V}_{\boldsymbol{e}_{i,v}} \end{vmatrix}$$

$$\vec{\boldsymbol{u}} = A\vec{\boldsymbol{s}}$$

$$\vec{\boldsymbol{u}} \in \mathbb{Z}_q^{2d} \quad \vec{\boldsymbol{s}} \in \mathbb{Z}_q^{n} \quad A \in \mathbb{Z}_q^{2d \times n} \quad n = (n'+3)d$$

Proving the shortness of secret values for each ciphertext individually is not cost-effective. The main reason is that proving shortness of $\vec{\boldsymbol{r}}$ will be repeated for each ciphertext. One would look for amortized or batch-proofs to solve the problem. However, it is also possible to reconstruct $\mathbf{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{u}}$ relation for all commitments at once. More specifically, it is possible to concatenate all matrix equations into a single bigger equation as below.

$$\begin{vmatrix} \boldsymbol{t}_{u_0^{(1)}} \\ \boldsymbol{t}_{v_0^{(1)}} \\ \boldsymbol{t}_{u_0^{(2)}} \\ \boldsymbol{t}_{v_0^{(2)}} \\ \vdots \\ \boldsymbol{t}_{u_0^{(N)}} \\ \boldsymbol{t}_{v_0^{(N)}} \end{vmatrix} = \begin{vmatrix} \vec{\boldsymbol{b}}_{1,1} & \dots & \vec{\boldsymbol{b}}_{1,n'} & pk.\boldsymbol{a} & \mathbf{0} & \cdots & \mathbf{0} \\ \vec{\boldsymbol{b}}_{N+1,1} & \dots & \vec{\boldsymbol{b}}_{N+1,n'} & pk.\boldsymbol{b} & \mathbf{0} & \cdots & \mathbf{0} \\ \vec{\boldsymbol{b}}_{2,1} & \dots & \vec{\boldsymbol{b}}_{2,n'} & \mathbf{0} & pk.\boldsymbol{a} & \cdots & \mathbf{0} \\ \vec{\boldsymbol{b}}_{N+2,1} & \dots & \vec{\boldsymbol{b}}_{N+2,n'} & \mathbf{0} & pk.\boldsymbol{b} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vec{\boldsymbol{b}}_{N,1} & \dots & \vec{\boldsymbol{b}}_{N,n'} & \mathbf{0} & \mathbf{0} & \cdots & pk.\boldsymbol{a} \\ \vec{\boldsymbol{b}}_{2N,1} & \dots & \vec{\boldsymbol{b}}_{2N,n'} & \mathbf{0} & \mathbf{0} & \cdots & pk.\boldsymbol{b} \end{vmatrix} \mathbf{I}_{2N} \begin{vmatrix} \vec{\boldsymbol{r}}_1 \\ \vdots \\ \vec{\boldsymbol{r}}_{n'} \\ \boldsymbol{r}_{1,E} \\ \boldsymbol{r}_{2,E} \\ \vdots \\ \boldsymbol{r}_{N,E} \\ \boldsymbol{e}_{1,u} \\ \boldsymbol{e}_{1,v} \\ \boldsymbol{e}_{2,u} \\ \boldsymbol{e}_{2,v} \\ \vdots \\ \boldsymbol{e}_{N,u} \\ \boldsymbol{e}_{N,v} \end{vmatrix} \tag{4}$$

Here, $\mathbf{I}_{2N}$ is $2N \times 2N$ identity matrix with diagonal elements being polynomial $\mathbf{1}$. Finally, we transfer the equation (4) from $\mathcal{R}_q$ to $\mathbb{Z}_q$ domain.

$$\begin{vmatrix} \mathcal{V}_{\boldsymbol{t}_{u_0^{(1)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(1)}}} \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(2)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(2)}}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(N)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(N)}}} \end{vmatrix} = \begin{vmatrix} \mathcal{M}_{\vec{\boldsymbol{b}}_{1,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{1,n'}} & \mathcal{M}_{pk.\boldsymbol{a}} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ \mathcal{M}_{\vec{\boldsymbol{b}}_{N+1,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{1,n'}} & \mathcal{M}_{pk.\boldsymbol{b}} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ \mathcal{M}_{\vec{\boldsymbol{b}}_{2,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{2,n'}} & \mathbf{0}_d & \mathcal{M}_{pk.\boldsymbol{a}} & \cdots & \mathbf{0}_d \\ \mathcal{M}_{\vec{\boldsymbol{b}}_{N+2,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{2,n'}} & \mathbf{0}_d & \mathcal{M}_{pk.\boldsymbol{b}} & \cdots & \mathbf{0}_d \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}_{N,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{N,n'}} & \mathbf{0}_d & \mathbf{0}_d & \cdots & \mathcal{M}_{pk.\boldsymbol{a}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}_{2N,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}_{2N,n'}} & \mathbf{0}_d & \mathbf{0}_d & \cdots & \mathcal{M}_{pk.\boldsymbol{b}} \end{vmatrix} \begin{vmatrix} \mathbf{I}_{2Nd} \end{vmatrix} \begin{vmatrix} \mathcal{V}_{\vec{\boldsymbol{r}}_1} \\ \vdots \\ \mathcal{V}_{\vec{\boldsymbol{r}}_{n'}} \\ \boldsymbol{r}_{1,E} \\ \mathcal{V}_{\boldsymbol{r}_{2,E}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{r}_{N,E}} \\ \mathcal{V}_{\boldsymbol{e}_{1,u}} \\ \mathcal{V}_{\boldsymbol{e}_{1,v}} \\ \mathcal{V}_{\boldsymbol{e}_{2,u}} \\ \mathcal{V}_{\boldsymbol{e}_{2,v}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,u}} \\ \mathcal{V}_{\boldsymbol{e}_{N,v}} \end{vmatrix} \tag{5}$$

The last equality has the form $A\vec{s} = \vec{u}$, too. This time $n = (n' + 3N)d$, so that $\vec{u} \in \mathbb{Z}_q^{2Nd} \quad \vec{s} \in \mathbb{Z}_q^n \quad A \in \mathbb{Z}_q^{2Nd \times n}$.

Finally, we employ optimization technique described in Section 3.4 and set commitment vectors as $\vec{\boldsymbol{b}}_i = \vec{\mathbf{0}}_\mu \| \vec{\boldsymbol{e}}_i \| \vec{\boldsymbol{b}}'_i$ where $\vec{\boldsymbol{e}}_i$ are $9N + 1$ dimensional standard basis vectors and $\vec{\boldsymbol{b}}' \in \mathcal{R}_q^\lambda$. Changing rows and simplifying (5), we get

$$\begin{vmatrix} \mathcal{V}_{\boldsymbol{t}_{u_0^{(1)}}} \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(N)}}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(N)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(1)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(2)}}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(N)}}} \end{vmatrix} = \left[\begin{vmatrix} \mathbf{0}_{2Nd \times \mu d} \end{vmatrix} \begin{vmatrix} \mathbf{I}_{2Nd} \end{vmatrix} \begin{vmatrix} \mathbf{0}_{2Nd \times (7N+1)d} \end{vmatrix} \begin{vmatrix} \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+1,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+1,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+2,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+2,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2N,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2N,\lambda}} \end{vmatrix} \begin{vmatrix} \mathcal{M}_{pk.\boldsymbol{a}} \\ \mathcal{M}_{pk.\boldsymbol{b}} \end{vmatrix} \otimes \mathbf{I}_N \begin{vmatrix} \mathbf{I}_{2Nd} \end{vmatrix} \right] \begin{vmatrix} \mathcal{V}_{\vec{\boldsymbol{r}}_1} \\ \vdots \\ \mathcal{V}_{\vec{\boldsymbol{r}}_{n'}} \\ \mathcal{V}_{\boldsymbol{r}_{1,E}} \\ \mathcal{V}_{\boldsymbol{r}_{2,E}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{r}_{N,E}} \\ \mathcal{V}_{\boldsymbol{e}_{1,u}} \\ \mathcal{V}_{\boldsymbol{e}_{2,u}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,u}} \\ \mathcal{V}_{\boldsymbol{e}_{1,v}} \\ \mathcal{V}_{\boldsymbol{e}_{2,v}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,v}} \end{vmatrix}$$

As multiplying with zero matrix doesn't affect at all, without loss of generality it can be removed from equation. Then, polynomials with index in the range $0 \leq i \leq \mu$ and $2N \leq i \leq 9N + 1$ in commitment randomness should also be removed. The final form of the equation will thus be

$$\begin{vmatrix} \mathcal{V}_{\boldsymbol{t}_{u_0^{(1)}}} \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(N)}}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{t}_{u_0^{(N)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(1)}}} \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(2)}}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{t}_{v_0^{(N)}}} \end{vmatrix} = \left[\mathbf{I}_{2Nd} \begin{vmatrix} \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+1,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+1,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+2,1}} & \dots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N+2,\lambda}} \end{vmatrix} \begin{vmatrix} \mathcal{M}_{pk.\boldsymbol{a}} \\ \mathcal{M}_{pk.\boldsymbol{b}} \end{vmatrix} \otimes \mathbf{I}_N \middle| \mathbf{I}_{2Nd} \right] \begin{vmatrix} \mathcal{V}_{\vec{\boldsymbol{r}}_{\mu+1}} \\ \vdots \\ \mathcal{V}_{\vec{\boldsymbol{r}}_{n'}} \\ \mathcal{V}_{\boldsymbol{r}_{1,E}} \\ \mathcal{V}_{\boldsymbol{r}_{2,E}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{r}_{N,E}} \\ \mathcal{V}_{\boldsymbol{e}_{1,u}} \\ \mathcal{V}_{\boldsymbol{e}_{2,u}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,u}} \\ \mathcal{V}_{\boldsymbol{e}_{1,v}} \\ \mathcal{V}_{\boldsymbol{e}_{2,v}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,v}} \end{vmatrix} .$$

In the last equation, $n' = \lambda + 2N$ and as before $n = (n' + 3N)d$. Having an equation of the form $A\vec{s} = \vec{u}$, the prover follows steps in [ENS20, Figure 3] to generate zero-knowledge proof of correctness of zero-encryption ciphertexts. From now on, we call this proof *shortness proof*, as we have seen it is mathematically equivalent to prove shortness of encryption terms.

Next, $\mathcal{P}$ commits to permutation vector using the same commitment randomness as before.

$$\boldsymbol{t}_{\pi(i)} = \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{r}} \rangle + \pi(i)$$

At this point, the prover sends $\vec{\boldsymbol{t}}_0, \boldsymbol{t}_{u_0^{(i)}}, \boldsymbol{t}_{v_0^{(i)}}, \boldsymbol{t}_{\pi(i)}$ commitments and shortness proof to the verifier in order to get random challange $\boldsymbol{\alpha}$ which is used in (1), (2) and (3) arguments. If expanded, those arguments contain intermediate terms $\boldsymbol{\alpha}^{\pi(i)}, \boldsymbol{\alpha}^{\pi(i)}\boldsymbol{u}_0^{(i)}$ and $\boldsymbol{\alpha}^{\pi(i)}\boldsymbol{v}_0^{(i)}$. Therefore, the prover also calculates commitments to them.

$$\begin{aligned} \boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}} &= \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)} \\ \boldsymbol{t}_{4N+i} &= \langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)}\boldsymbol{u}_0^{(i)} \\ \boldsymbol{t}_{5N+i} &= \langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)}\boldsymbol{v}_0^{(i)} \end{aligned}$$

Then, the verifier receives last $3N$ commitments and returns challenges $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ sampled from $\mathcal{C}$. The prover continues to calculate commitments to remaining intermediate terms and sends to the verifier. $\boldsymbol{\Pi}$ stores partial products in the equation (1) at every step and its initial value is $\boldsymbol{1}$.

$$\begin{aligned}
\boldsymbol{t}_{6N+i} &= \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma} \\
\boldsymbol{t}_{7N+i} &= \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\Pi} \\
\boldsymbol{t}_{8N+i} &= \langle \vec{\boldsymbol{b}}_{8N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\Pi}(\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma}) \\
\boldsymbol{\Pi} &= \boldsymbol{\Pi} \cdot (\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma})
\end{aligned}$$

Finally, $\mathcal{P}$ is ready to start proving relations (1), (2) and (3) among RLWE ciphertexts. Remaining part of the protocol looks like other lattice based zero-knowledge protocols. The prover samples k masking vectors $\vec{\boldsymbol{y}}_i$ from discrete Gaussian distribution with standard deviation $\varsigma = 11k\kappa\|\vec{\boldsymbol{r}}\|_2$ where κ is a bound on the l_1 norm of elements in $\mathcal{C}$. Next, k prover commitments $\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0\vec{\boldsymbol{y}}_i$ are calculated and sent to the verifier to get new $(4N+4)k$ challenges $\boldsymbol{\epsilon}_i$. Then, the prover calculates four garbage terms $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ and a final garbage commitment $\boldsymbol{t}_{9N+1}$. In Figure 5 they correspond to long expressions. With a careful inspection, one can see that indeed they are amortized linear and product relations of expanded arguments (1), (2) and (3). At the next step, the prover sends these garbage terms to the verifier and waits for the final challange polynomial $\boldsymbol{c}$. As a last step, rejection sampling is performed on $\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ to make sure that $\vec{\boldsymbol{y}}_i$ hides $\vec{\boldsymbol{r}}$ without leaking any information about commitment randomness. Finally, $\vec{\boldsymbol{z}}_i$ vectors are sent to the verifier. The verifier runs verification equations depicted in Figure 6. The whole protocol is shown in Figure 5.

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\boldsymbol{u}^{(i)}, \boldsymbol{u}_0^{(i)}, \boldsymbol{u}'^{(i)} \in \mathcal{R}_q$		$\boldsymbol{u}^{(i)}, \boldsymbol{u}'^{,(i)}$
$\boldsymbol{v}^{(i)}, \boldsymbol{v}_0^{(i)}, \boldsymbol{v}'^{(i)} \in \mathcal{R}_q$		$\boldsymbol{v}^{(i)}, \boldsymbol{v}'^{,(i)}$
$\kappa = \mu + \lambda + 9N + 1$		
$\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu \times \kappa}$; $\vec{\boldsymbol{b}}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{9N+1} \in \mathcal{R}_q^{\kappa}$		$\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{9N+1}$
$\pi = \text{Perm}(N)$		
$\vec{\boldsymbol{r}} \in \chi_2^{\kappa d}$;		
$\vec{\boldsymbol{t}}_0 = \boldsymbol{B_0}\vec{\boldsymbol{r}}$		
For $i = 1, \ldots N$		
$\boldsymbol{t}_{u_0^{(i)}} = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \boldsymbol{u}_0^{(i)}$		
$\boldsymbol{t}_{v_0^{(i)}} = \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{v}_0^{(i)}$		
$\boldsymbol{t}_{\pi(i)} = \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{r}} \rangle + \pi(i)$		
Shortness proof Σ_1		
	$\xrightarrow{\vec{\boldsymbol{t}}_0, \boldsymbol{t}_{\pi(i)}, \boldsymbol{t}_{u_0^{(i)}}, \boldsymbol{t}_{v_0^{(i)}}, \Sigma_1}$	$\boldsymbol{\alpha} \in \mathcal{C}$
	$\xleftarrow{\boldsymbol{\alpha}}$	
for $i = 1, \ldots N$:		
$\boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}} = \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)}$		
$\boldsymbol{t}_{4N+i} = \langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)} \boldsymbol{u}_0^{(i)}$		
$\boldsymbol{t}_{5N+i} = \langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)} \boldsymbol{v}_0^{(i)}$		
	$\xrightarrow{\boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}}, \boldsymbol{t}_{4N+i}, \boldsymbol{t}_{5N+i}}$	$\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{C}$
	$\xleftarrow{\boldsymbol{\beta}, \boldsymbol{\gamma}}$	
$\boldsymbol{\Pi} = 1$		
for $i = 1, \ldots N$:		
$\boldsymbol{t}_{6N+i} = \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma}$		
$\boldsymbol{t}_{7N+i} = \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\Pi}$		
$\boldsymbol{t}_{8N+i} = \langle \vec{\boldsymbol{b}}_{8N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\Pi}(\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma})$		
$\boldsymbol{\Pi} = \boldsymbol{\Pi} \cdot (\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma})$		
	$\xrightarrow{\boldsymbol{t}_{6N+i}, \boldsymbol{t}_{7N+i}, \boldsymbol{t}_{8N+i},}$	
For $i = 0, \ldots, k-1$:		
$\vec{\boldsymbol{y}}_i \xleftarrow{\$} D_{\varsigma}^{ld}$		
$\vec{\boldsymbol{w}}_i = \boldsymbol{B}_0 \vec{\boldsymbol{y}}_i$		
	$\xrightarrow{\vec{\boldsymbol{w}}_i}$	
	$\xleftarrow{\boldsymbol{\epsilon}}$	$\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_{(4N+4)k} \in \mathcal{R}_q$
$\boldsymbol{v}_1 = \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{iN+j} \sigma^{-i} \Big(\boldsymbol{\beta} \langle \vec{\boldsymbol{b}}_{2N+j}, \vec{\boldsymbol{y}}_i \rangle + \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle - \langle \vec{\boldsymbol{b}}_{6N+j}, \vec{\boldsymbol{y}}_i \rangle \Big)$		
$\boldsymbol{v}_2 = \langle \vec{\boldsymbol{b}}_{9N+1}, \vec{\boldsymbol{y}}_0 \rangle + \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{Nk+iN+j} \sigma^{-i} (\langle \vec{\boldsymbol{b}}_{6N+j}, \vec{\boldsymbol{y}}_i \rangle \langle \vec{\boldsymbol{b}}_{7N+j}, \vec{\boldsymbol{y}}_i \rangle) +$		
$+ \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{2Nk+iN+j} \sigma^{-i} (\langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle \langle \vec{\boldsymbol{b}}_{j}, \vec{\boldsymbol{y}}_i \rangle) +$		
$+ \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{3Nk+iN+j} \sigma^{-i} (\langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle \langle \vec{\boldsymbol{b}}_{N+j}, \vec{\boldsymbol{y}}_i \rangle)$		
$\boldsymbol{t}_{9N+1} = \langle \vec{\boldsymbol{b}}_{9N+1}, \vec{\boldsymbol{r}} \rangle + \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{Nk+iN+j} \sigma^{-i} \big(\langle \vec{\boldsymbol{b}}_{8N+j}, \vec{\boldsymbol{y}}_i \rangle -$		
$- \Pi \langle \vec{\boldsymbol{b}}_{6N+j}, \vec{\boldsymbol{y}}_i \rangle - (\boldsymbol{\beta}\pi(j) + \boldsymbol{\alpha}^{\pi(j)} - \boldsymbol{\gamma}) \langle \vec{\boldsymbol{b}}_{7N+j}, \vec{\boldsymbol{y}}_i \rangle \big) +$		
$+ \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{2Nk+iN+j} \sigma^{-i} \big(\langle \vec{\boldsymbol{b}}_{4N+j}, \vec{\boldsymbol{y}}_i \rangle - \boldsymbol{\alpha}^{\pi(j)} \langle \vec{\boldsymbol{b}}_{j}, \vec{\boldsymbol{y}}_i \rangle - \boldsymbol{u}_0^{(j)} \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle \big) +$		
$+ \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{3Nk+iN+j} \sigma^{-i} \big(\langle \vec{\boldsymbol{b}}_{5N+j}, \vec{\boldsymbol{y}}_i \rangle - \boldsymbol{\alpha}^{\pi(j)} \langle \vec{\boldsymbol{b}}_{N+j}, \vec{\boldsymbol{y}}_i \rangle - \boldsymbol{v}_0^{(j)} \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle \big)$		
$\boldsymbol{v}_3 = \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{4Nk+2i+1} \sigma^{-i} \big(\sum_{j=1}^{N} \boldsymbol{u}^{(j)\prime} \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle - \sum_{j=1}^{N} \langle \vec{\boldsymbol{b}}_{4N+j}, \vec{\boldsymbol{y}}_i \rangle \big) +$		
$+ \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{4Nk+2i+2} \sigma^{-i} \big(\sum_{j=1}^{N} \boldsymbol{v}^{(j)\prime} \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i \rangle - \sum_{j=1}^{N} \langle \vec{\boldsymbol{b}}_{5N+j}, \vec{\boldsymbol{y}}_i \rangle \big)$		
$\boldsymbol{v}_4 = \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{(4N+2)k+i} \sigma^{-i} (\langle \vec{\boldsymbol{b}}_{9N}, \vec{\boldsymbol{y}}_i \rangle) + \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{(4N+3)k+i} \sigma^{-i} (\langle \vec{\boldsymbol{b}}_{7N+1}, \vec{\boldsymbol{y}}_i \rangle)$		
	$\xrightarrow{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{t}_{9N+1}}$	
	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \xleftarrow{\$} C$
For $i = 0, \ldots k-1$:		
$\vec{\boldsymbol{z}}_i = \vec{\boldsymbol{y}}_i + \sigma^i(\boldsymbol{c}) \vec{\mathbf{r}}$		
If Rej $\big((\vec{\boldsymbol{z}}_i), (\sigma^i(\boldsymbol{c}) \vec{\mathbf{r}}), \mathbf{s} \big) = 1$, abort	$\xrightarrow{\vec{\boldsymbol{z}}_i}$	
		Verify

Figure 5. ZK-proof of shuffle

Verify

Verify Shortness proof Σ_1

For $i = 0, \dots k-1$:

$||\vec{\boldsymbol{z}}_i||_2 \overset{?}{\leq} \beta = \mathsf{s}\sqrt{2ld}$

$\boldsymbol{B}_0\vec{\boldsymbol{z}}_i \overset{?}{=} \vec{\boldsymbol{w}}_i + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{t}}_0$

For $j = 1, \dots N$:

$\boldsymbol{f}_i^{u_0^{(j)}} = \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{u_0^{(j)}}$

$\boldsymbol{f}_i^{v_0^{(i)}} = \langle \vec{\boldsymbol{b}}_{N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{v_0^{(j)}}$

$\boldsymbol{f}_i^{\pi(j)} = \langle \vec{\boldsymbol{b}}_{2N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{\pi(j)}$

$\boldsymbol{f}_i^{\boldsymbol{\alpha}^{\pi(j)}} = \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(j)}}$

$\boldsymbol{f}_i^{4N+j} = \langle \vec{\boldsymbol{b}}_{4N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{4N+j}$

$\boldsymbol{f}_i^{5N+j} = \langle \vec{\boldsymbol{b}}_{5N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{5N+j}$

$\boldsymbol{f}_i^{6N+j} = \langle \vec{\boldsymbol{b}}_{6N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{6N+j}$

$\boldsymbol{f}_i^{7N+j} = \langle \vec{\boldsymbol{b}}_{7N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{7N+j}$

$\boldsymbol{f}_i^{8N+j} = \langle \vec{\boldsymbol{b}}_{8N+j}, \vec{\boldsymbol{z}}_i \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{t}_{8N+j}$

$\boldsymbol{f}_{9N+1} = \langle \vec{\boldsymbol{b}}_{9N+1}, \vec{\boldsymbol{z}}_0 \rangle - \boldsymbol{c}\boldsymbol{t}_{9N+1}$

$$\sum_{i=0}^{k-1}\sum_{j=1}^{N} \boldsymbol{\epsilon}_{iN+j}\sigma^{-i}\Big(\boldsymbol{\beta}\boldsymbol{f}_i^{\pi(j)} + \boldsymbol{f}_i^{\boldsymbol{\alpha}^{\pi(j)}} - \boldsymbol{f}_i^{6N+j} + \sigma^i(\boldsymbol{c})\boldsymbol{\gamma}\Big) \overset{?}{=} \boldsymbol{v}_1$$

$$\begin{aligned}\sum_{i=0}^{k-1}\sum_{j=1}^{N} \boldsymbol{\epsilon}_{Nk+iN+j}\sigma^{-i}(\boldsymbol{f}_i^{6N+j}\boldsymbol{f}_i^{7N+j} + \sigma^i(\boldsymbol{c})\boldsymbol{f}_i^{8N+j}) + \\ + \sum_{i=0}^{k-1}\sum_{j=1}^{N} \boldsymbol{\epsilon}_{2Nk+iN+j}\sigma^{-i}(\boldsymbol{f}_i^{\boldsymbol{\alpha}^{\pi(j)}}\boldsymbol{f}_i^{u_0^{(j)}} + \sigma^i(\boldsymbol{c})\boldsymbol{f}_i^{4N+j}) + \\ + \sum_{i=0}^{k-1}\sum_{j=1}^{N} \boldsymbol{\epsilon}_{3Nk+iN+j}\sigma^{-i}(\boldsymbol{f}_i^{\boldsymbol{\alpha}^{\pi(j)}}\boldsymbol{f}_i^{v_0^{(j)}} + \sigma^i(\boldsymbol{c})\boldsymbol{f}_i^{5N+j}) + \boldsymbol{f}_{9N+1} \overset{?}{=} \boldsymbol{v}_2\end{aligned}$$

$$\begin{aligned}&M_1 = \sum_{i=1}^{N} \boldsymbol{\alpha}^i\boldsymbol{u}^{(i)} \quad M_2 = \sum_{i=1}^{N} \boldsymbol{\alpha}^i\boldsymbol{v}^{(i)} \\ &\sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{4Nk+2i+1}\sigma^{-i}\Big(\sum_{j=1}^{N} \boldsymbol{u}^{(j)\prime}\boldsymbol{f}_i^{\boldsymbol{\alpha}^{\pi(j)}} - \sum_{j=1}^{N}\boldsymbol{f}_i^{4N+j} + \sigma^i(\boldsymbol{c})M_1\Big) + \\ &\qquad + \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{4Nk+2i+2}\sigma^{-i}\Big(\sum_{j=1}^{N} \boldsymbol{v}^{(j)\prime}\boldsymbol{f}_i^{\boldsymbol{\alpha}^{\pi(j)}} - \sum_{j=1}^{N}\boldsymbol{f}_i^{5N+j} + \sigma^i(\boldsymbol{c})M_2\Big) \overset{?}{=} \boldsymbol{v}_3\end{aligned}$$

$$\begin{aligned}&\Pi = \prod_{j=1}^{N}(\boldsymbol{\beta}j + \boldsymbol{\alpha}^j - \boldsymbol{\gamma}) \\ &\sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{(4N+2)k+i}\sigma^{-i}(\boldsymbol{f}_i^{9N} + \sigma^i(\boldsymbol{c})\Pi) + \sum_{i=0}^{k-1} \boldsymbol{\epsilon}_{(4N+3)k+i}\sigma^{-i}(\boldsymbol{f}_i^{7N+1} + \sigma^i(\boldsymbol{c})) \overset{?}{=} \boldsymbol{v}_4\end{aligned}$$

Figure 6. Verification equations

Completeness In case of non-aborting transcript due to rejection sampling, the honest verifier $\mathcal{V}$ is convinced with overwhelming probability. Observe that distributions of vectors $\vec{\boldsymbol{z}}_i$ are independent of each other and have statistical distance 2^{-100} to $D_{\mathtt{s}}^{ld}$ due to Rejection Sampling lemma. Therefore by Lemma 2 they are bounded by $\beta_1 = \mathtt{s}\sqrt{2(\lambda+\mu+9N+1)d}$. The remaining four verification equations in Figure 6 regarding $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \boldsymbol{\nu}_3$ and $\boldsymbol{\nu}_4$ are straightforward to verify. Similarly, this reasoning can be applied to shortness proof but with $\beta_2 = \mathtt{s}\sqrt{2(\lambda+\mu+5N+\lambda+3)d}$(See [ENS20] for more detail).

Zero-knowledge Zero-knowledge property of proof of shortness protocol is given in [ENS20]. Indeed, following the same steps, it is possible to simulate this protocol as well. First, sample $\vec{\boldsymbol{z}}_i \leftarrow D_{\mathtt{s}}^{ld}$, which are statistically close to the non-aborting trasncript by Rejection Sampling Lemma. Next, again by the same lemma $\sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}$ are independent of the $\vec{\boldsymbol{z}}_i$ and thus simulator choose $\sigma^i(\boldsymbol{c}) = \boldsymbol{c}_i' \xleftarrow{\$} C$, $\vec{\boldsymbol{r}} \leftarrow \chi_2^{ld}$ like a honest prover. Now, simulator can calculate $\vec{\boldsymbol{w}}_i$ which is uniquely determined by previous variables. Other challenges $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{C}$ are independent of each other, thus they can also be randomly chosen. Straightforwardly, the simulator computes $\vec{\boldsymbol{t}}_0$. The rest of commitments can be uniformly sampled from $\mathcal{R}_q$ as by the MLWE assumption they will be indistinguishable from real MLWE samples. Finally, remaining equations of $\boldsymbol{\nu}_i$ are deterministic functions of $\vec{\boldsymbol{t}}, \vec{\boldsymbol{z}}_i$ and $\sigma^i(\boldsymbol{c})$.

Soundness Soundness relation for proof of shortness protocol is described in detail in [ENS20] which is similar to the proof for protocol in Figure 5. Consider the extractor given in [ALS20] which can extract weak openings after rewinding the protocol l/k times and get $\vec{\boldsymbol{r}}^\star$ and $\vec{\boldsymbol{y}}^\star$ or finds $\mathsf{MSIS}_{8d\beta_1}$ solution for $\boldsymbol{B}_0$. It can also extract messages simply from commitment relations.

$$
\begin{aligned}
\boldsymbol{t}_{u_0^{(i)}} &= \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_0^{(i)\star} \\
\boldsymbol{t}_{v_0^{(i)}} &= \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_1^{(i)\star} \\
\boldsymbol{t}_{\pi(i)} &= \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_2^{(i)\star}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}} &= \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{r}}\star \rangle + \boldsymbol{m}_3^{(i)\star} \\
\boldsymbol{t}_{4N+i} &= \langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{r}}^{\star} \rangle + \boldsymbol{m}_4^{(i)\star} \\
\boldsymbol{t}_{5N+i} &= \langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{r}}^{\star} \rangle + \boldsymbol{m}_5^{(i)\star} \\
\boldsymbol{t}_{6N+i} &= \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{r}}^{\star} \rangle + \boldsymbol{m}_6^{(i)\star} \\
\boldsymbol{t}_{7N+i} &= \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{r}}^{\star} \rangle + \boldsymbol{m}_7^{(i)\star} \\
\boldsymbol{t}_{8N+i} &= \langle \vec{\boldsymbol{b}}_{8N+i}, \vec{\boldsymbol{r}}^{\star} \rangle + \boldsymbol{m}_8^{(i)\star} \\
\boldsymbol{t}_{9N+1} &= \langle \vec{\boldsymbol{b}}_{9N+1}, \vec{\boldsymbol{r}}^{\star} \rangle + \boldsymbol{m}_9^{\star}
\end{aligned}
$$

Setting $\vec{\boldsymbol{z}}_i^{\star} = \vec{\boldsymbol{y}}_i^{\star} + \sigma^i(\boldsymbol{c})\vec{\boldsymbol{r}}^{\star}$, masked openings are defined below.

$$
\begin{aligned}
\boldsymbol{f}_i^{u_0^{(j)}} &= \langle \vec{\boldsymbol{b}}_j, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_0^{(j)\star} \\
\boldsymbol{f}_i^{v_0^{(j)}} &= \langle \vec{\boldsymbol{b}}_{N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_1^{(j)\star} \\
\boldsymbol{f}_i^{\pi(j)} &= \langle \vec{\boldsymbol{b}}_{2N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_2^{(j)\star} \\
\boldsymbol{f}_i^{\alpha_0^{\pi(j)}} &= \langle \vec{\boldsymbol{b}}_{3N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \boldsymbol{c}\boldsymbol{m}_3^{(j)\star} \\
\boldsymbol{f}_i^{4N+j} &= \langle \vec{\boldsymbol{b}}_{4N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_4^{(j)\star} \\
\boldsymbol{f}_i^{5N+j} &= \langle \vec{\boldsymbol{b}}_{5N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_5^{(j)\star} \\
\boldsymbol{f}_i^{6N+j} &= \langle \vec{\boldsymbol{b}}_{6N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_6^{(j)\star} \\
\boldsymbol{f}_i^{7N+j} &= \langle \vec{\boldsymbol{b}}_{7N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_7^{(j)\star} \\
\boldsymbol{f}_i^{8N+j} &= \langle \vec{\boldsymbol{b}}_{8N+j}, \vec{\boldsymbol{y}}_i^{\star} \rangle - \sigma^i(\boldsymbol{c})\boldsymbol{m}_8^{(j)\star} \\
\boldsymbol{f}^{9N+1} &= \langle \vec{\boldsymbol{b}}_{9N+1}, \vec{\boldsymbol{y}}_0^{\star} \rangle - \boldsymbol{c}\boldsymbol{m}_9^{(j)\star}
\end{aligned}
$$

Now, substituting those terms to their respective places in verification equations and doing algebraic simplifications, verification equations for $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ and $\boldsymbol{v}_4$ yields

$$
\boldsymbol{c} \sum_{i=0}^{k-1} \sum_{j=1}^{N} \boldsymbol{\epsilon}_{iN+j} \sigma^{-i} \Big(\boldsymbol{\beta}\boldsymbol{m}_2^{(j)\star} + \boldsymbol{m}_3^{(j)\star} - \boldsymbol{m}_6^{(j)\star} + \boldsymbol{\gamma} \Big) = 0
$$

$$\boldsymbol{c}^2\Big(\sum_{i=0}^{k-1}\sum_{j=1}^{N}\boldsymbol{\epsilon}_{Nk+iN+j}\sigma^{-i}(\boldsymbol{m}_6^{(j)\star}\boldsymbol{m}_7^{(j)\star}-\boldsymbol{m}_8^{(j)\star})+$$
$$+\sum_{i=0}^{k-1}\sum_{j=1}^{N}\boldsymbol{\epsilon}_{2Nk+iN+j}\sigma^{-i}(\boldsymbol{m}_0^{(j)\star}\boldsymbol{m}_3^{(j)\star}-\boldsymbol{m}_4^{(j)\star})+$$
$$+\sum_{i=0}^{k-1}\sum_{j=1}^{N}\boldsymbol{\epsilon}_{3Nk+iN+j}\sigma^{-i}(\boldsymbol{m}_1^{(j)\star}\boldsymbol{m}_3^{(j)\star}-\boldsymbol{m}_5^{(j)\star})\Big)=0$$

$$\boldsymbol{c}\Big(\sum_{i=0}^{k-1}\boldsymbol{\epsilon}_{4Nk+2i+1}\sigma^{-i}\big(M_1-\sum_{j=1}^{N}\boldsymbol{u}^{(j)\prime}\boldsymbol{m}_3^{(j)\star}-\sum_{j=1}^{N}\boldsymbol{m}_4^{(j)\star}\big)+$$
$$+\sum_{i=0}^{k-1}\boldsymbol{\epsilon}_{4Nk+2i+2}\sigma^{-i}\big(M_2-\sum_{j=1}^{N}\boldsymbol{v}^{(j)\prime}\boldsymbol{m}_3^{(j)\star}-\sum_{j=1}^{N}\boldsymbol{m}_5^{(j)\star}\big)\Big)=0$$

$$\boldsymbol{c}\Big(\sum_{i=0}^{k-1}\boldsymbol{\epsilon}_{(4N+2)k+i}\sigma^{-i}(\Pi-\boldsymbol{m}_8^{(N)\star})+\sum_{i=0}^{k-1}\boldsymbol{\epsilon}_{(4N+3)k+i}\sigma^{-i}(\mathbf{1}-\boldsymbol{m}_7^{(1)\star})\Big)=0$$

In [ALS20, Theorem 5.1], the cheating probability of similar arguments are proven to be bounded $\epsilon < (3p)^k$. That means, if $\boldsymbol{\beta}\boldsymbol{m}_2^{(j)\star}+\boldsymbol{m}_3^{(j)\star}-\boldsymbol{m}_6^{(j)\star}+\boldsymbol{\gamma}\neq 0$ for some j, then the probability of above equation being true is bounded by $(3p)^k$. Similarly, with the same probability bound, we get $\boldsymbol{m}_0^{(j)\star}\boldsymbol{m}_3^{(j)\star}-\boldsymbol{m}_4^{(j)\star}\neq 0$; $\boldsymbol{m}_1^{(j)\star}\boldsymbol{m}_3^{(j)\star}-\boldsymbol{m}_5^{(j)\star}\neq 0$ and $\boldsymbol{m}_6^{(j)\star}\boldsymbol{m}_7^{(j)\star}-\boldsymbol{m}_8^{(j)\star}\neq 0$ altogether, or $\sum_{j=1}^{N}\boldsymbol{u}^{(j)\prime}\boldsymbol{m}_3^{(j)\star}-\sum_{j=1}^{N}\boldsymbol{m}_4^{(j)\star}-M_1\neq 0$ and $\sum_{j=1}^{N}\boldsymbol{v}^{(j)\prime}\boldsymbol{m}_3^{(j)\star}-\sum_{j=1}^{N}\boldsymbol{m}_5^{(j)\star}-M_2\neq 0$; or $\boldsymbol{m}_8^{(N)\star}-\Pi\neq 0$.

Combining all extracted relations we re-establish

$$\prod_j^N(\boldsymbol{\beta}\boldsymbol{m}_2^{(j)\star}+\boldsymbol{m}_3^{(j)\star}-\boldsymbol{\gamma})=\Pi=\prod_j^N(\boldsymbol{\beta}j+\boldsymbol{\alpha}^j-\boldsymbol{\gamma})$$
$$\sum_j^N\boldsymbol{m}_3^{(j)\star}(\boldsymbol{u}^{(j)\prime}-\boldsymbol{m}_0^{(j)\star})=M_1=\sum_{i=1}^{N}\boldsymbol{\alpha}^i\boldsymbol{u}^{(i)}$$
$$\sum_j^N\boldsymbol{m}_3^{(j)\star}(\boldsymbol{v}^{(j)\prime}-\boldsymbol{m}_1^{(j)\star})=M_2=\sum_{i=1}^{N}\boldsymbol{\alpha}^i\boldsymbol{v}^{(i)}$$

Mix-Node Security Once more, we refer to [CMM19] where mix-node security is proved using game based approach. Indeed, by following exactly the same steps, and only replacing statistical closeness of Game 0 and Game 1 with computational closeness under $\mathsf{MLWE}_{8d\beta_2}$ assumption guaranteeing shortness error terms in RLWE encryptions, it is possible to show that the advantage of adversary over random guessing is bounded:

$$\epsilon = \mathbf{Adv}^{sec}_{\mathcal{A}}(\kappa) \leq \epsilon_{MLWE} + 2^{-100} + \epsilon_{RLWE} \ .$$

3.4 Non-interactivity and proof size

The protocol in Figure 5 is made non-interactive with the help of standard Fiat-Shamir technique. In other words, challenges are computed by the prover by hashing all previous messages and public information. Furthermore, instead of sending $\vec{\boldsymbol{w}}_i, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ which are used as input to the hash function to generate challenges, the standard technique is to send hash output and let the verifier recompute those values from verification equations and check that hashes of computed input terms match with the prover's hash. Thus, it is enough to send the commitment $\vec{\boldsymbol{t}}_0 \| \boldsymbol{t}_1 \| \cdots \| \boldsymbol{t}_{9N}$, garbage term $\boldsymbol{t}_{9N+1}$ and vectors $\vec{\boldsymbol{z}}_i$. A polynomial in $\mathcal{R}_q$ consists of d coefficients less than q, so it takes $d\lfloor \log q \rfloor$ bits at most. $\vec{\boldsymbol{t}}_0$ and $\vec{\boldsymbol{z}}_i$ for $i = 1, \ldots, k$ consist of μ and $\lambda + \mu + 9N + 1$ polynomials, respectively. A formulate to calculate the full cost of shortness proof is given in [ENS20]. Combining all of these, the size of accepting trasncript for our protocol is

$$\begin{aligned}
&(\mu + 9N + 1)d\lfloor \log q \rfloor + k(\lambda + \mu + 9N + 1)d\lfloor \log q \rfloor + 256 + \\
&+ (\lambda + \mu + 5N + 4)d\lfloor \log q \rfloor + k(2\lambda + \mu + 5N + 3)d\lfloor \log q \rfloor + 256 = \\
&= 14N(k+1)d\lfloor \log q \rfloor + (k(3\lambda + 2\mu + 4) + \lambda + 2\mu + 5)d\lfloor \log q \rfloor + 512
\end{aligned}$$

Overall, size of proof of shuffle protocol is linearly dependent on the number of ciphertexts (i.e. votes in the voting scenario). However, the number of public variables, such as commitment keys, is increasing quadratically. A possible optimization method is to choose common shared seed and derive all public polynomials using that seed.

Another possible place for optimization is to choose public variables in a specific format such as $\boldsymbol{B}_0 = [\mathbf{I}_\mu | \boldsymbol{B}'_0]$ where $\boldsymbol{B}'_0 \in \mathcal{R}_q^{\mu \times (\lambda + 9N + 1)}$ and vectors $\vec{\boldsymbol{b}}_i = \vec{\mathbf{0}}_\mu \| \vec{\boldsymbol{e}}_i \| \vec{\boldsymbol{b}}'_i$ where $\vec{\boldsymbol{e}}_i$ is the i-th standard basis vector of length $9N+1$ and $\vec{\boldsymbol{b}}'_i \in \mathcal{R}_q^\lambda$ as suggested in [LNS20], so that total number of uniform polynomials will be linear in N. (This optimization is already taken into account in the size of shortness proof transcript while constructing shortness proof)

3.5 Instantiation

Here we propose a valid parameter set for proof of a shuffle protocol. Parameters have to instantiated in a way that the protocol achieves 128 bit classical soundness and post-quantum encryption security of RLWE is at least that much. For Module SIS security,

$8d\beta_1 < q$ and $8d\beta_2 < q$. Coefficients of secret key and error terms used in RLWE encryption are sampled uniformly in $\{-1, 0, 1\}$, i.e $\chi_1 = \mathcal{U}(\{-1, 0, 1\}^d)$. Similarly, distribution C and χ_2 are defined on the same set: $Pr(x = 1) = Pr(x = -1)$ and $Pr(x = 0) = 1/2$ in C and $Pr(x = 0) = 6/16$ in χ_2. We find that for $q \approx 2^{32}$, mixing node is secure up to 10 voters which is insufficient. For this reason and in order to easily represent coefficients with primary data types, we choose $q \approx 2^{64}$. Then, using LWE and SIS security estimator script[3] we get that for $d = 4096, \lambda = 1, \mu = 1$ Hermite factor for $\mathsf{MLWE}_{\lambda,\chi_2}$ with ternary noise is 1.0027 and $\mathsf{MSIS}_{8d\beta}$ has root Hermite factor 1.0027. Finally, by Lemma 3 in [ALS20], $p \approx -62.$, which implies that $k = 2$ is enough for the desired soundness level.

3.6 Performance and Security

We estimate performance of proof of shuffle protocol in terms of expensive operations. Sampling challenges uniformly random from $\mathcal{C}$ or χ_1 is not very hard but from discrete Gaussian distribution with large deviation is time-consuming. To solve this, we suggest using constant time Gaussian sampler by Zhao *et al.* [ZSS20]. Then, the only expensive operation is polynomial multiplication in $\mathcal{R}_q$. When the ring is fully splitting, multiplication can be handled in NTT domain in linear steps. In Figure 5 we see that the protocol uses $O(N^2)$ multiplication operations due to $18N$ inner products between vectors of length $\lambda + \mu + 9N + 1$. However, applying optimization trick in Section 3.4, this dependency becomes linear in N. Because complexity of polynomial multiplication depends only ring structure, it can be assumed to be constant. Thus, the time complexity of the protocol becomes linear in the number of voters.

Post-quantum security of Fiat-Shamir transform has not been fully proven in quantum random oracle model (QROM) yet. Several works on this researh area restricted definitions for security properties. For example, computionally binding commitment schemes can be insecure against quantum attacks, as shown in [ARU14]. Collapse binding is a stronger security property which allows to construct quantum argument of knowledge [Unr16]. The BDLOP commitment scheme used in our protocol has not been shown to satisfy collapse-binding property. But because SIS hash functions are collapse-binding [LZ19], hopefully one can prove for Module-SIS based BDLOP commitments as well. Another main challenge is to prove security of mutli-round Fiat-Shamir [DFM20] in QROM. Until these problems are solved, unfortunately, we cannot claim full post-quantum security of non-interactive protocol described in Section 3.4. An alternative solution is Unruh transform [Unr15], but applying it will result undesirably large protocol size.

However, the interactive protocol in Figure 5 will be *potentially* post-quantum secure. In the online voting context, election auditors can be assumed to be honest verifiers. They

[3] `https://github.com/pq-crystals/security-estimates`

can be restricted to have access to powerful quantum device during mixing procedure in order to prevent them obtain secret permutation vector. After successfully verified mixing phase is over, RLWE ciphertexts can be publicly shared at no risk due to post-quantum security level of chosen parameters.

While the protocol given in Figure 5 proves arguments (1), (2) and (3) with negligible soundness error for carefully chosen parameters such as in Section 3.5, arguments themselves are not sound in case of fully splitting rings. The main reason is that regardless of the construction, challenge sets for $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ will not be sufficiently large. First, there are at most q pairwise different polynomials whose difference is invertible. Therefore, the probability bound in Lemma 3 will not be negligible. Even relaxing invertibility requirement does not help substantially. However, repeating the whole protocol several times can be a way to overcome this problem.

4 Implementation results

To show the protocol in action, a small, yet fully functional, console-based voting mock-up application is built using C++ language. There are no separate server or client applications. This allows us to run experiments quickly for thousands of voters in parallel. However, the basic software development principles are adopted for further developments, allowing us to create independent services that have already been isolated by their roles (voter/client, Election Set-Up, *Ballot box, Mix Net, Decryption oracle, Tallier*). The source code is provided alongside the thesis and available on demand[4].

The application is developed on top of several helpful libraries. NFLLib [MBG+16] is used for faster calculations on lattices, which includes NTT transformations. This library has other prerequisites (cmake, GMP, and Mpfr) as well. Then, `fips202.h` and `fips202.h` files contains public domain implementation of SHAKE extendable-output functions and SHA-3 hash functions from FIPS 202[5]. We used SHAKE256 as a hash function in the Fiat-Shamir transform. As NFLlib is already SIMD optimized, we did not employ advanced vectorization techniques explicitly. However, the fast discrete Gaussian sampler in that library takes a long time and occupies lots of memory in the initialization phase as it is building a look-up table with a very large standard deviation. Instead, we used constant-time discrete Gaussian sampler from recent work [ZSS20]. During compilation, `-O3` and `-march=native` flags are set.

All cryptographic components have been implemented from scratch. These include RLWE encryption scheme [LPR13], commitment scheme and zero-knowledge proof of opening and linear relations [BDL+18], and zero-knowledge proof of knowledge of short exact solution to unstructured ternary equation in $\mathbb{Z}_q$ [ENS20].

In *Set-up phase*, the application initializes services mentioned above and generates a public and secret RLWE encryption key pair. For simplicity, the secret key is not divided into secret shares among multiple trustees. The public key is reachable in the whole application context by any function, whereas the secret key is visible to *Mix Net* (hence, mix nodes) and *Decryption oracle* only. After this phase is finished, the central election controller declares election is open meaning *Ballot Box* has started to accept encrypted ballots. *Voter* can make use of the public key to privately encrypt the personal choice of candidate, and send the ciphertext to the *Ballot Box*. Again for simplicity and due to they are out of the context of this work, digital signatures are omitted. During experiments, thousands of voters are instantiated, made a random choice, and cast votes in parallel. For correctness and debugging purposes, these random choices are logged. Later, central election controller closes election, i.e *Ballot Box* is not accepting ballots anymore.

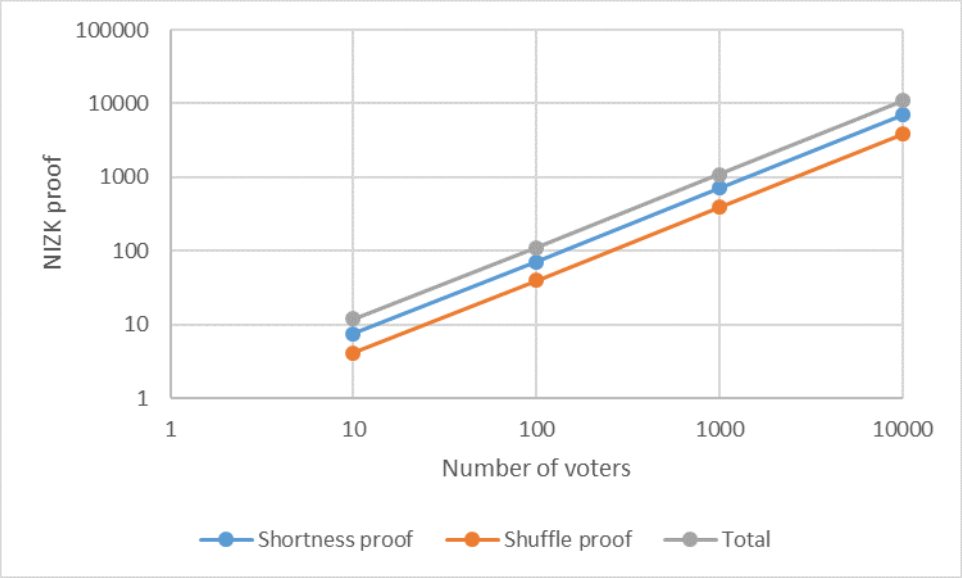
Figure 7 and 8 gives information on the running time of shuffling, proof of shuffle, and verification of proof for the varied number of voters. As it is obvious from the figure,

[4]Interested people send an email to `valeh.farzaliyev@ut.ee`

[5]`https://github.com/PQClean/PQClean/blob/master/common/fips202.h`

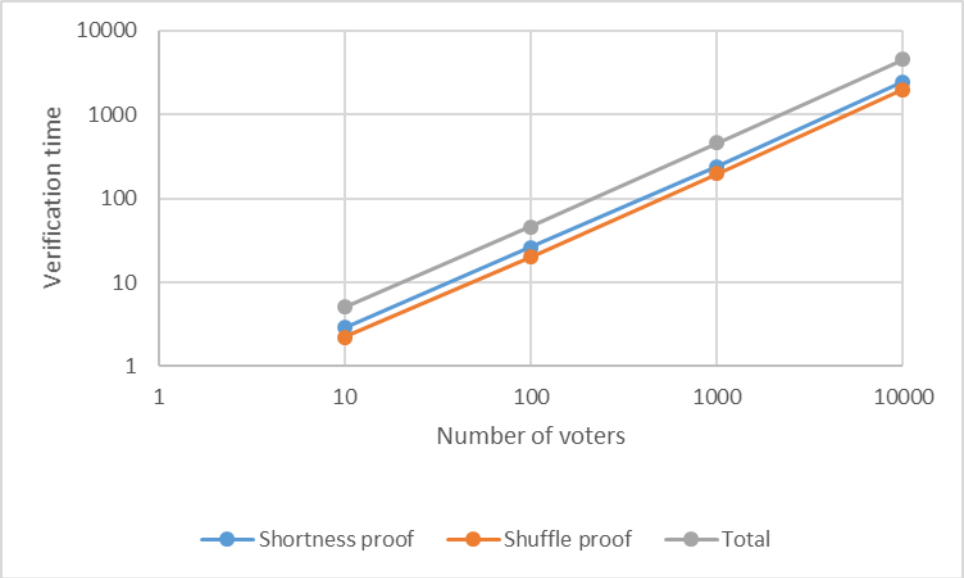
the linear relationship is observed, which is expected. Similarly, the time to generate proof of shuffle is linearly increasing with the number of mix nodes for a fixed number of voters.

All experiments have been performed on a 2.2GHz Intel Haswell CPU with 64GBs of RAM. NFLlib uses AVX2 or SSE optimization techniques if any of them exists in CPU architecture. For the CPU model I used, it supports the AVX2 instruction set.



N	Shortness proof (seconds)	Shuffle proof (seconds)	Total	Size
10	7.5	4.1	12	13.875MB
100	71	40	111	132MB
1000	708	394	1090	1.28GB
10000	7000	3879	10859	12.8GB

Figure 7. Proof generation time



N	Shortness proof (seconds)	Shuffle proof (seconds)	Total
10	2.9	2.2	5.1
100	26	20	46
1000	240	199	458
10000	2450	1989	4579

Figure 8. Proof verification time

5 Post-Quantum Voting Scheme

We have successfully shown how to design a practical zero-knowledge proof of shuffle for Costa *et al.* mixing node using post-quantum secure primitives only. However, it has reduced soundness and lacks concrete security proof in QROM. Assuming these problems have been solved, we briefly sketch a way for a lattice-based post-quantum secure voting scheme. For simplicity, we consider what should be changed in an internet voting platform, called IVXV, used in Estonian elections.

IVXV is a verifiable shuffling based voting protocol in which threshold ElGamal encryption is used to encrypt digital ballots. Before elections are started, encryption keys are generated and distributed among several electoral authorities. Voters also append their digital signatures to encrypted ballots using RSA signatures or Elliptic Curve Digital Signature Algorithm. Later, once ballots are stored in the read-only bulletin board, they are shuffled and proof of correct shuffle is generated using Verificatum software. After several rounds of shuffling, fresh ciphertexts are decrypted only in the presence of at least half of the key shares and zero-knowledge proof of correct decryption is produced. External auditors can verify published zero-knowledge proofs to ensure that there was no fraudulent activity. Besides, in order to reduce coercion risks, a mobile ballot verification app is built for voters. In case a voter is suspected that his or her choice is not cast as intended, they can scan a special QR code displayed right after finishing voting. The QR code contains vote id and randomness used in ElGamal encryption. Thus, without a need for a secret key, the mobile app can recover plaintext from encrypted ballot stored in the bulletin board. The detailed security aspects of IVXV platform are well-explained in the official website.[6]

The post-quantum alternative of IVXV is possible using lattice-based cryptography. Moreover, we suggest using RLWE encryption scheme instead of ElGamal for three reasons. First, similar to ElGamal, it is IND-CPA secure encryption scheme with relatively short key and ciphertext size compared to other post-quantum secure encryption methods. The second reason is that using RLWE allows us to utilize other lattice-based tools even if they are based on different assumptions (such as MLWE and MSIS). Finally, the mobile verification procedure need not be changed at all - sending encryption randomness is enough to recover encrypted message with overwhelming probability. Observe that for a RLWE ciphertext $(\boldsymbol{u}, \boldsymbol{v})$

$$\boldsymbol{v} = pk.\boldsymbol{b} \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \lfloor \frac{q}{2} \rceil \boldsymbol{m} \implies \lfloor \frac{q}{2} \rceil \boldsymbol{m} = \boldsymbol{v} - pk.\boldsymbol{b} \cdot \boldsymbol{r} - \boldsymbol{e}_2$$

Because, $\boldsymbol{e}_2$ is a small noise term, $\boldsymbol{m}$ can be recovered from the difference $\boldsymbol{v} - pk.\boldsymbol{b} \cdot \boldsymbol{r}$.

Choosing a post-quantum secure digital signature is not as hard as choosing other primitives. The reason is that the digital signature can be realized independently and

[6] https://www.valimised.ee/sites/default/files/uploads/eng/IVXV-UK-1.0-eng.pdf

need not be lattice-based. For example, one would use any round 3 finalists of NIST PQC standardization contest[7].

Threshold LWE is still an active research area. The suitable protocol would be a work by Damgård *et al.* in which they attempted to generate multiparty computation protocol secure against active adversaries based on RLWE assumption [DPSZ12, Appendix D]. The final piece is distributed decryption which is also given in [DPSZ12].

This is a general idea of how ivxv voting protocol can be made quantum resistant. Of course, all components have to be modified in a way that they can work together. The hardest part would be finding a concrete parameter set with provable strong security properties.

[7] https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions

6 Conclusion

While the digitalization trend continues, it had already faced the biggest threat by quantum computers. These computers are potentially very powerful so that they can break almost all modern asymmetric cryptography used in digital banking, secure browsing, communications, and even online voting. Fortunately, many people are working on post-quantum secure alternatives. Lattice-based cryptography is the most promising candidate to replace current standards.

In this thesis, we have presented a practical zero-knowledge proof of shuffle for lattice-based mixing networks suitable for medium-scale elections. Our protocol is built on techniques from recent academic works on lattice-based zero-knowledge proofs including boosting soundness via Galois automorphisms. The resulting scheme has linear memory cost and time complexity. The average run time per vote is one second.

The highly parallelizable nature of lattice operations allows for faster implementations using many CPU and/or GPU cores concurrently. Although our simple conceptual implementation needs several days to verify the validity of a mixing node in case of million voters, this can be significantly reduced down to possibly comparable to manual tallying with the help of parallel programming approaches, such as OpenMP SIMD [FFLM20] and GPU [DS15]. This is a performance optimization task and most likely will be solved in near future considering the uprising trend of parallel computing.

We have also described why fully splitting rings are not a good choice for compact proofs. However, partially splitting rings require extra effort while proving security relations and software implementations. Lattice-based zero-knowledge proofs constructed over partially splitting rings recently gained attention. It is an interesting research question whether new techniques can be integrated to our proof of shuffle protocol to remove the soundness barrier and we let it open for future work.

References

[Ajt98] Miklós Ajtai. The shortest vector problem in L_2 is *NP*-hard for randomized reductions (extended abstract). In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 10–19. ACM, 1998.

[ALS20] Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical Product Proofs for Lattice Commitments. In Daniele Micciancio and Thomas Ristenpart, editors, *Proceedings of CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 470–499. Springer, 2020.

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 474–483. IEEE Computer Society, 2014.

[BDL+18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More Efficient Commitments from Structured Lattice Assumptions. In Dario Catalano and Roberto De Prisco, editors, *Proceedings of SCN 2018*, volume 11035 of *LNCS*, pages 368–385. Springer, 2018.

[BHM20] Xavier Boyen, Thomas Haines, and Johannes Mueller. A verifiable and practical lattice-based decryption mix net with external auditing. Cryptology ePrint Archive, Report 2020/115, 2020. `https://eprint.iacr.org/2020/115`.

[BKLP15] Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *Proceedings ESORICS 2015 Part I*, volume 9326 of *LNCS*, pages 305–325. Springer, 2015.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[CMM17] Núria Costa, Ramiro Martínez, and Paz Morillo. Proof of a Shuffle for Lattice-Based Cryptography. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevicius, editors, *Proceedings of NordSec 2017*, volume 10674 of *LNCS*, pages 280–296. Springer, 2017.

[CMM19] Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-Based Proof of a Shuffle. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B.

Rønne, and Massimiliano Sala, editors, *Proceedings of Financial Cryptography and Data Security - FC 2019*, volume 11599 of *LNCS*, pages 330–346. Springer, 2019.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 602–631. Springer, 2020.

[dPLNS17] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical Quantum-Safe Voting from Lattices. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of ACM CCS 2017*, pages 1565–1581. ACM, 2017.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[DS15] Wei Dai and Berk Sunar. cuHE: A Homomorphic Encryption Accelerator Library. In Enes Pasalic and Lars R. Knudsen, editors, *Proceedings of BalkanCryptSec 2015*, volume 9540 of *LNCS*, pages 169–186. Springer, 2015.

[ENS20] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings. *IACR Cryptol. ePrint Arch.*, 2020:518, 2020.

[FFLM20] Pierre Fortin, Ambroise Fleury, François Lemaire, and Michael Monagan. High performance SIMD modular arithmetic for polynomial evaluation. working paper or preprint, April 2020.

[GE19] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits, 2019.

[Ger20] Micha German. Making votes count with internet voting. *Political Behavior*, 2020.

[GMSS99] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Pierre Seifert. Approximating shortest lattice vectors is not harder than approximating closest lattice vectors. *Inf. Process. Lett.*, 71(2):55–61, 1999.

[HM20] Thomas Haines and Johannes Mueller. Sok: Techniques for verifiable mix nets. Cryptology ePrint Archive, Report 2020/490, 2020. `https://eprint.iacr.org/2020/490`.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, Dec 1982.

[LNS20] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical Lattice-Based Zero-Knowledge Proofs for Integer Relations. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *Proceedings of ACM CCS 2020 0*, pages 1051–1070. ACM, 2020.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. *J. ACM*, 60(6):43:1–43:35, 2013.

[LS18] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Proceedings of EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 204–224. Springer, 2018.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Proceedings of EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 326–355. Springer, 2019.

[MBG+16] Carlos Aguilar Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrède Lepoint. Nfllib: Ntt-based fast lattice library. In Kazue Sako, editor, *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, volume 9610 of *Lecture Notes in Computer Science*, pages 341–356. Springer, 2016.

[PR05] Chris Peikert and Alon Rosen. Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices. *Electron. Colloquium Comput. Complex.*, (158), 2005.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[SE94] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1):181–199, Aug 1994.

[Sho99] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.*, 41(2):303–332, 1999.

[Str18] Martin Strand. A Verifiable Shuffle for the GSW Cryptosystem. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *Financial Cryptography and Data Security 2018, Revised Selected Papers*, volume 10958 of *LNCS*, pages 165–180. Springer, 2018.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 755–784. Springer, 2015.

[Unr16] Dominique Unruh. Computationally binding quantum commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 497–527. Springer, 2016.

[Wil17] Jan Willemson. Bits or paper: Which should get to carry your vote? In *Electronic Voting. Second International Joint Conference, E-Vote-ID 2017, Bregenz, Austria, October 24-27, 2017, Proceedings*, Security and Cryptology, pages 292–305. Springer International Publishing, 2017.

[Woj93] Banaszczyk Wojciech. New bounds in some transference theorems in the geometry of numbers. volume 296, pages 625–635, 1993.

[ZSS20] Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. COSAC: COmpact and Scalable Arbitrary-Centered Discrete Gaussian Sampling over Integers. In Jintai Ding and Jean-Pierre Tillich, editors, *Proceedings of PQCrypto 2020*, volume 12100 of *LNCS*, pages 284–303. Springer, 2020.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Valeh Farzaliyev**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Towards Practical Post-Quantum Voting Protocol:Shorter Exact Lattice-Based Proof of a Shuffle,
   *(*title of thesis)

   supervised by Jan Willemson and Dominique Peer Ghislain Dr Unruh.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Valeh Farzaliyev
14/01/2021