Efficient Verifiable Mixnets from Lattices, Revisited

Jonathan Bootle[1], Vadim Lyubashevsky[1], and Antonio Merino-Gallardo[1,2⋆]

[1] IBM Research Europe, Zurich, Switzerland
{jbt,vad}@zurich.ibm.com
[2] Hasso-Plattner-Institute, University of Potsdam, Potsdam, Germany
antonio@m-g.es

Abstract. Mixnets are powerful building blocks for providing anonymity in applications like electronic voting and anonymous messaging. The encryption schemes upon which traditional mixnets are built, as well as the zero-knowledge proofs used to provide verifiability, will, however, soon become insecure once a cryptographically-relevant quantum computer is built. In this work, we construct the most compact verifiable mixnet that achieves privacy and verifiability through encryption and zero-knowledge proofs based on the hardness of lattice problems, which are believed to be quantum-safe.

A core component of verifiable mixnets is a proof of shuffle. The starting point for our construction is the proof of shuffle of Aranha et al. (CT-RSA 2021). We first identify an issue with the soundness proof in that work, which is also present in the adaptation of this proof in the mixnets of Aranha et al. (ACM CCS 2023) and Hough et al. (IACR CiC 2025). The issue is that one cannot directly adapt classical proofs of shuffle to the lattice setting due to the splitting structure of the rings used in lattice-based cryptography. This is not just an artifact of the proof, but a problem that manifests itself in practice, and we successfully mount an attack against the implementation of the first of the mixnets. We fix the problem and introduce a general approach for proving shuffles in splitting rings that can be of independent interest.

The efficiency improvement of our mixnet over prior work is achieved by switching from re-encryption mixnets (as in the works of Aranha et al. and Hough et al.) to decryption mixnets with very efficient layering based on the hardness of the LWE and LWR problems over polynomial rings. The ciphertexts in our scheme are smaller by approximately a factor of 10X and 2X over the aforementioned instantiations, while the linear-size zero-knowledge proofs are smaller by a factor of 4X and 2X.

⋆ Work done while at ETH Zurich.

1 Introduction

A mixnet is a multi-party protocol that allows users to anonymously send messages. Introduced by Chaum in 1981 [19], mixnets are a fundamental building block for applications like electronic voting [19], anonymous messaging [18, 19, 24, 36, 39], and electronic cash [38].

There are two main approaches for realizing mixnets. The original construction by Chaum is what we today call a *decryption* mixnet. Users encrypt their messages under several layers of encryption and send the ciphertexts to the first server in the mixnet. The mixing servers participate one after the other in a chain-like manner. They remove one layer of encryption from every ciphertext, change their order (or *shuffle* them) according to a secret permutation, and forward them to the next server. The last server can finally publish the fully-decrypted messages. The idea is that as long as one of the servers performs its shuffling permutation honestly and keeps it secret, an adversary cannot trace back which message corresponds to which user.

A drawback of Chaum's construction is that the size of the ciphertexts grows with the number of servers in the mixnet. This was resolved by the construction of Park, Itoh and Kurosawa [49] with their *re-encryption* mixnets. These mixnets are tailored to encryption schemes that are re-randomizable, i.e., that allow modifying the randomness of a ciphertext without changing the encrypted message, and while only having access to the public key of the scheme. A notable example is the ElGamal encryption scheme [25]. In these mixnets, users encrypt their messages under a single layer of encryption, and again send the ciphertexts to the first mixing server. Rather than decrypting and shuffling, the servers proceed by *re-randomizing* and shuffling the ciphertexts. The last server in the mixnet finally publishes the ciphertexts, whose order looks random as long as one of the servers keeps their shuffling permutation hidden. This mixing phase is followed by a distributed decryption phase in order to obtain the messages in the clear. While it avoids the ciphertext growth of the decryption mixnet, the re-encryption mixnet requires an encryption scheme that has efficient distributed decryption.

Besides their privacy guarantees, mixnets are usually expected to provide verifiability, i.e., some method to verify that the mixing servers have not manipulated the ciphertexts. This is crucial, for instance, in applications like electronic voting, so as to ensure that the published votes at the end indeed correspond to the ones the voters submitted to the mixnet. While several different techniques have been proposed for adding verifiability to a mixnet (see [33] for an analysis of different approaches), zero-knowledge proofs provide the strongest guarantees. In essence, the mixing servers publish a cryptographic proof that their output list of ciphertexts is related to the input list of ciphertexts exactly as specified by the protocol, while not revealing any information about their secret shuffling permutation. The core of these proofs are the so-called proofs of correct shuffle.

A proof of correct shuffle, in its most general form, demonstrates that two lists contain the same elements without revealing the permutation that links them. These proofs can then be tailored to a particular mixnet. For instance, in a

decryption mixnet, a mixing server can prove that their output list of ciphertexts is a permutation of the decryption of their input list of ciphertexts. Shuffling proofs have been the focus of study of many works [1,8,29,48,50]. A particularly efficient paradigm was introduced by Neff [48], where the lists are represented by polynomials that have the messages as roots, and the proof comes down to showing that two polynomials are equal. Most of these proofs of shuffle have been tailored to ElGamal ciphertexts, mainly because of their suitability to re-encryption mixnets.

With the aim of achieving quantum-safe mixnets, several works construct mixnets or proofs of shuffle based on lattices. While the proof of shuffle from Costa, Martínez and Morillo [21] is constructed for lattice ciphertexts, its soundness relies on the discrete logarithm problem. The proof of Strand [53], in turn, is tailored to fully homomorphic encryption. Boyen, Haines and Müller [16] construct an efficient lattice-based decryption mixnet whose verifiability relies on auditors (who can detect if the servers cheat a lot, but would not be able to detect, with high probability, a server who makes a few illegal substitutions) so as to avoid the use of zero-knowledge proofs.

Three proofs of shuffle relying on lattice-based zero-knowledge proofs are proposed for use in re-encryption mixnets: the one by Herranz, Martínez and Sánchez, that represent the permutation using a permutation network; and the ones by Costa, Martínez, Morillo [22], and Aranha, Baum, Gjøsteen, Silde and Tunge [5], that follow Neff's polynomial approach. The proof from [22] is improved and implemented by Farzaliyev, Willemson and Kaasik [26]. The work by Aranha, Baum, Gjøsteen and Silde [4], that extends the proof of shuffle from [5], is the first that deals with adding verifiability to the distributed decryption phase required by re-encryption mixnets. This is also achieved in the subsequent work of Hough, Sandsbråten and Silde [37]. However, the introduction of distributed decryption comes at the cost of large parameters and therefore large output sizes.

1.1 Contributions

We construct a verifiable lattice-based *decryption* mixnet that features a low ciphertext expansion, namely around 3KB per layer of encryption, while being friendly to lattice-based proofs.

Proof of Shuffle. One of the core components of the verifiable mixnet is a proof of shuffle. We start by demonstrating an attack against the soundness of the proof of shuffle from [4], originated in [5], and also present in [37]. We then introduce a general approach for proving shuffles in the rings used in lattice-based cryptography (see Lemma 5).

If we have lists of elements $(a_1, \ldots, a_N)$ and $(b_1, \ldots, b_N)$, where all the a_i, b_i are in some field $\mathbb{F}$, then the two lists are a permutation of each other if and only if over $\mathbb{F}[X]$, we have

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \ .$$

The above implication, however, does not hold in case a_i, b_i are in a ring $\mathcal{R}$ that is not a field, and this was the place where the error was made in [4,5,37][3]. In $\mathcal{R} = \mathbb{Z}_q[x]/(x^n+1)$, which is the most common ring used in lattice cryptography, for all choices of prime q and integer n, this ring is never a field because $x^n + 1$ always factors into at least two non-trivial polynomials over $\mathbb{Z}_q[x]$. When $\mathcal{R}$ is not a field, one could have $a_1 \cdot a_2 = b_1 \cdot b_2$, without (a_1, a_2) being a permutation of (b_1, b_2). In [4,5,37], the proof of shuffle is used inside a voting protocol, and we implement an attack against the implementation in [4] showing that the shufflers could indeed modify votes and still have the zero-knowledge proofs go through.

The idea for fixing this problem is to instead define a permutation π over indices $(1, \ldots, N)$ and link it to the permutation over the messages in the ring. Let $(\sigma_1, \ldots, \sigma_N)$ be a permutation of $(1, \ldots, N)$. Then, as long as the difference between every two elements in $\{1, \ldots, N\}$ is invertible, proving

$$\prod_{i=1}^{N}(a_i + i \cdot X_1 - X_2) = \prod_{i=1}^{N}(b_i + \sigma_i \cdot X_1 - X_2)$$

over $\mathcal{R}[X_1, X_2]$ implies that $(a_1, \ldots, a_N)$ and $(b_1, \ldots, b_N)$ are related by the same permutation.[4]

Having established the multiplicative relation needed to prove that one list is a permutation of the other, we use it as follows: given a list of ciphertexts $(\mathtt{Enc}(a_1), \ldots, \mathtt{Enc}(a_N))$, we create a permuted list of a_i, and then prove that the permuted list and the decryption of the first list satisfy the above multiplicative relation. The product expression is inspired by the lattice-based proof of shuffle of Costa, Martínez and Morillo [22] adapted from Bayer-Groth's classical proof [8], but some of our design decisions (see Section 4.2) make the proof noticeably more efficient.

Mixnet. To obtain an efficient decryption mixnet (we'll use two mix servers as a running example here), we would like to send $\mathtt{Enc}_{pk_1}(\mathtt{Enc}_{pk_2}(m))$ to the network, then have each of the servers remove their encryption layer in turn. We don't want to exclusively use public key encryption, because public key lattice ciphertexts have a fairly large expansion in practice, and this would increase the size of the plaintext exponentially with the number of layers. Instead, a common technique is to use a combination of symmetric encryption $\mathtt{Sym}$ with public key encryption and for symmetric keys k_1 and k_2, send

$$\mathtt{Enc}_{pk_1}(k_1), \mathtt{Sym}_{k_1}\left(\mathtt{Enc}_{pk_2}(k_2), \mathtt{Sym}_{k_2}(m)\right) \quad . \tag{1}$$

[3] We were able to identify the issue in these works based on an earlier observation by Katerina Sotiraki about how this approach fails outside unique factorization domains [51].

[4] We can use another set instead of $1, \ldots, N$ provided that the pairwise differences between set elements are invertible. For example, in our setting when working over rings $\mathbb{Z}_q[x]/(x^n + 1)$, we replace i with a polynomial whose coefficients are a binary representation of i.

Note that if the symmetric scheme does not expand its ciphertext, then the size of the resulting mixnet ciphertexts increases only linearly in the number of layers, because the public key encryption is never nested. In order to make the above efficiently compatible with our lattice-based zero-knowledge proofs, we need to have a "lattice-friendly" symmetric encryption scheme that does not have ciphertext expansion. Our solution is to use a stream cipher based on the hardness of the Learning with Rounding (LWR) problem. For example, if we let k_1 be a vector with small coefficients, then we can expand it into an arbitrary-length vector by computing $r = \lceil A \cdot k_1 \rfloor$ where the number of rows of the matrix A determines how many samples we obtain. Then (1) becomes something akin to

$$\mathtt{Enc}_{pk_1}(k_1), \lceil A_1 \cdot k_1 \rfloor + \left(\mathtt{Enc}_{pk_2}(k_2), \lceil A_2 \cdot k_2 \rfloor + m\right) \quad . \tag{2}$$

Since proving correct encryption, and also rounding, can be done fairly efficiently using lattice-based proof systems (e.g. [11, 41]), the nested encryption scheme in (2) is compatible with the rest of the lattice-based protocol. We can further improve (2) by minimizing the size of k_i that we need to encrypt. Because the public key encryption scheme is over a "large" modulus ≈ 3000 as in Kyber [14], $\mathtt{Enc}_{pk_1}(k_1)$ is over a ring with this modulus, and then presumably $\lceil A_1 \cdot k_1 \rfloor$ needs to be over the same ring because we are adding the ciphertext to it. This would require k_1 to be quite large because its dimension will need to be large enough for LWR to be hard in a ring with modulus ≈ 3000. The way to keep k_i small is to apply the LWR assumption over a small ring (with modulus e.g. 256) and use the result as the secret for the LWR (or LWE) assumption over the bigger ring. This is described in more detail in Section 3.

As an example application of our decryption mixnet, we instantiate it with parameters suitable for an electronic voting scheme (see Section 5), and compare it to the ones from [4] and [37] [5], which used re-encryption mixnets. Because using a decryption mixnet allows us to use a much smaller modulus for the encryption scheme, we are able to decrease the ciphertext size by a factor of approximately 10X and 2X, if one uses linear-sized lattice-based ZK proofs, then the proof sizes decrease by approximately a factor of 4X and 2X (see Figure 4). This shows that unless the state of the art in distributed decryption improves substantially for lattice-based schemes, decryption mixnets seem like the more compact option over re-encryption mixnets.

2 Preliminaries

2.1 Notation

For a positive integer N we define $[N] = \{1, 2, \ldots, N\}$ and denote by Perm_N the set of permutations of $[N]$. We define an equivalence relation $\sim_P$ such that two lists $(a_1, \ldots, a_N)$ and $(b_1, \ldots, b_N)$ satisfy $(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N)$ if, and

[5] Even though those schemes are lacking soundness, as we demonstrated, we believe that they can be fixed using our techniques and have very similar proof sizes to those stated in the papers.

only if, they are related by a permutation, i.e., there exists $\pi \in \mathrm{Perm}_N$ such that $a_i = b_{\pi(i)}$ for all $i \in [N]$.

We use the notation $a \leftarrow b$ to assign to a the value of b. We denote by $a \xleftarrow{\$} S$ the uniform sampling from a finite set S, and similarly $a \xleftarrow{\$} \chi$ to sample according to a distribution χ. If an algorithm is deterministic, we use $a \leftarrow \mathcal{A}(b)$ to assign to a the result of the algorithm $\mathcal{A}$ on input b. For probabilistic algorithms we use $a \xleftarrow{\$} \mathcal{A}(b)$ instead. We use the acronym PPT to describe a probabilistic polynomial-time algorithm. A security parameter 1^ω with $\omega \in \mathbb{N}$ is implicitly provided as input to all the algorithms in this document.

A function $f : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for all $c \in \mathbb{N}$ there exists $\omega_c \in \mathbb{N}$ such that for all $\omega > \omega_c$, we have $|f(\omega)| < \frac{1}{\omega^c}$. Throughout this document, when we say that some quantity is negligible, we mean that it is a negligible function of the implicit security parameter 1^ω. In the context of probabilities, we write $a \approx b$ to mean that $a - b$ is negligible.

For all $a \in \mathbb{Q}$, we denote by $\lfloor a \rfloor$ the largest integer smaller than or equal to a, by $\lceil a \rceil$ the smallest integer larger than or equal to a, and by $\lceil a \rfloor$ the closest integer to a, with ties broken up.

Ring of integers. Let $\mathbb{Z}_q$ be the ring of integers modulo q. For all $w \in \mathbb{Z}_q$ we denote by $|w|$ the absolute value of w when written as an element of $\{-\lfloor q/2 \rfloor, \ldots, 0, \ldots, \lceil q/2 \rceil - 1\}$. This way, we define the ℓ_∞ and ℓ_2 norms of a vector $\mathbf{w} \in \mathbb{Z}_q^N$ as $\|\mathbf{w}\|_\infty := \max_{i \in [N]} |\mathbf{w}[i]|$ and $\|\mathbf{w}\| := \sqrt{|\mathbf{w}[1]|^2 + \cdots + |\mathbf{w}[N]|^2}$, where $\mathbf{w}[i]$ denotes the i-th element of the vector $\mathbf{w}$.

Given two moduli q_1, q_2, we define a rounding operation from $\mathbb{Z}_{q_1}$ to $\mathbb{Z}_{q_2}$ that maps $w \in \mathbb{Z}_{q_1}$ to $\lceil w \rfloor_{q_1 \to q_2} := \lceil (w \cdot q_2)/q_1 \rfloor \mod q_2$.

2.2 Polynomial Rings

Notation. For a modulus q and $n > 1$ a power of 2, we denote by $\mathcal{R}_q$ the quotient ring

$$\mathcal{R}_q := \mathbb{Z}_q[x]/(x^n + 1) \ .$$

Unless otherwise stated, we will work over a prime modulus q. Elements of $\mathcal{R}_q$ are denoted by plain lower-case letters, e.g. $a \in \mathcal{R}_q$, vectors by bold lower-case letters, e.g. $\boldsymbol{a} \in \mathcal{R}_q^N$, and matrices by bold upper-case letters, e.g. $\mathbf{A} \in \mathcal{R}_q^{N \times M}$. We use bracket notation to index the elements of vectors and matrices, e.g. $\boldsymbol{a}[i]$ and $\mathbf{A}[i][j]$, counting from 1.

Every element of $\mathcal{R}_q$ can be uniquely represented as a polynomial of degree at most $n-1$ over $\mathbb{Z}_q$, so we define its ℓ_∞ and ℓ_2 norms as the norms of the vector of coefficients. We naturally extend this to a vector $\boldsymbol{w} \in \mathcal{R}_q^N$ as

$$\|\boldsymbol{w}\|_\infty := \max_{i \in [N]} \|\boldsymbol{w}[i]\|_\infty \,, \quad \text{and} \quad \|\boldsymbol{w}\| := \sqrt{\|\boldsymbol{w}[1]\|^2 + \cdots + \|\boldsymbol{w}[N]\|^2}.$$

Given two moduli q_1, q_2, we define a rounding operation $\lceil \cdot \rfloor_{q_1 \to q_2}$ from $\mathcal{R}_{q_1}$ to $\mathcal{R}_{q_2}$, simply by applying the rounding operation $\lceil \cdot \rfloor_{q_1 \to q_2}$ from $\mathbb{Z}_{q_1}$ to $\mathbb{Z}_{q_2}$ to each

of the coefficients. We also naturally extend this operation to vectors over the rings, by applying the rounding to each of the ring elements.

Binary elements and binomial distribution. We introduce the following notation for working with ring elements with binary coefficients.

- $\texttt{is\_bin} : \mathcal{R}_q^* \to \{\mathsf{True}, \mathsf{False}\}$. Predicate that takes a vector over $\mathcal{R}_q$ and evaluates to True if all the coefficients of all its elements are in $\{0, 1\}$, and to False otherwise.
- $\texttt{int\_to\_bin} : \{0, \ldots, 2^n - 1\} \to \mathcal{R}_q$. Function that maps an integer in the domain to the ring element whose coefficients are the binary representation of the integer, with the least significant bit in the constant coefficient.
- $\texttt{ring\_to\_bin} : \mathcal{R}_q^N \times \mathbb{N} \to (\mathcal{R}_q^N)^*$. Binary decomposition of (a vector of) ring elements. For $\boldsymbol{d} \in \mathcal{R}_q^N$ with coefficients in $[-2^{b-1}, 2^{b-1})$ we denote by

  $$(\boldsymbol{d}^{(0)}, \ldots, \boldsymbol{d}^{(b-1)}) \leftarrow \texttt{ring\_to\_bin}(\boldsymbol{d}, b)$$

  the decomposition of $\boldsymbol{d}$ into b binary vectors whose coefficients are the binary decomposition of the coefficients of $\boldsymbol{d}$ in two's complement.
- $\texttt{Stack} : (\mathcal{R}_q^N)^b \to \mathcal{R}_q^{Nb}$. Function that stacks b vectors into a vector in $\mathcal{R}_q^{Nb}$.
- $\mathbf{G}_N^{(b)} := (\,\mathrm{I}_N \,|\, 2\mathrm{I}_N \,|\, 4\mathrm{I}_N \,|\, \cdots \,|\, 2^{b-2}\mathrm{I}_N| \,-2^{b-1}\mathrm{I}_N)$. Gadget matrix. We have

  $$\boldsymbol{d} = \mathbf{G}_N^{(b)} \cdot \texttt{Stack}(\texttt{ring\_to\_bin}(\boldsymbol{d}, b)) \ .$$

  We write $\mathbf{G}^{(b)}$ for short when N can be inferred from the context.
- B_η: centered binomial distribution for some positive integer η. We first define it over the integers as

  $$\mathsf{B}_\eta^\star := \left\{ \sum_{i=1}^{\eta} \boldsymbol{a}[i] - \sum_{i=1}^{\eta} \boldsymbol{b}[i] \ : \ \boldsymbol{a}, \boldsymbol{b} \xleftarrow{\$} \{0,1\}^\eta \right\} \ .$$

  We extend it to the ring $\mathcal{R}_q$: we write $w \xleftarrow{\$} \mathsf{B}_\eta$ to denote that we are sampling an element of $\mathcal{R}_q$ by sampling each of its coefficients independently according to $\mathsf{B}_\eta^\star$. We write $\boldsymbol{w} \xleftarrow{\$} \mathsf{B}_\eta^N$ to obtain a vector with elements sampled independently according to B_η. Another way to express this works by defining a matrix

  $$\mathbf{B}_N^{(\eta)} := (\underbrace{\mathrm{I}_N \,|\, \cdots \,|\, \mathrm{I}_N}_{\eta \text{ times}} \,|\, \underbrace{-\mathrm{I}_N \,|\, \cdots \,|\, -\mathrm{I}_N}_{\eta \text{ times}}) \in \mathbb{Z}_q^{N \times 2\eta N} \ .$$

  If we have a vector $\boldsymbol{d}^\dagger \in \mathcal{R}_q^{2\eta N}$ with binary coefficients that have been sampled independently and uniformly at random, then we can obtain a binomially distributed vector as $\boldsymbol{w} \leftarrow \mathbf{B}_N^{(\eta)} \boldsymbol{d}^\dagger$. We write $\mathbf{B}^{(\eta)}$ for short when the dimension N can be inferred from the context.

General Facts. We state some results about polynomial rings that we will use throughout this work. One core result is a corollary of the Chinese Remainder Theorem (CRT) that allows us to see the quotient ring $\mathcal{R}_q$ as a product of smaller rings, as many as the number of irreducible factors of $x^n + 1$ over $\mathbb{Z}_q[x]$.

Fact 1. *Let $f(x) = f_1(x)f_2(x)\cdots f_t(x)$ be a decomposition into irreducible factors $f_i \in \mathbb{Z}_q[x]$ of a polynomial $f \in \mathbb{Z}_q[x]$. Then, we have the following ring isomorphism*

$$\mathbb{Z}_q[x]/(f(x)) \cong (\mathbb{Z}_q[x]/(f_1(x))) \times \cdots \times (\mathbb{Z}_q[x]/(f_t(x))) \ ,$$

where each of the factors $\mathbb{Z}_q[x]/(f_i(x))$ is a field. In particular, the isomorphism consists in reducing modulo each of the $f_i(x)$.

A recurrent theme will be that of obtaining invertible elements. The following result provides a simple and sufficient condition for a polynomial to be invertible, namely that it has small norm. Additionally, the result gives us the decomposition of $x^n + 1$ in terms of the value of q.

Lemma 1 ([42, Corollary 1.2]). *Let $n \geq k > 1$ be powers of 2 and $q = 2k+1 \pmod{4k}$ be a prime. Then the polynomial $x^n + 1$ factors as*

$$x^n + 1 \equiv \prod_{j=1}^{k} (x^{n/k} - r_j) \pmod{q}$$

for distinct $r_j \in \mathbb{Z}_q^*$ where $x^{n/k} - r_j$ are irreducible in the ring $\mathbb{Z}_q[x]$. Furthermore, any $y \in \mathbb{Z}_q[x]/(x^n + 1)$ that satisfies either

$$0 < \|y\|_\infty < \frac{1}{\sqrt{q}} \cdot q^{1/k} \quad \text{or} \quad 0 < \|y\| < q^{1/k}$$

has an inverse in $\mathbb{Z}_q[x]/(x^n + 1)$.

Finally, we state (a generalized version of) the Schwartz-Zippel Lemma, a powerful tool for proving the equality of polynomials.

Lemma 2 ([22, Lemma 1]). *Let $p \in \mathcal{R}[X_1, X_2, \ldots, X_k]$ be a non-zero polynomial of total degree $d \geq 0$ over a commutative ring $\mathcal{R}$. Let $\mathcal{D}$ be a finite subset of $\mathcal{R}$ such that none of the differences between two elements of $\mathcal{D}$ is a divisor of 0 and let $r_1, r_2, \ldots, r_k$ be selected at random independently and uniformly from $\mathcal{D}$. Then*

$$\Pr[p(r_1, r_2, \ldots, r_k) = 0] \leq \frac{d}{|\mathcal{D}|} \ .$$

While Lemma 2 has a probabilistic formalization, it is sometimes convenient to use a counting argument. The following lemma is also proven within the proof in [22, Lemma 1], this time for a univariate polynomial.

Lemma 3. *Let $p \in \mathcal{R}[X]$ be a non-zero polynomial of total degree $d \geq 0$ over a commutative ring $\mathcal{R}$. Let $\mathcal{D}$ be a finite subset of $\mathcal{R}$ such that none of the differences between two elements of $\mathcal{D}$ is a divisor of 0. Then p has at most d roots in $\mathcal{D}$.*

2.3 Hardness Assumptions

We state the lattice computational problems Module-Learning with Errors (MLWE), Module-Learning with Rounding (MLWR) and Module-Short Integer Solution (MSIS). We note that in MLWE we fix the distribution of the secrets $\mathbf{s}$ and $\mathbf{e}$ to be the centered binomial distribution B_η, in line with Kyber [14], whereas in MLWR we instead fix the distribution of the secret $\mathbf{s}$ to be binary.

Definition 1 ($\mathrm{MLWE}_{N,M,\eta}$)**.** *The* ***Module-LWE*** *problem with parameters $N, M > 0$ and $0 < \eta < q$ asks the adversary $\mathcal{A}$ to distinguish between the following two distributions:*

- $\mathcal{D}_0 := \{(\mathbf{A}, \mathbf{As} + \mathbf{e}) \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{N\times M};\ \mathbf{s} \xleftarrow{\$} \mathsf{B}_\eta^M;\ \mathbf{e} \xleftarrow{\$} \mathsf{B}_\eta^N\}$.
- $\mathcal{D}_1 := \{(\mathbf{A}, \mathbf{u}) \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{N\times M};\ \mathbf{u} \xleftarrow{\$} \mathcal{R}_q^N\}$.

We denote the advantage of $\mathcal{A}$ in solving $\mathrm{MLWE}_{N,M,\eta}$ *by*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{MLWE}}_{N,M,\eta}(\mathcal{A}) :=& |\Pr[b = 0 \ : \ (\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{D}_0;\ b \xleftarrow{\$} \mathcal{A}(\mathbf{A}, \mathbf{t})] \\ &- \Pr[b = 0 \ : \ (\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{D}_1;\ b \xleftarrow{\$} \mathcal{A}(\mathbf{A}, \mathbf{t})]|.\end{aligned}$$

We say that $\mathrm{MLWE}_{N,M,\eta}$ *is hard if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathrm{Adv}^{\mathrm{MLWE}}_{N,M,\eta}(\mathcal{A})$ *is negligible.*

Definition 2 ($\mathrm{MLWR}_{N,M,q_1,q_2}$)**.** *The* ***Module-LWR*** *problem with parameters $N, M, q_1, q_2 > 0$ asks the adversary $\mathcal{A}$ to distinguish between the following two distributions:*

- $\mathcal{D}_0 := \{(\mathbf{A}, \lceil \mathbf{As} \rfloor_{q_1 \to q_2}) \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_{q_1}^{N\times M};\ \mathbf{s} \xleftarrow{\$} \{0,1\}^M\}$.
- $\mathcal{D}_1 := \{(\mathbf{A}, \lceil \mathbf{u} \rfloor_{q_1 \to q_2}) \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_{q_1}^{N\times M};\ \mathbf{u} \xleftarrow{\$} \mathcal{R}_{q_1}^N\}$.

We denote the advantage of $\mathcal{A}$ in solving $\mathrm{MLWR}_{N,M,q_1,q_2}$ *by*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{MLWR}}_{N,M,q_1,q_2}(\mathcal{A}) :=& |\Pr[b = 0 \ : \ (\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{D}_0;\ b \xleftarrow{\$} \mathcal{A}(\mathbf{A}, \mathbf{t})] \\ &- \Pr[b = 0 \ : \ (\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{D}_1;\ b \xleftarrow{\$} \mathcal{A}(\mathbf{A}, \mathbf{t})]| \ .\end{aligned}$$

We say that $\mathrm{MLWR}_{N,M,q_1,q_2}$ *is hard if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathrm{Adv}^{\mathrm{MLWR}}_{N,M,q_1,q_2}(\mathcal{A})$ *is negligible.*

Definition 3 ($\mathrm{MSIS}_{N,M,B}$)**.** *The* ***Module-SIS*** *problem with parameters $N, M > 0$ and $0 < B < q$ asks to find, given $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{N\times M}$, a vector $\mathbf{z} \in \mathcal{R}_q^M$ such that $0 < \|\mathbf{z}\| \le B$ and $\mathbf{Az} = 0$. We denote the advantage of an algorithm $\mathcal{A}$ in solving* $\mathrm{MSIS}_{N,M,B}$ *by*

$$\mathrm{Adv}^{\mathrm{MSIS}}_{N,M,B}(\mathcal{A}) = \Pr[0 < \|\mathbf{z}\| \le B \wedge \mathbf{Az} = 0 \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{N\times M};\ \mathbf{z} \xleftarrow{\$} \mathcal{A}(\mathbf{A})] \ .$$

We say that $\mathrm{MSIS}_{N,M,B}$ *is hard if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathrm{Adv}^{\mathrm{MSIS}}_{N,M,B}(\mathcal{A})$ *is negligible.*

2.4 MLWE Encryption

We describe a PKE scheme MLPKE based on the hardness of the MLWE problem stated in Section 2.3. We follow the presentation of the PKE that underlies Kyber [14].

- $\mathsf{MLPKE.KGen}()$: output the key-pair $\mathsf{sk} \leftarrow \boldsymbol{s}$, $\mathsf{pk} \leftarrow (\mathbf{A}, \boldsymbol{t} = \mathbf{A}\boldsymbol{s} + \boldsymbol{e}_1)$, where $\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{k_{\mathrm{LWE}} \times k_{\mathrm{LWE}}}$ and $\boldsymbol{s}, \boldsymbol{e_1} \xleftarrow{\$} \mathsf{B}_\eta^{k_{\mathrm{LWE}}}$.
- $\mathsf{MLPKE.Enc}_{\mathsf{pk}}(m)$: take a message $m \in \mathcal{R}_q$ with binary coefficients. Sample $\boldsymbol{r}, \boldsymbol{e}_2 \xleftarrow{\$} \mathsf{B}_\eta^{k_{\mathrm{LWE}}}$ and $e_3 \xleftarrow{\$} \mathsf{B}_\eta$, and output the ciphertext

$$c \leftarrow (\boldsymbol{u}^T = \boldsymbol{r}^T\mathbf{A} + \boldsymbol{e}_2^T, v = \boldsymbol{r}^T\boldsymbol{t} + e_3 + \left\lceil \frac{q}{2} \right\rfloor m) \ .$$

- $\mathsf{MLPKE.Dec}_{\mathsf{sk}}(c)$: compute the ring element $\tilde{m} = v - \mathbf{u}^T\mathbf{s}$. Each coefficient of the plaintext $m \in \mathcal{R}_q$ is set to 0 or to 1 by checking whether the corresponding coefficient in $\tilde{m}$ is closer to 0 or to $q/2$.

If the parameters are set so that $2k_{\mathrm{LWE}}\eta^2 + \eta < q/4$, the decryption procedure will indeed return the correct plaintext. In some settings it might be fine just to have *approximate* correctness. In that case, one can set the parameters so that the failure probability is as low as desired (see, e.g., [40, Section 2.3.2]).

As previously stated, the security of the scheme can be reduced to the MLWE problem. For a proof see, e.g., [40, Section 2.3.1].

Theorem 1. *If the problem* $\mathrm{MLWE}_{k_{\mathrm{LWE}}, k_{\mathrm{LWE}}, \eta}$ *is hard, then the scheme* MLPKE *is IND-CPA secure.*

While we have only defined the scheme MLPKE for messages that are a single ring element, we extend it to support vectors of ring elements, simply by computing a separate ciphertext for each of the elements in the vector.

2.5 Ajtai Commitments

We describe a commitment scheme $\mathsf{Com}^\star = (\mathsf{Setup}^\star, \mathsf{Commit}^\star, \mathsf{Verify}^\star)$ based on the hardness of the MSIS and MLWE problems stated in Section 2.5. It is a version of the original construction implicitly introduced by Ajtai in [3].

- $\mathsf{Setup}^\star()$: choose parameters: MSIS rank k_{SIS}, message length L, and binomial parameter η. Set $B := \sqrt{nL + n2k_{\mathrm{SIS}}\eta^2}$. The message space, randomness space, commitment space and randomness distribution are defined as:

$$S_M := \{\mathbf{m} \in \mathcal{R}_q^L \ : \ \mathtt{is\_bin}(\mathbf{m})\}, \ S_R := \{\mathbf{r} \in \mathcal{R}_q^{2k_{\mathrm{SIS}}} \ : \ \|\mathbf{r}\|_\infty \leq \eta\} \ ,$$

$$S_C := \mathcal{R}_q^{k_{\mathrm{SIS}}} \ , \ \chi := \mathsf{B}_\eta^{2k_{\mathrm{SIS}}} \ .$$

  Additionally, sample matrices $\mathbf{A_1} \xleftarrow{\$} \mathcal{R}_q^{k_{\mathrm{SIS}} \times L}$ and $\mathbf{A_2} \xleftarrow{\$} \mathcal{R}_q^{k_{\mathrm{SIS}} \times 2k_{\mathrm{SIS}}}$. The public parameters are set to be:

$$pp := (k_{\mathrm{SIS}}, L, \eta, B, \mathbf{A_1}, \mathbf{A_2}) \ .$$

– $\mathsf{Commit}^\star(m; r)$: sample $\mathbf{r} \xleftarrow{\$} \mathsf{B}_\eta^{2k_{\mathrm{SIS}}}$ and output the commitment

$$\mathbf{c} \leftarrow \mathbf{A_1 m} + \mathbf{A_2 r} \ .$$

– $\mathsf{Verify}^\star(c, m, r)$: check that

$$\left\| \begin{pmatrix} \mathbf{m} \\ \mathbf{r} \end{pmatrix} \right\| \leq B \ \wedge \ \mathbf{c} = \mathbf{A_1 m} + \mathbf{A_2 r} \ .$$

If the checks succeed, output 1; otherwise output 0.

Note that we have parametrized the scheme $\mathsf{Com}^\star$ in terms of the length of the messages to be committed. If one wants to have a single instantiation of $\mathsf{Com}^\star$ for messages of different lengths, it is enough to instantiate it for the maximum length needed and then pad the other messages with zeros.

The analysis of the correctness, binding and hiding properties of the scheme $\mathsf{Com}^\star$ can be found in Appendix B.

2.6 Mixnets

The two main realizations of mixnets are decryption and re-encryption mixnets. In this paper, we only work with decryption mixnets.

Decryption mixnets. The original mixnet introduced by Chaum in 1981 [19] is what we today call a *decryption* mixnet. Let $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme such that the ciphertexts are part of the message space, i.e., $\mathcal{C} \subseteq \mathcal{M}$. A decryption mixnet is composed of ρ mixing servers with public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_\rho \in \mathcal{PK}$. There are N users that submit their encrypted messages to the mixnet. In particular, the i-th user performs a layered encryption of their message $m_i \in \mathcal{M}$ of the form

$$\mathbf{c}_{i,1} := \mathsf{Enc}_{\mathsf{pk}_1}(\mathsf{Enc}_{\mathsf{pk}_2}(\cdots(\mathsf{Enc}_{\mathsf{pk}_\rho}(m_i)))) \ .$$

If we denote $\mathbf{c}_{i,\rho+1} := m_i$, then the ciphertexts can also be recursively defined as $\mathbf{c}_{i,j} := \mathsf{Enc}_{\mathsf{pk}_j}(\mathbf{c}_{i,j+1})$ for all $j \in [\rho]$. The procedure of the mixnet is as follows. All the users submit their outermost ciphertexts $\mathbf{c}_{i,1}$ to the first server in the mixnet. The servers take part in the process one after the other in a chain-like manner. The j-th server takes each ciphertext $\mathbf{c}_{i,j}$ and decrypts it. This is done for every user, hence obtaining the list of ciphertexts $(\mathbf{c}_{1,j+1}, \ldots, \mathbf{c}_{N,j+1})$. Next, the server shuffles this list according to a freshly sampled permutation and forwards it to the $(j+1)$-th server. In the end, the last server obtains the decrypted messages and publishes them once shuffled. The idea is that, as long as one of the shuffling permutations is kept secret, it is not possible to link the published messages to the ciphertexts that were initially submitted by the users.

Verifiability. As described so far, plain mixnets do not provide all the guarantees that one would expect for many applications. For instance, a mixing server could simply replace one of the ciphertexts in the mixnet by another of their choice. In the context of e-voting, this would translate to replacing votes. This is the reason why mixnets are upgraded to have *verifiability*, i.e., some mechanism by which they can ensure that all the servers have behaved according to the specified protocol. Several approaches have been studied to add verifiability to a mixnet. Haines and Müller [33] conduct a detailed comparison of several such techniques, identifying aspects like their trust assumptions, verifiability levels or cryptographic primitives that they rely on.

Some of these techniques offer a *relaxed* verifiability, where some (small amount) of manipulations might go undetected. For instance, the lattice-based decryption mixnet of Boyen, Haines and Müller [16] involves external auditors who inject some dummy messages into the mixnet and check that the mixing servers have not manipulated them. The idea is that if a mixing server wants to manipulate some ciphertexts and happens to choose one from the auditors, then this action will be detected. Allowing for certain amount of manipulations, these techniques are typically quite efficient.

Instead, we are interested in the mixnets that achieve verifiability through Zero-Knowledge Proofs (ZKPs), as they provide the strongest guarantees. In these, all the servers provide a cryptographic proof that they have indeed proceeded according to the specified protocol. These mixnets offer perfect verifiability, up to the negligible soundness error of the ZKPs. The core component of these proofs is the proof of correct shuffle. Essentially, the servers need to prove that their input and output lists of ciphertexts are indeed connected by a permutation, while keeping this permutation private. In particular, in decryption mixnets, the mixing servers need to prove that the output ciphertexts are a permutation of the decryption of the input ciphertexts.

2.7 Related Work

The first proof of correct shuffle was proposed by Sako and Kilian in 1995 [50]. Since then, many proofs of shuffle have been developed, mostly with the aim of achieving verifiability for mixnets. They can be grouped in terms of the way in which they represent the secret permutation.

Abe and Hoshino constructed a proof of shuffle [1, 2] representing the permutation using a Beneš network. A Beneš network is composed of gates that take two inputs and either leave them in the same order or swap them. Any permutation can be represented using such a network, with the number of gates in the network being in the order of $N \log N$ gates for permuting N elements. The proof of shuffle works by proving knowledge of the configuration of the gates that results in the correct secret permutation. More recently, Herranz, Martínez and Sánchez [35] followed this paradigm to obtain a lattice-based proof of shuffle, proving the network via the proof of circuit satisfiability from [7].

Another approach was proposed by Furukawa and Sako [29], representing the permutation using matrices. In particular, any permutation on N elements

can be represented by an $N \times N$ binary matrix with exactly one 1 in each row and column. For a proof of shuffle, one can commit to the matrix, prove that it is of the right form, and prove that the output vector is indeed obtained by multiplying the input vector by the matrix. This approach was subsequently improved [28, 32, 54, 55]. In 2017, Costa, Martínez and Morillo [21] applied this technique to a lattice-based encryption scheme, but using commitments whose binding property relies on the discrete logarithm problem.

Lastly, another classical and generally more efficient approach for proving a shuffle is due to Neff [48]. It relies on the invariance of polynomials under the permutation of their roots: two polynomials can be defined, each having as roots the messages of each of the lists; the proof consists in proving that those two polynomials are equal. Several works improved on this approach [30, 31], most notably the efficient proof of shuffle from Bayer and Groth [8]. This approach generally leads to more efficient constructions, mainly because of the complexity of the instance size: compare the degree N polynomials to the $N \log N$ gates or the $N \times N$ matrices. One caveat is that, while the elements of the permutation matrices and the gates are binary, and therefore short and easy to commit to using lattice-based schemes, the coefficients of the polynomials might be larger.

Several lattice-based proofs of shuffle have been proposed in recent years following Neff's paradigm [48]. Strand [53] constructs a shuffling proof tailored to a fully homomorphic encryption scheme. Costa, Martínez and Morillo [22] adapt the proof of shuffle from [8]. This proof is subsequently improved and implemented by Farzaliyev, Willemson and Kaasik [26]. In turn, Aranha, Baum, Gjøsteen, Silde and Tunge [5] construct a proof of shuffle with techniques based on the original [48]. This proof is extended by Aranha, Baum, Gjøsteen and Silde [4], who also introduce a proof of correct distributed decryption for BGV ciphertexts [17], hence obtaining a verifiable re-encryption mixnet that also deals with the final decryption of the messages. The more recent re-encryption mixnet by Hough, Sandsbråten and Silde [37], while also relying on the proof of shuffle from [5], improves on [4] by switching to the single-element NTRU ciphertexts [52] with optimized parameters, and by proving a tighter bound on the drowning noise required for distributed decryption. It is that distributed decryption, however, that imposes the use of large parameters on [4,37], ultimately worsening the efficiency of their constructions.

3 A decryption mixnet from lattices

3.1 Large lattice parameters in re-encryption mixnets

Re-encryption mixnets re-randomize and shuffle the ciphertexts so that it is not possible to link the output ciphertexts to the input ones. A subsequent distributed decryption phase is needed to recover the original messages. In lattice-based schemes, however, distributed decryption generally requires large parameters, hence it negates the benefits of having no ciphertext expansion in re-encryption mixnets, and also leads to less efficient proofs.

Distributed decryption requires large parameters. Given a ciphertext (u, v) and a secret key s, standard decryption proceeds by computing $v - \langle u, s \rangle$ and then rounding the noise away to obtain the plaintext.[6] If the secret key is shared additively as $s = \sum_i s_i$, each party with a share s_i can compute a partial result $\langle u, s_i \rangle$. Combining all these partial results, the value $\langle u, s \rangle$ can be recovered, hence decryption can be completed in a distributed way. However, the value $\langle u, s_i \rangle$ leaks information about the secret share s_i, so instead of publishing it directly, each party publishes a noisy version $\langle u, s_i \rangle + \tilde{e}$. When combining the partial results, a noisy version of $\langle u, s \rangle$ will now be recovered, but the parameters of the scheme can be adjusted (enlarged) so that the rounding in decryption will successfully return the original plaintext. This technique for securing distributed decryption, known as "noise flooding", "noise drowning" or the addition of "smudging noise", was introduced in [9], with a noise $\tilde{e}$ that is exponentially large in the security parameter, hence also requiring an exponentially large modulus.

Recent works propose ways to construct lattice-based schemes that feature distributed decryption with a polynomial modulus [15, 20, 44]. One of the main insights from [44] is that, instead of considering fixed ciphertext noise, the distribution of the ciphertext noise can be taken into account when choosing the distribution of the smudging noise. Hence, their approach relies on the ciphertexts being fresh encryptions with noise from the specified distribution. In re-encryption mixnets, however, several mixing servers add encryptions of 0 to the users' ciphertexts. Although the servers prove that the randomness they are adding is small, it could still potentially come from a different distribution. For this reason, we believe that the approach of [44] is not directly applicable to the mixnet setting. As for [15,20], they construct FHE schemes that can perform distributed decryption with a polynomial modulus. They do so by following a Rényi divergence analysis to prove that polynomial-size smudging noise is sufficient to achieve game-based security for distributed decryption. It would be interesting to investigate whether any of these techniques can be used to construct verifiable schemes suitable for re-encryption mixnets.

Impact on ciphertext size. The main motivation for the introduction of re-encryption mixnets in [49] is to avoid the ciphertext length expansion of decryption mixnets. In the discrete logarithm setting, re-encryption mixnets indeed have smaller ciphertexts and better overall efficiency. We argue that this is not necessarily the case in the lattice setting, because of the large parameters imposed by distributed decryption. As an example, we look at the lattice-based re-encryption mixnet from [4]. Their application of the noise flooding techinque for distributed decryption leads them to use a large modulus $q \approx 2^{78}$, as well as polynomials of large degree $n = 4096$. Their BGV ciphertexts [17] are composed of two such polynomials, hence have a size of $2n \log_2 q \approx 80$KB. Another example is the re-encryption mixnet from [37], that also relies on noise flooding, but improves on [4] by switching to smaller NTRU ciphertexts [52] with a smaller

[6] We use the MLWE encryption scheme from Section 2.4 as a running example, but similar ideas apply to the BGV [17] and NTRU [52] ciphertexts used in [4, 37].

modulus $q \approx 2^{59}$, resulting in ciphertexts of 15KB. In turn, in Section 3.2 we construct a decryption mixnet that features ciphertexts of smaller size than those of [4, 37], even when accounting for several layers of encryption. In particular, each layer of encryption results in approximately 3KB (see Section 5). We do so while preserving the zero-knowledge friendliness of the mixnet, so that it can be made verifiable with efficient lattice-based proofs.

Impact on proof efficiency. When adding verifiability to a mixnet via zero-knowledge proofs, the efficiency of these proofs—measured in terms of proof size, and prover and verifier complexity—plays a crucial role in the overall efficiency of the mixnet. The smaller moduli of decryption mixnets allows us to add verifiability at a lower cost (see Section 5).

3.2 Our decryption mixnet

In a decryption mixnet with ρ mixing servers, each user encrypts their message m_i in a layered way as

$$\mathbf{c}_{i,1} \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}_1}(\mathsf{Enc}_{\mathsf{pk}_2}(\cdots(\mathsf{Enc}_{\mathsf{pk}_\rho}(m_i))))$$

or, equivalently, $\mathbf{c}_{i,j} \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}_j}(\mathbf{c}_{i,j+1})$ for all $j \in [\rho]$, where $\mathbf{c}_{i,\rho+1} := m_i$. The servers participate one after the other removing one layer of encryption and shuffling the ciphertexts, so the last server finally publishes the shuffled messages.

The mixnet gets determined by a concrete instantiation of public-key encryption scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$. Since the ciphertext length increases with the number of layers, low ciphertext length expansion is desirable. A common solution [23] is to use a hybrid scheme where a short secret is encrypted under a standard PKE scheme (e.g. lattice-based) and the plaintext is encrypted under a one-time SE scheme that uses the secret as key. Note that, instead of the PKE scheme, one could directly use a Key Encapsulation Mechanism (KEM) that generates and encapsulates the secret to be used as key. The decryption mixnet of [16] follows this approach, combining a lattice-based KEM with a SE scheme based on AES. Since we want to rely on zero-knowledge proofs to make the mixnet verifiable, we need to replace AES with a one-time SE scheme that is friendlier to lattice proof systems.

With that in mind, we construct a lattice-based one-time SE scheme OTSE and combine it with the PKE scheme MLPKE from Section 2.4 to obtain a hybrid PKE scheme HPKE, described in Figure 1.

The one-time symmetric scheme OTSE, described in Figure 2, is defined over the following key, message and ciphertext space:

$$\mathcal{K} = \{\bar{\mathbf{s}} \in \mathcal{R}_q^{k_{\mathrm{LWR}}} \mid \mathtt{is\_bin}(\bar{\mathbf{s}})\} \quad \text{and} \quad \mathcal{M} = \mathcal{C} = \mathcal{R}_q^L \ ,$$

where $k_{\mathrm{LWR}}, L \geq 1$. The scheme OTSE is also parametrized in terms of the binomial parameter η and on the following two matrices:

$$\mathbf{H} \xleftarrow{\$} \mathcal{R}_{2^\zeta}^{(k_{\mathrm{LWE}}+L)\times k_{\mathrm{LWR}}} \quad \text{and} \quad \mathbf{H}' = (\mathbf{H}'' \mid \mathrm{I}_L) \in \mathcal{R}_q^{L\times(k_{\mathrm{LWE}}+L)} \ ,$$

$\underline{\mathsf{HPKE}.\mathrm{KGen}()}$
01 $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{MLPKE}.\mathsf{KGen}()$
02 Return $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{HPKE}.\mathsf{Enc}_{\mathsf{pk}}(m)}$
03 $\overline{\mathbf{s}} \xleftarrow{\$} \mathsf{OTSE}.\mathrm{KGen}()$
04 $\mathbf{c}' \xleftarrow{\$} \mathsf{MLPKE}.\mathsf{Enc}_{\mathsf{pk}}(\overline{\mathbf{s}})$
05 $\mathbf{c}^\star \leftarrow \mathsf{OTSE}.\mathsf{Enc}_{\overline{\mathbf{s}}}(m)$
06 Return $(\mathbf{c}', \mathbf{c}^\star)$

$\underline{\mathsf{HPKE}.\mathsf{Dec}_{\mathsf{sk}}((\mathbf{c}', \mathbf{c}^\star))}$
07 $\overline{\mathbf{s}} \leftarrow \mathsf{MLPKE}.\mathsf{Dec}_{\mathsf{sk}}(\mathbf{c}')$
08 $m \leftarrow \mathsf{OTSE}.\mathsf{Dec}_{\overline{\mathbf{s}}}(\mathbf{c}^\star)$
09 Return m

Fig. 1. Description of the PKE scheme HPKE. The scheme MLPKE is the MLWE-based PKE scheme from Section 2.4, and the scheme OTSE is described in Figure 2.

$\underline{\mathsf{OTSE}.\mathrm{KGen}()}$
01 $\overline{\mathbf{s}} \xleftarrow{\$} \{0,1\}^{k_{\mathrm{LWR}}}$
02 Return $\overline{\mathbf{s}}$

$\underline{\mathsf{OTSE}.\mathsf{Enc}_{\overline{\mathbf{s}}}(m)}$
03 $\mathbf{d} \leftarrow \lceil \mathbf{H}\overline{\mathbf{s}} \rfloor_{2^\zeta \to 2^{2\eta}}$
04 $(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d}, 2\eta)$
05 $\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)})$
06 $\mathbf{a} \leftarrow \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger$
07 $\mathbf{c}^\star \leftarrow m + \mathbf{a}$
08 Return $\mathbf{c}^\star$

$\underline{\mathsf{OTSE}.\mathsf{Dec}_{\overline{\mathbf{s}}}(\mathbf{c}^\star)}$
09 $\mathbf{d} \leftarrow \lceil \mathbf{H}\overline{\mathbf{s}} \rfloor_{2^\zeta \to 2^{2\eta}}$
10 $(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d}, 2\eta)$
11 $\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)})$
12 $\mathbf{a} \leftarrow \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger$
13 $m \leftarrow \mathbf{c}^\star - \mathbf{a}$
14 Return m

Fig. 2. Description of the one-time SE scheme OTSE. The functions `ring_to_bin` and `Stack` are defined in Section 2.2, as well as the binomial matrix $\mathbf{B}^{(\eta)}$. The matrices $\mathbf{H}$ and $\mathbf{H}'$ are public parameters of the scheme.

where $\zeta > 2\eta, k_{\mathrm{LWE}} \geq 1$ and $\mathbf{H}'' \xleftarrow{\$} \mathcal{R}_q^{L \times k_{\mathrm{LWE}}}$.

The intuition for the construction of OTSE is to expand a random seed $\overline{\mathbf{s}}$ into a one-time pad $\mathbf{a}$ that masks the message. The first half of the expansion consists in generating a uniformly random vector of binary elements $\mathbf{d}^\dagger$ based on the MLWR assumption. Multiplying by the binomial matrix $\mathbf{B}^{(\eta)}$ turns this vector

into a binomially distributed one, that is finally used to compute a uniformly distributed vector $\mathbf{a}$ over $\mathcal{R}_q$ based on the MLWE assumption.

3.3 Verifiability

We extend our mixnet with zero-knowledge proofs to achieve verifiability. This way, at any point after the output of the mixnet is published, anyone can check that the output messages are indeed the same as the ones in the ciphertexts submitted to the mixnet. We do this by having a public board where

- each user publishes their ciphertext along with a proof of correct encryption, i.e., a proof of knowledge of a plaintext such that the ciphertext is a correct layered encryption of it, and
- each server publishes their output list of ciphertexts, along with a proof of correct decryption of the input ciphertexts and a proof that the output is indeed related to the decrypted input by a permutation.

Servers take their input list of ciphertexts from the published output of the previous server (or from the users in the case of the first server). In order to ensure that servers do not try to decrypt malformed ciphertexts, they will check that all the users' and previous servers' proofs indeed verify correctly before starting the decryption procedure.

The proofs of correct encryption/decryption are specific to our construction and will be covered in this section. We present the proof of correct shuffling separately Section 4, as we believe it will find other applications.

Proof of correct decryption. Each mixing server needs to prove that it decrypted all of its input ciphertexts correctly. As a preliminary step, this requires a proof of knowledge of the secret key $\mathbf{s} \in \mathcal{R}_q^{k_{\mathrm{LWE}}}$ related to the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{t}) \in \mathcal{R}_q^{k_{\mathrm{LWE}} \times k_{\mathrm{LWE}}} \times \mathcal{R}_q^{k_{\mathrm{LWE}}}$. The main relation to prove is

$$\mathbf{A}\mathbf{s} + \mathbf{e} = \mathbf{t} \mod q \ ,$$

where $\mathbf{e} \in \mathcal{R}_q^{k_{\mathrm{LWE}}}$ is the secret error used to construct $\mathbf{t}$. Additionally, we need to ensure that the entries of vectors $\mathbf{s}$ and $\mathbf{e}$ are ring elements with small coefficients, by proving that their ℓ_2 norm is not too large. Aside from the proof of knowledge of the secret key, the server needs to prove the correct decryption of each user's ciphertexts according to HPKE. In fact, it is enough for the server to prove, starting from the ciphertexts $(\mathbf{c}'_i, \mathbf{c}^\star_i)$, the correct decryption of the seed $\overline{\mathbf{s}}_i$ contained in $\mathbf{c}'_i$, and its expansion into the value $\mathbf{d}^\dagger_i$ from OTSE. The correct expansion of $\mathbf{d}^\dagger_i$ into the pad $\mathbf{a}_i$ that masks the plaintext is implicitly proven in the proof of shuffle from Section 4.4. Since each seed $\overline{\mathbf{s}}_i$ is made of k_{LWR} ring elements, we consider $\mathbf{c}'_i$ to be a list of k_{LWR} ciphertexts $(\mathbf{c}'_{i,1}, \ldots, \mathbf{c}'_{i,k_{\mathrm{LWR}}})$.

Given the ciphertexts $(\mathbf{c}'_{i,1}, \ldots, \mathbf{c}'_{i,k_{\mathrm{LWR}}}) = \mathsf{MLPKE.Enc}_{\mathsf{pk}}(\overline{\mathbf{s}}_i)$, the server needs to prove the correctness of:

1. Decryption of the seed $\bar{\mathbf{s}}_i \in \mathcal{R}_q^{k_{\mathrm{LWR}}}$. Denoting the secret decryption errors by $\mathbf{e}'_{i,1}, \ldots, \mathbf{e}'_{i,k_{\mathrm{LWR}}}$, the relations to prove are

$$\mathbf{e}'_{i,j} = {\mathbf{c}'_{i,j}}^T \begin{pmatrix} -\mathbf{s} \\ 1 \end{pmatrix} - \left\lceil \frac{q}{2} \right\rfloor \bar{\mathbf{s}}_i[j] \mod q, \ \left\| \mathbf{e}'_{i,j} \right\|_\infty < q/4$$

   for $j \in [k_{\mathrm{LWR}}]$.
2. Computation of $\mathbf{d}_i \leftarrow \lceil \mathbf{H}\bar{\mathbf{s}}_i \rfloor_{2^\zeta \to 2^{2\eta}}$. This can be proven by showing that

$$2^{\zeta-2\eta}\mathbf{d}_i + \mathbf{r}_i = \mathbf{H}\bar{\mathbf{s}}_i \mod 2^\zeta \ , \tag{3}$$

   where $\mathbf{r}_i$ is the rounding error. Additionally, we need to bound the coefficients of this error to ensure that they are in the range $[-\frac{2^\zeta}{2\cdot 2^{2\eta}}, \frac{2^\zeta}{2\cdot 2^{2\eta}})$. For this, the server can prove that they can be written in $\zeta - 2\eta$ bits.
3. Binary decomposition of $\mathbf{d}_i$, that is, $\mathbf{d}_i = \mathbf{G}^{(2\eta)}\mathbf{d}_i^\dagger$, and $\texttt{is\_bin}(\mathbf{d}_i^\dagger)$.

Instead of committing to elements that will be binary decomposed, we can optimize the proof by only committing to their binary decomposition and substituting it in the relations where those elements appear. This way, the binary decompositions are implicitly proven.

We now make the private elements that need to be committed explicit, along with the norm bounds/binary constraints that we need to prove about them:

- $\mathbf{s}, \mathbf{e} \in \mathcal{R}_q^{k_{\mathrm{LWE}}}$: secret and error such that $\mathbf{As} + \mathbf{e} = \mathbf{t} \mod q$, where $(\mathbf{A}, \mathbf{t})$ is the public key of the server. They must satisfy $\|\mathbf{s}\| \leq \delta$ and $\|\mathbf{e}\| \leq \delta$, where δ depends on the binomial parameter η from MLPKE.
- $\bar{\mathbf{s}}_i \in \mathcal{R}_q^{k_{\mathrm{LWR}}}$ for all $i \in [N]$: seed made of k_{LWR} **binary** ring elements.
- $\mathbf{e}'_{i,j} \in \mathcal{R}_q$ for all $i \in [N]$, $j \in [k_{\mathrm{LWR}}]$: decryption errors for the seed ciphertexts. They must satisfy $\left\| \mathbf{e}'_{i,j} \right\|_\infty < q/4$.
- $\mathbf{d}_i^\dagger \in \mathcal{R}_q^{2\eta(k_{\mathrm{LWE}}+L)}$ for all $i \in [N]$: **binary** decomposition of $\mathbf{d}_i$, i.e., $\mathbf{G}^{(2\eta)}\mathbf{d}_i^\dagger = \lceil \mathbf{H}\bar{\mathbf{s}}_i \rfloor_{2^{2\eta}}$.
- $\mathbf{r}_i^\dagger \in \mathcal{R}_q^{(\zeta-2\eta)(k_{\mathrm{LWE}}+L)}$ for all $i \in [N]$: **binary** decomposition of the rounding error, i.e., $\mathbf{G}^{(\zeta-2\eta)}\mathbf{r}_i^\dagger = \mathbf{r}_i$.

The relations that the mixing server needs to prove are:

1. Knowledge of the secret key associated to the public key $(\mathbf{A}, \mathbf{t})$:

$$\mathbf{As} + \mathbf{e} = \mathbf{t} \mod q \ .$$

2. Decryption of the seed $\bar{\mathbf{s}}_i$ for all $i \in [N]$:

$$\mathbf{e}'_{i,j} = {\mathbf{c}'_{i,j}}^T \begin{pmatrix} -\mathbf{s} \\ 1 \end{pmatrix} - \left\lceil \frac{q}{2} \right\rfloor \bar{\mathbf{s}}_i[j] \mod q \ \ \forall j \in [k_{\mathrm{LWR}}] \ .$$

3. Correct rounding of $\mathbf{H}\bar{\mathbf{s}}_i$ for all $i \in [N]$:

$$2^{\zeta-2\eta}\mathbf{G}^{(2\eta)}\mathbf{d}_i^\dagger + \mathbf{G}^{(\zeta-2\eta)}\mathbf{r}_i^\dagger = \mathbf{H}\bar{\mathbf{s}}_i \mod 2^\zeta \ .$$

Proof of correct encryption. Each user needs to prove knowledge of a message m_i such that the outermost ciphertext $\mathbf{c}_{i,1}$ that they submit to the mixnet is indeed a correct layered encryption of it. In particular, they need to prove the correct encryption of ρ seeds under MLPKE, and the correct encryption of the messages under OTSE using the seeds as keys. The proofs of correct encryption of the seeds consist of simple linear proofs (the computation of the (u, v) ciphertext components). As for the encryption under OTSE, it involves a proof of correct rounding exactly the same as the one in the decryption proof described above, plus the additional relation:

$$\mathbf{c}^\star = m + \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger \mod q \ .$$

3.4 Security

The mixnet users perform a layered encryption of their messages under the PKE scheme HPKE. We can show that HPKE is IND-CPA secure based on lattice hardness assumptions. The idea of the proof is as follows. Recall that HPKE is constructed as a hybrid scheme combining the PKE scheme MLPKE from Section 2.4 (used as a KEM), and the one-time SE scheme OTSE from Section 3. According to the standard KEM/DEM composition theorem (see, e.g., [34, Theorem 3.1]), it suffices to prove that MLPKE is IND-CPA secure, which is exactly Theorem 1, and that OTSE is IND-OT, which follows from the MLWR and MLWE assumptions.

Theorem 2. *If* $\mathrm{MLWR}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}$ *and* $\mathrm{MLWE}_{L,k_{\mathrm{LWE}},\eta}$ *are hard, then* OTSE *is IND-OT secure. In particular, given a PPT adversary* $\mathcal{A}$ *against the IND-OT security of* OTSE*, we construct PPT algorithms* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathrm{Adv}^{\mathit{IND\text{-}OT}}_{\mathsf{OTSE}}(\mathcal{A}) \leq 2\left(\mathrm{Adv}^{\mathrm{MLWR}}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}(\mathcal{B}_1) + \mathrm{Adv}^{\mathrm{MLWE}}_{L,k_{\mathrm{LWE}},\eta}(\mathcal{B}_2)\right) \ .$$

The proof of Theorem 2 is given in Appendix C.

Strictly speaking, OTSE is a family of one-time SE schemes parametrized by the public matrices $\mathbf{H}$ and $\mathbf{H}'$. We prove OTSE secure based on the fact that $\mathbf{H}$ and $\mathbf{H}'$ were randomly sampled.

We also discuss the possibility of security beyond IND-CPA for HPKE, in particular, security in the presence of a decryption oracle (IND-CCA). Mixnet users produce proofs of plaintext knowledge with their ciphertexts. Following the Naor-Yung paradigm [45], we can achieve IND-CCA security by attaching a second encryption of the plaintext under a different public key, along with a second proof of plaintext knowledge (proving the plaintext is the same as for the other encryption), effectively doubling the size of the ciphertext. As an optimization, it would be interesting to analyze whether the shared randomness technique from [12] is applicable in this case. This technique would allow reuse of the seed from OTSE in the symmetric part of the encryption, so only the encryption of the seed with its corresponding proof would actually need to be duplicated. In this case, IND-CCA security could be achieved without a significant overhead in the ciphertext size.

In this work we focus on the IND-CPA security, since our mixnet is a replacement for re-encryption mixnets with re-randomizable PKEs that are IND-CPA but not IND-CCA secure.

4 Lattice-Based Proof of Shuffle

4.1 The Issue with Non-Unique Factorization

In this section, we discuss Neff's approach to proving that two lists of messages have been shuffled [48], and explain why a direct adaptation of this approach to the lattice setting (such as in [4, 5, 37]) does not work and leads to practical attacks.

The idea of Neff's approach is to construct two polynomial products that have the messages as roots, and then prove the equality of the polynomials. The unique factorization of such polynomials over fields implies that the messages are related by a permutation. The essence of the approach is contained in Lemma 4.

Lemma 4. *Let $\mathbb{F}$ be a field and $N \in \mathbb{N}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \mathbb{F}^N$. If*

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \tag{4}$$

over $\mathbb{F}[X]$, then $(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N)$.

Proof. The field $\mathbb{F}$ is a unique factorization domain, hence so is the polynomial ring $\mathbb{F}[X]$. Therefore, given two decompositions of a polynomial over $\mathbb{F}[X]$ into irreducible factors, the factors must be the same up to order and multiplication by units. Since all the factors have the same leading coefficient, those units are the identity. □

Several works on proofs of shuffle use variants of this approach, most notably Bayer and Groth [8]. With the transition to quantum-safe cryptography, it is natural to try to adapt this strategy to lattice-based proofs. The proof of [5], adapted in [4, 37], directly adopted Neff's approach in the lattice setting. They implicitly rely on Lemma 4, but in the quotient rings from lattices instead of in finite fields. In particular, they work over a ring of the form $\mathbb{Z}_q[x]/(x^n + 1)$, where q is chosen so that the ring splits into t factors for $t > 1$. This means that the ring does not have unique factorization and that Lemma 4 does not actually hold, leading to an attack on the soundness of the proof of shuffle used in [4, 5, 37].

In order to introduce the attack, it is helpful to understand the gap between the permutation result proven in Lemma 4 and what we can actually prove over the splitting ring used in [4, 5, 37]. Fact 1 implies that their ring is isomorphic to the cartesian product of t fields, i.e., we can see each ring element as a tuple of t CRT components of field elements. Hence, we can apply Lemma 4 separately over each of those fields to obtain t permutations that relate the lists over each

of the CRT components. The issue is, however, that those permutations might not be the same across the different components. In that case, the lists will not actually be related by a permutation when seen over the full ring.

The attack is then straight-forward. We want to construct two lists of messages that are not related by a permutation, but for which the product from Equation (4) holds. We start with any two ring elements that are different in the first and second CRT components. We then create two new messages by swapping the values of the first CRT component of the original elements. These messages will be different from the original ones, but the polynomial product will still be the same. This means that the shuffling proofs from [4,5,37] will verify for these pairs of messages, even though they are not permuted. We have successfully tested the attack against the implementation of the shuffling proof from [4], and can be accessed in github.com/amerigal/shuffle-attack. While this shows that their proof of shuffle is unsound, it is also important to analyze the impact of the attack in the context of the e-voting protocol where the proof is embedded. In [4], the proof of shuffle is performed by the mixing servers of a re-encryption mixnet to show that they have correctly shuffled and re-randomized the ciphertexts that encrypt the votes. Following the presented attack, a malicious server can choose some ciphertexts, permute them in some CRT components, and still provide a valid proof of shuffling.

What can a server achieve by permuting ciphertexts in some CRT components? It is not clear how they could leverage this to transform an encrypted vote into another vote of their choice. In any case, they certainly have the ability to invalidate votes in an undetectable way. Note that the encryption schemes used in these re-encryption mixnets do not have integrity protection, as they need to allow for re-randomization. Hence the targeted ciphertexts with the modified CRT components will still be valid ciphertexts; they will just decrypt to some unpredictable sequence of bits. Depending on how the votes are encoded, they might turn out to be other (random) votes, or invalid votes. This is specially concerning when the first mixing server is malicious, as they know the identity of the voter associated to each ciphertext.

4.2 Fixing the Shuffle

Neff's approach [48] does not work in the lattice setting, which uses rings without unique factorization. A natural solution would be to work instead over finite fields, e.g. by focusing on lattice cryptography over quotient rings that do not split, as is the case with NTRU Prime [10]. However, the most common choice of ring in lattice cryptography is $\mathbb{Z}_q[x]/(x^n + 1)$ for an odd prime q and $n > 2$ a power of 2, used e.g., in the standardized ML-KEM [47] and ML-DSA [46]. This ring is *never* a field because $x^n + 1$ factors for all choices of odd prime q (see [40, Lemma 8]).

Therefore, to build shuffle proofs that work well with standardized schemes, we modify the product used in Neff's approach, generalizing and improving ideas from the work of Costa, Martínez and Morillo [22].

In the ring setting, the standard product equality from Lemma 4 only implies that the messages are permuted within each of the fields in which the ring splits, but not necessarily according to the same permutation. To fix this, we add more terms to the factors of the product to ensure the consistency of the permutation, and analyze the new polynomial expression in Lemma 5, which works over any ring which factors into integral domains, and in particular, over $\mathcal{R}_q$ and the quotient rings used in lattice-based cryptography.

Lemma 5. *Let $\mathcal{R} := \mathcal{R}_1 \times \cdots \times \mathcal{R}_t$ be a product of integral domains $\mathcal{R}_i$. Let $N \in \mathbb{N}$. Let $\mathcal{D} \subseteq \mathcal{R}$ and an injective map $g : [N] \to \mathcal{D}$ such that $u - v$ is invertible for all distinct $u, v \in \mathcal{D}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \mathcal{R}^N$. If*

$$\prod_{i=1}^{N}(a_i + g(i) \cdot X_1 - X_2) = \prod_{i=1}^{N}(b_i + \sigma_i \cdot X_1 - X_2) \tag{5}$$

over $\mathcal{R}[X_1, X_2]$, where $\sigma_i \in \mathcal{D}$ for all $i \in [N]$, then

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N) \ .$$

In particular, it follows that $\sigma_i = g(\pi(i))$ where $\pi \in \mathrm{Perm}_N$ is a permutation such that $b_i = a_{\pi(i)}$ for all $i \in [N]$.

Proof. Let $j \in [N]$. Looking at the products of Equation (5) as polynomials in X_2, the term $(a_j + g(j) \cdot X_1) \in \mathcal{R}[X_1]$ is a root of the left hand side over $\mathcal{R}[X_1][X_2]$, so it also has to be a root of the right hand side. Therefore,

$$\prod_{i=1}^{N}(b_i + \sigma_i \cdot X_1 - a_j - g(j) \cdot X_1) = 0$$

or, equivalently,

$$\prod_{i=1}^{N}((\sigma_i - g(j)) \cdot X_1 + (b_i - a_j)) = 0 \ .$$

Let $\ell \in [t]$. Since the factor $\mathcal{R}_\ell$ is an integral domain, so is the polynomial ring $\mathcal{R}_\ell[X_1]$. Hence, there exists $i_{j,\ell} \in [N]$ such that

$$(\sigma_{i_{j,\ell}} - g(j)) \cdot X_1 + (b_{i_{j,\ell}} - a_j) = 0$$

over $\mathcal{R}_\ell[X_1]$, i.e., when looking at the ℓ-th component of the ring element. Therefore, $\sigma_{i_{j,\ell}} = g(j)$ and $b_{i_{j,\ell}} = a_j$ over $\mathcal{R}_\ell$.

The map $j \to i_{j,\ell}$ is the permutation that relates the lists. We need to argue that the map is well defined, i.e., that $i_{j,\ell}$ does not depend on ℓ, and that it is indeed a permutation.

We claim that the equality $\sigma_{i_{j,\ell}} = g(j)$ holds not only over $\mathcal{R}_\ell$, but also over the full ring $\mathcal{R}$. Otherwise, $\sigma_{i_{j,\ell}} - g(j)$ would be a zero divisor, as it would have a 0 in the ℓ-th component, but would not be 0 over $\mathcal{R}$. However, since $\sigma_{i_{j,\ell}}, g(j) \in \mathcal{D}$, their difference would have to be invertible, leading to a contradiction. Therefore,

$\sigma_{i_{j,\ell}} = g(j)$ over the ring $\mathcal{R}$. Since this is true for all $\ell \in [t]$ and $j \in [N]$ and g is injective, we have $i_{j,1} = i_{j,2} = \cdots = i_{j,t}$ for all $j \in [N]$.

Denoting $i_j := i_{j,1}$, we have obtained that $\sigma_{i_j} = g(j)$ and $b_{i_j} = a_j$ for all $j \in [N]$. The injectivity of g implies that the i_j are distinct. Hence, the mapping $j \to i_j$ is indeed the permutation that we were looking for. □

Lemma 5 is inspired by the product expression from the lattice-based proof of shuffle of Costa, Martínez and Morillo [22] adapted from Bayer-Groth's classical proof [8]. Their product can be seen as an instantiation of Lemma 5. In [22], it is correctly analyzed that, despite the non-unique factorization of their ring, the product from [8] still implies that the lists of messages have been correctly permuted. It is worth noting that [22], following [8], uses the product to prove the permutation of some auxiliary values and then uses those to prove the permutation of the messages. It is actually not necessary to perform both of these two steps in the lattice setting, and we instead use the product to prove the permutation of the messages directly.

Another advantage of Lemma 5 over the proof from [22] is that the lemma captures the properties that the product needs to satisfy to ensure the permutation, hence allowing for different and more suitable choices of terms that still satisfy the properties. For instance, the product from [22] is an instantiation of Lemma 5 where $\mathcal{R} = \mathcal{R}_q$, $\mathcal{D}$ is the set of ring elements with degree 0 and the function g maps each integer to the ring element with that value as constant coefficient. This choice not only limits the number of messages to the size of the modulus (which we typically want to be small for efficiency), but also involves proving that certain ring elements have degree 0 (for the prover to show that the additional elements are in $\mathcal{D}$). In our proof of shuffle, we instead choose $\mathcal{D}$ to be the set of polynomials with binary coefficients and g mapping each integer to the polynomial whose coefficients are the binary representation of the integer. This choice allows for 2^n messages, and we set $n = 256$ as done, e.g., in ML-KEM [47]. Additionally, we can use efficient lattice-based proofs to show that the ring elements have binary coefficients.

Working with small messages. The simpler product from Lemma 4 can actually be used in the lattice setting in the special case that the pairwise differences between messages are invertible. This holds for messages with small norm, as their differences also have small norm and Lemma 1 implies that they are invertible. This approach is not suitable for verifiable mixnets, as their proofs of shuffle are meant to prove the permutation of ciphertexts, that can be arbitrary ring elements. In any case, as it might be useful for other protocols, we state the corresponding lemma and its proof in Appendix D.

4.3 How to Prove the Product

Proving that a list of messages has been permuted can be reduced to proving an equality of polynomial products over the ring, e.g., applying Lemma 5. We now explain how we prove the equality of such polynomial products.

Using the Schwartz-Zippel lemma, it suffices to prove the equation on a random evaluation point chosen by the verifier. This converts an equality of products of polynomials of ring elements into an equality of products of ring elements:

$$\prod_{i=1}^{N} \bar{a}_i = \prod_{i=1}^{N} \bar{b}_i \ , \tag{6}$$

for $(\bar{a}_1, \ldots, \bar{a}_N), (\bar{b}_1, \ldots, \bar{b}_N) \in \mathcal{R}_q^N$, where at least one of the tuples is committed. We reduce this to the task of proving only linear relations, following the approach of Neff [48] and its adaptation to the lattice setting in the shuffling proof of Aranha, Baum, Gjøsteen, Silde and Tunge [5].

A simplified version of the method is as follows. Suppose that we have committed to the $\bar{a}_i$ and $\bar{b}_i$ and want to prove Equation (6). The prover sends the auxiliary value

$$u_i := (-1)^i \prod_{j=1}^{i} \frac{\bar{a}_j}{\bar{b}_j}$$

to the verifier for each $i \in [N-1]$. Note that we require the $\bar{b}_i$ to be invertible, but this will generally be the case in our protocols. It is now enough to prove the N linear equations

$$\begin{aligned}
&(1)\ \bar{a}_1 + u_1 \bar{b}_1 = 0 \ , \\
&(2)\ u_{i-1}\bar{a}_i + u_i \bar{b}_i = 0 \ \ \forall i \in \{2, \ldots, N-1\} \ , \\
&(3)\ u_{N-1}\bar{a}_N + (-1)^N \bar{b}_N = 0 \ ,
\end{aligned}$$

which together imply Equation (6). It is not hard to see that the u_i do indeed satisfy these N equations. The idea for the soundness of this approach is that a non-trivial solution $(1, u_1, \ldots, u_{N-1})$ to a linear system closely related to the equations above implies that its determinant has to be 0, where the determinant is $\prod_{i=1}^{N} \bar{a}_i - \prod_{i=1}^{N} \bar{b}_i$. Hence, we have reduced the equality of products from Equation (6) to N linear equations on the committed values $\bar{a}_i$ and $\bar{b}_i$.

This simplified method will not give a zero-knowledge protocol because the u_i leak information about the $\bar{a}_i$ and $\bar{b}_i$. The solution is to include random masking values. For this, the verifier sends a random challenge γ, and the prover samples masks $\theta_i \xleftarrow{\$} \mathcal{R}_q$ for all $i \in [N-1]$. The coefficients u_i are now computed as

$$u_i := (-1)^i \gamma \prod_{j=1}^{i} \frac{\bar{a}_i}{\bar{b}_i} + \theta_i$$

for all $i \in [N-1]$. The masks θ_i mean that the u_i no longer reveal any information about the $\bar{a}_i$ and $\bar{b}_i$. However, the introduction of the masks changes the linear system. The new linear system to be proven is

$$\begin{aligned}
&(1)\ \gamma\bar{a}_1 + u_1 \bar{b}_1 = \theta_1 \bar{b}_1 \ , \\
&(2)\ u_{i-1}\bar{a}_i + u_i \bar{b}_i = \theta_{i-1}\bar{a}_i + \theta_i \bar{b}_i \ \ \forall i \in \{2, \ldots, N-1\} \ , \\
&(3)\ u_{N-1}\bar{a}_N + (-1)^N \gamma \bar{b}_N = \theta_{N-1}\bar{a}_N \ .
\end{aligned}$$

The prover will commit to the terms on the right hand side of the system and prove to the verifier that each linear equation is satisfied.

Intuitively, the soundness of this approach relies on the prover being able to provide two sets of solutions u_i, u'_i for different challenges γ, γ'. This means that the prover can obtain a solution $(\gamma - \gamma', u_1 - u'_1, \ldots, u_{N-1} - u'_{N-1})$ to the homogeneous version of the system, whose determinant is $\prod_{i=1}^{N} \bar{a}_i - \prod_{i=1}^{N} \bar{b}_i$. The verifier's challenge γ is needed this time to ensure that the first component of the solution is non-zero, hence the solution is non-trivial, so the determinant has to be 0. We actually require the solution to be non-trivial over each of the fields the ring splits into, but this will follow from the choice of challenge space. See the proof of Theorem 5 in Appendix E for more details.

4.4 Our Proof of Shuffle

The verifiability of our decryption mixnet requires the servers to provide a proof of shuffle for the ciphertexts they decrypt. We construct such a proof by relying on Lemma 5 and proving the product equality using Neff's linear approach discussed in the previous section.

Recall that a mixing server receives N ciphertexts of the form

$$(\mathbf{c}'_i,\ \mathbf{c}^\star_i = \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger_i + \mathbf{c}_i)$$

for all $i \in [N]$, where the first component $\mathbf{c}'_i$ encrypts a seed, the binary vector $\mathbf{d}^\dagger_i$ is the result of expanding that seed, and the matrices $\mathbf{H}', \mathbf{B}^{(\eta)}$ transform that binary vector into a ring element used to mask the inner ciphertext $\mathbf{c}_i$. The output of the server is a permutation of the inner ciphertexts $(\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N) = (\mathbf{c}_{\pi(1)}, \ldots, \mathbf{c}_{\pi(N)})$. The server needs to prove that the output list and the list $(\mathbf{c}^\star_1 - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger_1, \ldots, \mathbf{c}^\star_N - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger_N)$ are indeed related by a permutation, where the permutation π is kept secret and the vectors $\mathbf{d}^\dagger_i$ are committed inside $\mathbf{D}$. We use Ajtai's commitment scheme $\mathsf{Com}^\star = (\mathsf{Setup}^\star, \mathsf{Commit}^\star, \mathsf{Verify}^\star)$ described in Section 2.5. Accordingly, the parameter generator algorithm $\mathcal{G}$ will be $\mathsf{Setup}^\star$, and the prover and verifier will implicitly receive the public parameters of the commitment scheme.

The relation to be proven is:

$$\mathsf{R}_{\text{shuffle}} := \left\{ \begin{array}{c} \mathbf{x} = ((\mathbf{c}^\star_1, \ldots, \mathbf{c}^\star_N), (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), \mathbf{D}), \\ \mathbf{w} = (\pi, (\mathbf{d}^\dagger_1, \cdots, \mathbf{d}^\dagger_N), \mathbf{r}_{\mathbf{D}}) \end{array} \middle| \begin{array}{c} \mathsf{Verify}^\star(\mathbf{D}, (\mathbf{d}^\dagger_1, \ldots, \mathbf{d}^\dagger_N), \mathbf{r}_{\mathbf{D}}) = 1 \\ \wedge\ \hat{\mathbf{c}}_i = \mathbf{c}^\star_{\pi(i)} - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger_{\pi(i)} \\ \wedge\ \mathtt{is\_bin}(\mathbf{d}^\dagger_i)\ \forall i \in [N] \end{array} \right\} .$$

It is important to ensure the consistency of the $\mathbf{d}^\dagger_i$ between this proof of shuffle and the proof of correct decryption from Section 3.3, namely by using the same commitment $\mathbf{D}$ in both. Note that the correctness of that commitment and the binary nature of the $\mathbf{d}^\dagger_i$ only has to be proven once.

The proofs of shuffling that we have been discussing work with lists of ring elements, whereas we are now dealing with lists of *vectors* of ring elements.

We simply compress those vectors into single ring elements by applying the Schwartz-Zippel lemma. Accordingly, the verifier sends a challenge $\lambda \in \mathcal{R}_q$ and we set

$$z_i = \sum_{j=1}^{L} (\mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger)[j] \cdot \lambda^{j-1} \quad \text{and} \quad \hat{z}_i = \sum_{j=1}^{L} \hat{\mathbf{c}}_i[j] \cdot \lambda^{j-1}$$

for all $i \in [N]$, where L is the length of the vectors. We can then rely on Lemma 5 to show that the lists $(z_1, \ldots, z_N), (\hat{z}_1, \ldots, \hat{z}_N) \in \mathcal{R}_q^N$ are related by the secret permutation π.

We instantiate the ring $\mathcal{R}_q$ with a prime $q \equiv 5 \pmod 8$. According to Lemma 1, for this choice of modulus, the polynomial $x^n + 1$ factors into two irreducible polynomials $x^n + 1 = p_1 \cdot p_2 \mod q$, where $\deg(p_1) = \deg(p_2) = n/2$. Fact 1 gives us the isomorphism $\mathcal{R}_q \cong \mathbb{Z}_q[x]/(p_1) \times \mathbb{Z}_q[x]/(p_2)$, where both of the factors are fields.

In order to instantiate Lemma 5, we need to determine a set $\mathcal{D}$ where the differences of its elements are invertible and an injective map $g : [N] \to \mathcal{D}$. We choose $\mathcal{D}$ to be the set of ring elements with binary coefficients, and $g = \texttt{int\_to\_bin} : [N] \to \mathcal{R}_q^n$ mapping each integer to the ring element whose coefficients are the binary decomposition of that integer. The differences of distinct elements of $\mathcal{D}$ have infinity norm at most 2, hence Lemma 1 implies that such differences are invertible.

We also require the challenge space to be such that the differences of distinct elements are invertible. However, nothing needs to be proven about these elements, as they are publicly known, hence we do not need to take them of small norm. We want a large set that is easy to sample from. For this reason we set the challenge space to be $\mathcal{C} := \{p \in \mathcal{R}_q \mid \deg(p) < n/2\}$.

The proof of shuffle is described in Protocol 1. It is presented as an interactive proof, but can be made non-interactive using the Fiat-Shamir transformation.

Theorem 3. *If* $\mathrm{ZK}^\star$ *has statistical completeness with completeness error* ϵ*, then Protocol 1 has statistical completeness with completeness error* $\epsilon + 2Nq^{-n/2}$*.*

Theorem 4. *If* $\mathrm{ZK}^\star$ *is Special Honest Verifier Zero-Knowledge (SHVZK), then Protocol 1 is also SHVZK.*

Theorem 5. *Suppose that* $\mathrm{ZK}^\star$ *has witness extended emulation with knowledge error* κ*. Then Protocol 1 has witness extended emulation with knowledge error at most* $2(N+1)^2((L-1)N+1)\kappa + ((L+1)N+1)q^{-n/2}$*.*

The proofs of Theorem 3, Theorem 4 and Theorem 5 can be found in Appendix E.

$\mathcal{P}(\mathbf{x}, \mathbf{w})$		$\mathcal{V}(\mathbf{x})$
For $1 \le i \le N$:		
$\sigma_i \leftarrow \texttt{int\_to\_bin}(\pi(i))$		
$\mathbf{P} \leftarrow \mathsf{Commit}^\star(\sigma_1, \ldots, \sigma_N; \mathbf{r_P})$	$\xrightarrow{\quad\mathbf{P}\quad}$	
	$\xleftarrow{\quad\alpha, \beta, \lambda\quad}$	$\alpha, \beta, \lambda \xleftarrow{\$} \mathcal{C}$
$\mathbf{\Lambda} \leftarrow (1, \lambda, \ldots, \lambda^{L-1})$		
For $1 \le i \le N$:		
$z_i \leftarrow \langle \mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger, \mathbf{\Lambda} \rangle$		
$\hat{z}_i \leftarrow \langle \hat{\mathbf{c}}_i, \mathbf{\Lambda} \rangle$		
$h_i \leftarrow z_i + \texttt{int\_to\_bin}(i) \cdot \beta - \alpha$		
$\hat{h}_i \leftarrow \hat{z}_i + \sigma_i \cdot \beta - \alpha$		
For $1 \le i \le N-1$:		
$\theta_i \xleftarrow{\$} \mathcal{R}_q$		
$w_1 \leftarrow \theta_1 h_1$		
For $2 \le i \le N-1$:		
$w_i \leftarrow \theta_{i-1}\hat{h}_i + \theta_i h_i$		
$w_N \leftarrow \theta_{N-1}\hat{h}_N$		
For $1 \le i \le N$:		
Compute $\mathbf{w}_i^\dagger$ s.t. $w_i = \mathbf{G}^{(\lceil \log q \rceil)}\mathbf{w}_i^\dagger$		
$\mathbf{W} \leftarrow \mathsf{Commit}^\star(\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger; \mathbf{r_W})$	$\xrightarrow{\quad\mathbf{W}\quad}$	
	$\xleftarrow{\quad\gamma\quad}$	$\gamma \xleftarrow{\$} \mathcal{C}$
For $1 \le i \le N-1$:		
$u_i \leftarrow (-1)^i \gamma \prod_{j=1}^{i} \frac{\hat{h}_j}{h_j} + \theta_i$		
	$\xrightarrow{\quad u_1, \ldots, u_{N-1}\quad}$	

$$
(\mathrm{ZK}^\star)\left[\begin{array}{c|l}
\begin{array}{c}\mathbf{x} = ((\mathbf{c}_1^\star, \ldots, \mathbf{c}_N^\star),\\ (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N),\\ (u_1, \ldots, u_{N-1}),\\ \mathbf{D}, \mathbf{P}, \mathbf{W}, \alpha, \beta, \lambda, \gamma),\\ \mathbf{w} = ((\mathbf{d}_1^\dagger, \cdots, \mathbf{d}_N^\dagger),\\ (\sigma_1, \ldots, \sigma_N),\\ (\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger),\\ \mathbf{r_D}, \mathbf{r_P}, \mathbf{r_W})\end{array} &
\begin{array}{l}\text{Denoting } h_i, \hat{h}_i \text{ as computed above then}\\
(1)\ \ \gamma\hat{h}_1 + u_1 h_1 = \mathbf{G}^{(\lceil \log q \rceil)}\mathbf{w}_1^\dagger\\
(2)\ \ u_{i-1}\hat{h}_i + u_i h_i = \mathbf{G}^{(\lceil \log q \rceil)}\mathbf{w}_i^\dagger \quad \forall i \in \{2, \ldots, N-1\}\\
(3)\ \ u_{N-1}\hat{h}_N + (-1)^N \gamma h_N = \mathbf{G}^{(\lceil \log q \rceil)}\mathbf{w}_N^\dagger\\
\\
\texttt{is\_bin}(\mathbf{d}_i^\dagger) \wedge \texttt{is\_bin}(\sigma_i) \wedge \texttt{is\_bin}(\mathbf{w}_i^\dagger)\ \forall i \in [N]\\
\wedge \mathsf{Verify}^\star(\mathbf{D}, (\mathbf{d}_1^\dagger, \ldots, \mathbf{d}_N^\dagger), \mathbf{r_D}) = 1\\
\wedge \mathsf{Verify}^\star(\mathbf{P}, (\sigma_1, \ldots, \sigma_N), \mathbf{r_P}) = 1\\
\wedge \mathsf{Verify}^\star(\mathbf{W}, (\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger), \mathbf{r_W}) = 1\end{array}
\end{array}\right]
$$

Protocol 1: Proof of shuffle for one mix server.

5 Mixnet size

We provide an estimation for the size of our decryption mixnet and compare it to the state-of-the-art re-encryption mixnets from Aranha, Baum, Gjøsteen and Silde [4], and Hough, Sandsbråten and Silde [37]. While their proofs of shuffle would have to be updated with the fix we present in Section 4.2, the overhead should not be significant. We start by giving a concrete choice of parameters for our construction in Figure 3.

q	n	k_{LWE}	k_{LWR}	k_{SIS}	η	ζ
3109	256	3	2	3	2	6

Fig. 3. Our choice of parameters aiming for at least 128 bits of security.

The elements that contribute to the size of our decryption mixnet are: (1) the ciphertexts submitted by the users, as well as those outputted by the mixing servers in every layer, and (2) the proofs of correct decryption and shuffle that the mixing servers perform. A separate cost is that of the proofs of correct encryption submitted by the users, that only need to be checked by the mixing servers, and that we discuss at the end. Figure 4 summarizes the comparison with [4, 37]. A more detailed comparison follows.

	Ciphertext	Proof	Total
[4]	80	$290 + 157$	2188
[37]	15	$115 + 85$	875
Our	8	103	444

Fig. 4. Comparison to [4] and [37] in terms of ciphertext, proof, and total mixnet size. All the quantities are in KB per user. The ciphertexts in [4, 37] are constant size, whereas for our work we give the average for 4 layers. The proof size is given per server: in [4, 37] we distinguish between the shuffle & re-randomization proof, and the distributed decryption proof, whereas for our work we report on the single shuffle & decryption proof, giving an average size for 4 layers. The total size is computed for a setting with 4 servers. See the discussion for more details.

Ciphertext size. We recall that the users encrypt their messages in a layered way using HPKE. The PKE scheme HPKE, described in Section 3.2, is a hybrid scheme where a seed $\bar{\mathbf{s}} \in \mathcal{R}_q^{k_{\mathrm{LWR}}}$ is encrypted under the MLWE-based encryption from Section 2.4, resulting in ciphertexts made of $k_{\mathrm{LWR}} \cdot (k_{\mathrm{LWE}} + 1)$ ring elements. The seed is then used as key in OTSE to generate a one-time pad that masks the ciphertext/message from the previous layer. Starting with a message $m \in \mathcal{R}_q$

and performing ρ layers of encryption results in ciphertexts made of $\rho \cdot k_{\text{LWR}} \cdot (k_{\text{LWE}} + 1) + 1$ ring elements. The size of an element in $\mathcal{R}_q = \mathbb{Z}_{3109}[x]/(x^{256} + 1)$ is 384B. This means that the ciphertext starts with the message size of 384B and each layer of encryption increases the size by 3072B. A common scenario of $\rho = 4$ mixing servers, as suggested in [4, 37], results in ciphertexts of 12.7KB. While that is the size of the input ciphertexts to the first mixing server, a layer of encryption of 3KB is removed by each server. This results in an average ciphertext size per layer of around 8KB. We can compare this to the 80KB and 15KB constant-size ciphertexts from the mixnets of [4] and [37].

Proof size (servers). Each of the mixing servers in our decryption mixnet performs a proof of correct shuffle and decryption, essentially proving that their output list of ciphertexts/messages is obtained by removing one layer of encryption and permuting their input list of ciphertexts. We compute the size of these proofs using the zero-knowledge linear proofs from LaZer [43], a library for lattice-based proofs. In a scenario of 4 mixing servers, the proofs that the servers perform have a size per user of 156KB, 120KB, 85KB, and 49KB. This results in an average proof size per server and per user of 103KB. An analogous setting in the re-encryption mixnets of [4, 37] would be composed of 4 mixing servers and 4 decryption servers. Each of the mixing servers in [4] performs a proof of re-randomization and shuffle of 290KB per user, and each of the decryption servers outputs a proof of 157KB per user. For the mixnet of [37] the proof sizes are 115KB and 85KB respectively.[7]

Mixnet size. The size per user of a decryption mixnet with 4 mixing servers is given by the 4 proofs computed by the servers, the 4 layers of ciphertexts, and the finally outputted plaintexts. This gives a total of 444KB for our mixnet. In the re-encryption mixnets of [4] and [37], we need to account for 4 shuffling proofs, 4 decryption proofs and 5 layers of ciphertexts (the input layer, and the output of each mixing server), resulting in a total of 2188KB and 875KB respectively. The plaintexts in these mixnets are an implicit output in the decryption proofs.

We note that a significant improvement can be achieved in all three mixnets by using a succinct proof system like LaBRADOR [11]. Its proofs can aggregate many relations and range from 40KB to 60KB in total, with a very slow asymptotic growth. While LaBRADOR does not directly provide zero-knowledge, [11] states that this property can be achieved by combining their proof system with a simple protocol that masks the witness, and that results in a proof size not much larger than that of the original system. Overall, one could obtain proofs of a few hundred KB accounting for *all* the users in the mixnet. With such proofs, the size of the mixnet would be overwhelmingly dominated by the size of the ciphertexts.

[7] Compared to [37, Table 1], we exclude the ciphertext size from the proof of shuffle, so as to make more explicit the contribution of the ciphertext size to the total size of the mixnet.

Proof size (users). Along with the ciphertext that encrypts their message, each user needs to submit a proof of correct encryption, i.e., a proof of knowledge of the plaintext contained in the layered encryption. These proofs are only present at the beginning and are checked by all the servers. Each proof involves several linear relations per layer of encryption, that we again implement in LaZer [43], obtaining a proof size per user of 343KB. While these proofs are noticeably large, we stress that this cost is of a different kind than the ones outlined before. The proofs of correct encryption only need to be checked by the mixing servers, so as to make sure that no ciphertext is malformed; or else decryption could potentially leak information about their secret keys. If some user or third party wants to verify the correctness of the procedure of the mixnet, only the proofs of correct shuffle and decryption are relevant. For this reason, the proofs of plaintext knowledge can be deleted once the servers have verified them. We also note that the size of these proofs could be decreased by using a succinct proof system like LaBRADOR [11]. As for [4, 37], they do not report on the sizes of their so-called ballot proofs, but they should be smaller because of the single layer of encryption of the simpler schemes that they use.

Acknowledgments. We would like to thank Katerina Sotiraki for the observation that the polynomial approach to proving permutations does not work over rings which are not unique factorization domains. We also thank the anonymous reviewers from PKC2025 for their useful feedback. This work was supported by the EU H2020 ERC Project 101002845 PLAZA.

References

1. Abe, M.: Mix-networks on permutation networks. In: Lam, K.Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT'99. LNCS, vol. 1716, pp. 258–273. Springer, Berlin, Heidelberg (Nov 1999). https://doi.org/10.1007/978-3-540-48000-6_21
2. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation networks. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 317–324. Springer, Berlin, Heidelberg (Feb 2001). https://doi.org/10.1007/3-540-44586-2_23
3. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., van Emde Boas, P., Nielsen, M. (eds.) ICALP 99. LNCS, vol. 1644, pp. 1–9. Springer, Berlin, Heidelberg (Jul 1999). https://doi.org/10.1007/3-540-48523-6_1
4. Aranha, D.F., Baum, C., Gjøsteen, K., Silde, T.: Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. p. 1467–1481. CCS '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3616683, https://doi.org/10.1145/3576915.3616683
5. Aranha, D.F., Baum, C., Gjøsteen, K., Silde, T., Tunge, T.: Lattice-based proof of shuffle and applications to electronic voting. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 227–251. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75539-3_10
6. Attema, T., Cramer, R., Kohl, L.: A compressed Σ-protocol theory for lattices. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826,

pp. 549–579. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_19
7. Baum, C., Bootle, J., Cerulli, A., del Pino, R., Groth, J., Lyubashevsky, V.: Sublinear lattice-based zero-knowledge arguments for arithmetic circuits. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 669–699. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_23
8. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Berlin, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_17
9. Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 201–218. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_13
10. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU prime: Reducing attack surface at low cost. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 235–260. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-72565-9_12
11. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from module-SIS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 518–548. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_17
12. Biagioni, S., Masny, D., Venturi, D.: Naor-Yung paradigm with shared randomness and applications. In: Zikas, V., De Prisco, R. (eds.) SCN 16. LNCS, vol. 9841, pp. 62–80. Springer, Cham (Aug / Sep 2016). https://doi.org/10.1007/978-3-319-44618-9_4
13. Bootle, J., Chiesa, A., Sotiraki, K.: Sumcheck arguments and their applications. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 742–773. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_26
14. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018). https://doi.org/10.1109/EuroSP.2018.00032
15. Boudgoust, K., Scholl, P.: Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 371–404. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8721-4_12
16. Boyen, X., Haines, T., Müller, J.: A verifiable and practical lattice-based decryption mix net with external auditing. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 336–356. Springer, Cham (Sep 2020). https://doi.org/10.1007/978-3-030-59013-0_17
17. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS 2012. pp. 309–325. ACM (Jan 2012). https://doi.org/10.1145/2090236.2090262
18. Chaum, D., Das, D., Javani, F., Kate, A., Krasnova, A., de Ruiter, J., Sherman, A.T.: cMix: Mixing with minimal real-time asymmetric cryptographic operations. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17International Conference on Applied Cryptography and Network Security. LNCS, vol. 10355, pp. 557–578. Springer, Cham (Jul 2017). https://doi.org/10.1007/978-3-319-61204-1_28

19. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (feb 1981). https://doi.org/10.1145/358549.358563, https://doi.org/10.1145/358549.358563
20. Chowdhury, S., Sinha, S., Singh, A., Mishra, S., Chaudhary, C., Patranabis, S., Mukherjee, P., Chatterjee, A., Mukhopadhyay, D.: Efficient threshold FHE with application to real-time systems. Cryptology ePrint Archive, Report 2022/1625 (2022), https://eprint.iacr.org/2022/1625
21. Costa, N., Martínez, R., Morillo, P.: Proof of a shuffle for lattice-based cryptography. In: Lipmaa, H., Mitrokotsa, A., Matulevičius, R. (eds.) NordSec 2017. LNCS, vol. 10674, pp. 280–296. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70290-2_17
22. Costa, N., Martínez, R., Morillo, P.: Lattice-based proof of a shuffle. In: Bracciali, A., Clark, J., Pintore, F., Rønne, P.B., Sala, M. (eds.) FC 2019 Workshops. LNCS, vol. 11599, pp. 330–346. Springer, Cham (Feb 2019). https://doi.org/10.1007/978-3-030-43725-1_23
23. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003). https://doi.org/10.1137/S0097539702403773
24. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type III anonymous remailer protocol. In: 2003 IEEE Symposium on Security and Privacy. pp. 2–15. IEEE Computer Society Press (May 2003). https://doi.org/10.1109/SECPRI.2003.1199323
25. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Berlin, Heidelberg (Aug 1984). https://doi.org/10.1007/3-540-39568-7_2
26. Farzaliyev, V., Willemson, J., Kaasik, J.K.: Improved lattice-based mix-nets for electronic voting. In: Park, J.H., Seo, S.H. (eds.) ICISC 21. LNCS, vol. 13218, pp. 119–136. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-031-08896-4_6
27. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
28. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Transactions **88-A**, 172–188 (2005). https://doi.org/10.1093/ietfec/E88-A.1.172
29. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Berlin, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_22
30. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Berlin, Heidelberg (Jan 2003). https://doi.org/10.1007/3-540-36288-6_11
31. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Berlin, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_22
32. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Berlin, Heidelberg (Apr 2007). https://doi.org/10.1007/978-3-540-71677-8_25
33. Haines, T., Müller, J.: SoK: Techniques for verifiable mix nets. In: Jia, L., Küsters, R. (eds.) CSF 2020 Computer Security Foundations Symposium. pp. 49–64. IEEE Computer Society Press (2020). https://doi.org/10.1109/CSF49147.2020.00012

34. Herranz, J., Hofheinz, D., Kiltz, E.: Some (in)sufficient conditions for secure hybrid encryption. Inf. Comput. **208**(11), 1243–1257 (nov 2010). https://doi.org/10.1016/j.ic.2010.07.002
35. Herranz, J., Martínez, R., Sánchez, M.: Shorter lattice-based zero-knowledge proofs for the correctness of a shuffle. In: Bernhard, M., Bracciali, A., Gudgeon, L., Haines, T., Klages-Mundt, A., Matsuo, S., Perez, D., Sala, M., Werner, S. (eds.) FC 2021 Workshops. LNCS, vol. 12676, pp. 315–329. Springer, Berlin, Heidelberg (Mar 2021). https://doi.org/10.1007/978-3-662-63958-0_27
36. van den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: scalable private messaging resistant to traffic analysis. In: Proceedings of the 25th Symposium on Operating Systems Principles. p. 137–152. SOSP '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2815400.2815417
37. Hough, P., Sandsbråten, C., Silde, T.: More efficient lattice-based electronic voting from NTRU. IACR Communications in Cryptology **1**(4) (2025). https://doi.org/10.62056/a69qudhdj
38. Jakobsson, M., M'Raïhi, D.: Mix-based electronic payments. In: Tavares, S.E., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 157–173. Springer, Berlin, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48892-8_13
39. Kwon, A., Corrigan-Gibbs, H., Devadas, S., Ford, B.: Atom: Horizontally scaling strong anonymity. In: Proceedings of the 26th Symposium on Operating Systems Principles. p. 406–422. SOSP '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132755
40. Lyubashevsky, V.: Basic lattice cryptography: The concepts behind kyber (ML-KEM) and dilithium (ML-DSA). Cryptology ePrint Archive, Report 2024/1287 (2024), https://eprint.iacr.org/2024/1287
41. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 71–101. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_3
42. 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.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 204–224. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_8
43. Lyubashevsky, V., Seiler, G., Steuer, P.: The LaZer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 3125–3137. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3690330
44. Micciancio, D., Suhl, A.: Simulation-secure threshold PKE from LWE with polynomial modulus. Cryptology ePrint Archive, Paper 2023/1728 (2023), https://eprint.iacr.org/2023/1728
45. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press (May 1990). https://doi.org/10.1145/100216.100273
46. National Institute of Standards and Technology (NIST): Module-lattice-based digital signature standard. Tech. Rep. Federal Information Processing Standards Publications (FIPS PUBS) 204, U.S. Department of Commerce, Washington, D.C. (2024). https://doi.org/10.6028/NIST.FIPS.204
47. National Institute of Standards and Technology (NIST): Module-lattice-based key-encapsulation mechanism standard. Tech. Rep. Federal Information Processing Standards Publications (FIPS PUBS) 203, U.S. Department of Commerce, Washington, D.C. (2024). https://doi.org/10.6028/NIST.FIPS.203

48. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Reiter, M.K., Samarati, P. (eds.) ACM CCS 2001. pp. 116–125. ACM Press (Nov 2001). https://doi.org/10.1145/501983.502000
49. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT'93. LNCS, vol. 765, pp. 248–259. Springer, Berlin, Heidelberg (May 1994). https://doi.org/10.1007/3-540-48285-7_21
50. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.J. (eds.) EUROCRYPT'95. LNCS, vol. 921, pp. 393–403. Springer, Berlin, Heidelberg (May 1995). https://doi.org/10.1007/3-540-49264-X_32
51. Sotiraki, K.: Personal communication with Jonathan Bootle (January 2022)
52. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 27–47. Springer, Berlin, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_4
53. Strand, M.: A verifiable shuffle for the GSW cryptosystem. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) FC 2018 Workshops. LNCS, vol. 10958, pp. 165–180. Springer, Berlin, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-662-58820-8_12
54. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 10. LNCS, vol. 6055, pp. 100–113. Springer, Berlin, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-12678-9_7
55. Wikström, D.: A commitment-consistent proof of a shuffle. In: Boyd, C., Nieto, J.M.G. (eds.) ACISP 09. LNCS, vol. 5594, pp. 407–421. Springer, Berlin, Heidelberg (Jul 2009). https://doi.org/10.1007/978-3-642-02620-1_28

Appendix

A Cryptographic Primitives

A.1 One-time Symmetric Encryption

Definition 4. *A **one-time Symmetric Encryption (SE) scheme** SE, defined over a key space $\mathcal{K}$, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, consists of a triple of algorithms $\mathsf{SE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:*

- $\mathsf{KGen}()$*: key generation is a PPT algorithm that outputs a key* $\mathsf{K} \in \mathcal{K}$.
- $\mathsf{Enc}_{\mathsf{K}}(m)$*: encryption is a deterministic polynomial time algorithm that takes a key* $\mathsf{K} \in \mathcal{K}$ *and a message* $m \in \mathcal{M}$ *and outputs a ciphertext* $c \in \mathcal{C}$.
- $\mathsf{Dec}_{\mathsf{K}}(c)$*: decryption is a deterministic polynomial time that takes a key* $\mathsf{K} \in \mathcal{K}$ *and a ciphertext* $c \in \mathcal{C}$*, and outputs a message* $m \in \mathcal{M}$ *or an error, denoted by* $\perp$.

The scheme SE *is said to be **correct** if for all keys* K *output by* KGen *and for all messages* $m \in \mathcal{M}$ *it holds that*

$$\mathsf{Dec}_{\mathsf{K}}(\mathsf{Enc}_{\mathsf{K}}(m)) = m \ .$$

We also write $\mathsf{SE.KGen}$, $\mathsf{SE.Enc}$ and $\mathsf{SE.Dec}$ to make SE more explicit.

A standard notion for a one-time SE is that of IND-OT security, where the adversary submits two equal-length messages, is given the encryption of one of them, and then needs to determine which one was encrypted. Since we are dealing with a *one-time* scheme, we only allow the adversary to submit messages once. This notion is captured in the Game IND-OT$(\mathcal{A}, \mathsf{SE})$, depicted in Figure 5.

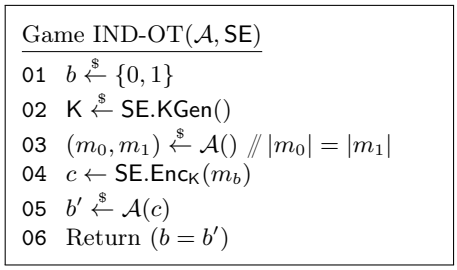

Fig. 5. Game capturing the IND-OT security of a one-time SE scheme.

We define the advantage of an adversary $\mathcal{A}$ in this game as follows:

$$\mathrm{Adv}^{\text{IND-OT}}_{\mathsf{SE}}(\mathcal{A}) := 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right| \ .$$

Definition 5. *A one-time SE scheme* SE *is said to be* ***IND-OT secure*** *if, for all PPT adversaries, the advantage* $\mathrm{Adv}^{IND\text{-}OT}_{\mathsf{SE}}(\mathcal{A})$ *is negligible.*

A.2 Public-Key Encryption

Definition 6. *A* ***Public-Key Encryption (PKE) scheme*** PKE*, defined over a key space* $\mathcal{SK} \times \mathcal{PK}$*, message space* $\mathcal{M}$ *and ciphertext space* $\mathcal{C}$*, consists of a triple of algorithms* $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following syntax:*

- $\mathsf{KGen}()$*: key generation is a PPT algorithm that outputs a key-pair consisting of secret and public key* $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{SK} \times \mathcal{PK}$*.*
- $\mathsf{Enc}_{\mathsf{pk}}(m)$*: encryption is a PPT algorithm that takes a public key* $\mathsf{pk} \in \mathcal{PK}$ *and a message* $m \in \mathcal{M}$ *and outputs a ciphertext* $c \in \mathcal{C}$*.*
- $\mathsf{Dec}_{\mathsf{sk}}(c)$*: decryption is a deterministic polynomial time algorithm that takes a secret key* $\mathsf{sk} \in \mathcal{SK}$ *and a ciphertext* $c \in \mathcal{C}$ *and outputs a message* $m \in \mathcal{M}$ *or an error, denoted by* $\perp$*.*

The scheme PKE *is said to be* ***approximately correct*** *if for all* $m \in \mathcal{M}$*,*

$$\Pr[m = m' \ : \ (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(); \ c \xleftarrow{\$} \mathsf{Enc}_{\mathsf{pk}}(m); \ m' \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)] \approx 1.$$

Game IND-CPA($\mathcal{A}$, PKE)

01 $b \xleftarrow{\$} \{0,1\}$
02 $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{PKE.KGen}()$
03 $(m_0, m_1) \xleftarrow{\$} \mathcal{A}(\mathsf{pk})$ // $|m_0| = |m_1|$
04 $c \xleftarrow{\$} \mathsf{PKE.Enc}_{\mathsf{pk}}(m_b)$
05 $b' \xleftarrow{\$} \mathcal{A}(\mathsf{pk}, c)$
06 Return $(b = b')$

Fig. 6. Game capturing the IND-CPA security of a PKE scheme.

We also write PKE.KGen, PKE.Enc and PKE.Dec to make PKE more explicit.

A standard security notion for PKE schemes is that of IND-CPA security. Accordingly, we consider the Game IND-CPA depicted in Figure 6.

We define the advantage of an adversary $\mathcal{A}$ in this game as follows:

$$\mathrm{Adv}^{\text{IND-CPA}}_{\mathsf{PKE}}(\mathcal{A}) := 2 \cdot \left| \Pr[b = b'] - \frac{1}{2} \right| \ .$$

Definition 7. *A public-key encryption scheme* PKE *is said to be* ***IND-CPA secure*** *if, for all PPT adversaries $\mathcal{A}$, the advantage* $\mathrm{Adv}^{IND\text{-}CPA}_{\mathsf{PKE}}(\mathcal{A})$ *is negligible.*

A.3 Commitment Schemes

Definition 8. *A* ***commitment scheme*** Com *is a triple of algorithms* Com = (Setup, Commit, Verify) *with the following syntax:*

- Setup()*: a PPT algorithm that outputs public parameters pp describing a message space S_M, randomness space S_R, commitment space S_C, and an efficiently sampleable probability distribution χ over S_R.*
- Commit$(pp, m; r)$*: a PPT algorithm that takes public parameters pp, a message $m \in S_M$ and samples randomness $r \xleftarrow{\$} \chi$. The output is a commitment $c \in S_C$.*
- Verify(pp, c, m, r)*: a deterministic polynomial time algorithm that takes public parameters pp, a commitment $c \in S_C$, a message $m \in S_M$ and randomness $r \in S_R$. The output is a bit $b \in \{0,1\}$ that indicates "accept" when $b = 1$ and "reject" otherwise.*

For succinctness, we will sometimes omit the public parameters *pp* as input to Commit and Verify, but they will implicitly receive them.

Definition 9 (Correctness). *A commitment scheme* Com =(Setup, Commit, Verify) *is said to be* ***correct*** *if for all adversaries $\mathcal{A}$,*

$$\Pr[\mathsf{Verify}(c, m, r) = 1 \ : \ pp \xleftarrow{\$} \mathsf{Setup}(); \ m \xleftarrow{\$} \mathcal{A}(pp); \ c \xleftarrow{\$} \mathsf{Commit}(pp, m; r)] = 1.$$

Definition 10 (Hiding). *A commitment scheme* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ *is said to be computationally* ***hiding*** *if for all stateful PPT adversaries* $\mathcal{A}$

$$\left| \Pr \left[\begin{array}{c} b = b' \ : \ pp \xleftarrow{\$} \mathsf{Setup}(); (m_0, m_1) \xleftarrow{\$} \mathcal{A}(pp); \\ b \xleftarrow{\$} \{0,1\}; \ c \xleftarrow{\$} \mathsf{Commit}(pp, m_b; r); \ b' \xleftarrow{\$} \mathcal{A}(c) \end{array} \right] - \frac{1}{2} \right| \approx 0 \ .$$

Definition 11 (Binding). *A commitment scheme* $\mathsf{Com} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ *is said to be computationally* ***binding*** *if for all PPT adversaries* $\mathcal{A}$

$$\Pr \left[\begin{array}{c} m_0 \neq m_1 \wedge \mathsf{Verify}(c, m_0, r_0) = \mathsf{Verify}(c, m_1, r_1) = 1 \\ : \ pp \xleftarrow{\$} \mathsf{Setup}(); \ (m_0, r_0, m_1, r_1, c) \xleftarrow{\$} \mathcal{A}(pp) \end{array} \right] \approx 0 \ .$$

Moreover, Com *is strongly computationally binding if we replace* $m_0 \neq m_1$ *in the definition above with* $(m_0, r_0) \neq (m_1, r_1)$.

A.4 Proof Systems

We follow the presentation in [7]. Let R be a ternary relation, where the first input will contain some public parameters pp. If $(pp, \mathbf{x}, \mathbf{w}) \in \mathsf{R}$, we say that $\mathbf{w}$ is a *witness* for the instance $\mathbf{x}$.

A proof system $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ for the relation R consists of a parameter generator $\mathcal{G}$, and two interactive and stateful PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ called the prover and verifier. We denote the interaction between prover and verifier by $(\Sigma, b) \xleftarrow{\$} \langle \mathcal{P}(pp, \mathbf{x}, \mathbf{w}), \mathcal{V}(pp, \mathbf{x}) \rangle$, where the output consists of the communication transcript Σ and the verifier's decision bit b, with $b = 0$ meaning reject and $b = 1$ meaning accept.

We introduce now the notions of completeness, knowledge soundness, special soundness, and zero-knowledge for a proof system.

Definition 12 (Statistical completeness). *The proof system* Π *has* ***statistical completness*** *with completeness error* $\epsilon : \mathbb{N} \to [0,1]$ *if for all adversaries* $\mathcal{A}$*, we have that*

$$\Pr \left[\begin{array}{c} (pp, \mathbf{x}, \mathbf{w}) \in \mathsf{R} \wedge b = 0 \ : \\ pp \xleftarrow{\$} \mathcal{G}(); \ (\mathbf{x}, \mathbf{w}) \xleftarrow{\$} \mathcal{A}(pp); \ (\Sigma, b) \xleftarrow{\$} \langle \mathcal{P}(pp, \mathbf{x}, \mathbf{w}), \mathcal{V}(pp, \mathbf{x}) \rangle \end{array} \right] \leq \epsilon(\omega) \ .$$

Definition 13 (Witness-extended emulation). *The proof system* Π *has* ***witness-extended emulation*** *with knowledge soundness error* $\kappa : \mathbb{N} \to [0,1]$ *if for all deterministic polynomial time* $\mathcal{P}^*$*, there exists an expected polynomial time extractor* $\mathcal{E}$ *such that for all deterministic polynomial-time stateful adversaries* $\mathcal{A}$*, we have*

$$\left| \Pr \left[\begin{array}{c} \mathcal{A}(\Sigma) = 1 : \ pp \xleftarrow{\$} \mathcal{G}(); \ (\mathbf{x}, \mathbf{w}') \xleftarrow{\$} \mathcal{A}(pp); \\ (\Sigma, b) \xleftarrow{\$} \langle \mathcal{A}(pp, \mathbf{x}, \mathbf{w}'), \mathcal{V}(pp, \mathbf{x}) \rangle \end{array} \right] \right.$$
$$\left. - \Pr \left[\begin{array}{c} \mathcal{A}(\Sigma) = 1 \text{ and if } \Sigma \text{ is accepting then } (pp, \mathbf{x}, \mathbf{w}) \in \mathsf{R} : \\ pp \xleftarrow{\$} \mathcal{G}(); \ (\mathbf{x}, \mathbf{w}') \xleftarrow{\$} \mathcal{A}(pp); (\Sigma, \mathbf{w}) \xleftarrow{\$} \mathcal{E}^{\mathcal{A}(pp, \mathbf{x}, \mathbf{w}')}(pp, \mathbf{x}, \mathbf{w}') \end{array} \right] \right| \leq \kappa(\omega) \ ,$$

where the superscript on the extractor $\mathcal{E}$ *means that* $\mathcal{E}$ *has oracle access to* $\mathcal{P}^*$*, i.e., that* $\mathcal{E}$ *can execute* $\mathcal{P}^*$ *as a subroutine for challenges of its choice.*

Definition 14 (Argument of knowledge). *The proof system $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is called an* ***argument of knowledge*** *for the relation R if it is statistically complete and computationally knowledge sound.*

We say that the protocol is *public coin* if all the messages sent by the verifier are its challenges, chosen uniformly at random independently of the prover's messages.

Definition 15 (Tree of transcripts). *An $(k_1, \ldots, k_t)$-**tree of transcripts** for a public-coin protocol is a set of $\prod_{i=1}^{t} k_i$ transcripts arranged in a tree such that vertices correspond to prover messages and edges correspond to verifier challenges. Each node at depth i has k_i child edges labelled with distinct challenges. This way, each transcript corresponds to exactly one root-to-leaf path. We say that the tree is **accepting** if the verifier would have accepted every transcript.*

Lemma 6 ([13], adapted from [6, Lemma 5]). *Let $k_1, \ldots, k_t \colon \mathbb{N} \to \mathbb{N}$ be functions such that $k := \prod_{i=1}^{t} k_i$ is upper bounded by a polynomial. Let Π be a $(2t+1)$-message public-coin interactive argument in which the verifier $\mathcal{V}$ samples each challenge uniformly at random from a challenge set of size $K \geq \max_{i \in [t]} k_i$. There is an algorithm Tree such that for any malicious prover $\mathcal{A}$ for Π that makes $\mathcal{V}$ accept with probability at least ϵ, $\mathsf{Tree}^{\mathcal{A}}(pp, \mathbf{x})$ runs in expected time at most K (where running $\mathcal{P}^*$ takes unit time) and produces a $(k_1, \ldots, k_t)$-tree of accepting transcripts for $\mathbf{x}$ with probability at least $\epsilon - \frac{\sum_{i=1}^{t}(k_i - 1)}{K}$ (otherwise producing a tree that is incomplete or contains non-accepting transcripts). Further, the first transcript produced by Tree (corresponding to the first leaf in the tree) is distributed according to $\langle \mathcal{A}, \mathcal{V} \rangle$.*

Definition 16 (Special honest-verifier zero-knowledge). *A public-coin argument of knowledge $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ is said to be **statistical special honest-verifier zero-knowledge** (SHVZK) if there exists a PPT simulator $\mathcal{S}$ such that for all interactive and stateful adversaties $\mathcal{A}$, the difference between the probabilities*

$$\Pr\left[\begin{array}{c} (pp, \mathbf{x}, \mathbf{w}) \in \mathsf{R} \wedge b' = 1 \ : \ pp \xleftarrow{\$} \mathcal{G}(); \\ (\mathbf{x}, \mathbf{w}, \xi) \xleftarrow{\$} \mathcal{A}(pp); \ (\Sigma, b) \xleftarrow{\$} \langle \mathcal{P}(pp, \mathbf{x}, \mathbf{w}), \mathcal{V}(pp, \mathbf{x}; \xi) \rangle \ ; \ b' \xleftarrow{\$} \mathcal{A}(\Sigma) \end{array}\right]$$

and

$$\Pr\left[\begin{array}{c} (pp, \mathbf{x}, \mathbf{w}) \in \mathsf{R} \wedge b' = 1 \ : \ pp \xleftarrow{\$} \mathcal{G}(); \\ (\mathbf{x}, \mathbf{w}, \xi) \xleftarrow{\$} \mathcal{A}(pp); \ \Sigma \xleftarrow{\$} \mathcal{S}(pp, \mathbf{x}, \xi); \ b' \xleftarrow{\$} \mathcal{A}(\Sigma) \end{array}\right]$$

is negligible, where ξ denotes the randomness used by the verifier.

So far we have been referring to *interactive* argument systems. A standard approach to turn public coin interactive protocols into *non-interactive* ones is via the Fiat-Shamir transform [27]. The idea is that the prover can compute each challenge as a hash of the instance, previous messages, public parameters, etc. The resulting proof consists of all the messages, that the verifier can use along with the instance to recompute the challenges and perform the same checks that it would perform in the interactive version of the protocol.

B Ajtai Commitments

Theorem 6. *The commitment scheme* $\mathsf{Com}^\star$ *is correct.*

Proof. The equation $\mathbf{c} = \mathbf{A_1 m} + \mathbf{A_2 r}$ is trivially verified by a properly computed commitment. As for the L_2 norm, a correct message-randomness vector with maximum L_2 norm would be, for instance, one where all the coefficients in $\mathbf{m}$ are set to 1 and all the coefficients in $\mathbf{r}$ are set to η. The norm of such a vector is exactly $\sqrt{nL + n2k_{\text{SIS}}\eta^2}$, which is the value that B is set to be. □

Theorem 7. *If the problem* $\text{MSIS}_{k_{\text{SIS}},L+2k_{\text{SIS}},2B}$ *is hard, then the scheme* $\mathsf{Com}^\star$ *is strongly computationally binding.*

Proof. Suppose a PPT adversary $\mathcal{A}$ outputs $(c, m_0, r_0, m_1, r_1) \xleftarrow{\$} \mathcal{A}(pp)$ such that

$$\mathsf{Verify}^\star(\mathbf{m_0}, \mathbf{r_0}, \mathbf{c}) = \mathsf{Verify}^\star(\mathbf{m_1}, \mathbf{r_1}, \mathbf{c}) = 1 \ ,$$

i.e.,

$$\left\| \begin{pmatrix} \mathbf{m_0} \\ \mathbf{r_0} \end{pmatrix} \right\| \leq B \ \wedge \ \left\| \begin{pmatrix} \mathbf{m_1} \\ \mathbf{r_1} \end{pmatrix} \right\| \leq B$$

and

$$\mathbf{A_1 m_0} + \mathbf{A_2 r_0} = \mathbf{c} = \mathbf{A_1 m_1} + \mathbf{A_2 r_1} \ .$$

Then,

$$(\mathbf{A_1} \mid \mathbf{A_2}) \begin{pmatrix} \mathbf{m_0} - \mathbf{m_1} \\ \mathbf{r_0} - \mathbf{r_1} \end{pmatrix} = \mathbf{0} \ ,$$

where $(\mathbf{A_1} \mid \mathbf{A_2}) \in \mathcal{R}_q^{k_{\text{SIS}} \times (L+2k_{\text{SIS}})}$ is a uniformly random matrix, and

$$\left\| \begin{pmatrix} \mathbf{m_0} - \mathbf{m_1} \\ \mathbf{r_0} - \mathbf{r_1} \end{pmatrix} \right\| \leq 2B \ .$$

Therefore, the adversary has obtained a solution for $\text{MSIS}_{k_{\text{SIS}},L+2k_{\text{SIS}},2B}$. □

Theorem 8. *If the problem* $\text{MLWE}_{k_{\text{SIS}},k_{\text{SIS}},\eta}$ *is hard, then the scheme* $\mathsf{Com}^\star$ *is computationally hiding.*

Proof. Given $\mathbf{A_2} \xleftarrow{\$} \mathcal{R}_q^{k_{\text{SIS}} \times 2k_{\text{SIS}}}$ and $\mathbf{r} \xleftarrow{\$} \mathsf{B}_\eta^{2k_{\text{SIS}}}$, the value $(\mathbf{A_2}, \mathbf{A_2 r})$ is indistinguishable from random based on the hardness of $\text{MLWE}_{k_{\text{SIS}},k_{\text{SIS}},\eta}$.

Now if $\mathbf{A_2 r}$ is indistinguishable from random, so is the commitment

$$\mathbf{c} \leftarrow \mathbf{A_1 m} + \mathbf{A_2 r} \ ,$$

independently of the message $\mathbf{m}$. □

C Proof of Theorem 2

Theorem 2. *If the problems* $\mathrm{MLWR}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}$ *and* $\mathrm{MLWE}_{L,k_{\mathrm{LWE}},\eta}$ *are hard, then* OTSE *is IND-OT secure. In particular, given a PPT adversary* $\mathcal{A}$ *against the IND-OT security of* OTSE*, we construct PPT algorithms* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *such that*

$$\mathrm{Adv}^{\mathit{IND\text{-}OT}}_{\mathsf{OTSE}}(\mathcal{A}) \leq 2\left(\mathrm{Adv}^{\mathrm{MLWR}}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}(\mathcal{B}_1) + \mathrm{Adv}^{\mathrm{MLWE}}_{L,k_{\mathrm{LWE}},\eta}(\mathcal{B}_2)\right) \ .$$

Proof. We will follow a game-hopping proof strategy. Let $\mathsf{G}_0 :=$ Game IND-OT$(\mathcal{A}, \mathsf{OTSE})$. This game is depicted in Figure 7. We modify it into games G_1 and G_2, depicted in Figure 8.

Game G_0

$b \xleftarrow{\$} \{0,1\}$
$\bar{\mathbf{s}} \xleftarrow{\$} \{0,1\}^{k_{\mathrm{LWR}}}$
$(m_0, m_1) \xleftarrow{\$} \mathcal{A}() \ /\!/ \ m_0, m_1 \in \mathcal{R}_q^L$
$\mathbf{d} \leftarrow \lceil \mathbf{H}\bar{\mathbf{s}} \rfloor_{2^\zeta \to 2^{2\eta}}$
$(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d}, 2\eta)$
$\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)})$
$\mathbf{a} \leftarrow \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger$
$\mathbf{c}^\star \leftarrow m_b + \mathbf{a}$
$b' \xleftarrow{\$} \mathcal{A}(\mathbf{c}^\star)$
Return $(b = b')$

Fig. 7. Game G_0 is the Game IND-OT$(\mathcal{A}, \mathsf{OTSE})$.

We can express the advantage of the adversary $\mathcal{A}$ in breaking the IND-OT security of OTSE in terms of the described games:

$$\begin{aligned}
\frac{1}{2}\mathrm{Adv}^{\mathrm{IND\text{-}OT}}_{\mathsf{OTSE}}(\mathcal{A}) &= \left|\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_0] - \frac{1}{2}\right| \\
&\leq |\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_0] - \Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_1]| \\
&+ |\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_1] - \Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_2]| \\
&+ \left|\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_2] - \frac{1}{2}\right| \ . \qquad (7)
\end{aligned}$$

We proceed to analyze each of the terms separately.

Game $\mathsf{G}_0 \to$ Game G_1.

We describe in Fig. 9 an adversary $\mathcal{B}_1$ against MLWR that runs $\mathcal{A}$ as a subroutine and whose advantage is

$$\mathrm{Adv}^{\mathrm{MLWR}}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}(\mathcal{B}_1) = |\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_0] - \Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_1]| \ .$$

The adversary $\mathcal{B}_1$ receives as input $(\mathbf{A}, \mathbf{t}) \xleftarrow{\$} \mathcal{D}_{\bar{b}}$ for some bit $\bar{b}$, where

Game G_1

$b \xleftarrow{\$} \{0,1\}$
$\bar{\mathbf{s}} \xleftarrow{\$} \{0,1\}^{k_{\mathrm{LWR}}}$
$(m_0, m_1) \xleftarrow{\$} \mathcal{A}() \;/\!/\; m_0, m_1 \in \mathcal{R}_q^L$
$\boxed{\mathbf{d} \xleftarrow{\$} \mathcal{R}_{2^{2\eta}}^{k_{\mathrm{LWE}}+L}}$
$(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d}, 2\eta)$
$\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)})$
$\mathbf{a} \leftarrow \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger$
$\mathbf{c}^\star \leftarrow m_b + \mathbf{a}$
$b' \xleftarrow{\$} \mathcal{A}(\mathbf{c}^\star)$
Return $(b = b')$

Game G_2

$b \xleftarrow{\$} \{0,1\}$
$\bar{\mathbf{s}} \xleftarrow{\$} \{0,1\}^{k_{\mathrm{LWR}}}$
$(m_0, m_1) \xleftarrow{\$} \mathcal{A}() \;/\!/\; m_0, m_1 \in \mathcal{R}_q^L$
$\mathbf{d} \xleftarrow{\$} \mathcal{R}_{2^{2\eta}}^{k_{\mathrm{LWE}}+L}$
$(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d}, 2\eta)$
$\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)})$
$\boxed{\mathbf{a} \xleftarrow{\$} \mathcal{R}_q^L}$
$\mathbf{c}^\star \leftarrow m_b + \mathbf{a}$
$b' \xleftarrow{\$} \mathcal{A}(\mathbf{c}^\star)$
Return $(b = b')$

Fig. 8. Games G_1 and G_2. The squared boxes mark the modified line with respect to the previous game.

Adversary $\mathcal{B}_1(\mathbf{A}, \mathbf{t})$

Instantiate OTSE with $\mathbf{H} \leftarrow \mathbf{A}$.
$b \xleftarrow{\$} \{0,1\}$
$(m_0, m_1) \xleftarrow{\$} \mathcal{A}() \;/\!/\; m_0, m_1 \in \mathcal{R}_q^L$
$\mathbf{d} \leftarrow \mathbf{t}$
$(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d}, 2\eta)$
$\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(2\eta-1)})$
$\mathbf{a} \leftarrow \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger$
$\mathbf{c}^\star \leftarrow m_b + \mathbf{a}$
$b' \xleftarrow{\$} \mathcal{A}(\mathbf{c}^\star)$
Return $(b \oplus b')$

Fig. 9. Adversary $\mathcal{B}_1$ against MLWR. The input $(\mathbf{A}, \mathbf{t})$ is the challenge received from the MLWR challenger.

- $\mathcal{D}_0 := \{(\mathbf{A}, \lceil \mathbf{A}\mathbf{s} \rfloor_{2^\zeta \to 2^{2\eta}}) \;:\; \mathbf{A} \xleftarrow{\$} \mathcal{R}_{2^\zeta}^{(k_{\mathrm{LWE}}+L)\times k_{\mathrm{LWR}}};\ \mathbf{s} \xleftarrow{\$} \{0,1\}^{k_{\mathrm{LWR}}}\}$, and
- $\mathcal{D}_1 := \{(\mathbf{A}, \lceil \mathbf{u} \rfloor_{2^\zeta \to 2^{2\eta}}) \;:\; \mathbf{A} \xleftarrow{\$} \mathcal{R}_{2^\zeta}^{(k_{\mathrm{LWE}}+L)\times k_{\mathrm{LWR}}};\ \mathbf{u} \xleftarrow{\$} \mathcal{R}_{2^\zeta}^{k_{\mathrm{LWE}}+L}\}$

and needs to guess $\bar{b}$. Note that since $2^{2\eta}$ divides 2^ζ, the element $\mathbf{u}$ can be directly sampled in the target ring, so we can equivalently express $\mathcal{D}_1$ as

$$\mathcal{D}_1 = \{(\mathbf{A}, \mathbf{u}) \;:\; \mathbf{A} \xleftarrow{\$} \mathcal{R}_{2^\zeta}^{(k_{\mathrm{LWE}}+L)\times k_{\mathrm{LWR}}};\ \mathbf{u} \xleftarrow{\$} \mathcal{R}_{2^{2\eta}}^{k_{\mathrm{LWE}}+L}\}.$$

Recall that the matrix $\mathbf{H}$ in OTSE is sampled uniformly at random. Hence, when $\bar{b}$ is 0, the distribution of $(\mathbf{A}, \mathbf{t})$ is exactly that of $(\mathbf{H}, \mathbf{d})$ in G_0, whereas when $\bar{b}$ is 1, it is exactly that of $(\mathbf{H}, \mathbf{d})$ in G_1. This means that $\mathcal{B}_1$ perfectly simulates G_0 when $\bar{b} = 0$ and G_1 when $\bar{b} = 1$.

Therefore, denoting by $\bar{b}'$ the guess of $\mathcal{B}_1$, we have that

$$\begin{aligned}\mathrm{Adv}^{\mathrm{MLWR}}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}(\mathcal{B}_1) &= \left|\Pr[\bar{b}'=0 \mid \bar{b}=0] - \Pr[\bar{b}'=0 \mid \bar{b}=1]\right| \\ &= \left|\Pr[(b\oplus b')=0 \mid \bar{b}=0] - \Pr[(b\oplus b')=0 \mid \bar{b}=1]\right| \\ &= \left|\Pr[b=b' \mid \bar{b}=0] - \Pr[b=b' \mid \bar{b}=1]\right| \\ &= |\Pr[b=b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_0] - \Pr[b=b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_1]| \ .\end{aligned}$$

Game $\mathsf{G}_1 \to$ Game G_2.

We now describe in Fig. 10 an adversary $\mathcal{B}_2$ against MLWE that runs $\mathcal{A}$ as a subroutine and whose advantage is

$$\mathrm{Adv}^{\mathrm{MLWE}}_{L,k_{\mathrm{LWE}},\eta}(\mathcal{B}_2) = |\Pr[b=b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_1] - \Pr[b=b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_2]| \ .$$

Adversary $\mathcal{B}_2(\mathbf{A},\mathbf{t})$

Instantiate OTSE with $\mathbf{H}' \leftarrow (\mathbf{A} \mid \mathrm{I}_L)$.

01 $b \xleftarrow{\$} \{0,1\}$
02 $(m_0,m_1) \xleftarrow{\$} \mathcal{A}() \ /\!/ \ m_0,m_1 \in \mathcal{R}_q^L$
03 $\mathbf{d} \xleftarrow{\$} \mathcal{R}_{2^{2\eta}}^{k_{\mathrm{LWE}}+L}$
04 $(\mathbf{d}^{(0)},\ldots,\mathbf{d}^{(2\eta-1)}) \leftarrow \mathtt{ring\_to\_bin}(\mathbf{d},2\eta)$
05 $\mathbf{d}^\dagger \leftarrow \mathtt{Stack}(\mathbf{d}^{(0)},\ldots,\mathbf{d}^{(2\eta-1)})$
06 $\mathbf{a} \leftarrow \mathbf{t}$
07 $\mathbf{c}^\star \leftarrow m_b + \mathbf{a}$
08 $b' \xleftarrow{\$} \mathcal{A}(\mathbf{c}^\star)$
09 Return $(b \oplus b')$

Fig. 10. Adversary $\mathcal{B}_2$ against MLWE. The input $(\mathbf{A},\mathbf{t})$ is the challenge received from the MLWE challenger.

The adversary $\mathcal{B}_2$ receives as input $(\mathbf{A},\mathbf{t}) \xleftarrow{\$} \mathcal{D}_{\bar{b}}$ for some bit $\bar{b}$, where

- $\mathcal{D}_0 = \{(\mathbf{A},\mathbf{A}\mathbf{s}+\mathbf{e}) \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{L\times k_{\mathrm{LWE}}};\ \mathbf{s} \xleftarrow{\$} \mathsf{B}_\eta^{k_{\mathrm{LWE}}};\ \mathbf{e} \xleftarrow{\$} \mathsf{B}_\eta^L\}$.
- $\mathcal{D}_1 = \{(\mathbf{A},\mathbf{u}) \ : \ \mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{L\times k_{\mathrm{LWE}}};\ \mathbf{u} \xleftarrow{\$} \mathcal{R}_q^L\}$.

and needs to guess $\bar{b}$.

Recall that in OTSE, the matrix $\mathbf{H}'$ is of the form $\mathbf{H}' = (\mathbf{H}'' \mid \mathrm{I}_L)$, where $\mathbf{H}'' \xleftarrow{\$} \mathcal{R}_q^{L\times k_{\mathrm{LWE}}}$. Hence the distribution of $\mathbf{H}''$ is exactly that of $\mathbf{A}$ in both $\mathcal{D}_0$ and $\mathcal{D}_1$. Accordingly, the adversary $\mathcal{B}_2$ sets $\mathbf{H}' \leftarrow (\mathbf{A} \mid \mathrm{I}_L)$.

Let's now look at the distribution of the pad $\mathbf{a}$. In G_1, it is computed as $\mathbf{a} \leftarrow \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^\dagger$. The vector $\mathbf{d}^\dagger$ is the result of stacking the bit decomposition of a uniformly random vector $\mathbf{d}$ over $\mathcal{R}_{2\eta}$. Since the modulus is a power of 2, the binary coefficients in the elements of $\mathbf{d}^\dagger$ are uniformly random. Hence, when multiplying by $\mathbf{B}^{(\eta)}$, the result is distributed exactly according to $\mathsf{B}_\eta^{k_{\mathrm{LWE}}+L}$. This

is then multiplied by $\mathbf{H}'$, so the distribution of $(\mathbf{H}', \mathbf{a})$ in G_1 is exactly that of $(\mathbf{A}, \mathbf{t})$ in $\mathcal{D}_0$. It is also the case that the distribution of $(\mathbf{H}', \mathbf{a})$ in G_2 is exactly that of $(\mathbf{A}, \mathbf{t})$ in $\mathcal{D}_1$, as they are both uniformly random.

Therefore, the adversary $\mathcal{B}$ perfectly simulates G_1 when $\bar{b} = 0$ and perfectly simulates G_2 when $\bar{b} = 1$. Consequently,

$$\begin{aligned}\mathrm{Adv}^{\mathrm{MLWE}}_{L,k_{\mathrm{LWE}},\eta}(\mathcal{B}_2) &= \left|\Pr[\bar{b}' = 0 \mid \bar{b} = 0] - \Pr[\bar{b}' = 0 \mid \bar{b} = 1]\right| \\ &= \left|\Pr[(b \oplus b') = 0 \mid \bar{b} = 0] - \Pr[(b \oplus b') = 0 \mid \bar{b} = 1]\right| \\ &= \left|\Pr[b = b' \mid \bar{b} = 0] - \Pr[b = b' \mid \bar{b} = 1]\right| \\ &= \left|\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_1] - \Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_2]\right| \ .\end{aligned}$$

Game G_2.

In G_2, the pad $\mathbf{a}$ that masks the message $\mathbf{m}_b$ is sampled uniformly at random, hence also the ciphertext $\mathbf{c}^\star \leftarrow \mathbf{m}_b + \mathbf{a}$ is uniformly random. In particular, the distribution of $\mathbf{c}^\star$ is independent of $\mathbf{m}_b$ and, hence, is independent of b. Therefore, the output b' of $\mathcal{A}$ is independent of b, so

$$\Pr[b = b' \mid \mathcal{A} \text{ plays in } \mathsf{G}_2] = \frac{1}{2} \ .$$

Substituting all the terms in Equation (7), we obtain that

$$\frac{1}{2}\mathrm{Adv}^{\text{IND-OT}}_{\mathsf{OTSE}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{MLWR}}_{k_{\mathrm{LWE}}+L,k_{\mathrm{LWR}},2^\zeta,2^{2\eta}}(\mathcal{B}_1) + \mathrm{Adv}^{\mathrm{MLWE}}_{L,k_{\mathrm{LWE}},\eta}(\mathcal{B}_2) \ . \quad \square$$

D Shuffle of small messages

Lemma 7 is an adaptation of Lemma 4 to the lattice setting, valid whenever the messages are small. Recall that by Lemma 1, the differences of such messages are invertible.

Lemma 7. *Let $\mathcal{R} := \mathbb{F}_1 \times \cdots \times \mathbb{F}_t$ be a product of fields $\mathbb{F}_i$. Let $N \in \mathbb{N}$ and let $\mathcal{D} \subseteq \mathcal{R}$ be a set such that $u - v$ is invertible for all distinct $u, v \in \mathcal{D}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \mathcal{D}^N$. If*

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X)$$

over $\mathcal{R}[X]$, then

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N) \ .$$

Proof. We can apply Lemma 4 in the field $\mathbb{F}_1$ to obtain a permutation π such that the equality $a_i = b_{\pi(i)}$ holds over $\mathbb{F}_1$, i.e., looking at the first component of the elements, for all $i \in [N]$.

We argue that the permutation is actually valid over the full ring $\mathcal{R}$, i.e., that $a_i = b_{\pi(i)}$ holds over $\mathcal{R}$ for all $i \in [N]$. Otherwise, there would exist $j \in [N]$ such that $a_j \neq b_{\pi(j)}$, so $a_j - b_{\pi(j)}$ would be a non-zero divisor of 0, as the difference is 0 over $\mathbb{F}_1$. However, since $a_j, b_{\pi(j)} \in \mathcal{D}$, their difference would also be invertible, leading to a contradiction. $\square$

One might think that Lemma 7 allows for another alternative in the general case: decompose arbitrary ring elements into smaller parts and prove the permutation of each of the parts separately using Lemma 7. The issue is, however, that we would need to ensure that all the permutations are consistent across the parts, in addition to proving the correctness of the decomposition. Given that overhead, it is generally better to use Lemma 5 directly.

E Security of the Proof of Shuffle

Theorem 3. *If* $\mathrm{ZK}^\star$ *has statistical completeness with completeness error* ϵ*, then Protocol 1 has statistical completeness with completeness error* $\epsilon + 2Nq^{-n/2}$*.*

Proof. The ϵ term in the completeness error comes directly from the completeness error of $\mathrm{ZK}^\star$. The $2Nq^{-n/2}$ term comes from the possibility that the u_i are not well defined, i.e., that h_i is non-invertible for some $i \in [N]$.

We can show that

$$\Pr\left[\bigcup_{i=1}^{N} h_i \text{ is non-invertible}\right] \leq 2Nq^{-n/2} \ .$$

We note that the h_i can be expressed as $h_i = y_i - \alpha$ for some $y_i \in \mathcal{R}_q$, for all $i \in [N]$, where $\alpha \xleftarrow{\$} \mathcal{C}$.

We recall that an element in the ring $\mathcal{R}_q \cong \mathbb{Z}_q[x]/(p_1) \times \mathbb{Z}_q[x]/(p_2)$ is invertible if, and only if, it is non-zero in each of the fields $\mathbb{Z}_q[x]/p_j$. With this in mind, by union bound we have that

$$\begin{aligned}
\Pr\left[\bigcup_{i=1}^{N} h_i \text{ is non-invertible}\right] &\leq \sum_{i=1}^{N} \Pr\left[h_i \text{ is non-invertible}\right] \\
&= \sum_{i=1}^{N} \Pr\left[\bigcup_{j=1}^{2} h_i \equiv 0 \mod p_j\right] \\
&\leq \sum_{i=1}^{N} \sum_{j=1}^{2} \Pr\left[h_i \equiv 0 \mod p_j\right] \ ,
\end{aligned}$$

where the probability is over the random sampling of

$$\alpha \xleftarrow{\$} \mathcal{C} := \{p \in \mathcal{R}_q \mid \deg(p) < n/2\} \ .$$

We fix $i \in [N], j \in [2]$ and analyze the probability $\Pr\left[h_i \equiv 0 \mod p_j\right]$. We have that

$$\Pr\left[h_i \equiv 0 \mod p_j\right] = \Pr\left[y_i - \alpha \equiv 0 \mod p_j\right] = \Pr\left[y_i \equiv \alpha \mod p_j\right] \ .$$

Since $\deg(p_j) < n/2$, there is exactly one possible value of $\alpha \in \mathcal{C}$ such that $y_i \equiv \alpha \pmod{p_j}$, where $|\mathcal{C}| = q^{n/2}$. Therefore,

$$\Pr\left[h_i \equiv 0 \mod p_j\right] = q^{-n/2} \ ,$$

hence,

$$\Pr\left[\bigcup_{i=1}^{N} h_i \text{ is non-invertible}\right] \leq 2Nq^{-n/2} \ .$$

Additionally, we can show that the assignment

$$u_i \leftarrow (-1)^i \gamma \prod_{j=1}^{i} \frac{\hat{h}_j}{h_j} + \theta_i$$

is such that

$$\gamma\hat{h}_1 + u_1 h_1 = \theta_1 h_1, \tag{8}$$
$$u_{i-1}\hat{h}_i + u_i h_i = \theta_{i-1}\hat{h}_i + \theta_i h_i \quad \forall i \in \{2, \ldots, N-1\} \quad \text{and} \tag{9}$$
$$u_{N-1}\hat{h}_N + (-1)^N \gamma h_N = \theta_{N-1}\hat{h}_N \ . \tag{10}$$

For Equation (8), we have that

$$\gamma\hat{h}_1 + u_1 h_1 = \gamma\hat{h}_1 + (-\gamma\hat{h}_1/h_1 + \theta_1)h_1 = \theta_1 h_1 \ .$$

Regarding Equation (9), for $i \in \{2, \ldots, N-1\}$ it holds that

$$\begin{aligned} u_{i-1}\hat{h}_i + u_i h_i &= \left((-1)^{i-1} \prod_{j=0}^{i-1} \frac{\hat{h}_j}{h_j} + \theta_{i-1}\right)\hat{h}_i + \left((-1)^i \prod_{j=0}^{i} \frac{\hat{h}_j}{h_j} + \theta_i\right) h_i \\ &= \theta_{i-1}\hat{h}_i + \theta_i h_i \ . \end{aligned}$$

Lastly, for Equation (10) we observe that $(\hat{h}_1 \cdots \hat{h}_N)/(h_1 \cdots h_{N-1}) = h_N$. This follows from the fact that $\prod_{i=1}^{N} h_i = \prod_{i=1}^{N} \hat{h}_i$, since

$$((z_1, \texttt{int\_to\_bin}(1)), \ldots, (z_N, \texttt{int\_to\_bin}(N))) \sim_P ((\hat{z}_1, \sigma_1), \ldots, (\hat{z}_N, \sigma_N)) \ .$$

Then,

$$\begin{aligned} u_{N-1}\hat{h}_N + (-1)^N \gamma h_N &= \left((-1)^{N-1}\gamma \prod_{j=1}^{N} \frac{\hat{h}_j}{h_j} + \theta_{N-1}\right)\hat{h}_N + (-1)^N \gamma h_N \\ &= (-1)^{N-1}\gamma h_N + \theta_{N-1}\hat{h}_N + (-1)^N \gamma h_N \\ &= \theta_{N-1}\hat{h}_N \ . \end{aligned}$$

□

Theorem 4. *If* $\mathrm{ZK}^\star$ *is Special Honest Verifier Zero-Knowledge (SHVZK), then Protocol 1 is also SHVZK.*

Proof. The simulator $\mathcal{S}$ receives as input an instance

$$\mathbf{x} = ((\mathbf{c}_1^\star, \ldots, \mathbf{c}_N^\star), (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), \mathbf{D})$$

for which there exists a valid witness $\mathbf{w}$ (that $\mathcal{S}$ ignores) such that $(\mathbf{x}, \mathbf{w}) \in \mathsf{R}_{\text{shuffle}}$. Additionally, $\mathcal{S}$ receives challenges $\alpha, \beta, \lambda, \gamma \in \mathcal{C}$ and the randomness ξ that the verifier uses in ZK*. The simulator needs to output a transcript that is indistinguishable to that of an honest execution of the protocol with those challenges, instance and randomness.

The simulator $\mathcal{S}$ sets its output as follows:

- $\mathbf{P} \xleftarrow{\$} \mathsf{Commit}^\star(0, \ldots, 0; \mathbf{r_P})$.
- $\mathbf{W} \xleftarrow{\$} \mathsf{Commit}^\star(\mathbf{0}, \ldots, \mathbf{0}; \mathbf{r_W})$.
- $u_i \xleftarrow{\$} \mathcal{R}_q$ for all $i \in [N-1]$.
- $\Sigma^\star$ obtained by running the simulator $\mathcal{S}^\star$ of ZK* on input the instance

$$\mathbf{x}' = ((\mathbf{c}_1^\star, \ldots, \mathbf{c}_N^\star), (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), (u_1, \ldots, u_{N-1}), \mathbf{D}, \mathbf{P}, \mathbf{W}, \alpha, \beta, \lambda, \gamma),$$

  and the verifier's randomness ξ .

We need to argue that these values are indeed indistinguishable from those generated in an honest execution of the protocol.

The commitments $\mathbf{P}$ and $\mathbf{W}$ are indistinguishable from the honestly computed ones because of the hiding property of $\mathsf{Com}^\star$.

The values u_i are computed in the protocol as

$$u_i \leftarrow (-1)^i \gamma \prod_{j=1}^{i} \frac{\hat{h}_j}{h_j} + \theta_i$$

where $\theta_i \xleftarrow{\$} \mathcal{R}_q$ for all $i \in [N-1]$. Hence, all the u_i look uniformly random, and so that is how $\mathcal{S}$ samples them.

Finally, the transcript $\Sigma^\star$ is indistinguishable from an honestly computed one because of the SHVZK property of ZK*. There is one technicality, though, since $\mathcal{S}^\star$ is only guaranteed to output indistinguishable transcripts for valid instances. However, the instance $\mathbf{x}'$ provided by $\mathcal{S}$ is generally not valid, namely because the linear relations from ZK* are likely not satisfied by the 0 polynomials that are inside the commitments. In any case, there exist elements $(\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger)$ such that the linear equations are satisfied, namely the binary decomposition of the values on the left hand side of the linear equations. Even though $\mathcal{S}$ cannot compute these values without knowing the witness, suppose that the commitment $\mathbf{W}$ was replaced by a commitment $\mathbf{W}'$ to $(\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger)$. In this case, $\mathcal{S}^\star$ would indeed output a valid transcript. Now if $\mathcal{S}^\star$ did not output a valid transcript when receiving $\mathbf{W}$, then the hiding property of $\mathsf{Com}^\star$ would be broken, as $\mathcal{S}^\star$ would be able to distinguish the commitments $\mathbf{W}$ and $\mathbf{W}^\star$. Therefore, the transcript $\Sigma^\star$ is indistinguishable from an honestly computed one based on the SHVZK of ZK* and the hiding property of $\mathsf{Com}^\star$. □

Theorem 5. *Suppose that* $\mathrm{ZK}^\star$ *has witness extended emulation with knowledge error* κ*. Then Protocol 1 has witness extended emulation with knowledge error at most* $2(N+1)^2((L-1)N+1)\kappa + ((L+1)N+1)q^{-n/2}$.

Proof. We design a knowledge extractor $\mathcal{E}$ for Protocol 1. Suppose that $\mathrm{ZK}^\star$ is a k-round public-coin interactive argument of knowledge. By Lemma 6, we know that there exists a tree-finding algorithm running in expected time at most $2(N+1)^2((L-1)N+1)\kappa$ executions of $\mathcal{A}$ which produces a $((L-1)N+1)$-$(N+1)$-$(N+1)$-2-1^k-tree of accepting transcripts with probability at least $\epsilon - ((L+1)N+1)q^{-n/2}$, where ϵ is the success probability of the adversary $\mathcal{A}$.

Remark 1. Strictly speaking, since we do not know the size of the sets from which the verifier samples their challenges in $\mathrm{ZK}^\star$, we rely on a generalization of Lemma 6 explained in [6], which addresses differently sized challenge sets. However, since we only sample a tree of transcripts with arity 1 for the challenges in $\mathrm{ZK}^\star$, there are no terms in the knowledge error depending on the challange spaces from $\mathrm{ZK}^\star$.

The $\mathcal{E}$ for Protocol 1 runs this tree-finding algorithm, but replaces the parts corresponding to $\mathrm{ZK}^\star$ with the knowledge extractor for $\mathrm{ZK}^\star$. Since the tree has $2(N+1)^2((L-1)N+1)$ leaves, this changes the success probability by at most $2(N+1)^2((L-1)N+1)\kappa$.

Removing the parts of the tree corresponding to $\mathrm{ZK}^\star$ but leaving the witness produced by its knowledge extractor gives a $((L-1)N+1)$-$(N+1)$-$(N+1)$-2 tree of transcripts for an instance $\mathbf{x} = ((\mathbf{c}_1^\star, \ldots, \mathbf{c}_N^\star), (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), \mathbf{D})$ with a witness for the relation proved by ZK^* at each leaf. Namely, at each leaf, there is an instance

$$\mathbf{x}' = ((\mathbf{c}_1^\star, \ldots, \mathbf{c}_N^\star), (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), (u_1, \ldots, u_{N-1}), \mathbf{D}, \mathbf{P}, \mathbf{W}, \alpha, \beta, \lambda, \gamma) \ ,$$

for ZK^*, and a witness

$$\mathbf{w}' = ((\mathbf{d}_1^\dagger, \cdots, \mathbf{d}_N^\dagger), (\sigma_1, \ldots, \sigma_N), (\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger), \mathbf{r}_\mathbf{D}, \mathbf{r}_\mathbf{P}, \mathbf{r}_\mathbf{W})$$

satisfying the relations from $\mathrm{ZK}^\star$. In particular, we have

$$\mathtt{is\_bin}(\mathbf{d}_i^\dagger) \wedge \mathtt{is\_bin}(\sigma_i) \wedge \mathtt{is\_bin}(\mathbf{w}_i^\dagger) \ \forall i \in [N] \ ,$$

$$\begin{aligned} \mathsf{Verify}^\star(\mathbf{D}, (\mathbf{d}_1^\dagger, \ldots, \mathbf{d}_N^\dagger), \mathbf{r}_\mathbf{D}) &= \mathsf{Verify}^\star(\mathbf{P}, (\sigma_1, \ldots, \sigma_N), \mathbf{r}_\mathbf{P}) \\ &= \mathsf{Verify}^\star(\mathbf{W}, (\mathbf{w}_1^\dagger, \ldots, \mathbf{w}_N^\dagger), \mathbf{r}_\mathbf{W}) = 1 \ , \end{aligned}$$

and denoting

$$z_i := \sum_{j=1}^{L} (\mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger)[j] \cdot \lambda^{j-1}, \ \ \hat{z}_i := \sum_{j=1}^{L} \hat{\mathbf{c}}_i[j] \cdot \lambda^{j-1} \ ,$$

$$h_i := z_i + \mathtt{int\_to\_bin}(i) \cdot \beta - \alpha, \ \ \hat{h}_i := \hat{z}_i + \sigma_i \cdot \beta - \alpha \ , \ \ w_i := \mathbf{G}^{(\lceil \log q \rceil)}\mathbf{w}_i^\dagger \ ,$$

then

$$\begin{aligned} &\gamma\hat{h}_1 + u_1 h_1 = w_1 \ , \\ &u_{i-1}\hat{h}_i + u_i h_i = w_i \quad \forall i \in \{2, \ldots, N-1\} \ , \quad \text{and} \\ &u_{N-1}\hat{h}_N + (-1)^N \gamma h_N = w_N \ . \end{aligned} \tag{11}$$

Fix any leaf of this tree. The leaf has a sibling leaf which corresponds to a challenge $\gamma' \neq \gamma$ and an instance

$$\mathbf{x}'' = ((\mathbf{c}_1^\star, \ldots, \mathbf{c}_N^\star), (\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_N), (u'_1, \ldots, u'_{N-1}), \mathbf{D}, \mathbf{P}, \mathbf{W}, \alpha, \beta, \lambda, \gamma') \ ,$$

with witness

$$\mathbf{w}'' = ((\widetilde{\mathbf{d}_1^\dagger}, \cdots, \widetilde{\mathbf{d}_N^\dagger}), (\widetilde{\sigma_1}, \ldots, \widetilde{\sigma_N}), (\widetilde{\mathbf{w}_1^\dagger}, \ldots, \widetilde{\mathbf{w}_N^\dagger}), \widetilde{\mathbf{r}_\mathbf{D}}, \widetilde{\mathbf{r}_\mathbf{P}}, \widetilde{\mathbf{r}_\mathbf{W}})$$

satisfying the relations from ZK*. In particular, the witness satisfies

$$\begin{aligned} \mathsf{Verify}^\star(\mathbf{D}, (\widetilde{\mathbf{d}_1^\dagger}, \ldots, \widetilde{\mathbf{d}_N^\dagger}), \widetilde{\mathbf{r}_\mathbf{D}}) &= \mathsf{Verify}^\star(\mathbf{P}, (\widetilde{\sigma_1}, \ldots, \widetilde{\sigma_N}), \widetilde{\mathbf{r}_\mathbf{P}}) \\ &= \mathsf{Verify}^\star(\mathbf{W}, (\widetilde{\mathbf{w}_1^\dagger}, \ldots, \widetilde{\mathbf{w}_N^\dagger}), \widetilde{\mathbf{r}_\mathbf{W}}) = 1 \end{aligned}$$

for the same commitments $\mathbf{D}, \mathbf{P}, \mathbf{W}$. If $\mathbf{w}' \neq \mathbf{w}''$, then the extractor can break the strong binding property of the commitment scheme. Hence, we can assume that $\mathbf{w}' = \mathbf{w}''$. In particular, it must also hold that

$$\begin{aligned} &\gamma\hat{h}_1 + u'_1 h_1 = w_1, \\ &u'_{i-1}\hat{h}_i + u'_i h_i = w_i \quad \forall i \in \{2, \ldots, N-1\} \quad \text{and} \\ &u'_{N-1}\hat{h}_N + (-1)^N \gamma h_N = w_N. \end{aligned} \tag{12}$$

Subtracting both sets of equations (11) and (12), we get that

$$\begin{aligned} &(\gamma - \gamma')\hat{h}_1 + (u_1 - u'_1)h_1 = 0, \\ &(u_{i-1} - u'_{i-1})\hat{h}_i + (u_i - u'_i)h_i = 0 \quad \forall i \in \{2, \ldots, N-1\} \quad \text{and} \\ &(u_{N-1} - u'_{N-1})\hat{h}_N + (-1)^N(\gamma - \gamma')h_N = 0 \ . \end{aligned}$$

Equivalently,

$$\underbrace{\begin{pmatrix} \hat{h}_1 & h_1 & 0 & \cdots & 0 \\ 0 & \hat{h}_2 & h_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \hat{h}_{N-1} & h_{N-1} \\ (-1)^N h_N & 0 & \cdots & 0 & \hat{h}_N \end{pmatrix}}_{\mathbf{H}} \underbrace{\begin{pmatrix} \gamma - \gamma' \\ u_1 - u'_1 \\ \vdots \\ u_{N-2} - u'_{N-2} \\ u_{N-1} - u'_{N-1} \end{pmatrix}}_{\mathbf{c}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \ .$$

We can show that $\det(\mathbf{H}) = 0$. Recall that we denote the decomposition of the ring into fields as $\mathcal{R}_q \cong \mathbb{Z}_q[x]/p_1 \times \mathbb{Z}_q[x]/p_2$. Let $j \in \{1, 2\}$. Since $\gamma \neq \gamma'$ and

$\deg(\gamma - \gamma') < n/2$, it follows that $\mathbf{c} \not\equiv \mathbf{0} \mod p_j$. Hence, $\mathbf{c}$ is a non-zero vector such that $\mathbf{Hc} \equiv 0$ over the field $\mathbb{Z}_q[x]/p_j$, so it must be that $\det(\mathbf{H}) \equiv 0 \mod p_j$. That holds for both $j \in \{1, 2\}$, so we obtain that $\det(\mathbf{F}) = 0$ in $\mathcal{R}_q$.

Let's now express the determinant of $\mathbf{H}$ in terms of the $\hat{h}_i$'s and h_i's. We can consider the submatrices

$$\mathbf{H}_1 = \begin{pmatrix} \hat{h}_2 & h_2 & \cdots & 0 & 0 \\ 0 & \hat{h}_3 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \hat{h}_{N-1} & h_{N-1} \\ 0 & 0 & \cdots & 0 & \hat{h}_N \end{pmatrix} \quad , \quad \mathbf{H}_2 = \begin{pmatrix} h_1 & 0 & \cdots & 0 & 0 \\ \hat{h}_2 & h_2 & \cdots & 0 & 0 \\ 0 & \hat{h}_3 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \hat{h}_{N-1} & h_{N-1} \end{pmatrix} .$$

Applying the Laplace expansion of the determinant to the first column of $\mathbf{H}$, we get that

$$0 = \det(\mathbf{H}) = \hat{h}_1 \det(\mathbf{H}_1) + (-1)^{2N+1} h_N \det(\mathbf{H}_2) = \prod_{i=1}^{N} \hat{h}_i - \prod_{i=1}^{N} h_i \ .$$

Consequently, $\prod_{i=1}^{N} \hat{h}_i = \prod_{i=1}^{N} h_i$, i.e.,

$$\prod_{i=1}^{N} (z_i + \texttt{int\_to\_bin}(i) \cdot \beta - \alpha) = \prod_{i=1}^{N} (\hat{z}_i + \sigma_i \cdot \beta - \alpha) \ . \tag{13}$$

Now let $\beta_1 := \beta$. The parent node of the previous two nodes, corresponding to the prover's message $\mathbf{W}$, has N siblings, with edges corresponding to challenges β_ℓ for $\ell \in \{2, \ldots, N+1\}$. The nature of the tree implies that $\beta_{\ell_1} \neq \beta_{\ell_2}$ for all $\ell_1 \neq \ell_2$. For those N siblings we can apply the same proof strategy to obtain witnesses satisfying the same product equality as in Equation (6). Since all these nodes share the same commitments $\mathbf{D}$ and $\mathbf{P}$, those witnesses must contain the same elements $(\mathbf{d}_1^\dagger, \cdots, \mathbf{d}_N^\dagger), (\sigma_1, \ldots, \sigma_N)$, or else the extractor can break the binding property of the commitment scheme.

Overall, we have that

$$\prod_{i=1}^{N} (z_i + \texttt{int\_to\_bin}(i) \cdot \beta_\ell - \alpha) = \prod_{i=1}^{N} (\hat{z}_i + \sigma_i \cdot \beta_\ell - \alpha)$$

for all $\ell \in [N+1]$. Lemma 3 implies that

$$\prod_{i=1}^{N} (z_i + \texttt{int\_to\_bin}(i) \cdot X_1 - \alpha) = \prod_{i=1}^{N} (\hat{z}_i + \sigma_i \cdot X_1 - \alpha)$$

since the difference of those polynomials is a polynomial of degree N in $\mathcal{R}_q[X_1]$ for which we have found $N+1$ roots.

We now let $\alpha_1 := \alpha$ and focus on the parent of those $N+1$ nodes. This node also has $N+1$ siblings, with edges corresponding to challenges α_ℓ for

$\ell \in \{2, \ldots, N+1\}$, such that $\alpha_{\ell_1} \neq \alpha_{\ell_2}$ for all $\ell_1 \neq \ell_2$. We can apply the same argument as before to obtain that

$$\prod_{i=1}^{N}(z_i + \texttt{int\_to\_bin}(i) \cdot X_1 - \alpha_\ell) = \prod_{i=1}^{N}(\hat{z}_i + \sigma_i \cdot X_1 - \alpha_\ell)$$

for all $\ell \in [N+1]$, and Lemma 3 again implies that

$$\prod_{i=1}^{N}(z_i + \texttt{int\_to\_bin}(i) \cdot X_1 - X_2) = \prod_{i=1}^{N}(\hat{z}_i + \sigma_i \cdot X_1 - X_2) \ .$$

Substituting the values of z_i and $\hat{z}_i$, we have obtained that

$$\begin{aligned}&\prod_{i=1}^{N}\left(\sum_{j=1}^{L}(\mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger)[j] \cdot \lambda^{j-1} + \texttt{int\_to\_bin}(i) \cdot X_1 - X_2\right)\\ =&\prod_{i=1}^{N}\left(\sum_{j=1}^{L}\hat{\mathbf{c}}_i[j] \cdot \lambda^{j-1} + \sigma_i \cdot X_1 - X_2\right) \ . \end{aligned} \tag{14}$$

We finally let $\lambda_1 := \lambda$ and focus on the parent of those last $N+1$ nodes. This node has $(L-1)N$ siblings, with edges corresponding to challenges λ_ℓ for $\ell \in \{2, \ldots, (L-1)N+1\}$, such that $\lambda_{\ell_1} \neq \lambda_{\ell_2}$ for all $\ell_1 \neq \ell_2$. Again for those $(L-1)N$ nodes, and the respective subtrees that hang from then, we can apply the same argument to show that Eq. (14) also has to hold for the challenges λ_ℓ, with the same witness values $\mathbf{d}_i^\dagger, \sigma_i$ beacuse of binding. Hence, we have that

$$\begin{aligned}&\prod_{i=1}^{N}\left(\sum_{j=1}^{L}(\mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger)[j] \cdot \lambda_\ell^{j-1} + \texttt{int\_to\_bin}(i) \cdot X_1 - X_2\right)\\ =&\prod_{i=1}^{N}\left(\sum_{j=1}^{L}\hat{\mathbf{c}}_i[j] \cdot \lambda_\ell^{j-1} + \sigma_i \cdot X_1 - X_2\right) \ . \end{aligned}$$

for all $\ell \in [(L-1)N+1]$. Applying Lemma 3 once more, we obtain that

$$\begin{aligned}&\prod_{i=1}^{N}\left(\sum_{j=1}^{L}(\mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger)[j] \cdot X_3^{j-1} + \texttt{int\_to\_bin}(i) \cdot X_1 - X_2\right)\\ =&\prod_{i=1}^{N}\left(\sum_{j=1}^{L}\hat{\mathbf{c}}_i[j] \cdot X_3^{j-1} + \sigma_i \cdot X_1 - X_2\right) \ . \end{aligned}$$

Denoting

$$f_i(X_3) := \sum_{j=1}^{L}(\mathbf{c}_i^\star - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}_i^\dagger)[j] \cdot X_3^{j-1} \quad \text{and} \quad \hat{f}_i(X_3) := \sum_{j=1}^{L}\hat{\mathbf{c}}_i[j] \cdot X_3^{j-1}$$

for all $i \in [N]$, we have that

$$\prod_{i=1}^{N} (f_i(X_3) + \texttt{int\_to\_bin}(i) \cdot X_1 - X_2) = \prod_{i=1}^{N} \left(\hat{f}_i(X_3) + \sigma_i \cdot X_1 - X_2 \right) \ ,$$

where $f_i(X_3), \hat{f}_i(X_3) \in \mathcal{R}_q[X_3]$.

Since $\mathcal{R}_q \cong \mathbb{Z}_q[x]/p_1 \times \mathbb{Z}_q[x]/p_2$, we also have that

$$\mathcal{R}_q[X_3] \cong (\mathbb{Z}_q[x]/p_1)[X_3] \times (\mathbb{Z}_q[x]/p_2)[X_3],$$

where each factor $(\mathbb{Z}_q[x]/p_j)[X_3]$ is an integral domain, as $\mathbb{Z}_q[x]/p_j$ is a field. Additionally, the set of binary ring elements is such that their differences have norm at most 2, so Lemma 1 implies that those differences are invertible. Therefore, we can apply Lemma 5 to obtain that

$$(f_1(X_3), \ldots, f_N(X_3)) \sim_P (\hat{f}_1(X_3), \ldots, \hat{f}_N(X_3)) \ ,$$

i.e., there exists $\pi \in \mathrm{Perm}_N$ such that $\hat{f}_i(X_3) = f_{\pi(i)}(X_3)$ for all $i \in [N]$, or equivalently,

$$\hat{\mathbf{c}}_i = \mathbf{c}^{\star}_{\pi(i)} - \mathbf{H}'\mathbf{B}^{(\eta)}\mathbf{d}^{\dagger}_{\pi(i)}$$

for all $i \in [N]$. The extractor can hence output $\mathbf{w} := (\pi, (\mathbf{d}^{\dagger}_1, \cdots, \mathbf{d}^{\dagger}_N), \mathbf{r}_{\mathbf{D}})$. $\square$