Received: 22 January 2022 | Revised: 12 July 2022 | Accepted: 28 July 2022

IET Information Security

DOI: 10.1049/ise2.12089

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Improved lattice-based mix-nets for electronic voting

Valeh Farzaliyev[1,2,3] | Jan Willemson[1,2] | Jaan Kristjan Kaasik[1,3]

[1]Cybernetica AS, Tartu, Estonia

[2]STACC OÜ, Tartu, Estonia

[3]Tartu University, Tartu, Estonia

Correspondence

Jan Willemson, Cybernetica AS, Narva mnt 20, Tartu 51009, Estonia.
Email: jan.willemson@gmail.com

Funding information

European Regional Development Fund, Grant/Award Number: EU48684; Estonian Research Council, Grant/Award Number: PRG920

Abstract

Mix-networks were first proposed by Chaum in the late 1970s–early 1980s as a general tool for building anonymous communication systems. Classical mix-net implementations rely on standard public key primitives (e.g., ElGamal encryption) that will become vulnerable when a sufficiently powerful quantum computer will be built. Thus, there is a need to develop quantum-resistant mix-nets. This article focuses on the application case of electronic voting where the number of votes to be mixed may reach hundreds of thousands or even millions. We propose an improved architecture for lattice-based post-quantum mix-nets featuring more efficient zero-knowledge proofs while maintaining established security assumptions. Our current implementation scales up to 100,000 votes, still leaving a lot of room for future optimisation.

KEYWORDS

electronic voting, implementation, lattice-based post-quantum cryptography, mix-nets, zero-knowledge proofs

1 | INTRODUCTION

Voting is the main mechanism of public opinion polling utilised, for example, in the context of general elections. Traditionally, voting has happened in a controlled location (polling station) to ease electoral management and reduce potential fraud.

However, by the beginning of the 21st century, peoplehave become more mobile than ever before, so taking all the electorate into one place for a short period of time has become increasingly challenging. This challenge has been amplified by the recent COVID-19 outburst that has brought along the need to avoid gathering people in small spaces.

Thus, the need for the methods of remote voting has increased significantly. For example, during the 2020 U.S. presidential elections, more than 65 million votes were sent in by post. Even though there seems to be little evidence of direct fraud, the extent of postal voting still caused a lot of controversy and discussion.

Indeed, the unreliability of postal services may raise questions about what to do with late votes, voter identification of postal votes is not particularly strong, and due to voting in an uncontrolled environment, it is hard to guarantee voting privacy and coercion resistance.

Such problems motivate a search for alternatives, with remote electronic (Internet) voting being one of the prime candidates.

The votes stored on and transmitted via digital media are, contrary to paper votes, not directly perceivable by humans. Thus, the central problem of remote electronic voting is the independent verifiability of all the actions. In this paper, we are going to focus on a particular method of ensuring verifiability of the central voting system, since this is potentially the most critical point of failure.

What makes central server-side verification challenging is the need to also maintain the privacy of the votes. There are two main approaches used to implement privacy-preserving verifiable electronic voting systems—homomorphic tallying and mixing the votes before decryption [1]. There are a number of implementations known for both of these approaches, typically relying on some form of homomorphic encryption, for example, the Paillier or the ElGamal scheme [2].

However, the classical asymmetric algorithms used in these implementations are known to become weak due to Shor's algorithm once a sufficiently capable quantum computer will be built [3]. Thus, looking for post-quantum alternatives is a necessity.

In recent years, both post-quantum homomorphic tallying [4, 5] and mixing [6–8] have been studied. Mix-networks were first proposed by Chaum in the late 1970s–early 1980s [9] as a general tool for building anonymous communication systems. In this paper, we will concentrate on quantum-resistant mix-nets, aiming at improving their efficiency in terms of the number of votes they are able to shuffle in a given time period.

As the most expensive part of a cryptographic mix-net is the generation and verification of zero-knowledge proofs of correct operation, we concentrate on improving these proofs. Technically, we build upon the recently proposed protocol by Costa *et al.* [9], applying amortisation techniques for linear and product relations described in the LANES framework [10–12] and using a commitment scheme by Baum *et al.* [13].

As a result, we design a purely lattice-based zero-knowledge proof of a shuffle for a lattice-based mixing scheme that can be scaled up to about 100,000 votes. We instantiate the protocol with specific parameters such that the protocol achieves 128-bit soundness and 180-bit post-quantum encryption security level. Finally, we provide a proof-of-concept implementation of the proposed scheme and benchmark its practical performance.

The structure of this paper is as follows. In Section 2, we specify notation and Preliminaries used in the construction of the protocol and its security proof. The protocol itself is presented in Section 3. Implementation and experimental results are presented in Section 4.3. Finally, Section 5 draws some conclusions and sets directions for future work. Details of the proofs can be found in the Appendices.

This is an extended version of paper that appeared at the 24th Annual International Conference on Information Security and Cryptology. Compared to the conference version, the zero-knowledge proof of a shuffle protocol has been modified by appending an argument that the extracted permutation vector is indeed a vector of integers. The authors would like to thank members of the academic community for pointing to this shortcoming. Additionally, further analysis revealed that some intermediate terms are committed to twice in the protocol. Removing those redundant commitments resulted in shorter proof size, increasing the protocol efficiency especially for larger number of voters. Also, a number of typos have been fixed and wording has been improved in many places to improve readability of the current journal version of the paper.

2 | PRELIMINARIES

2.1 | Notation

For a prime q, let $\mathbb{Z}_q$ be the ring of integers modulo q, with its elements considered in the interval $\left[-\frac{q-1}{2}, \frac{q-1}{2}\right]$, and let $\mathbb{Z}_n^\times$ denote the group of invertible elements modulo n. $\lfloor x \rceil$ represents the closest integer to x in $\mathbb{Z}_q$. Vectors over $\mathbb{Z}_q$ are denoted as $\overrightarrow{v} \in \mathbb{Z}_q^m$ and matrices over $\mathbb{Z}_q$ are denoted by regular capital letters (e.g. A) unless explicitly stated otherwise. 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 $\overrightarrow{\boldsymbol{b}}$. Matrices over $\mathcal{R}$ or $\mathcal{R}_q$ are bold upper-case letters, for example, $\boldsymbol{B}$. By default, all vectors and their concatenations are column vectors. More precisely, an 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 $a_i \in \mathbb{Z}_q$. Especially for ring $\mathcal{R}_q$, the same element can be represented as a matrix in $\mathbb{Z}_q$ when it is a multiplicand:

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

l_2 and l_∞ norms are defined as usual:

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

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

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

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

2.2 | Splitting rings

In this work, we set $q - 1 \equiv 2l \bmod 4l$, so that $X^d + 1$ splits into l irreducible polynomials of degree d/l, that is,

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

where ζ is primitive $2l$th root of unity in $\mathbb{Z}_q$ and $\boldsymbol{\varphi}_{\boldsymbol{i}} = X^{d/l} - \zeta^{2i-1}$. 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 $l = d$.

The Number Theoretic Transform (NTT) of a polynomial $\boldsymbol{p} \in \mathcal{R}_q$ is defined as

$$\mathbf{NTT}(\boldsymbol{p}) = \begin{bmatrix} \hat{p}_0 \\ \vdots \\ \hat{p}_{l-1} \end{bmatrix} \text{ where } \hat{\boldsymbol{p}}_{i-1} = \boldsymbol{p} \bmod \boldsymbol{\varphi}_i.$$

By Chinese Remainder Theorem, there exists a unique inverse transformation—Inverse NTT—such that **INTT** (**NTT**($\boldsymbol{p}$)) = $\boldsymbol{p}$. Also, NTT allows the computing of the product of two polynomials faster and saves time in other operations.

$$ab = \mathbf{INTT}(\mathbf{NTT}(a) \circ \mathbf{NTT}(b))$$
$$\mathbf{NTT}(a + b) = \mathbf{NTT}(a) + \mathbf{NTT}(b)$$

Here $\circ$ is the component-wise multiplication operation.

2.3 | Ring-LWE encryption, module SIS/LWE

In our constructions, we will rely on hardness of Ring Learning With Errors (Ring-LWE, RLWE) [14] and Module Learning With Errors (Module-LWE, MLWE) or Module Short Integer Solution (Module-SIS, MSIS) [15, 16] problems.

Definition 1 [RLWE$_\chi$] In the decisional Ring-LWE problem with an error distribution χ over $\mathcal{R}$, the probabilistic polynomial time (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. Ring-LWE assumption is that search-RLWE and/or decisional-RLWE problem is hard for any probabilistic polynomial time adversaries.

We implement the encryption scheme described in ref. [14]. Let χ_1 be error distribution over $\mathcal{R}$ where each coefficient is sampled from $\{-1, 0, 1\}$.

- *KeyGen*: 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 = (pk.$\boldsymbol{a}$, pk.$\boldsymbol{b}$) = ($\boldsymbol{a}$, $\boldsymbol{a}\cdot\boldsymbol{s}$ + $\boldsymbol{e}$) and private key as $\boldsymbol{s}$.
- *Encryption*: 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 are used. Then the ciphertext is a pair of polynomials ($\boldsymbol{u}$, $\boldsymbol{v}$) such that

$$\boldsymbol{u} = pk.\boldsymbol{a} \cdot \boldsymbol{r} + \boldsymbol{e}_1,$$

$$\boldsymbol{v} = pk.\boldsymbol{b} \cdot \boldsymbol{r} + \boldsymbol{e}_2 + \left\lfloor \frac{q}{2} \right\rfloor \boldsymbol{z}.$$

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

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

If each coefficient of the resulting polynomial is close to 0, set the respective coefficient of the 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 are indistinguishable from uniformly random elements. Thus, unless one can solve the decisional-RLWE problem, all ciphertexts look uniform and no information can be extracted about the plaintext.

Definition 2 [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 $(\boldsymbol{A}, \overrightarrow{\boldsymbol{t}}) \xleftarrow{\$} \mathcal{R}_q^{m\times n} \times \mathcal{R}_q^m$ from $(\boldsymbol{A}, \boldsymbol{A}\overrightarrow{\boldsymbol{s}} + \overrightarrow{\boldsymbol{e}})$ for $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{m\times n}$, a secret vector $\overrightarrow{\boldsymbol{s}} \leftarrow \chi^n$ and an error vector $\overrightarrow{\boldsymbol{e}} \leftarrow \chi^m$.

Definition 3 [MSIS$_{m,n,\beta}$] The goal in the Module-SIS problem with parameters $n, m > 0$ and $0 < \beta < q$ is to find $\overrightarrow{\boldsymbol{x}} \in \mathcal{R}_q^m$ for a given matrix $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$ such that $\boldsymbol{A}\overrightarrow{\boldsymbol{x}} = \overrightarrow{\boldsymbol{0}}$ mod q and $0 < \|\overrightarrow{\boldsymbol{x}}\|_\infty < \beta$.

In practical security estimations, the parameter m in Definitions 2 and 3 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 respectively.

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 invertibility in this ring and the number of residue fields it splits into [17, 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 between 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 invertibility 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 ref. [10, 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, in ref. [10] it is estimated that $p = 2^{-31.44}$ 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 between[†] any two challenges $\overline{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c}'$ is non-invertible with probability at most $p^{d/l}$.

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 ref. [10, Lemma 3.3], it is shown that if $\boldsymbol{c} \leftarrow \mathcal{C}$, the distribution of coefficients of $\mathbf{c}$ 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, in ref. [10] it is estimated that $p = 2^{-31.44}$ for $l = d = 128$, $q \approx 2^{32}$. An element $\boldsymbol{c}$ in splitting ring $\mathcal{R}_q$ is non-invertible when $\mathbf{c}$ mod $\boldsymbol{\varphi}_{\mathrm{i}} = 0$ for any $i = 1, \ldots, l$. Then the difference between any two challenges $\overline{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{c}'$ is non-invertible with probability at most $p^{d/l}$.

2.5 | Error distribution and rejection sampling

Security of RLWE and MLWE problems depends on the error distribution. The original security proofs [14, 15] assumed the errors from discrete spherical Gaussian distribution. However, in literature we can find different choices such as centred binomial distribution [18, 19] or uniform distribution in a small interval [20]. We use the former for sampling randomness in MLWE and the latter for randomness and error terms in RLWE.

Rejection sampling. It is a common practice to hide secret commitment randomness $\overrightarrow{\boldsymbol{r}} \in \mathcal{R}_q^{\kappa}$ in another vector $\overrightarrow{\boldsymbol{z}}$ without leaking any information about $\overrightarrow{\boldsymbol{r}}$. For this purpose, we use uniform rejection sampling technique from ref. [19]. In the protocol, the prover samples a 'masking' vector $\overrightarrow{\boldsymbol{y}}$ using uniform distribution in $[-\delta + 1, \delta]$. Upon receiving the challenge $\boldsymbol{c} \xleftarrow{\$} \mathcal{C}$ by the verifier, the prover responds with $\overrightarrow{\boldsymbol{z}} = \overrightarrow{\boldsymbol{y}} + \boldsymbol{c}\overrightarrow{\boldsymbol{r}}$. The dependency of $\overrightarrow{\boldsymbol{z}}$ on $\overrightarrow{\boldsymbol{r}}$ is removed if $\|\overrightarrow{\boldsymbol{z}}\|_\infty < \delta - \beta$ where $\|\boldsymbol{c}\overrightarrow{\boldsymbol{r}}\|_\infty \leq \beta$. Otherwise, the prover rejects the masked vector and aborts the protocol to start over again.

The expected number of repetitions M required by rejection sampling can be estimated by

$$1/M = \left(\frac{2(\delta - \beta) - 1}{2\delta - 1}\right)^{\kappa d} \approx e^{-\kappa d \beta/\delta}.$$

For more details see ref. [19]. The parameter δ is typically chosen so that the expected value of M is small (say, 2 or 3).

2.6 | Commitment scheme

In this work, we will be using a variant of BDLOP commitment scheme [13]. Let, $\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu\times(\mu+\lambda+1)}$, $\overrightarrow{\boldsymbol{b}}_1 \in \mathcal{R}_q^{\mu+\lambda+1}$ and $\overrightarrow{\boldsymbol{r}} \leftarrow \chi_2^{(\mu+\lambda+1)d}$. The commitment of a single message $\boldsymbol{m} \in \mathcal{R}_q$ is a pair $(\overrightarrow{\boldsymbol{t}}_0, \boldsymbol{t}_1)$ where

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

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

Definition 4 A weak opening for the commitment $\overrightarrow{\boldsymbol{t}} = \overrightarrow{\boldsymbol{t}}_0 \| \boldsymbol{t}_1$ consists of l polynomials $\overline{\mathbf{c}}_i \in \mathcal{R}_q$, randomness vector $\overrightarrow{\boldsymbol{r}}^\star$ over $\mathcal{R}_q$ and a message $\boldsymbol{m}^\star \in \mathcal{R}_q$ such that

$$\|\overline{\mathbf{c}}_i\|_1 \leq 2d \text{ and } \overline{\mathbf{c}}_i \bmod \boldsymbol{\varphi}_i \neq 0 \text{ for all } 1 \leq i \leq l,$$
$$\|\overline{\mathbf{c}}_i\overrightarrow{\boldsymbol{r}}^\star\|_\infty \leq 2\beta \text{ for all } 1 \leq i \leq l,$$
$$\boldsymbol{B}_0\overrightarrow{\boldsymbol{r}}^\star = \overrightarrow{\boldsymbol{t}}_0,$$
$$\langle \overrightarrow{\boldsymbol{b}}_1, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}^\star = \boldsymbol{t}_1.$$

The BDLOP commitment scheme is proven to be binding also with respect to the weak opening in ref. [10, Lemma 4.3].

2.7 | Generalised Schwartz-Zippel lemma

The generalised Schwartz-Zippel lemma is stated as follows [9, Appendix A].

Lemma 1 *Let p ∈ R[x_1, x_2, …, x_n] be a non-zero polynomial of total degree d ≥ 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, ..., r_n) = 0]* ≤ d/|S|.

In general, it is not trivial to construct the set S. A polynomial in $\mathcal{R}_q$ is a zero divisor when at least one of its NTT coefficients is zero. Thus, the difference between two elements is not a divisor of zero when they do not have a 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 do not have common NTT coefficients and construct the set S. We fix this set to be $\mathcal{S} = \left\{\boldsymbol{f} \in \mathcal{R}_q \mid \deg \boldsymbol{f} < d/l\right\}$.

2.8 | Mix-node security

Costa et al. [9] proposed a stronger 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 $z^{\pi(j_\mathcal{A})}$ be the message before and after running the algorithm.

Definition 5 Let J be a uniform random variable taking values in [1, …, 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, ∀c, $\exists\kappa_0$ s.t if $\kappa > \kappa_0$:

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

3 | IMPROVED MIX-NODE

Our proof of shuffle protocol is based on Costa et al.'s work [9]. Assume that there are N RLWE ciphertexts $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ encrypted with public key $(\boldsymbol{pk.a}, \boldsymbol{pk.b})$ to be shuffled. A mixing node will generate secret random zero encryption ciphertexts $(\boldsymbol{u}_{i,0}, \boldsymbol{v}_{i,0})$ and permutation $\boldsymbol{\pi}$, and output $\left(\boldsymbol{u}_i', \boldsymbol{v}_i'\right)$ such that

$$\left(\boldsymbol{u}_{i,0}, \boldsymbol{v}_{i,0}\right) = \left(pk.\boldsymbol{a} \cdot \boldsymbol{r}_{E,i} + \boldsymbol{e}_{u,i}, pk.\boldsymbol{b} \cdot \boldsymbol{r}_{E,i} + \boldsymbol{e}_{v,i} + 0\right)$$
$$\left(\boldsymbol{u}'_i, \boldsymbol{v}'_i\right) = \left(\boldsymbol{u}_{\pi(i)} + \boldsymbol{u}_{i,0}, \boldsymbol{v}_{\pi(i)} + \boldsymbol{v}_{i,0}\right)$$

where $\boldsymbol{r}_{E,i}$, $\boldsymbol{e}_{u,i}$, $\boldsymbol{e}_{v,i} \leftarrow \chi_1$ for all $i = 1, \ldots, N$. We extend the proof in ref. [9] for any splitting rings in Appendix 7 to show that if $\boldsymbol{\pi}$ is a valid permutation, then for any $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{S}$ the equation

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

holds due to generalised Schwartz-Zippel lemma with small cheating probability. Furthermore,

$$\sum_{i=1}^{N} \boldsymbol{\alpha}^i \boldsymbol{u}_i = \sum_{i=1}^{N} \boldsymbol{\alpha}^{\pi(i)} \left(\boldsymbol{u}'_i - \boldsymbol{u}_{i,0}\right), \tag{2}$$

$$\sum_{i=1}^{N} \boldsymbol{\alpha}^i \boldsymbol{v}_i = \sum_{i=1}^{N} \boldsymbol{\alpha}^{\pi(i)} \left(\boldsymbol{v}'_i - \boldsymbol{v}_{i,0}\right). \tag{3}$$

One can think of (2) and (3) as two polynomials with coefficients in $\mathcal{R}_q$ evaluated at the same point $\boldsymbol{\alpha}$. Again, due to generalised 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 ref. [9] to be enough to argue for the correctness of a shuffle.

The protocol in ref. [9] uses a commitment scheme from ref. [21] to prove the aforementioned arguments mainly due to the existence of zero-knowledge proofs for linear and multiplicative relations for the commitment scheme. We recap 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 amortised 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 $\boldsymbol{\pi}$ and sends the commitment to the verifier again. Committing to permutation vector means committing to $\boldsymbol{\pi}(1)$, …, $\boldsymbol{\pi}(\mathrm{N})$. Then, $\mathcal{V}$ samples a polynomial $\boldsymbol{\alpha}$ from the challenge set and sends it back to the prover. Following to that, $\mathcal{P}$ calculates commitments to $\boldsymbol{\alpha}^{\pi(1)}$, …, $\boldsymbol{\alpha}^{\pi(\mathrm{N})}$. 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 ref. (1) using the committed values. Next, again by the 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. [9] mention that it is possible to use amortisation techniques described in ref. [6] to 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 the performance and concrete security level of the protocol.

We solve both issues by replacing the commitment scheme with a variant of the Module SIS/LWE based commitment scheme from ref. [13]. This allows us to use more efficient zero-knowledge arguments for proving linear and product relations between committed messages [10, 12]. Those protocols are short, efficient and have no extra cost when amortised over many relations. Besides, there is no need to repeat the protocol several times to get desired soundness properties. Nevertheless, as we change the mathematical setting, there is a need for additional careful analysis of security.

For example, another change we introduce is regarding challenge sets. Previously, prime modulus q was required to satisfy $q \equiv 3 \bmod 8$, which implies that the ring $\mathcal{R}_q$ splits only into two residue fields. This condition is required to define a concrete sufficiently large set of challenge polynomials of which any of the differences between two elements in this set is invertible. Now, we relax this restriction and allow q to split into more than 2 residue fields. In general, if the ring $\mathcal{R}_q$ splits into l levels, one can construct such a set with cardinality $q^{d/1}$ which can be sufficiently large for some values of q, d and l.

Now we proceed to describe our protocol.

First, let μ and λ be rank of secure MSIS and MLWE instances, respectively, $q - 1 \equiv 2l \bmod 4l$ be such that $\mathcal{R}_q$ is a partially splitting ring and $\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu \times (\mu+\lambda+\eta+8N+2)}$, $\overrightarrow{\boldsymbol{b}}_1, \overrightarrow{\boldsymbol{b}}_2, \ldots$ $\overrightarrow{\boldsymbol{b}}_{8N+\eta+2} \in \mathcal{R}_q^{\mu+\lambda+\eta+8N+2}$. Furthermore, set $q^{d/1} \approx 2^{256}$ and $\beta'_i = \delta_i - \beta_i - 1$ for $i = 1, 2$.

Theorem 1 *The protocol in Figure 1 is statistically complete, computationally honest verifier zero-knowledge under the Module-LWE assumption, computationally special-sound under the Module-SIS assumption and is a computationally secure mix-node under* RLWE_{χ_1} *and* $\mathrm{MSIS}_{\mu,8d\beta'_2}$ *assumptions. That is, if p is the maximum probability over* $\mathbb{Z}_q$ *of the coefficients of* $\mathbf{c} \bmod X^{d/l} - \zeta$*, then*

- *For completeness, in case of non-aborting transcript due to rejection sampling, the honest verifier* $\mathcal{V}$ *is always convinced.*
- *For zero-knowledge, there exists a simulator* Sim *that, without access to secret information, outputs a simulation of accepting the transcript of the protocol. Any adversary capable of distinguishing an actual transcript from a simulated one with an advantage* ϵ also has an advantage ϵ *in distinguishing* $MLWE_{\lambda,\chi_2}$ within the same running time.
- *For soundness, there is an extractor* $\mathcal{E}$ *with rewindable black-box access to a deterministic prover* $\mathcal{P}^\star$ *that convinces* $\mathcal{V}$ *with probability* $\epsilon \geq (6\mathrm{p})^{d/1} + q^{-\eta}$, either outputting a weak opening for commitment

$$\overrightarrow{\boldsymbol{t}} = \overrightarrow{\boldsymbol{t}}_0 \| \boldsymbol{t}_{u_0^{(i)}} \| \boldsymbol{t}_{v_0^{(i)}} \| \boldsymbol{t}_{\boldsymbol{\pi}(i)} \| \boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}} \| \boldsymbol{t}_{4N+1} \| \ldots \| \boldsymbol{t}_{8N+2} \| \boldsymbol{t}_{g_1} \| \ldots \| \boldsymbol{t}_{g_\eta}$$

such that extracted messages satisfy equations (1), (2) and (3), or being able to solve $MSIS_{\mu,8d\beta'_1}$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\boldsymbol{u}_i, \boldsymbol{u}_{i,0}, \boldsymbol{u}'_i \in \mathcal{R}_q$		$\boldsymbol{u}_i, \boldsymbol{u}'_i$
$\boldsymbol{v}_i, \boldsymbol{v}_{i,0}, \boldsymbol{v}'_i \in \mathcal{R}_q$		$\boldsymbol{v}_i, \boldsymbol{v}'_i$
$\kappa = \mu + \lambda + \eta + 8N + 2$		
$\boldsymbol{B}_0 \in \mathcal{R}_q^{\mu \times \kappa};\ \vec{\boldsymbol{b}}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{8N+\eta+2} \in \mathcal{R}_q^{\kappa}$		$\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \ldots, \vec{\boldsymbol{b}}_{8N+\eta+2}$
$\pi = \mathrm{Perm}(N)$		
$\vec{\boldsymbol{r}} \in \chi_2^{\kappa d};$		
$\vec{\boldsymbol{t}}_0 = \boldsymbol{B_0} \vec{\boldsymbol{r}}$		
For $i = 1, \ldots, N$		
$\quad \boldsymbol{t}_{u_{i,0}} = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \boldsymbol{u}_{i,0}$		
$\quad \boldsymbol{t}_{v_{i,0}} = \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{v}_{i,0}$		
$\quad \boldsymbol{t}_{\pi(i)} = \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{r}} \rangle + \pi(i)$		
Shortness proof Σ_1		
	$\xrightarrow{\vec{\boldsymbol{t}}_0, \boldsymbol{t}_{\pi(i)}, \boldsymbol{t}_{u_{i,0}}, \boldsymbol{t}_{v_{i,0}}, \Sigma_1}$	$\boldsymbol{\alpha} \in \mathcal{S}$
	$\xleftarrow{\boldsymbol{\alpha}}$	
For $i = 1, \ldots, N$:		
$\quad \boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}} = \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)}$		
$\quad \boldsymbol{t}_{4N+i} = \langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)} \boldsymbol{u}_{i,0}$		
$\quad \boldsymbol{t}_{5N+i} = \langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\alpha}^{\pi(i)} \boldsymbol{v}_{i,0}$		
	$\xrightarrow{\boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}}, \boldsymbol{t}_{4N+i}, \boldsymbol{t}_{5N+i}}$	$\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{S}$
	$\xleftarrow{\boldsymbol{\beta}, \boldsymbol{\gamma}}$	
$\boldsymbol{\Pi}_1 = 1$		
for $i = 1, \ldots, N$:		
$\quad \boldsymbol{t}_{6N+i} = \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma}$		
$\quad \boldsymbol{t}_{7N+i} = \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\Pi}_i$		
$\quad \boldsymbol{\Pi}_{i+1} = \boldsymbol{\Pi}_i \cdot (\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \boldsymbol{\gamma})$		
$\boldsymbol{t}_{8N+1} = \langle \vec{\boldsymbol{b}}_{8N+1}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{\Pi}_{N+1}$		
	$\xrightarrow{\boldsymbol{t}_{6N+i}, \boldsymbol{t}_{7N+i}, \boldsymbol{t}_{8N+1}}$	
$\vec{\boldsymbol{g}} \overset{\$}{\leftarrow} \mathcal{S}^{\eta}$		
For $j = 1, \ldots, \eta$		
$\quad \boldsymbol{t}_{g_j} = \langle \vec{\boldsymbol{b}}_{8N+2+j}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}_j$	$\xrightarrow{\boldsymbol{t}_{g+1}, \boldsymbol{t}_{g_2}, \ldots, \boldsymbol{t}_{g_\eta}}$	$\boldsymbol{\theta}_{11}, \ldots, \boldsymbol{\theta}_{\eta N} \overset{\$}{\leftarrow} \mathbb{Z}_q$
	$\xleftarrow{\boldsymbol{\theta}_{11}, \ldots, \boldsymbol{\theta}_{\eta N}}$	
For $j = 1, \ldots, \eta$:		
$\quad \boldsymbol{h}_j = \sum_{i=1}^{N} \boldsymbol{\theta}_{ji} \pi(i) + \boldsymbol{g}_j$	$\xrightarrow{\vec{\boldsymbol{h}}}$	
$\vec{\boldsymbol{y}} \overset{\$}{\leftarrow} [-\delta_1 + 1, \delta_1]^{\kappa d}$		
$\vec{\boldsymbol{w}} = \boldsymbol{B}_0 \vec{\boldsymbol{y}}$		
	$\xrightarrow{\vec{\boldsymbol{w}}}$	
	$\xleftarrow{\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_{(4N+4)}}$	$\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2, \ldots, \boldsymbol{\epsilon}_{(4N+4)} \in \mathcal{R}_q$
$\boldsymbol{v}_1 = \sum_{i=1}^{N} \boldsymbol{\epsilon}_i \Big(\boldsymbol{\beta} \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{y}} \rangle + \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle - \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{y}} \rangle \Big)$		
$\boldsymbol{v}_2 = \langle \vec{\boldsymbol{b}}_{8N+2}, \vec{\boldsymbol{y}} \rangle + \sum_{i=1}^{N} \boldsymbol{\epsilon}_{N+i} (\langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{y}} \rangle \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{y}} \rangle) +$		
$+ \sum_{i=1}^{N} \boldsymbol{\epsilon}_{2N+i} (\langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle) +$		
$+ \sum_{i=1}^{N} \boldsymbol{\epsilon}_{3N+i} (\langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{y}} \rangle)$		
$\boldsymbol{t}_{8N+2} = \langle \vec{\boldsymbol{b}}_{8N+2}, \vec{\boldsymbol{r}} \rangle + \sum_{i=1}^{N} \boldsymbol{\epsilon}_{N+i} \big(\langle \vec{\boldsymbol{b}}_{7N+i+1}, \vec{\boldsymbol{y}} \rangle -$		
$- \boldsymbol{\Pi}_i \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{y}} \rangle - (\boldsymbol{\beta}\pi(i) + \boldsymbol{\alpha}^{\pi(i)} - \gamma) \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{y}} \rangle \big) +$		
$+ \sum_{i=1}^{N} \boldsymbol{\epsilon}_{2N+i} \big(\langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{\alpha}^{\pi(i)} \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle - \boldsymbol{u}_{i,0} \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle \big) +$		
$+ \sum_{i=1}^{N} \boldsymbol{\epsilon}_{3N+i} \big(\langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{\alpha}^{\pi(i)} \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{y}} \rangle - \boldsymbol{v}_{i,0} \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle \big)$		
$\boldsymbol{v}_3 = \boldsymbol{\epsilon}_{4N+1} \big(\sum_{i=1}^{N} \boldsymbol{u}'_i \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle - \sum_{i=1}^{N} \langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{y}} \rangle \big) +$		
$\quad + \boldsymbol{\epsilon}_{4N+2} \big(\sum_{i=1}^{N} \boldsymbol{v}'_i \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{y}} \rangle - \sum_{i=1}^{N} \langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{y}} \rangle \big)$		
$\boldsymbol{v}_4 = \boldsymbol{\epsilon}_{4N+3} (\langle \vec{\boldsymbol{b}}_{9N}, \vec{\boldsymbol{y}} \rangle) + \boldsymbol{\epsilon}_{4N+4} (\langle \vec{\boldsymbol{b}}_{7N+1}, \vec{\boldsymbol{y}} \rangle)$		
	$\xrightarrow{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{4+\eta}, \boldsymbol{t}_{8N+2}}$	
For $j = 1, \ldots, \eta$		
$\quad \boldsymbol{v}_{4+j} = \sum_{i=1}^{N} \boldsymbol{\theta}_{ji} \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{y}} \rangle + \langle \vec{\boldsymbol{b}}_{8N+2+j}, \vec{\boldsymbol{y}} \rangle$	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \overset{\$}{\leftarrow} C$
$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\, \vec{\mathbf{r}}$		
If $\lVert \vec{\boldsymbol{z}} \rVert_\infty \geq \delta_1 - \beta_1,$ abort	$\xrightarrow{\vec{\boldsymbol{z}}}$	**Verify**

FIGURE 1 ZK-proof of shuffle.

- *And finally, an adversary with an advantage ϵ over random guessing has also an advantage over MSIS$_{\mu,8d\beta'_2}$ and/or RLWE$_{\chi_1}$ problems with probability at least ϵ.*

Proof. Completeness. Observe that in a non-aborting transcript vector $\vec{\boldsymbol{z}}$ is bounded by $\delta_1 - \beta_1$. The remaining verification equations in Figure 2 regarding $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, …, $\boldsymbol{v}_{4+\eta}$ are

```
Verify
________________________________________
Verify Shortness proof $\Sigma_1$

$||\vec{\boldsymbol{z}}||_\infty \overset{?}{<} \delta_1 - \beta_1$
$\boldsymbol{B}_0\vec{\boldsymbol{z}} \overset{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\,\vec{\boldsymbol{t}}_0$

For $i = 1, \ldots N$ :
    $\boldsymbol{f}^{u_0^{(i)}} = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{u_{i,0}}$
    $\boldsymbol{f}^{v_0^{(i)}} = \langle \vec{\boldsymbol{b}}_{N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{v_{i,0}}$
    $\boldsymbol{f}^{\pi(i)} = \langle \vec{\boldsymbol{b}}_{2N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{\pi(i)}$
    $\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}} = \langle \vec{\boldsymbol{b}}_{3N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{\boldsymbol{\alpha}^{\pi(i)}}$
    $\boldsymbol{f}^{4N+i} = \langle \vec{\boldsymbol{b}}_{4N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{4N+i}$
    $\boldsymbol{f}^{5N+i} = \langle \vec{\boldsymbol{b}}_{5N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{5N+i}$
    $\boldsymbol{f}^{6N+i} = \langle \vec{\boldsymbol{b}}_{6N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{6N+i}$
    $\boldsymbol{f}^{7N+i} = \langle \vec{\boldsymbol{b}}_{7N+i}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{7N+i}$
$\boldsymbol{f}_{8N+1} = \langle \vec{\boldsymbol{b}}_{8N+1}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{9N+1}$
$\boldsymbol{f}_{8N+2} = \langle \vec{\boldsymbol{b}}_{8N+2}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{8N+2}$
 For $j = 1, \ldots, \eta$
    $\boldsymbol{f}_{g_j} = \langle \vec{\boldsymbol{b}}_{8N+2+j}, \vec{\boldsymbol{z}} \rangle - \boldsymbol{ct}_{g_j}$

$\sum_{i=1}^N \boldsymbol{\epsilon}_i\left(\boldsymbol{\beta f}^{\pi(i)} + \boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}} - \boldsymbol{f}^{6N+i} + \boldsymbol{c\gamma}\right) \overset{?}{=} \boldsymbol{v}_1$

$\sum_{i=1}^N \boldsymbol{\epsilon}_{N+i}(\boldsymbol{f}^{6N+i}\boldsymbol{f}^{7N+i} + \boldsymbol{cf}^{7N+i+1})+$
        $+\sum_{i=1}^N \boldsymbol{\epsilon}_{2N+i}(\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}}\boldsymbol{f}^{u_0^{(i)}} + \boldsymbol{cf}^{4N+i})+$
        $+\sum_{i=1}^N \boldsymbol{\epsilon}_{3N+i}(\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}}\boldsymbol{f}^{v_0^{(i)}} + \boldsymbol{cf}^{5N+i}) + \boldsymbol{f}_{8N+2} \overset{?}{=} \boldsymbol{v}_2$

$M_1 = \sum_{i=1}^N \boldsymbol{\alpha}^i\boldsymbol{u}_i \quad M_2 = \sum_{i=1}^N \boldsymbol{\alpha}^i\boldsymbol{v}_i$
$\boldsymbol{\epsilon}_{4N+1}\big(\sum_{i=1}^N \boldsymbol{u}'_i\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}} - \sum_{i=1}^N\boldsymbol{f}^{4N+i} + \boldsymbol{c}M_1\big)+$
        $+\boldsymbol{\epsilon}_{4N+2}\big(\sum_{i=1}^N \boldsymbol{v}'_i\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}} - \sum_{i=1}^N\boldsymbol{f}^{5N+i} + \boldsymbol{c}M_2\big) \overset{?}{=} \boldsymbol{v}_3$

$\Pi = \prod_{i=1}^N(\boldsymbol{\beta}i + \boldsymbol{\alpha}^i - \boldsymbol{\gamma})$
$\boldsymbol{\epsilon}_{4N+3}(\boldsymbol{f}^{8N+1} + \boldsymbol{c}\Pi) + \boldsymbol{\epsilon}_{4N+4}(\boldsymbol{f}^{7N+1} + \boldsymbol{c}) \overset{?}{=} \boldsymbol{v}_4$

 For $j = 1, \ldots, \eta$
    Check $\deg \boldsymbol{h}_j \leq d/l - 1$
    $\sum_{i=1}^N \boldsymbol{\theta}_{ji}\boldsymbol{f}^{\pi(i)} + \boldsymbol{f}_{g_j} + \boldsymbol{ch}_j \overset{?}{=} \boldsymbol{v}_{4+j}$
```

FIGURE 2 Verification equations.

straightforward to verify. Similarly, proof of shortness protocol is complete.

Zero-knowledge. Zero-knowledge property of proof of shortness protocol is given in ref. [12]. Indeed, following the same steps, it is possible to simulate this protocol as well. First, sample $\vec{\boldsymbol{z}} \xleftarrow{\$} \left[-\left(\delta_1 - \beta_1\right) + 1, \delta_1 - \beta_1 - 1\right]^{\kappa d}$, which is the distribution of $\vec{\boldsymbol{z}}$ in non-aborting transcript. Next, due to rejection sampling step, $\boldsymbol{c}\vec{\boldsymbol{r}}$ is independent of $\vec{\boldsymbol{z}}$ and thus the simulator chooses $\boldsymbol{c} \xleftarrow{\$} C$ like an honest verifier. Now, the simulator can calculate $\vec{\boldsymbol{w}}$ which is uniquely determined by previous variables. In honest transcript, each polynomial in $\vec{\boldsymbol{h}}$ is in $\mathcal{S}$, hence the simulator samples η polynomials randomly from that set. Other challenges $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\theta}_{ji} \in \mathcal{S}$ 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 $\boldsymbol{v}_i$ are deterministic functions of $\vec{\boldsymbol{t}}_i$, $\vec{\boldsymbol{z}}$, $\vec{\boldsymbol{h}}$ and $\boldsymbol{c}$.

Soundness. The soundness relation for proof of shortness protocol is described in detail in ref. [12] and is similar to the proof for a protocol in Figure 1. Consider the extractor given in ref. [10] which can extract weak openings after rewinding the protocol l 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.

$$
\begin{aligned}
&\text{For } i = 1, \ldots, N\\
&\qquad \boldsymbol{t}_{u_{i,0}} = \langle \overrightarrow{\boldsymbol{b}}_i, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_0^{(i)\star}\\
&\qquad \boldsymbol{t}_{v_{i,0}} = \langle \overrightarrow{\boldsymbol{b}}_{N+i}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_1^{(i)\star}\\
&\qquad \boldsymbol{t}_{\pi(i)} = \langle \overrightarrow{\boldsymbol{b}}_{2N+i}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_2^{(i)\star}\\
&\qquad \boldsymbol{t}_{\boldsymbol{\alpha}^{\pi(i)}} = \langle \overrightarrow{\boldsymbol{b}}_{3N+i}, \overrightarrow{\boldsymbol{r}}\ \star \rangle + \boldsymbol{m}_3^{(i)\star}\\
&\qquad \boldsymbol{t}_{4N+i} = \langle \overrightarrow{\boldsymbol{b}}_{4N+i}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_4^{(i)\star}\\
&\qquad \boldsymbol{t}_{5N+i} = \langle \overrightarrow{\boldsymbol{b}}_{5N+i}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_5^{(i)\star}\\
&\qquad \boldsymbol{t}_{6N+i} = \langle \overrightarrow{\boldsymbol{b}}_{6N+i}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_6^{(i)\star}\\
&\qquad \boldsymbol{t}_{7N+i} = \langle \overrightarrow{\boldsymbol{b}}_{7N+i}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_7^{(i)\star}\\
&\boldsymbol{t}_{8N+1} = \langle \overrightarrow{\boldsymbol{b}}_{8N+1}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_{8N+1}^\star\\
&\boldsymbol{t}_{8N+2} = \langle \overrightarrow{\boldsymbol{b}}_{8N+2}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{m}_9^\star\\
&\text{For } j = 1, \ldots, \eta\\
&\qquad \boldsymbol{t}_{\boldsymbol{g}_j} = \langle \overrightarrow{\boldsymbol{b}}_{8N+2+j}, \overrightarrow{\boldsymbol{r}}^\star \rangle + \boldsymbol{g}_j^\star
\end{aligned}
$$

Setting $\overrightarrow{\boldsymbol{z}}^\star = \overrightarrow{\boldsymbol{y}}^\star + \boldsymbol{c}\overrightarrow{\boldsymbol{r}}^\star$, masked openings are defined below.

$$
\begin{aligned}
&\text{For } i = 1, \ldots, N\\
&\qquad \boldsymbol{f}^{u_{i,0}} = \langle \overrightarrow{\boldsymbol{b}}_i, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_0^{(i)\star}\\
&\qquad \boldsymbol{f}^{v_{i,0}} = \langle \overrightarrow{\boldsymbol{b}}_{N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_1^{(i)\star}\\
&\qquad \boldsymbol{f}^{\pi(i)} = \langle \overrightarrow{\boldsymbol{b}}_{2N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_2^{(i)\star}\\
&\qquad \boldsymbol{f}^{\alpha^{\pi(i)}} = \langle \overrightarrow{\boldsymbol{b}}_{3N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_3^{(i)\star}\\
&\qquad \boldsymbol{f}^{4N+i} = \langle \overrightarrow{\boldsymbol{b}}_{4N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_4^{(i)\star}\\
&\qquad \boldsymbol{f}^{5N+i} = \langle \overrightarrow{\boldsymbol{b}}_{5N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_5^{(i)\star}\\
&\qquad \boldsymbol{f}^{6N+i} = \langle \overrightarrow{\boldsymbol{b}}_{6N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_6^{(i)\star}\\
&\qquad \boldsymbol{f}^{7N+i} = \langle \overrightarrow{\boldsymbol{b}}_{7N+i}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_7^{(i)\star}\\
&\boldsymbol{f}_{8N+1} = \langle \overrightarrow{\boldsymbol{b}}_{8N+1}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_{8N+1}^\star\\
&\boldsymbol{f}_{8N+2} = \langle \overrightarrow{\boldsymbol{b}}_{8N+2}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{m}_9^\star\\
&\text{For } j = 1, \ldots, \eta\\
&\qquad \boldsymbol{f}_{\boldsymbol{g}_j} = \langle \overrightarrow{\boldsymbol{b}}_{8N+2+j}, \overrightarrow{\boldsymbol{y}}^\star \rangle - \boldsymbol{c}\boldsymbol{g}_j^\star
\end{aligned}
$$

Now, let's substitute those terms to their respective places in verification equations. After simplifications (c.f Appendix 8) and following the argument in [10, Theorem 5.1], for some i, $\Pr[\boldsymbol{\beta}\boldsymbol{m}_2^{(i)\star} + \boldsymbol{m}_3^{(i)\star} - \boldsymbol{m}_6^{(i)\star} + \boldsymbol{\gamma} \neq 0] = \epsilon < (3p)^{d/l}$. Similarly, with the same probability bound, we get $\boldsymbol{m}_0^{(i)\star}\boldsymbol{m}_3^{(i)\star} - \boldsymbol{m}_4^{(i)\star} \neq 0$; $\boldsymbol{m}_1^{(i)\star}\boldsymbol{m}_3^{(i)\star} - \boldsymbol{m}_5^{(i)\star} \neq 0$ and $\boldsymbol{m}_6^{(i)\star}\boldsymbol{m}_7^{(i)\star} - \boldsymbol{m}_7^{(i+1)\star} \neq 0$ altogether, independently $\sum_{i=1}^N \boldsymbol{u}_i'\boldsymbol{m}_3^{(i)\star} - \sum_{i=1}^N \boldsymbol{m}_4^{(i)\star} - M_1 \neq 0$ and $\sum_{i=1}^N \boldsymbol{v}_i'\boldsymbol{m}_3^{(i)\star} - \sum_{i=1}^N \boldsymbol{m}_5^{(i)\star} - M_2 \neq 0$; also $\boldsymbol{m}_{8N+1}^\star - \Pi \neq 0$; and $\boldsymbol{m}_7^{(1)\star} - 1 \neq 0$; finally $\sum_{i=1}^N \boldsymbol{\theta}_{ij}\boldsymbol{m}_2^{(i)\star} + \boldsymbol{g}_j^\star - \boldsymbol{h}_j \neq 0$.

Combining all extracted relations we obtain

$$
\begin{aligned}
\prod_i^N (\boldsymbol{\beta}\boldsymbol{m}_2^{(i)\star} + \boldsymbol{m}_3^{(i)\star} - \boldsymbol{\gamma}) &= \Pi = \prod_i^N (\boldsymbol{\beta} i + \boldsymbol{\alpha}^i - \boldsymbol{\gamma})\\
\sum_i^N \boldsymbol{m}_3^{(i)\star}(\boldsymbol{u}_i' - \boldsymbol{m}_0^{(i)\star}) &= M_1 = \sum_{i=1}^N \boldsymbol{\alpha}^i \boldsymbol{u}_i,\\
\sum_i^N \boldsymbol{m}_3^{(i)\star}(\boldsymbol{v}_i' - \boldsymbol{m}_1^{(i)\star}) &= M_2 = \sum_{i=1}^N \boldsymbol{\alpha}^i \boldsymbol{v}_i.
\end{aligned}
$$

Mix-Node Security. Once more, we refer to ref. [9] where mix-node security is proved using a game-based approach. 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 an adversary over random guessing is bounded by

$$
\epsilon = \mathbf{Adv}_{\mathcal{A}}^{\mathrm{sec}}(\kappa) \leq \epsilon_{MLWE} + \epsilon_{RLWE}. \qquad \square
$$

4 | PERFORMANCE

4.1 | Non-interactive zero knowledge proof size and optimisations

The protocol in Figure 1 can be made non-interactive with the help of Fiat-Shamir transformation. In other words, challenges are computed by the prover by hashing all previous messages and public information. Furthermore, instead of sending $\overrightarrow{\boldsymbol{w}}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{4+\eta}$ which are used as inputs to the hash function to generate challenges, the standard technique is to send the hash output and let the verifier recompute those values from verification equations and check that the hashes of the computed input terms match with the prover's hash. Thus, it is enough to send the commitment $\overrightarrow{\boldsymbol{t}}_0 \| \boldsymbol{t}_1 \| \cdots \| \boldsymbol{t}_{8N+2} \| \boldsymbol{t}_{g_1} \| \ldots \| \boldsymbol{t}_{g_\eta}$ and vectors $\overrightarrow{\boldsymbol{z}}, \overrightarrow{\boldsymbol{h}}$. For the latter term, it is enough to send only first d/l coefficients for each component. A polynomial in $\mathcal{R}_q$ consists of d coefficients less than q, so it takes $d\lfloor \log\ q \rfloor$ bits at most. $\overrightarrow{\boldsymbol{t}}_0$ and $\overrightarrow{\boldsymbol{z}}$ consist of μ and $\lambda + \mu + 8N + 2 + \eta$ polynomials respectively. There are η polynomials in $\overrightarrow{\boldsymbol{h}}$, therefore it is enough to send $\eta d\lfloor \log\ q \rfloor / l$ bits. The full cost of shortness proof is analysed in Appendix 9. Combining all of these, the size of accepting transcript for our protocol is

$$
\begin{aligned}
&(\mu + 8N + 2 + \eta)d\lfloor \log q \rfloor + (\lambda + \mu + 8N + 2 + \eta)d\lfloor \log q \rfloor +\\
&\quad + \eta \frac{d}{l}\lfloor \log q \rfloor + 256 + (2\lambda + 10N)\frac{d^2}{l}\lfloor \log q \rfloor +\\
&\quad + (\lambda + 2\mu + 7)d\lfloor \log q \rfloor + 256 = \left(16 + \frac{10d}{l}\right)Nd\lfloor \log q \rfloor +\\
&\quad + (2\lambda(1 + d/l) + 4\mu + 11 + \eta(2 + 1/l))d\lfloor \log q \rfloor + 512.
\end{aligned}
$$

Overall, the size of the 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 optimisation method is to choose a common shared seed and derive all the public polynomials using that seed.

Another possible place for optimisation is to choose public variables in a specific format such as $\boldsymbol{B}_0 = \left[\mathbf{I}_\mu | \boldsymbol{B}'_0\right]$ where $\boldsymbol{B}'_0 \in \mathcal{R}_q^{\mu\times(\lambda+8N+2+\eta)}$ and vectors $\overrightarrow{\boldsymbol{b}}_i = \overrightarrow{\mathbf{0}}_\mu \| \overrightarrow{\boldsymbol{e}}_i \| \overrightarrow{\boldsymbol{b}}'_i$ where $\overrightarrow{\boldsymbol{e}}_i$ is the ith standard basis vector of length $8N + 2 + \eta$ and $\overrightarrow{\boldsymbol{b}}'_i \in \mathcal{R}_q^\lambda$ as suggested in ref. [12], so that total number of uniform polynomials will be linear in N. (This optimisation is already taken into account in the size of shortness proof transcript in Appendix 9.)

4.2 | Choosing parameters

We want to instantiate the protocol parameters 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, $8d\left(\delta_1 - \beta_1 - 1\right) = 8d\beta'_1 < q$ and $8d\left(\delta_2 - \beta_2 - 1\right) = 8d\beta'_2 < q$. Coefficients of secret key and error terms used in RLWE encryption are sampled uniformly in $\{-1, 0, 1\}$, that is, $\chi_1 = \mathcal{U}(\{-1, 0, 1\}^d)$. 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 $q \approx 2^{63}$. Then, using LWE and SIS security estimator script[1] we get that for $\beta_1 = \beta_2 = d = 4096$, $\lambda = \mu = 1$ and $\delta_1 = \delta_2 = 2^{45}$ ($M \leq 2$ for $N < 10^5$ voters) Hermite factor for $\mathsf{MLWE}_{\lambda,\chi_2}$ with ternary noise is 1.0029 and $\mathsf{MSIS}_{8d\beta'_{1,2}}$ has root Hermite factor 1.003. Finally, by Lemma 3 in ref. [10], $p \approx 2^{-62}$, which implies that $d/l = 2$ is enough for the desired soundness level. However, following the analysis in Appendix 7 we set $d/l = 4$ and $\eta = 2$.

4.3 | Implementation and benchmarks

We can estimate the performance of proof of shuffle protocol in terms of expensive operations. Sampling challenges uniformly random from $\mathcal{C}, \chi_1$ or in interval $[-\delta_1 + 1, \delta_1]$ is not complex. Thus, the only expensive operation is polynomial multiplication in $\mathcal{R}_q$. When the ring is fully splitting, multiplication can be handled in the NTT domain in a linear number of steps. But, due to the large soundness error, we avoid using such rings. In ref. [17], authors show the performance of NTT-based polynomial multiplication in partially splitting rings. We believe that their optimised implementation can further improve overall protocol running time. In Figure 1, we see that the protocol uses $O(N^2)$ multiplication operations due to $16N$ inner products between vectors of length $\lambda + \mu + 8N + 2 + \eta$. However, applying the optimisation trick in Section 4.1, this dependency becomes linear in N. Because the complexity of polynomial multiplication depends only on the ring structure, it can be assumed to be constant. Thus, the time complexity of the protocol becomes linear in the number of voters.

As a proof of concept, the proposed scheme is implemented in C language and made publicly available.[2] The polynomial operations are borrowed from Kyber/Dilithium reference implementations and modified afterwards for chosen parameters. SHAKE128 is used as a hash function for generating challenges. In Table 1, the average runtime to generate and verify the proof of shuffle protocol is given. Tests are run on 8th generation Intel i5-8250u CPU with 3.4 GHz maximum clock speed and 16 GB RAM.

Relying on the numbers shown in Table 1, in case the number of voters is 100,000, we can expect the proofs to take about 152,000 s (approximately 42.2 h) and the proof size to be about 1.34 TB, which is still manageable. We note that our implementation has not been heavily optimised. In order to go beyond the 100,000 order of magnitude, further optimisations are needed.

In the existing literature, a few other lattice-based e-voting protocols are proposed aiming at practical performance. EVOLVE [6] performs about 10 times faster than our implementation using a highly optimised mathematical library. Correctness, privacy and consistency of EVOLVE scheme are based on only hardness of MLWE and MSIS problems which is also the case for our protocol. However, EVOLVE is a homomorphic tally-based protocol, limiting its potential usage scenarios. The decryption mix-net-based voting solution by Boyen et al. [22] avoids using Non-Interactive Zero-knowledge proofs and bases security claims on trusted public audits. As a result, their proposed system achieves very fast results, but they need to trust the auditors is a significant restriction. To the best of our knowledge, the fastest fully lattice-based proof of correct shuffle is presented in ref. [23] where the authors use the shuffle of known values technique. The problem here is that the shuffle server can break the privacy of voters if the ballot box, decrypted ballots and shuffle proofs are made public. The proposed verifiable shuffle protocol is 5 times faster (33 ms per voter) than EVOLVE scheme benchmarked on an almost two times more powerful CPU.

Recently, a follow-up paper [24] was published by the same authors, presenting a verifiable shuffle protocol on BGV ciphertexts without decrypting. While their shuffle proof is significantly smaller than ours, it takes almost the same amount of time to generate the proof with their highly optimised implementation, whereas our implementation still allows for significant performance improvements. The advantage of their

TABLE 1 Performance table of our implementation of the protocol in Figure 1

	Shortness proof	Shuffle proof	Whole proof	Proof size
Per voter	1.52 s/1. 5 s	17 ms/13 ms	1.54 s/1.51 s	14 MB

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

[2] https://github.com/Valeh2012/ilmx

TABLE 2 Comparison table

	Election type	ZK proofs	Prover time per voter	Proof size per voter	Optimised implementation	Extra assumptions
EVOLVE [6]	Homomorphic tallying	Yes	0.15 s	-	Yes	No
Boyen et al. [22]	Decryption mix-net	No	132 µs	2 MB	Yes	Yes
Aranha et al. [23]	Reencryption mix-net	Yes	33 ms	22 KB	Yes	Yes
Aranha et al. [24]	Reencryption mix-net	Yes	≈1 s	370 KB	Yes	No
Costa et al. [9]	Reencryption mix-net	Yes	-	-	-	No
This work	Reenccryption mix-net	Yes	1.54 s	14 MB	No	No

scheme is that it takes much less time to verify the produced proofs.

All the considered protocols and their respective properties are summarised in Table 2.

4.4 | Post-quantum security

Post-quantum security of Fiat-Shamir transform has not been fully proven in the quantum random oracle model (QROM) yet. Several works in this research area restricted definitions for security properties. For example, computationally binding commitment schemes can be insecure against quantum attacks, as shown in ref. [25]. Collapse-binding is a stronger security property that allows to the construction of a quantum argument of knowledge [26]. The BDLOP commitment scheme used in our protocol has not been shown to satisfy the collapse-binding property. But because SIS hash functions are collapse-binding [27], hopefully one can prove for Module-SIS based BDLOP commitments as well. Another main challenge is to prove the security of mutli-round Fiat-Shamir [28] in the QROM. Until these problems are solved, unfortunately, we cannot claim full post-quantum security of the non-interactive protocol described in Section 4.1. An alternative solution is Unruh transform [29], but applying it will result in reduced performance.

However, the interactive protocol in Figure 1 will be potentially post-quantum secure. In the online voting context, election auditors can be assumed to be honest verifiers. They can be restricted to have access to the powerful quantum device during the mixing procedure in order to prevent them obtain the secret permutation vector. After the successfully verified mixing phase is over, RLWE ciphertexts can be publicly shared at no risk due to the post-quantum security level of chosen parameters.

5 | CONCLUSIONS AND FURTHER WORK

In this work, we have presented an improved lattice-based proof of shuffle protocol for secure mix-nets. The resulting scheme has linear memory cost and time complexity. As a result, we can potentially handle mixing up to 100,000 values. This is a significant landmark considering our motivating example case of mixing electronic votes.

The performance of the protocol can be improved even further with the help of parallel programming approaches. For example, with OpenMP SIMD [30] computations can be distributed to multiple processors, and at each of them, eight polynomial coefficients can be processed at a time on 512-bit wide registers using AVX512 instruction set. Another approach is to use GPUs as they are much faster than CPUs in matrix calculations [15]. We expect the effect of such optimisations to be approximately one or two orders of magnitude, but establishing the exact amount will remain the subject for future work.

ACKNOWLEDGEMENTS

This paper has been supported by the Estonian Research Council under the grant number PRG920 and European Regional Development Fund through the grant number EU48684.

CONFLICT OF INTEREST

The author declares that there is no conflict of interest that could be perceived as prejudicing the impartiality of the research reported.

DATA AVAILABILITY STATEMENT

There is no research data as such, but there is the application source code freely available on GitHub.

ORCID

Jan Willemson https://orcid.org/0000-0002-6290-2099

REFERENCES

1. del Blanco, D.Y.M., Alonso, L.P., Alonso, J.A.H.: Review of Cryptographic Schemes applied to Remote Electronic Voting systems: remaining challenges and the upcoming post-quantum paradigm. Open Math. 16(1), 95–112 (2018). https://doi.org/10.1515/math-2018-0013
2. Peng, K., et al.: Multiplicative homomorphic E-voting. In: Canteaut, A., Viswanathan, K. (eds.) Proceedings of INDOCRYPT 2004, vol. 3348, pp. 61–72. Springer (2004)
3. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. 41(2), 303–332 (1999). https://doi.org/10.1137/s0036144598347011
4. Chillotti, I., et al.: A homomorphic LWE based E-voting scheme. In: Takagi, T. (ed.) Proceedings of PQCrypto 2016, vol. 9606, pp. 245–265. Springer (2016)
5. del Pino, R., et al.: Practical quantum-safe voting from lattices. In: Thuraisingham, B.M., et al. (eds.) Proceedings of ACM CCS 2017, pp. 1565–1581. ACM (2017)

6. Costa, N., Martínez, R., Morillo, P.: Proof of a shuffle for lattice-based cryptography. In: Lipmaa, H., Mitrokotsa, A., Matulevicius, R. (eds.) Proceedings of NordSec 2017, vol. 10674, pp. 280–296. Springer (2017)
7. Strand, M.: A verifiable shuffle for the GSW cryptosystem. In: Zohar, A., et al., (eds.) Financial Cryptography and Data Security 2018, Revised Selected Papers, vol. 10958, pp. 165–180. Springer (2018)
8. Costa, N., Martínez, R., Morillo, P.: Lattice-based proof of a shuffle. In: Bracciali, A., et al. (eds.) Proceedings of Financial Cryptography and Data Security - FC 2019, vol. 11599, pp. 330–346. Springer (2019)
9. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981). https://doi.org/10.1145/358549.358563
10. Attema, T., Lyubashevsky, V., Seiler, G.: Practical product proofs for lattice commitments. In: Micciancio, D., Ristenpart, T. (eds.) Proceedings of CRYPTO 2020, Part II, vol. 12171, pp. 470–499. Springer (2020)
11. Esgin, M.F., Nguyen, N.K., Seiler, G.: Practical exact proofs from lattices: new techniques to exploit fully-splitting rings. IACR Cryptol ePrint Arch 2020, 518 (2020). https://eprint.iacr.org/2020/518
12. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Practical lattice-based zero-knowledge proofs for integer relations. In: Ligatti, J., et al. (eds.) Proceedings of ACM CCS 2020 0, pp. 1051–1070. ACM (2020)
13. Baum, C., et al.: More efficient commitments from structured lattice assumptions. In: Catalano, D., Prisco, R.D. (eds.) Proceedings of SCN 2018, vol. 11035, pp. 368–385. Springer (2018)
14. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. J. ACM 60(6), 1–43 (2013). https://doi.org/10.1145/2535925
15. Dai, W., Sunar, B.: cuHE: a homomorphic encryption accelerator library. In: Pasalic, E., Knudsen, L.R. (eds.) Proceedings of BalkanCryptSec 2015, vol. 9540, pp. 169–186. Springer (2015)
16. Peikert, C., Rosen, A.: Efficient Collision-Resistant Hashing from Worst-Case Assumptions on Cyclic Lattices, vol. 158. Electron Colloquium Comput Complex (2005)
17. Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) Proceedings of EUROCRYPT 2018, Part I, vol. 10820, pp. 204–224. Springer (2018)
18. Alkim, E., et al.: Post-quantum key exchange – a new hope. IACR Cryptol ePrint Arch 2015, 1092 (2015)
19. Ducas, L., et al.: Crystals - dilithium: digital signatures from module lattices. IACR Cryptol ePrint Arch. 2017, 633 (2017)
20. Cabarcas, D., Göpfert, F., Weiden, P.: Provably secure LWE encryption with smallish uniform noise and secret. In: Emura, K., Hanaoka, G., Zhao, Y. (eds.) Proceedings of ASIAPKC'14, pp. 33–42. ACM (2014)
21. Benhamouda, F., et al.: Efficient zero-knowledge proofs for commitments from learning with errors over rings. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) Proceedings ESORICS 2015 Part I, vol. 9326, pp. 305–325. Springer (2015)
22. Boyen, X., Haines, T., Müller, J.: A verifiable and practical lattice-based decryption mix net with external auditing. In: Chen, L., et al. (eds.) Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 2020, Lecture Notes in Computer Science, vol. 12309, Springer pp. 336–356. (2020). https://doi.org/10.1007/978-3-030-59013-0_17
23. Aranha, D.F., et al.: Lattice-based proof of shuffle and applications to electronic voting. In: Paterson, K.G. (ed.) Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 2021, Lecture Notes in Computer Science. vol. 12704, Springer pp. 227–251. (2021). https://doi.org/10.1007/978-3-030-75539-3_10
24. Aranha, D.F., et al.: Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions (2022). https://eprint.iacr.org/2022/422.%20Cryptology%20ePrint%20Archive,%20Paper%202022/422 https://eprint.iacr.org/2022/422
25. Ambainis, A., Rosmanis, A., Unruh, D.: Quantum attacks on classical proof systems: the hardness of quantum rewinding. In: 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, pp. 474–483. IEEE Computer Society Philadelphia (2014). https://doi.org/10.1109/FOCS.2014.57
26. Unruh, D.: Computationally binding quantum commitments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 497–527. Springer Vienna (2016). https://doi.org/10.1007/978-3-662-49896-5_18
27. Liu, Q., Zhandry, M.: Revisiting post-quantum fiat-shamir. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 2019. Lecture Notes in Computer Science, vol. 11693, pp. 326–355. Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_12
28. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: multi-round fiat-shamir and more. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 2020. Lecture Notes in Computer Science, vol. 12172, pp. 602–631. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_21
29. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 2015. Lecture Notes in Computer Science, vol. 9057, pp. 755–784. Springer (2015). https://doi.org/10.1007/978-3-662-46803-6_25
30. Fortin, P., et al.: High Performance SIMD Modular Arithmetic for Polynomial Evaluation (2020). Working paper or preprint. https://hal.archives-ouvertes.fr/hal-02552673

How to cite this article: Farzaliyev, V., Willemson, J., Kaasik, J.K.: Improved lattice-based mix-nets for electronic voting. IET Inf. Secur. 17(1), 18–34 (2023). https://doi.org/10.1049/ise2.12089

APPENDIX

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

Assume that the challenges $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ are uniformly sampled from $\mathcal{S}$. Also, let $\boldsymbol{m}_i$ and $\boldsymbol{m}'_i$ be extracted messages from commitments to $\boldsymbol{i}$ and $\boldsymbol{\alpha}^i$ for $i = 1, \ldots, N$. We want to prove that

$$\prod_{i=1}^{N}(\boldsymbol{\beta i}+\boldsymbol{\alpha}^{i}-\boldsymbol{\gamma})=\prod_{i=1}^{N}(\boldsymbol{\beta m}_i+\boldsymbol{m}'_i-\boldsymbol{\gamma}) \quad (4)$$

$$\Rightarrow \boldsymbol{m}_i = \pi(i) \quad (5)$$

$$\text{and} \quad \boldsymbol{m}'_i = \boldsymbol{\alpha}^{\pi(i)} \quad (6)$$

for permutation vector π that was determined before commitments.

The product of polynomials in (4) can be considered as a polynomial with roots in $\mathcal{R}_q$ evaluated at $\boldsymbol{\gamma}$. Due to Schwartz-Zippel lemma, if two polynomials are equal at a random point, with probability higher than $1 - N/q^{d/l}$ they are equal everywhere, that is, they are the same polynomial. But, their roots are not necessarily equal unless $\mathcal{R}_q[X]$ is a unique factorisation domain, which is not the case. So, $\left\{\boldsymbol{\beta i}+\boldsymbol{\alpha}^{i}\right\}_{i=1,\ldots,N}$ and $\left\{\boldsymbol{\beta m}_i+\boldsymbol{m}'_i\right\}_{i=1,\ldots,N}$ are sets of roots of those polynomials. We want to prove that these two sets contain the same elements with permuted order.

Choose any root $\boldsymbol{\beta j}+\boldsymbol{\alpha}^{j}$ for some $j=1,\ldots,N$. Because, it is one of roots, we have

$$\prod_{i=1}^{N}\left(\boldsymbol{\beta m}_i+\boldsymbol{m}'_i-\left(\boldsymbol{\beta j}+\boldsymbol{\alpha}^{j}\right)\right)=0.$$

And for any prime ideal $\boldsymbol{\varphi_k}$, we have

$$\prod_{i=1}^{N}\left(\boldsymbol{\beta m}_i+\boldsymbol{m}'_i-\left(\boldsymbol{\beta j}+\boldsymbol{\alpha}^{j}\right)\right)\equiv 0 \bmod \boldsymbol{\varphi_k}.$$

The last equality is a kth coefficient of NTT transform of the product. Let $\check{\beta},\check{m}_i,\check{m}'_i,\check{j},\check{\alpha}^j$ be kth NTT coefficients of respective variables. We rewrite last equality in the NTT domain:

$$\prod_{i=1}^{N}\left(\check{\beta}\check{m}_i+\check{m}'_i-\left(\check{\beta}\check{j}+\check{\alpha}^j\right)\right)=0.$$

Because, multiplication and addition are performed over the field $\mathbb{Z}_q[X]/\left(\boldsymbol{\varphi_k}\right)$, it can only be 0 if at least one of multiplicands is zero. Set the index $i=i_{jk}$ when $\check{\beta}\check{m}_{i_{jk}}+\check{m}'_{i_{jk}}-\left(\check{\beta}\check{j}+\check{\alpha}^j\right)=0$. In the beginning we assumed that challenges are sampled after commitments. Then, rewriting the previous equation,

$$\check{\beta}\left(\check{m}_{i_{jk}}-\check{j}\right)+\left(\check{m}'_{i_{jk}}-\check{\alpha}^j\right)=0 \bmod q$$

has solution $\check{\beta}=\left(\check{m}_{i_{jk}}-\check{j}\right)^{-1}\left(\check{\alpha}^j-\check{m}'_{i_{jk}}\right)$ if $\check{m}_{i_{jk}}\neq j \bmod q$. Otherwise, $\check{m}_{i_{jk}}=\check{j}$ which means for every j, there exists $\boldsymbol{m}_i$ that has kth NTT coefficient equal to $\check{j}$ for some k. If we can show that for fixed j, i is same for all k, then applying inverse NTT, we get $\boldsymbol{m}_i=j$, and it follows $\boldsymbol{m}'_i=\boldsymbol{\alpha}^j$. Indeed, in the protocol, construction of $\boldsymbol{h}_1,\ldots,\boldsymbol{h}_\eta$ polynomials ensure that every committed polynomial $\boldsymbol{m}_i$ has degree less than d/l, therefore its NTT coefficients are equal to each other.

We have proven that, with some cheating probability, for some j, $\boldsymbol{\beta j}+\boldsymbol{\alpha}^{j}$ is included in $\left\{\boldsymbol{\beta m}_i+\boldsymbol{m}'_i\right\}_{i=1,\ldots,N}$. Also, recall that both i and j run from 1 to N, and as for two different $j\neq j'$, $i_j\neq i'_j$, we conclude that both sets of roots contain same elements. Then, denoting $j=\pi(i)$ we get (4).

Now, we would like to bound the cheating probability. If in at least one prime ideal $\check{m}_{i_{jk}}\neq j \bmod q$, then $\boldsymbol{m}_i$ is no more a constant polynomial. No matter what is its value, one cannot retrieve $\boldsymbol{m}_{\mathrm{i}}=\pi(j)$. By construction, probability of $\check{\beta}$ to be some value is equal to $1/q^{d/1}$. We found at most N^2 different solution in one prime ideal. Then, $\Pr[\check{m}_{i_{jk}}\neq\check{j}]\leq\frac{N^2}{q^{d/l}}$. Finally, $\Pr[\boldsymbol{m}_i\neq j]\leq\frac{lN^2}{q^{d/l}}$ which is negligible when $N=2^{20}\approx 10^6$, $d/l=4$ and $q\approx 2^{64}$.

Soundness proof

In the proof of Theorem 1, it has already been described how to extract masked openings $\boldsymbol{f}^{u_0^{(i)}},\boldsymbol{f}^{v_0^{(i)}},\boldsymbol{f}^{\pi(i)},\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}},\boldsymbol{f}^{4N+i}$, $\boldsymbol{f}^{5N+i}$, $\boldsymbol{f}^{6N+i},\boldsymbol{f}^{7N+i},\boldsymbol{f}^{8N+i}$ and $\boldsymbol{f}_{9N+1}$ for $i=1,\ldots,N$.

First, we simplify the verification equation for $\boldsymbol{v}_1$:

$$\sum_{i=1}^{N}\boldsymbol{\epsilon}_i\left(\boldsymbol{\beta f}^{\pi(i)}+\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}}-\boldsymbol{f}^{6N+i}+\boldsymbol{c\gamma}\right)=\boldsymbol{v}_1$$

$$\Rightarrow\sum_{i=1}^{N}\boldsymbol{\epsilon}_i\left(\boldsymbol{\beta}\langle\overrightarrow{\boldsymbol{b}}_{2N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle+\langle\overrightarrow{\boldsymbol{b}}_{3N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-\langle\overrightarrow{\boldsymbol{b}}_{6N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\right)+$$

$$+\boldsymbol{c}\sum_{i=1}^{N}\boldsymbol{\epsilon}_i\left(\boldsymbol{\gamma}-\boldsymbol{\beta m}_2^{(i)\star}-\boldsymbol{m}_3^{(i)\star}+\boldsymbol{m}_6^{(i)\star}\right)=\boldsymbol{v}_1$$

Because all ϵ_i are uniformly random, $\sum_{i=1}^{N}\boldsymbol{\epsilon}_i(\boldsymbol{\gamma}-\boldsymbol{\beta m}_2^{(i)\star}-\boldsymbol{m}_3^{(i)\star}+\boldsymbol{m}_6^{(i)\star})$ is also uniformly random. Either $\boldsymbol{c}$ happens to be such that the whole expression becomes equal to $\boldsymbol{v}_1$ that is known to the verifier before sending the challenge with probability $p^{d/l}$, or the relation $\boldsymbol{\beta m}_2^{(i)\star}+\boldsymbol{m}_3^{(i)\star}-\boldsymbol{\gamma}=\boldsymbol{m}_6^{(i)\star}$ holds.

Next, for $\boldsymbol{v}_2$

$$\langle\overrightarrow{\boldsymbol{b}}_{8N+2},\overrightarrow{\boldsymbol{y}}^{\star}\rangle+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{N+i}\left(\boldsymbol{f}^{6N+i}\boldsymbol{f}^{7N+i}+\boldsymbol{cf}^{7N+i+1}\right)+$$

$$+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{2N+i}\left(\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}}\boldsymbol{f}^{u_0^{(i)}}+\boldsymbol{cf}^{4N+i}\right)+$$

$$+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{3N+i}\left(\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}}\boldsymbol{f}^{v_0^{(i)}}+\boldsymbol{cf}^{5N+i}\right)+\boldsymbol{f}_{8N+2}=\boldsymbol{v}_2$$

$$\Rightarrow\langle\overrightarrow{\boldsymbol{b}}_{8N+2},\overrightarrow{\boldsymbol{y}}^{\star}\rangle+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{N+i}\left(\langle\overrightarrow{\boldsymbol{b}}_{6N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\langle\overrightarrow{\boldsymbol{b}}_{7N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\right)+$$

$$+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{2N+i}\left(\langle\overrightarrow{\boldsymbol{b}}_{u_0^{(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\langle\overrightarrow{\boldsymbol{b}}_{\alpha^{\pi(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\right)+$$

$$+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{3N+i}(\langle\overrightarrow{\boldsymbol{b}}_{v_0^{(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\langle\overrightarrow{\boldsymbol{b}}_{\alpha^{\pi(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle)+$$

$$+\boldsymbol{c}\Big(\sum_{i=1}^{N}\boldsymbol{\epsilon}_{N+i}(\langle\overrightarrow{\boldsymbol{b}}_{7N+i+1},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-\boldsymbol{m}_7^{(i)\star}\langle\overrightarrow{\boldsymbol{b}}_{6N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-$$

$$-\boldsymbol{m}_6^{(i)\star}\langle\overrightarrow{\boldsymbol{b}}_{7N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big)+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{2N+i}(\langle\overrightarrow{\boldsymbol{b}}_{4N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-$$

$$-\boldsymbol{m}_3^{(i)\star}\langle\overrightarrow{\boldsymbol{b}}_{u_0^{(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-\boldsymbol{m}_0^{(i)\star}\langle\overrightarrow{\boldsymbol{b}}_{\alpha^{\pi(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big)+$$

$$+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{3N+i}\Big(\langle\overrightarrow{\boldsymbol{b}}_{5N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-\boldsymbol{m}_3^{(i)\star}\langle\overrightarrow{\boldsymbol{b}}_{v_0^{(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle-$$

$$-\boldsymbol{m}_1^{(i)\star}\langle\overrightarrow{\boldsymbol{b}}_{\alpha^{\pi(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big)-\boldsymbol{m}_9^{\star}\Big)+$$

$$+\boldsymbol{c}^2\left(\sum_{i=1}^{N}\boldsymbol{\epsilon}_{N+i}\left(\boldsymbol{m}_6^{(i)\star}\boldsymbol{m}_7^{(i)\star}-\boldsymbol{m}_8^{(i+1)\star}\right)+\right.$$

$$+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{2N+i}\left(\boldsymbol{m}_0^{(i)\star}\boldsymbol{m}_3^{(i)\star}-\boldsymbol{m}_4^{(i)\star}\right)+$$

$$\left.+\sum_{i=1}^{N}\boldsymbol{\epsilon}_{3N+i}\left(\boldsymbol{m}_1^{(i)\star}\boldsymbol{m}_3^{(i)\star}\boldsymbol{m}_5^{(i)\star}\right)\right)=\boldsymbol{v}_2$$

The last equality is a quadratic equation in $\boldsymbol{c}$. According to [10], the probability of choosing $\boldsymbol{c}$ as one of solutions is less than $(3p)^{d/l}$. Therefore, with cheating probability $(3p)^{d/l}$, $\boldsymbol{m}_6^{(i)\star}\boldsymbol{m}_7^{(i)\star} = \boldsymbol{m}_7^{(i+1)\star}\boldsymbol{m}_0^{(i)\star}\boldsymbol{m}_3^{(i)\star} = \boldsymbol{m}_4^{(i)\star}\boldsymbol{m}_1^{(i)\star}\boldsymbol{m}_3^{(i)\star} = \boldsymbol{m}_5^{(i)\star}$.

Then, for $\boldsymbol{v}_3$

$$\boldsymbol{\epsilon}_{4N+1}\Big(\sum_{i=1}^{N}\boldsymbol{u}_i'\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}} - \sum_{i=1}^{N}\boldsymbol{f}^{4N+i} + \boldsymbol{c}M_1\Big)+$$

$$+\,\boldsymbol{\epsilon}_{4N+2}\Big(\sum_{i=1}^{N}\boldsymbol{v}_i'\boldsymbol{f}^{\boldsymbol{\alpha}^{\pi(i)}} - \sum_{i=1}^{N}\boldsymbol{f}^{5N+i} + \boldsymbol{c}M_2\Big) = \boldsymbol{v}_3$$

$$\Rightarrow \boldsymbol{\epsilon}_{4N+1}\Big(\sum_{i=1}^{N}\boldsymbol{u}_i'\langle\overrightarrow{\boldsymbol{b}}_{\alpha^{\pi(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle - \sum_{i=1}^{N}\langle\overrightarrow{\boldsymbol{b}}_{4N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big)+$$

$$+\,\boldsymbol{\epsilon}_{4N+2}\Big(\sum_{i=1}^{N}\boldsymbol{v}_i'\langle\overrightarrow{\boldsymbol{b}}_{\alpha^{\pi(i)}},\overrightarrow{\boldsymbol{y}}^{\star}\rangle - \sum_{i=1}^{N}\langle\overrightarrow{\boldsymbol{b}}_{5N+i},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big)+$$

$$+\,\boldsymbol{c}\Big(\boldsymbol{\epsilon}_{4N+1}\ \Big(M_1 - \sum_{i=1}^{N}\boldsymbol{u}_i'\boldsymbol{m}_3^{(i)\star} + \sum_{i=1}^{N}\boldsymbol{m}_4^{(i)\star}\Big)+$$

$$+\,\boldsymbol{\epsilon}_{4N+2}\Big(M_2 - \sum_{i=1}^{N}\boldsymbol{v}_i'\boldsymbol{m}_3^{(i)\star} + \sum_{i=1}^{N}\boldsymbol{m}_5^{(i)\star}\Big)\Big) = \boldsymbol{v}_3$$

Therefore, with probability $1 - p^{d/l}$, $M_1 = \sum_{i=1}^{N}\boldsymbol{u}_i'\boldsymbol{m}_3^{(i)\star} - \sum_{i=1}^{N}\boldsymbol{m}_4^{(i)\star}$ and $M_2 = \sum_{i=1}^{N}\boldsymbol{v}_i'\boldsymbol{m}_3^{(i)\star} - \sum_{i=1}^{N}\boldsymbol{m}_5^{(i)\star}$.

Finally, for $\boldsymbol{v}_4$

$$\boldsymbol{\epsilon}_{4N+3}\Big(\boldsymbol{f}^{9N} + \boldsymbol{c}\Pi\Big) + \boldsymbol{\epsilon}_{4N+4}\Big(\boldsymbol{f}^{7N+1} + \boldsymbol{c}\Big) = \boldsymbol{v}_4$$

$$\Rightarrow \boldsymbol{\epsilon}_{4N+3}\Big(\langle\overrightarrow{\boldsymbol{b}}_{9N},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big) + \boldsymbol{\epsilon}_{4N+4}\Big(\langle\overrightarrow{\boldsymbol{b}}_{7N+1},\overrightarrow{\boldsymbol{y}}^{\star}\rangle\Big)+$$

$$+\ \boldsymbol{c}\Big(\boldsymbol{\epsilon}_{4N+3}\Big(\Pi - \boldsymbol{m}_8^{(N)\star}\Big) + \boldsymbol{\epsilon}_{4N+4}\Big(\mathbf{1} - \boldsymbol{m}_7^{(1)\star}\Big)\Big) = \boldsymbol{v}_4$$

which means, as before, with the same probability, $\Pi = \boldsymbol{m}_8^{(N)\star}$ and $\boldsymbol{m}_7^{(1)\star} = \mathbf{1}$.

For $\boldsymbol{v}_{4+j}$ we have to do a different analysis. Even though, corresponding verification equations are correct except with probability $p^{d/l}$, the prover still have chance to pass those equations with invalid committed values yet obtain $\boldsymbol{h}_j$ in correct distribution, that is, first d/l coefficients will be zero. Thus, actual cheating probability is $\Pr[(\sum_{i=1}^{N}\boldsymbol{\theta}_{ji}\boldsymbol{m}_2^{i\star} + \boldsymbol{g}_j^{\star})_\mu - h_{j,\mu} = 0] = 1/q$ for any $\mu > d/l$ coefficient of $\boldsymbol{h}_j$. Moreover, as $1/q$ is not satisfactory, we decrease that bound by having η of such $\boldsymbol{h}_j$ limiting the cheating probability to be $1/q^{\eta}$.

In the end, $\boldsymbol{m}_0^{(i)\star}\boldsymbol{m}_3^{(i)\star} = \boldsymbol{m}_4^{(i)\star} \Rightarrow M_1 = \sum_{i=1}^{N}\boldsymbol{u}_i'\boldsymbol{m}_3^{(i)\star} - \sum_{i=1}^{N}\boldsymbol{m}_0^{(i)\star}\boldsymbol{m}_3^{(i)\star} = \sum_{i=1}^{N}\boldsymbol{m}_3^{(i)\star}\Big(\boldsymbol{u}_i' - \boldsymbol{m}_0^{(i)\star}\Big)$. Similarly, $\boldsymbol{m}_1^{(i)\star}\boldsymbol{m}_3^{(i)\star} = \boldsymbol{m}_5^{(i)\star} \Rightarrow M_2 = \sum_{i=1}^{N}\boldsymbol{v}_i'\boldsymbol{m}_3^{(i)\star} - \sum_{i=1}^{N}\boldsymbol{m}_1^{(i)\star}\boldsymbol{m}_3^{(i)\star} = \sum_{i=1}^{N}\boldsymbol{m}_3^{(i)\star}\Big(\boldsymbol{v}_i' - \boldsymbol{m}_1^{(i)\star}\Big)$.

Finally, we show how to restore relation (1). There are N extracted message triplets $\boldsymbol{m}_6^{(i)\star}\boldsymbol{m}_7^{(i)\star} = \boldsymbol{m}_7^{(i+1)\star}$. Also, $\boldsymbol{\beta}\boldsymbol{m}_2^{(i)\star} + \boldsymbol{m}_3^{(i)\star}\ + \boldsymbol{\gamma} = \boldsymbol{m}_6^{(i)\star}, \boldsymbol{m}_7^{(1)\star} = \mathbf{1}$ and $\Pi = \boldsymbol{m}_{8N+1}^{\star}$. Then, $\boldsymbol{m}_7^{(2)\star} = \boldsymbol{m}_7^{(1)\star}\boldsymbol{m}_6^{(2)\star} = \boldsymbol{m}_7^{(1)\star}\boldsymbol{m}_6^{(1)\star}\boldsymbol{m}_6^{(2)\star}$. Continuing this pattern, $\Pi = \boldsymbol{m}_{8N+1}^{\star} = \boldsymbol{m}_7^{(1)\star}\ \prod_{i=1}^{N}\boldsymbol{m}_6^{(i)\star} = \prod_{i=1}^{N}\Big(\boldsymbol{\beta}\boldsymbol{m}_2^{(i)\star} + \boldsymbol{m}_3^{(i)\star} + \boldsymbol{\gamma}\Big)$.

Overall, the malicious prover may convince the honest verifier with probability at most $(6\mathrm{p})^{d/1} + 1/q^{\eta}$.

Shortness proof

Here, we present a simple way to prove the correctness of RLWE zero encryptions. More precisely, we have to show that all error terms are sampled from χ_1, that is, all coefficients are in $\{-1, 0, 1\}$ and the plaintext is $\mathbf{0}$.

If an RLWE keypair is ($pk.\boldsymbol{a}$, $pk.\boldsymbol{b}$), the zero-encryption ciphertext is a pair of two polynomials as below:

$$\boldsymbol{u}_0 = pk.\boldsymbol{a}\cdot\boldsymbol{r}_E + \boldsymbol{e}_u,$$

$$\boldsymbol{v}_0 = pk.\boldsymbol{b}\cdot\boldsymbol{r}_E + \boldsymbol{e}_v + \mathbf{0}\cdot\lfloor\tfrac{q}{2}\rceil = pk.\boldsymbol{b}\cdot\boldsymbol{r}_E + \boldsymbol{e}_v,$$

where $\boldsymbol{r}_E$, $\boldsymbol{e}_u$, $\boldsymbol{e}_v \leftarrow \chi_1$.

Because only the mixing node has access to zero-encryptions, one cannot use the usual zero-knowledge proof of plaintext knowledge. Instead, the public verifier has access to the commitments to ciphertexts:

$$\boldsymbol{t}_1 = \langle\overrightarrow{\boldsymbol{b}}_1,\overrightarrow{\boldsymbol{r}}\rangle + \boldsymbol{u}_0,$$
$$\boldsymbol{t}_2 = \langle\overrightarrow{\boldsymbol{b}}_2,\overrightarrow{\boldsymbol{r}}\rangle + \boldsymbol{v}_0,$$

where $\overrightarrow{\boldsymbol{b}}_1, \overrightarrow{\boldsymbol{b}}_2 \in \mathcal{R}_q^{\mu+\lambda+2}$ and $\overrightarrow{\boldsymbol{r}} \leftarrow \chi_2^{(\mu+\lambda+2)d}$.

Substituting the ciphertext defined as above:

$$\boldsymbol{t}_1 = \langle\overrightarrow{\boldsymbol{b}}_1,\overrightarrow{\boldsymbol{r}}\rangle + pk.\boldsymbol{a}\cdot\boldsymbol{r}_E + \boldsymbol{e}_u,$$
$$\boldsymbol{t}_2 = \langle\overrightarrow{\boldsymbol{b}}_2,\overrightarrow{\boldsymbol{r}}\rangle + pk.\boldsymbol{b}\cdot\boldsymbol{r}_E + \boldsymbol{e}_v.$$

Or, rewriting as a matrix equation

$$\begin{vmatrix}\boldsymbol{t}_1\\ \boldsymbol{t}_2\end{vmatrix} = \begin{vmatrix}\overrightarrow{\boldsymbol{b}}_{1,1} & \dots & \overrightarrow{\boldsymbol{b}}_{1,n'} & pk.\boldsymbol{a} & \mathbf{1} & \mathbf{0}\\ \overrightarrow{\boldsymbol{b}}_{2,1} & \dots & \overrightarrow{\boldsymbol{b}}_{2,n'} & pk.\boldsymbol{b} & \mathbf{0} & \mathbf{1}\end{vmatrix}\begin{vmatrix}\overrightarrow{\boldsymbol{r}}_1\\ \vdots\\ \overrightarrow{\boldsymbol{r}}_{n'}\\ \boldsymbol{r}_E\\ \boldsymbol{e}_u\\ \boldsymbol{e}_v\end{vmatrix},$$

where $n' = \mu + \lambda + 2$. Observe that the last equation has form $\mathbf{A}\overrightarrow{\boldsymbol{s}} = \overrightarrow{\boldsymbol{u}}$. Proving that $\overrightarrow{\boldsymbol{s}}$ is short in this equation also proves that the commitment is a commitment to valid encryption of zero polynomial with the given public key. Unfortunately, there is no practical exact proof of short solution to a structured linear equation in $\mathcal{R}_q$. However, one can transfer the equation into a 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 [11]:

$$\begin{vmatrix} \mathcal{V}_{\boldsymbol{t}_1} \\ \mathcal{V}_{\boldsymbol{t}_2} \end{vmatrix} = \begin{vmatrix} \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{1,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{1,n'}} & \mathcal{M}_{pk.\boldsymbol{a}} & I_d & \mathbf{0}_d \\ \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{2,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{2,n'}} & \mathcal{M}_{pk.\boldsymbol{b}} & \mathbf{0}_d & I_d \end{vmatrix} \begin{vmatrix} \mathcal{V}_{\overrightarrow{\boldsymbol{r}}_1} \\ \vdots \\ \mathcal{V}_{\overrightarrow{\boldsymbol{r}}_{n'}} \\ \mathcal{V}_{\boldsymbol{r}_E} \\ \mathcal{V}_{\boldsymbol{e}_u} \\ \mathcal{V}_{\boldsymbol{e}_v} \end{vmatrix}$$

$$\overrightarrow{u} = A\overrightarrow{s}$$

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

Now, suppose there are N such zero-encryption ciphertexts. Proving the shortness of secret values for each ciphertext individually is not cost-effective. The main reason is that proving shortness of $\overrightarrow{\boldsymbol{r}}$ will be repeated for each ciphertext. One would look for amortised or batch-proofs to solve the problem. However, it is also possible to reconstruct $\mathbf{A}\overrightarrow{\boldsymbol{s}} = \overrightarrow{\boldsymbol{u}}$ relation in a way that for $N = 1$ it will give exactly the same equation as above.

This can be done by adding additional rows and columns to matrix $\mathbf{A}$. We have to change index notation a bit. Let $\boldsymbol{t}_{i,u}$, $\boldsymbol{t}_{i,v}$ be commitments for $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$ of ith ciphertext respectively. Similarly, *let* $\overrightarrow{\boldsymbol{b}}_{i,u}$ and $\overrightarrow{\boldsymbol{b}}_{i,v}$ be polynomial vectors used in those commitments. $\boldsymbol{r}_{i,E}$, $\boldsymbol{e}_{i,u}$ and $\boldsymbol{e}_{i,v}$ are parameters of each zero-encryption we want to prove that they are short. Then, final matrix will look like as in (7).

$$\begin{vmatrix} \boldsymbol{t}_{1,u} \\ \boldsymbol{t}_{1,v} \\ \boldsymbol{t}_{2,u} \\ \boldsymbol{t}_{2,v} \\ \vdots \\ \boldsymbol{t}_{N,u} \\ \boldsymbol{t}_{N,v} \end{vmatrix} = \begin{vmatrix} \overrightarrow{\boldsymbol{b}}_{1,u,1} & \dots & \overrightarrow{\boldsymbol{b}}_{1,u,n'} & pk.\boldsymbol{a} & \mathbf{0} & \cdots & \mathbf{0} \\ \overrightarrow{\boldsymbol{b}}_{1,v,1} & \dots & \overrightarrow{\boldsymbol{b}}_{1,v,n'} & pk.\boldsymbol{b} & \mathbf{0} & \cdots & \mathbf{0} \\ \overrightarrow{\boldsymbol{b}}_{2,u,1} & \dots & \overrightarrow{\boldsymbol{b}}_{2,u,n'} & \mathbf{0} & pk.\boldsymbol{a} & \cdots & \mathbf{0} \\ \overrightarrow{\boldsymbol{b}}_{2,v,1} & \dots & \overrightarrow{\boldsymbol{b}}_{2,v,n'} & \mathbf{0} & pk.\boldsymbol{b} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \overrightarrow{\boldsymbol{b}}_{N,u,1} & \dots & \overrightarrow{\boldsymbol{b}}_{N,u,n'} & \mathbf{0} & \mathbf{0} & \cdots & pk.\boldsymbol{a} \\ \overrightarrow{\boldsymbol{b}}_{N,v,1} & \dots & \overrightarrow{\boldsymbol{b}}_{N,v,n'} & \mathbf{0} & \mathbf{0} & \cdots & pk.\boldsymbol{b} \end{vmatrix} \quad \mathrm{I}_{2N} \begin{vmatrix} \overrightarrow{\boldsymbol{r}}_1 \\ \vdots \\ \overrightarrow{\boldsymbol{r}}_{n'} \\ \boldsymbol{r}_{1,E} \\ \boldsymbol{r}_{2,E} \\ \vdots \\ \boldsymbol{r}_{N,E} \\ \boldsymbol{e}_{1,u} \\ \boldsymbol{e}_{1,v} \\ \boldsymbol{e}_{2,u} \\ \boldsymbol{e}_{2,v} \\ \vdots \\ \boldsymbol{e}_{N,u} \\ \boldsymbol{e}_{N,v} \end{vmatrix}. \tag{7}$$

There, $\mathbf{I}_{2N}$ is a $2N \times 2N$ identity matrix with diagonal elements being polynomial $\mathbf{1}$. Moreover, we can transfer the equation (7) from $\mathcal{R}_q$ to $\mathbb{Z}_q$ domain and get equation (8)

$$\begin{vmatrix} \mathcal{V}_{\boldsymbol{t}_{1,u}} \\ \mathcal{V}_{\boldsymbol{t}_{1,v}} \\ \mathcal{V}_{\boldsymbol{t}_{2,u}} \\ \mathcal{V}_{\boldsymbol{t}_{2,v}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{t}_{N,u}} \\ \mathcal{V}_{\boldsymbol{t}_{N,v}} \end{vmatrix} = \begin{vmatrix} \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{1,u,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{1,u,n'}} & \mathcal{M}_{pk.\boldsymbol{a}} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{1,v,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{1,v,n'}} & \mathcal{M}_{pk.\boldsymbol{b}} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{2,u,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{2,u,n'}} & \mathbf{0}_d & \mathcal{M}_{pk.\boldsymbol{a}} & \cdots & \mathbf{0}_d \\ \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{2,v,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{2,v,n'}} & \mathbf{0}_d & \mathcal{M}_{pk.\boldsymbol{b}} & \cdots & \mathbf{0}_d \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{N,u,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{N,u,n'}} & \mathbf{0}_d & \mathbf{0}_d & \cdots & \mathcal{M}_{pk.\boldsymbol{a}} \\ \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{N,v,1}} & \dots & \mathcal{M}_{\overrightarrow{\boldsymbol{b}}_{N,v,n'}} & \mathbf{0}_d & \mathbf{0}_d & \cdots & \mathcal{M}_{pk.\boldsymbol{b}} \end{vmatrix} \quad \mathrm{I}_{2Nd} \begin{vmatrix} \mathcal{V}_{\overrightarrow{\boldsymbol{r}}_1} \\ \vdots \\ \mathcal{V}_{\overrightarrow{\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{8}$$

The equality in (8) has the form $A\vec{\mathrm{s}} = \vec{\mathrm{u}}$, too. This time, $n' = \lambda + \mu + 2N$ and $n = (n' + 3N)d$, so that $\vec{\mathrm{u}} \in \mathbb{Z}_q^{2Nd}$ $\vec{\mathrm{s}} \in \mathbb{Z}_q^{n}$ $A \in \mathbb{Z}_q^{2Nd\times n}$.

Finally, we employ the optimisation technique described in Section 4.1 and set commitment vectors as $\vec{\boldsymbol{b}} = \mathbf{0}_\mu \| \vec{\boldsymbol{e}}_i \| \vec{\boldsymbol{b}}'$ where $\vec{\boldsymbol{e}}_i$ are $2N$ dimensional standard basis vectors and $\vec{\boldsymbol{b}}' \in \mathcal{R}_q^{\lambda}$. Changing rows and simplifying (8), we get equation (9).

$$\begin{vmatrix} \mathcal{V}_{t_{1,u}} \\ \mathcal{V}_{t_{2,u}} \\ \vdots \\ \mathcal{V}_{t_{N,u}} \\ \mathcal{V}_{t_{1,v}} \\ \vdots \\ \mathcal{V}_{t_{2,v}} \\ \mathcal{V}_{t_{N,v}} \end{vmatrix} = \left[\begin{array}{c|c|c|c|c} \mathbf{0}_{2Nd\times\mu d} & \mathrm{I}_{2Nd} & \begin{vmatrix} \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,u,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,u,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,u,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,u,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,u,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,u,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,v,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,v,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,v,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,v,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,v,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,v,\lambda}} \end{vmatrix} & \begin{vmatrix} \mathcal{M}_{pk.a} \\ \mathcal{M}_{pk.b} \end{vmatrix} \otimes \mathrm{I}_N & \mathrm{I}_{2Nd} \end{array} \right] \begin{vmatrix} \mathcal{V}_{\vec{r}_1} \\ \vdots \\ \mathcal{V}_{\vec{r}_{n'}} \\ \mathcal{V}_{\boldsymbol{r}_{1,E}} \\ \mathcal{V}_{\boldsymbol{r}_{2,E}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{r}_{N,E}} \\ \mathcal{V}_{\boldsymbol{e}_{1,u}} \\ \mathcal{V}_{\boldsymbol{e}_{2,u}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,u}} \\ \mathcal{V}_{\boldsymbol{e}_{1,v}} \\ \mathcal{V}_{\boldsymbol{e}_{2,v}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,v}} \end{vmatrix} \tag{9}$$

As multiplying with zero matrix does not affect the result, without loss of generality it can be removed from equation (9). Then, first μ polynomials from commitment randomness should also be removed. The final form is the equation (10). In that equation, $n' = \lambda + 2N$ and as before $n = (n' + 3N)d$.

$$\begin{bmatrix} \mathcal{V}_{t_{1,u}} \\ \mathcal{V}_{t_{2,u}} \\ \vdots \\ \mathcal{V}_{t_{N,u}} \\ \mathcal{V}_{t_{1,v}} \\ \vdots \\ \mathcal{V}_{t_{2,v}} \\ \mathcal{V}_{t_{N,v}} \end{bmatrix} = \left[\begin{array}{c|c|c|c} \mathrm{I}_{2Nd} & \begin{vmatrix} \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,u,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,u,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,u,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,u,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,u,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,u,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,v,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{1,v,\lambda}} \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,v,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{2,v,\lambda}} \\ \vdots & \ddots & \vdots \\ \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,v,1}} & \cdots & \mathcal{M}_{\vec{\boldsymbol{b}}'_{N,v,\lambda}} \end{vmatrix} & \begin{vmatrix} \mathcal{M}_{pk.a} \\ \mathcal{M}_{pk.b} \end{vmatrix} \otimes \mathrm{I}_N & \mathrm{I}_{2Nd} \end{array} \right] \begin{vmatrix} \mathcal{V}_{\vec{r}_{\mu+1}} \\ \vdots \\ \mathcal{V}_{\vec{r}_{n'}} \\ \mathcal{V}_{\boldsymbol{r}_{1,E}} \\ \mathcal{V}_{\boldsymbol{r}_{2,E}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{r}_{N,E}} \\ \mathcal{V}_{\boldsymbol{e}_{1,u}} \\ \mathcal{V}_{\boldsymbol{e}_{2,u}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,u}} \\ \mathcal{V}_{\boldsymbol{e}_{1,v}} \\ \mathcal{V}_{\boldsymbol{e}_{2,v}} \\ \vdots \\ \mathcal{V}_{\boldsymbol{e}_{N,v}} \end{vmatrix} \tag{10}$$

Protocol

Esgin et al. [11] proposed an efficient lattice-based zero-knowledge proof system proving knowledge of a vector $\vec{s}$ with coefficients in $\{-1, 0, 1\}$ solution to a linear equation $A\vec{s} = \vec{u}$ in $\mathbb{Z}_q$. Their work is generally based on fully splitting rings, which we avoid for high soundness error. However, by applying transformations described in ref. [12, Appendix A.4], it is possible to work with rings that are not necessarily fully splitting. We merge the product argument from [11] to prove that the secret is a ternary vector, and the linear argument to show $A\vec{s} = \vec{u}$ in $\mathbb{Z}_q$ from ref. [12, Appendix A.4] into a final scheme depicted in Figure 3.

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
Inputs:		
$\vec{s} = \{\vec{s}_1, \dots, \vec{s}_n\} \in \{-1,0,1\}^{nl}$		
$\vec{s} = \mathbf{NTT}(\check{\boldsymbol{s}}_1)\Vert\dots\Vert\mathbf{NTT}(\check{\boldsymbol{s}}_n)$		
$A \in \mathbb{Z}_q^{ml\times nl}$		A
$\vec{u} = A\vec{s}$		$\vec{u}$
$\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \dots, \vec{\boldsymbol{b}}_{n+3} \in \mathcal{R}_q^{\lambda+\mu+n+3}$		$\boldsymbol{B}_0, \vec{\boldsymbol{b}}_1, \dots, \vec{\boldsymbol{b}}_{n+3}$
$\boldsymbol{g} \overset{\$}{\leftarrow} \{\boldsymbol{g} \in \mathcal{R}_q \mid g_0 = \dots = g_{d/l-1} = 0\}$		
$\vec{\boldsymbol{r}} \in \chi_2^{\lambda+\mu+n+3}$		
$\vec{\boldsymbol{t}}_0 = \boldsymbol{B}_0\vec{\boldsymbol{r}}$		
For $i = 1, \dots, n$:		
$\quad \boldsymbol{t}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{r}} \rangle + \check{\boldsymbol{s}}_i$		
$\boldsymbol{t}_{n+1} = \langle \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{r}} \rangle + \boldsymbol{g}$		
$\vec{\boldsymbol{y}} \overset{\$}{\leftarrow} [-\delta_2 + 1, \delta_2]^{(\lambda+\mu+n+3)d}$		
$\vec{\boldsymbol{w}} = \boldsymbol{B}_0\vec{\boldsymbol{y}}$	$\xrightarrow{\vec{\boldsymbol{t}}_0, \boldsymbol{t}_1, \dots, \boldsymbol{t}_{n+1}, \vec{\boldsymbol{w}}}$	
		$\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n \overset{\$}{\leftarrow} \mathcal{R}_q$
	$\xleftarrow{\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_n, \vec{\gamma}}$	$\vec{\gamma} = \{\vec{\gamma}_1, \dots, \vec{\gamma}_m\} \in (\mathbb{F}_q^{d/l})^{ml}$
$\boldsymbol{t}_{n+2} = \langle \vec{\boldsymbol{b}}_{n+2}, \vec{\boldsymbol{r}} \rangle +$		
$\langle \vec{\boldsymbol{b}}_{n+3}, \vec{\boldsymbol{y}} \rangle - \sum_{i=1}^{n} \boldsymbol{\alpha}_i (3\check{\boldsymbol{s}}_i \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle^2)$		
$\boldsymbol{t}_{n+3} = \langle \vec{\boldsymbol{b}}_{n+3}, \vec{\boldsymbol{r}} \rangle +$		
$\sum_{i=1}^{n} \boldsymbol{\alpha}_i ((3\check{\boldsymbol{s}}_i^2 - \boldsymbol{1}) \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle)$		
$\boldsymbol{v}_1 = \langle \vec{\boldsymbol{b}}_{n+2}, \vec{\boldsymbol{y}} \rangle + \sum_{i=1}^{n} \boldsymbol{\alpha}_i (\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{y}} \rangle^3)$		
For $j = 1, \dots, n$:		
$\quad \boldsymbol{p}_j = \mathbf{INTT}(\sum_{i=1}^{m} A_{i,j}^T \vec{\gamma}_i)$		
$\boldsymbol{h} = \boldsymbol{g} + \sum_{j=1}^{n} (\check{\boldsymbol{s}}_j \boldsymbol{p}_j) - \frac{\langle \vec{u}, \vec{\gamma} \rangle}{l}$		
$\boldsymbol{v}_2 = \langle \sum_{j=1}^{n} \boldsymbol{p}_j \vec{\boldsymbol{b}}_j + \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{y}} \rangle$	$\xrightarrow{\boldsymbol{t}_{n+2}, \boldsymbol{t}_{n+3}, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{h}}$	
	$\xleftarrow{\boldsymbol{c}}$	$\boldsymbol{c} \overset{\$}{\leftarrow} \mathcal{C}$
$\vec{\boldsymbol{z}} = \vec{\boldsymbol{y}} + \boldsymbol{c}\vec{\mathbf{r}}$		
If $\Vert\vec{\boldsymbol{z}}\Vert_\infty \geq \delta_2 - \beta_2$ abort		
	$\xrightarrow{\vec{\boldsymbol{z}}}$	
		$\Vert\vec{\boldsymbol{z}}\Vert_\infty \leq \delta_2 - \beta_2$
		$\boldsymbol{B}_0\vec{\boldsymbol{z}} \overset{?}{=} \vec{\boldsymbol{w}} + \boldsymbol{c}\vec{\boldsymbol{t}}_0$
		For $i = 1, \dots, n+3$:
		$\quad \boldsymbol{f}_i = \langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{z}} \rangle - \boldsymbol{c}\boldsymbol{t}_i$
		$\sum_{i=1}^{n} \alpha_i \big(\boldsymbol{f}_i(\boldsymbol{f}_i - \boldsymbol{c})(\boldsymbol{f}_i + \boldsymbol{c})\big)$
		$+\boldsymbol{f}_{n+2} + \boldsymbol{c}\boldsymbol{f}_{n+3} \overset{?}{=} \boldsymbol{v}_1$
		$h_0 \overset{?}{=} h_1 \overset{?}{=} \dots \overset{?}{=} h_{d/l-1} \overset{?}{=} 0$
		For $j = 1, \dots, n$:
		$\quad \boldsymbol{p}_j = \mathbf{INTT}(\sum_{i=1}^{m} A_{i,j}^T \vec{\gamma}_i)$
		$\boldsymbol{t}_f = \sum_{j=1}^{n} (\boldsymbol{t}_j \boldsymbol{p}_j) - \frac{\langle \vec{u}, \vec{\gamma} \rangle}{l}$
		$\langle \sum_{j=1}^{n} \boldsymbol{p}_j \vec{\boldsymbol{b}}_j + \vec{\boldsymbol{b}}_{n+1}, \vec{\boldsymbol{z}} \rangle \overset{?}{=}$
		$\overset{?}{=} \boldsymbol{v}_2 + \boldsymbol{c}(\boldsymbol{t}_f + \boldsymbol{t}_{n+1} - \boldsymbol{h})$

FIGURE 3 ZK-proof of shortness.

According to the NTT definition, coefficients of NTT transform are polynomials of degree less than d/l that can also be viewed as elements of $\mathbb{F}_q^{d/l}$. Define $\mathcal{S}_q = \Big\{p_0 + p_1X^{d/l} + \cdots + p_{l-1}X^{d-d/l} \in \mathcal{R}_q\Big\}$ and $\pi : \mathbb{Z}^{l\times d/l} \to \mathbb{Z}^l$, where

$$\pi\big((s_{1,1}, \dots, s_{1,d/l}), \dots, (s_{l,1}, \dots, s_{l,d/l})\big) = (s_{1,1}, \dots, s_{l,1}).$$

Let us show, that $\mathbf{s} \in \mathcal{S}_q$ if and only if all vectors in NTT($\mathbf{s}$) can have only first coordinate nonzero, therefore $\pi(\mathrm{NTT}(\mathbf{s})) \in \mathbb{Z}_q^l \subset \mathbb{F}_{q^{d/l}}^l$. Necessity part follows from NTT transform definition. For sufficiency, we fix $\boldsymbol{s}$ and NTT($\boldsymbol{s}$) with our assumptions. Then, there must exist constants $s_1, \dots, s_l$ such that $s_i \equiv \boldsymbol{s} \equiv \pi(\mathrm{NTT}(\boldsymbol{s}))_i \bmod X^{d/l} - \zeta_i$. Denote

$$n_i = X^{d/l} - \zeta_i, \quad n = \prod_{i=1}^{l} n_i, \quad m_i = n/n_i.$$

It's clear from definition of m_i, that $m_i \bmod n_i$ is a constant term. Also, for $i \neq j$ we have that gcd $(X^{d/l} - \zeta_j, X^{d/l} - \zeta_i) = 1$, as both of them are irreducible. Therefore $m_i \neq 0 \bmod n_i$ must hold, as otherwise gcd $(X^{d/l} - \zeta_j, X^{d/l} - \zeta_i) = n_i$ and deg $(n_i) > 1$. Therefore we can take the inverse of the constant term as in $\mathbb{Z}_q$ and denote $k_i = m_i^{-1} \bmod n_i$. We claim, that

$\mathbf{s} \equiv \sum_{i=1}^{l} s_i k_i m_i$ in $\mathcal{R}_q$ and $\sum_{i=1}^{l} s_i k_i m_i \in \mathcal{S}_q$. First, notice that for all $j \in \{1, \ldots, \mathrm{l}\}$, we have

$$\begin{aligned}\sum_{i=1}^{l} s_i k_i m_i &\equiv n_j \cdot \sum_{i=1, i\neq j}^{l} \left(s_i k_i \frac{n}{n_i n_j} \right) + s_j k_j m_j \\ &\equiv 0 \cdot \sum_{i=1, i\neq j}^{l} \left(s_i k_i \frac{n}{n_i n_j} \right) + s_j \cdot 1 \\ &\equiv s_j \\ &\equiv \mathbf{s} \quad \bmod \left(X^{d/l} - \zeta_j \right).\end{aligned}$$

From Chinese Remainder Theorem, it follows that $\mathbf{s} \equiv \sum_{i=1}^{l} s_i k_i m_i$ in $\mathcal{R}_q$. On the other hand, k_i, s_i are constant terms and m_i a polynomial in $\mathcal{S}_q$ for every i. First two claims follow from definitions, one can verify the last one by multiplying out all of the brackets in

$$m_i = \prod_{j=1, j\neq i}^{l} \left(X^{d/l} - \zeta_j \right).$$

Since $\mathcal{S}_q$ is a subring, it is closed under multiplication and addition. Hence $s_i, k_i, m_i \in \mathcal{S}_q$ for $i \in \{1, \ldots, l\}$ implies that $\sum_{i=1}^{l} s_i k_i m_i \in \mathcal{S}_q$.

Therefore, NTT transform of $\boldsymbol{s}_i \in \mathcal{S}_q$ encodes l coefficients of $\vec{s}_i \in \mathbb{Z}_q^l$. Then, for $\vec{s} = \left\{ \vec{s}_1, \ldots, \vec{s}_n \right\} \in \mathbb{Z}_q^{nl}$ there exists n polynomials in $\mathcal{S}_q$ such that $\vec{s} = \mathbf{NTT}(\boldsymbol{s}_1) \| \ldots \| \mathbf{NTT}(\boldsymbol{s}_n)$.

Let $A \in \mathbb{Z}_q^{ml \times nl}$, $\vec{s} \in \mathbb{Z}_q^{nl}$ and $A\vec{s} = \vec{u}$. Divide A into submatrices $A_{i,j} \in \mathbb{Z}_q^{l \times l}$

$$A = \begin{vmatrix} A_{1,1} & \ldots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{m,1} & \ldots & A_{m,n} \end{vmatrix}.$$

Using standard Fiat-Shamir technique and optimisations, communication cost of the non-interactive version of the proof in Figure 3 is (n/l + μ + 3)$d\lfloor \log\ q \rfloor$ + (λ + μ + n/l + 3)$d\lfloor \log\ q \rfloor$ + $d\lfloor \log\ q \rfloor$ + 256. Substituting n with (n' + 3N)d = (λ + 2N + 3N)d = (λ + 5N)d, the proof size is

$$(2\lambda + 10N)\frac{d^2}{l}\lfloor \log q \rfloor + (\lambda + 2\mu + 7)d\lfloor \log q \rfloor + 256.$$

The success probability of the cheating prover is bounded by $\epsilon < q^{-d/l}$, see ref. [12].