Proof of a shuffle for lattice-based cryptography

Nuria Costa[1], Ramiro Martínez[2], and Paz Morillo[2]

[1] Scytl Secure Electronic Voting
[2] Universitat Politècnica de Catalunya

Abstract. In this paper we present the first proof of a shuffle for lattice-based cryptography which can be used to build a universally verifiable mix-net capable of mixing votes encrypted with a post-quantum algorithm, thus achieving long-term privacy. Universal verifiability is achieved by means of the publication of a non-interactive zero knowledge proof of a shuffle generated by each mix-node which can be verified by any observer. This published data guarantees long-term privacy since its security is based on perfectly hiding commitments and also on the hardness of solving the Ring Learning With Errors (RLWE) problem, that is widely believed to be quantum resistant.

Keywords: mix-nets, evoting, post-quantum cryptographic protocol, RLWE encryption, proof of a shuffle.

1 Introduction

In the last years, several countries have been introducing electronic voting systems to improve their democratic processes: electronic voting systems provide more accurate and fast vote counts, reduce the logistic cost of organizing an election and can offer specific mechanisms for voters with disabilities to be able to cast their votes independently. In particular, internet voting systems provide voters with the chance to cast their votes from anywhere: their homes, hospitals, or even from foreign countries in case they are abroad at the time of the election.

Anonymity and verifiability are two fundamental requirements for internet voting systems that seem to be contradictory. Anonymity requires that the link between the vote and the voter who has cast it must remain secret during the whole process, while verifiability requires that all the steps of the electoral process - vote casting, vote storage and vote counting - can be checked by the voters, the auditors or external observers.

The different techniques used by the actual internet voting systems to achieve anonymity can be classified in three categories:

Blind signature: this method allows the voter to obtain a message signed by an authorized entity in such a way that this entity gets no information at all about the message. Consequently, votes are anonymized before being sent.

Homomorphic tallying: the votes are encrypted using a homomorphic cryptosystem and during the tallying phase they are aggregated. The resulting ciphertext is decrypted and the ballot count results are obtained. This anonymizes the votes at the end of the election since no vote is individually decrypted.

Mixing: the ciphertexts are permuted and re-encrypted in such a way that the correlation between the input and output of the mixing process is hidden and it is not possible to trace it back. This operation is called a *shuffle* and it is executed in a mixing network (*mix-net*) composed of mixing nodes each one performing in turns the same operation. This is done in order to be able to preserve the privacy of the process even if some nodes are dishonest: as long as one of the mix nodes remains faithful and does not reveal the secret permutation or re-encryption values, unlinkability is preserved. Notice that this method requires to provide a *proof of a shuffle* so that it can be checked that the contents of the output are the same as the contents of the input.

On the other hand, in order to build verifiable systems one key instrument is the Bulletin Board: a public place where all the audit information of the election (encrypted votes, election configuration, . . .) is published by authorized parties and can be verified by anyone: voters, auditors or third parties. However, once published in the Bulletin Board, it is not possible to ensure that all the copies are deleted after the election and the audit period ends, and long-term privacy may not be ensured by encryption algorithms used nowadays, for example due to efficient quantum computers. Learning how a person voted some years ago may have political, as well as personal implications (e.g. in case of family coercion).

Everlasting privacy is a recent research topic meaning that even if a computationally unbounded adversary exists, the voter's privacy is preserved. Several solutions have been proposed in order to address the problem exposed above and the majority of them use Pedersen commitments [31] to protect the information that is going to be published in the Bulletin Board. These commitments perfectly hide the committed message since its privacy does not depend on any computational assumption whose strength may be eroded in the future. Nevertheless most of these proposals require an anonymous channel to send additional information (for instance, the encrypted openings of the commitments) to the server either during the voting phase [28, 13] or during the authentication phase [23, 24]. Since these anonymous channels are difficult to implement, mix-nets are frequently used as an alternative. There are some proposals to construct universally verifiable mix-nets with everlasting privacy [6], where a mixing of commitments instead of ciphertexts is performed.

Nevertheless, our goal is to achieve long-term privacy, in which the voter's privacy is preserved against a polynomially bounded quantum capable adversary. Lattice-based cryptography [27] is maybe the most promising approach to get cryptosystems that will remain secure in the post-quantum era, and so it has become a very active area of research in the last years. The security of lattice-based cryptography is based on the worst-case problem meaning that breaking a lattice-based cryptosystem implies finding an efficient algorithm for solving any instance of the underlying lattice problem, for instance, the Shortest Vector Problem (SVP), the Closest Vector Problem (CVP) or the Shortest Independent Vector Problem (SIVP). There are several proposals to build lattice-based cryptosystems such as public key encryption schemes, digital signatures schemes, hash functions, Identity Based Encryption schemes or ZK proofs.

Since mix-nets are of paramount importance in an online voting scenario and lattice-based cryptography seems to be one of the main alternatives to achieve post-quantum security, we consider it necessary to have a system with a mix-net capable of shuffling lattice-based encryptions. As far as we know there is only one proposal of an e-voting protocol that uses lattice-based cryptography [9]. In the cited paper, the authors present an e-voting scheme that uses LWE-based fully homomorphic encryption in order to provide a homomorphic tally system. Nevertheless, to the best of our knowledge, there is no proposal for a lattice based e-voting scheme using mix-nets in the literature.

1.1 Related work

The first mix-net was introduced by Chaum [8] in 1981 where the plaintext is encrypted as many times as mixing nodes using RSA onions with random padding, and during the mixing process each node decrypts the outer layer and removes the random padding, so the last node obtains the original message. In 1993, Park *et al.* noticed that Chaum's mix-net required a ciphertext size proportional to the number of mixing nodes and proposed a re-encryption mix-net [30] where instead of concatening, they re-randomized the ciphertexts using a homomorphic cryptosystem like ElGamal. In this system decryption occurs after shuffling is finished, however they also proposed a different mix-net in the same paper [30] where each node performs partial decryption besides the shuffling. Two years later, Sako and Kilian [35] defined the property of *universal verifiability* and proposed the first universally verifiable mix-net, that provides a zero-knowledge proof of correct mixing that any observer can verify. Achieving efficient mixing proofs was the challenge of the late 1990s, where two solutions were proposed for an efficient universally verifiable mix-net [26, 1]. In 2001, Furukawa and Sako [16] proposed a proof of correct mixing more efficient than the previous ones, in this scheme each node uses a matrix to do the ciphertexts permutation and proves that this matrix is a permutation matrix. In the same year, Neff [29] introduced the fastest, fully-private, universally verifiable mix-net shuffle proof known so far, optimized and generalized by Groth in [18]. In 2004, Golle *et al.* [17] proposed a mix-net with universal re-encryption, that does not require that each mix node knows the public key of the ciphertexts they are mixing. This can be done with homomorphic cryptosystems like ElGamal. In the same year, Wikström [40] gave the first mix-net definition and implementation in the UC framework [7] as well as a simpler and efficient construction [41].

Adida and Wikström introduced a different mix-net approach [2, 3] motivated by the complexity of using mix-nets in elections. They proposed an offline precomputation technique in order to reduce the online computation complexity. However, the scheme [2] was quite inefficient while the construction in [3] was very efficient but reduced to a relatively small number of senders. In 2010 Terelius and Wikström [38] proposed a provably secure technique to prove the correctness of a cryptographic shuffle using simple shuffle arguments and two years later, Bayer and Groth, proposed an honest verifier zero-knowledge argument for the correctness of a shuffle of homomorphic encryptions that, compared with previous work, matched the lowest computation cost for the verifier. Nevertheless, as these non-interactive proofs are known in the random oracle model, several works have studied how to construct NIZK shuffle arguments in the Common Reference String (CRS) model without using random oracles [19, 22, 14, 15]. However, given that these CRS-based proposals are constructed for bilinear groups, we are going to use the approach presented in [38, 42] to build our proof of a shuffle. In [42] Wikström presented a mix-net based on homomorphic cryptosystems using the idea of permutation matrices. In the proposal, a proof of a shuffle is split in an offline and online phase that reduces significantly the computational complexity in the online part. More precisely, in the offline part the mixing node computes a commitment to the permutation matrix and proves in zero knowledge that it knows an opening for that commitment. In the online part, the node computes a commitment-consistent proof of a shuffle to demonstrate that the committed matrix has been used to shuffle the input.

To the best of our knowledge, the concept of using mix-nets for lattice-based cryptography is very new in the research literature, and as such, there are not many proposed schemes. There have been proposals for a lattice based universal re-encryption for mix-nets [36, 37] but none of them proposes a proof of a shuffle, which is essential for verifiable protocols.

1.2 Our contribution

We propose the first universally verifiable mix-net for a post-quantum cryptosystem. The mix-net receives at its input a set of messages encrypted using a RLWE encryption scheme [25] whose security is based on the hardness of solving the Learning With Errors problem over rings (RLWE problem) [33]. In the proposal, we show how to permute and re-encrypt RLWE encryptions and we also give the first proof of a shuffle that works for a lattice-based cryptosystem. This proof is based on what is proposed in [42] but it is not a direct adaptation of it, since we introduce a new technique to implement the last part of the proof that differs from what is presented in that article.

We split the proof of a shuffle into two protocols following Wikström's technique. In the offline part, the permutation and re-encryption parameters used to shuffle the ciphertexts are committed and it is demonstrated using zero knowledge proofs that these values meet certain properties and that the openings for the commitments are known. The zero-knowledge proofs used in this part satisfy special soundness and special honest verifier zero-knowledge [10]. The first property means that given two accepting conversations with identical first messages but different challenges, it is possible to extract a valid witness. Regarding the second property, it means that for a given challenge the verifier can be simulated.

In the online part, instead of computing a commitment-consistent proof of a shuffle, each mix node should compute a commitment to its output using the commitments calculated in the offline protocol taking advantage of the homomorphic property of both the commitment and encryption schemes. Finally, the node should reveal the opening of the output commitment in order to demonstrate that it has used the committed permutation and re-encryption values to do the shuffle. It is important to notice that we are not opening directly the commitments to the secret permutation neither to the secret re-encryption values but the commitments to a linear combination of them. The openings revealed by each node perfectly hide the secret values and no information is leaked that could compromise the privacy of the process. Commitments used to construct the proof are generalized versions of the Pedersen commitment, which is perfectly hiding and computationally binding under the discrete logarithm assumption and it is widely used to provide everlasting privacy. The reason why we use this commitment is for efficiency and simplicity, nevertheless since our protocol only requires a commitment that allows us to prove linear relations between committed elements, the protocol presented in this paper could be modified in order to use the commitment scheme proposed by Benhamouda *et al.* in [5]. This would allow us to construct a mix-net totally based on post-quantum cryptography. As this is a non-trivial modification we first show how to mix RLWE ciphertexts using Pedersen commitments and how to do it universally verifiable.

The organization of this paper is as follows. In section 2 we define the notation and review the cryptographic background that is necessary to understand the mix-net proposal. In section 3 we give the details about the shuffle of RLWE encryptions, and finally in section 4 we conclude the paper.

2 Preliminaries

In this section we present the notation that we are going to use throughout the paper and we also give some details about the cryptographic background required for the latter sections.

Standard notation regarding vectors and matrices will be used. Vectors will be represented by boldface lowercase roman letters (such as $\mathbf{v}$ or $\mathbf{w}$) and matrices will be represented by boldface uppercase roman letters (such as $\mathbf{M}$ or $\mathbf{A}$). Let $\langle\cdot,\cdot\rangle$ denote the standard inner product in $\mathbb{Z}_q^N$, given two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_q^N$ $\langle\mathbf{v}, \mathbf{w}\rangle$ is defined by means of $\sum_{i=1}^N v_i w_i$. When working with lattices we are going to follow the notation proposed in [25].

2.1 Ideal lattices

A lattice is a set of points in an n-dimensional space with a periodic structure. Given m-linearly independent vectors $\mathbf{a_1}, \dots, \mathbf{a_m} \in \mathbb{R}^n$, the rank m lattice generated by them is the set of vectors:

$$\mathcal{L}(\mathbf{a_1}, \dots, \mathbf{a_m}) = \left\{ \sum_{i=1}^m x_i \mathbf{a}_i : x_i \in \mathbb{Z} \right\}$$

We denote the basis of the lattice as $\mathbf{A} = (\mathbf{a_1}, \dots, \mathbf{a_m})$, i.e., the matrix whose columns are $\mathbf{a_1}, \dots, \mathbf{a_m}$. We are going to work with lattices that are *full-rank* ($n = m$), that is, the number of linearly independent vectors in the basis of the lattice is equal to the number of dimensions in which the lattice is embedded.

Definition 1. *An ideal lattice is a lattice defined by a basis $\boldsymbol{A}$ constructed with a vector $\mathbf{a} \in \mathbb{Z}^n$ iteratively multiplied by a transformation matrix $\mathbf{F} \in \mathbb{Z}^{n \times n}$ defined from a vector $\mathbf{f} \in \mathbb{Z}^n$ as follows.*

$$\mathbf{F} = \left[\begin{array}{ccc|c} 0 & \dots & 0 & -f_0 \\ \ddots & & & -f_1 \\ & I & & \vdots \\ & & \ddots & -f_{n-1} \end{array}\right]$$

The basis is defined as: $\mathbf{A} = \mathbf{F}^* \mathbf{a} = [\mathbf{a}, \mathbf{F}\mathbf{a}, \dots, \mathbf{F}^{n-1}\mathbf{a}]$.

Lattices that follow this particular structure have been named ideal lattices because they can be equivalently characterized as ideals of the ring of modular polynomials $R = \mathbb{Z}[x]/\langle f(x) \rangle$ where $f(x) = x^n + f_{n-1}x^{n-1} + \dots + f_0 \in \mathbb{Z}[x]$. That means that working on the polynomials domain modulo $f(x)$ is equivalent to working on the ideal lattice domain characterized by $\mathbf{F}$. We will use the ring where $f(x) = x^n + 1$, as proposed by [25]. When working in this ring, where we know that $f(x)$ is a cyclotomic polynomial for n a power of 2, one obtains the family of the so called *anti-cyclic integer lattices*, i.e., lattices in $\mathbb{Z}^n$ that are closed under the operation that cyclically rotates the coordinates and negates the cycled element. The vector $\mathbf{f}$ corresponding to $f(x) = x^n + 1$ is $\mathbf{f} = (1, 0, \dots, 0)$ and therefore the basis $\mathbf{A}$ is:

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

Notice that using ideal lattices we are able to express a rank n ideal lattice with only n values, rather than $n \times n$ as is the case for general lattices, which allows a more compact representation that requires less storage space.

Given a prime q, let R_q be $\mathbb{Z}_q[x]/\langle f(x) \rangle$. Henceforth we will write a either as a polynomial $a = a_1 + a_2 x + a_3 x^2 + \dots + a_n x^{n-1} \in R_q$ or as a vector with coefficients $(a_1, a_2, a_3, \dots, a_n) \in \mathbb{Z}_q^n$. Notice that given two polynomials $a \in R_q$ and $p \in R_q$, the product $a \cdot p$ in R_q is equivalent to the product of the matrix $\mathbf{A}$ with the vector $\mathbf{p} = (p_1, \dots, p_n)$. Working with the polynomial representation in R_q allows a speedup in operations commonly used in lattice-based schemes: polynomial multiplication can be performed in $\mathcal{O}(n \log n)$ scalar operations, and in parallel depth $\mathcal{O}(\log n)$, using the Fast Fourier Transform (FFT).

2.2 RLWE encryption scheme

Let R_q be the ring of integer polynomials $R_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ where n is a power of 2, q is a prime; and let χ_σ be a discretized Gaussian distribution with standard deviation $\sigma = \alpha q/\sqrt{2\pi}$.

Definition 2 (RLWE distribution). *Given the "secret" $s \in R_q$ and an error distribution χ_σ, the RLWE distribution $A_{s,\chi}$ over $R_q \times R_q$ consists of samples of the form $(a, b = a \cdot s + e) \in R_q \times R_q$ where $a \leftarrow R_q$ is chosen uniformly, and e is the error polynomial sampled from the error distribution χ_σ.*

Definition 3 (*Search* RLWE). *Given many samples $(a_i, b_i = a_i \cdot s + e_i) \in R_q \times R_q$ from the RLWE distribution $A_{s,\chi}$ the goal is to recover the "secret" $s \in R_q$ with high probability,*

Definition 4 (*Decision* RLWE). *Given a vector (a, b) the goal is to efficiently distinguish if it has been sampled uniformly at random from $R_q \times R_q$ or from a RLWE distribution $A_{s,\chi}$.*

Hardness of RLWE. For any large enough q, solving certain instantiations of the *search* RLWE problem is at least as hard as quantumly solving a corresponding $\mathsf{poly(n)}$-approximate Shortest Vector Problem (*approx*-SVP) on any ideal lattice. On the other hand, solving *decision* RLWE in any cyclotomic ring (for any $\mathsf{poly(n)}$-bounded prime $q = 1 \mod n$) is as hard as solving *search* RLWE. See [25] for the details about the hardness of RLWE and the quantum reduction from worst-case *approx*-SVP on ideal lattices to the *search* RLWE.

RLWE parameters. How to choose secure parameters for lattice based cryptosystems is still an open question, nevertheless there are some parameters' proposals in the literature that take into account various security levels or attacker types [34], that consider the requirements of security reductions [32] or that consider an upper bound on the decryption error probability [20].

A RLWE encryption scheme is a triplet (KeyGen, Enc, Dec) which operates on rings such as R_q for which the RLWE is difficult to be solved. The original definition of the algorithm requires choosing *small* elements in R_q from an error distribution at several points. For practical purposes we will construct these *small* elements by taking their coefficients from an error distribution χ_σ which will be a discrete Gaussian distribution with parameter σ, as defined above. The RLWE encryption scheme that we are going to use is that proposed in [25] which defines the following algorithms:

$\mathsf{KeyGen}()$: choose a uniformly random element $a \in R_q$ as well as two random *small* elements $s, e \in R_q$ from the error distribution. Output $sk = s$ as the secret key and the pair $pk = (a, b = a \cdot s + e) \in R_q \times R_q$ as the public key.

$\mathsf{Encrypt}(pk, z, r, e_1, e_2)$: to encrypt an n-bit message $z \in \{0,1\}^n$, we view it as an element of R_q by using its bits as the 0-1 coefficients of a polynomial. The encryption algorithm then chooses three random *small* elements $r, e_1, e_2 \in R_q$ from the error distribution and outputs the pair $(u, v) \in R_q \times R_q$, as the encryption of z: $(u, v) = (r \cdot a + e_1 \mod q,\ \ b \cdot r + e_2 + \lfloor \frac{q}{2} \rceil z \mod q) \in R_q \times R_q$

$\mathsf{Decrypt}(sk, (u, v))$: the decryption algorithm simply computes $v - u \cdot s = (r \cdot e - s \cdot e_1 + e_2) + \lfloor \frac{q}{2} \rceil z \mod q$.

For an appropiate choice of parameters (namely q and σ) the coefficients of $r \cdot e - s \cdot e_1 + e_2$ have magnitude less than $q/4$, so the bits of z can be recovered by rounding each coefficient of $v - u \cdot s$ back to either 0 or $\lfloor \frac{q}{2} \rceil$, whichever is closest modulo q. Notice that in the RLWE encryption scheme presented above the secret s is taken from the error distribution, as well as r, e_1 and e_2. This is done in order to build efficient encryption schemes and it is demonstrated in [4] that the hardness of the underlying problem is not affected by this change.

Security. The RLWE encryption scheme is semantically secure given the pseudorandomness of the RLWE samples [25]. Notice that both the public key (a, b) and the ciphertexts are RLWE samples, where the encrypted messages can be seen as the pairs $(a, u), (b, v) \in R_q \times R_q$ (ignoring the message component $\lfloor \frac{q}{2} \rceil z \mod q$) with secret r. Then, these values are pseudorandom. Consequently, the encryption of a message using the RLWE encryption scheme is indistinguishable from an element sampled uniformly at random from $R_q \times R_q$ as long as the number of samples containing the same secret r is polynomial on the security parameter. The number of nodes in a mix-net is fixed and every re-encryption is computed using different parameters r, so the security is not compromised.

For our proposal we will need not only to encrypt messages but also to re-encrypt them. Since an RLWE encryption scheme is an additive homomorphic cryptosystem, we can re-encrypt a message just adding to the original ciphertext the encryption of the neutral element, that is, the encryption of a polynomial whose coefficients are 0. Following the same argumentation as above we can conclude that semantic security in an RLWE encryption scheme implies semantic security under re-encryption.

$\mathsf{Reencrypt}((u, v), r', e'_1, e'_2)$: to re-encrypt an n-bit message z, the algorithm chooses three random *small* elements $r', e'_1, e'_2 \in R_q$ from the error distribution and outputs the pair $(u', v') = (u, v) + \mathsf{Encrypt}(pk, 0, r', e'_1, e'_2) \in R_q \times R_q$.

Decrypting this re-encrypted ciphertext we would obtain $v' - u' \cdot s = (r + r') + (e_2 + e'_2) - s \cdot (e_1 + e'_1) + \left\lfloor \frac{q}{2} \right\rceil z$. The plaintext is preserved but the error terms may grow after every homomorphic operation. In order to avoid decrypting errors the number of mixing nodes must be taken into account when choosing the parameters q and σ, such that the error is still small compared to q even after as many re-encryptions as mixing nodes we are planning to use.

2.3 Zero knowledge proofs

A zero-knowledge proof is a protocol between two parties, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, where the first tries to convince the second that it knows some secret w that satifies a public relation $(x, w) \in R$ (where x would be some public information), for instance, that $\mathcal{P}$ knows the discrete logarithm of a public element. This proof is done in such a way that the prover does not reveal any information beyond the fact that a certain statement is true.

Definition 5. *A two party protocol $(\mathcal{P}, \mathcal{V})$ is a Σ-protocol [12] for relation R if it is a three round public-coin protocol of the form:*

1. *The prover $\mathcal{P}$ sends a message t to the verifier $\mathcal{V}$.*
2. *$\mathcal{V}$ sends a random string e to $\mathcal{P}$.*
3. *$\mathcal{P}$ sends a response s to $\mathcal{V}$. The verifier decides to accept or reject the proof based on the protocol transcript (t, e, s).*

and the following requirements hold:

- ***Completeness:*** *if an honest prover $\mathcal{P}$ knows w satisfying the relation $(x, w) \in R$, then $\mathcal{V}$ always accepts.*
- ***Special soundness:*** *given two accepted protocol transcripts (t, e, s) and (t, e', s') where $e \neq e'$, there exists a Probabilistic Polynomial-Time (PPT) algorithm which outputs w such that $(x, w) \in R$.*
- ***Special honest-verifier zero-knowledge:*** *given a pair (x, e) there exists a PPT algorithm that outputs a valid protocol transcript (t, e, s) with the same probability distribution as transcripts between the honest $\mathcal{P}$ and $\mathcal{V}$.*

The underlying structure behind the zero-knowledge proofs constructed in our proposal is that of a Σ-protocol. As Terelius and Wikström mention in their article [38], we need to prove knowledge on how to open commitments such that the committed values satisfy a public polynomial relation.

$$\Sigma\text{-}\mathsf{proof}[\mathbf{e} \in \mathbb{Z}_q^N, s \in \mathbb{Z}_q | a = \mathsf{Com}(\mathbf{e}, s) \wedge f(\mathbf{e}) = e')] \tag{2}$$

We refer the reader to [38] for more details on how this can be done.

2.4 Pedersen commitments

Let p and q be large primes, $\mathbb{Z}_p^*$ a group of integers modulo $p = 2q + 1$ and $G_q \subset \mathbb{Z}_p^*$ a subgroup of order q where the discrete logarithm assumption holds. Given two independent generators $\{g, g_1\}$ of G_q, to commit to a message $x \in \mathbb{Z}_q$ using the Pedersen commitment scheme [31], choose a random $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_q$ and output $\mathsf{Com}(x, \alpha) = g^\alpha g_1^x$. In order to open this commitment simply reveal the values α and x. This scheme is *perfectly hiding* and *computationally binding* as long as the discrete logarithm problem is hard in G_q.

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

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

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

$$\mathsf{Com}(\mathbf{M}, \alpha_1, \alpha_2, \ldots, \alpha_N) = (\mathsf{Com}(\mathbf{m_1}, \alpha_1), \ldots, \mathsf{Com}(\mathbf{m_N}, \alpha_N)) \tag{3}$$

Due to the homomorphic property of the Pedersen commitment we can compute a commitment to the product of a matrix $\mathbf{M}$ by a vector $\mathbf{x}$ from the commitment to the matrix $\mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha}) = (c_{\boldsymbol{m_1}}, \ldots, c_{\boldsymbol{m_N}})$.

$$\prod_{j=1}^{N} c_{\mathbf{m_j}}^{x_j} = \prod_{j=1}^{N} \left(g^{\alpha_j} \prod_{i=1}^{N} g_i^{m_{i,j}} \right)^{x_j} = g^{\langle \boldsymbol{\alpha}, \mathbf{x} \rangle} \prod_{i=1}^{N} g_i^{\langle (m_{i,1}, \ldots, m_{i,N}), (x_1, \ldots, x_N) \rangle} \tag{4}$$

3 Shuffling Ring-LWE encryptions

In this section we first present an overview of the mixing protocol and then we explain in more detail how is it proved that the committed matrix is a permutation matrix, the random re-encryption paramenters are *small* and that all these values have been used to perform the shuffle.

Let $M_1, \ldots, M_k$ be the mix-nodes that participate in the mixing protocol and let N be the number of encrypted messages at the input of each node.

3.1 Protocol overview

Given the the ring R_q and the encryption scheme presented in section 2, and the matrices $\mathbf{A}$ and $\mathbf{B}$ constructed from vectors $\mathbf{a}$ and $\mathbf{b}$ (RLWE public key) following equation 1; we can express a ciphertext $(u^{(i)}, v^{(i)}) \in R_q^2$ as a vector of $2n$ elements $(\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}) = (u_1^{(i)}, \ldots, u_n^{(i)}, v_1^{(i)}, \ldots, v_n^{(i)}) \in \mathbb{Z}_q^{2n}$ and its re-encryption as:

$$\left(\boldsymbol{u}'^{(i)}\ \boldsymbol{v}'^{(i)}\right)^T = \left(\boldsymbol{u}^{(i)}\ \boldsymbol{v}^{(i)}\right)^T + \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} (\boldsymbol{r}'^{(i)})^T + \left(\boldsymbol{e}_1'^{(i)}\ \boldsymbol{e}_2'^{(i)}\right)^T \qquad \forall i \in [1, \ldots, N]$$

Following this notation and given a permutation π characterized by the matrix $\mathbf{M}$ and a set of re-encryption parameters $\left(\boldsymbol{r}'^{(i)}, \boldsymbol{e}_1'^{(i)}, \boldsymbol{e}_2'^{(i)}\right)$ for each one of the messages, we can express the shuffling of N RLWE encryptions as:

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

$$+ \underbrace{\begin{pmatrix} r_1'^{(1)} & \cdots & r_n'^{(1)} \\ \vdots & \ddots & \vdots \\ r_1'^{(N)} & \cdots & r_n'^{(N)} \end{pmatrix}}_{N\times n} \underbrace{\begin{pmatrix} a_1 & \cdots & a_n & b_1 & \cdots & b_n \\ -a_n & \cdots & a_{n-1} & b_2 & \cdots & b_{n-1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -a_2 & \cdots & a_1 & -b_2 & \cdots & b_1 \end{pmatrix}}_{n\times 2n} + \underbrace{\begin{pmatrix} e_{1,1}'^{(1)} & \cdots & e_{1,n}'^{(1)} & e_{2,1}'^{(1)} & \cdots & e_{2,n}'^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e_{1,1}'^{(N)} & \cdots & e_{1,n}'^{(N)} & e_{2,1}'^{(N)} & \cdots & e_{2,n}'^{(N)} \end{pmatrix}}_{N\times 2n}$$

$$\left(\mathbf{U}''\ \mathbf{V}''\right) = \mathbf{M}\left(\mathbf{U}\ \mathbf{V}\right) + \mathbf{R}'\left(\mathbf{A}^T\ \mathbf{B}^T\right) + \left(\mathbf{E_1'}\ , \mathbf{E_2'}\right) \tag{5}$$

A mix-net node should prove that it knows the matrices $\mathbf{M}, \mathbf{R}', \mathbf{E_1'}, \mathbf{E_2'}$ such that the output of the node $\left(\mathbf{U}''\ \mathbf{V}''\right)$ is the input $\left(\mathbf{U}\ \mathbf{V}\right)$ re-encrypted and permuted, without revealing any information about $\mathbf{M}, \mathbf{R}', \mathbf{E_1'}$ and $\mathbf{E_2'}$.

$$\Sigma\text{-proof}\left[\begin{array}{c|c} \begin{array}{c} \pi \\ \boldsymbol{r}'^{(1)}, \ldots, \boldsymbol{r}'^{(N)} \\ \boldsymbol{e}_1'^{(1)}, \ldots, \boldsymbol{e}_1'^{(N)} \\ \boldsymbol{e}_2'^{(1)}, \ldots, \boldsymbol{e}_2'^{(N)} \end{array} & \begin{array}{c} \left(\left(\boldsymbol{u}''^{(1)}, \boldsymbol{v}''^{(1)}\right), \ldots, \left(\boldsymbol{u}''^{(N)}, \boldsymbol{v}''^{(N)}\right)\right)^T \\ = \\ \begin{pmatrix} \mathsf{Re\text{-}encrypt}\left(\left(\boldsymbol{u}^{\pi(1)}, \boldsymbol{v}^{\pi(1)}\right), \boldsymbol{r}'^{(1)}, \boldsymbol{e}_1'^{(1)}, \boldsymbol{e}_2'^{(1)}\right)^T \\ \cdots \\ \mathsf{Re\text{-}encrypt}\left(\left(\boldsymbol{u}^{\pi(N)}, \boldsymbol{v}^{\pi(N)}\right), \boldsymbol{r}'^{(N)}, \boldsymbol{e}_1'^{(N)}, \boldsymbol{e}_2'^{(N)}\right)^T \end{pmatrix} \end{array} \end{array}\right]$$

Following Wikström's proposal we are going to split the proof into two protocols.

Offline phase

1. The mix-node M_j chooses a random permutation π_j characterized by the matrix $\boldsymbol{M}_j \in \mathbb{Z}_q^{N\times N}$, computes a matrix commitment $\mathsf{Com}(\mathbf{M}_j, \boldsymbol{\alpha}_{m_j})$ and publishes it. It also proves knowledge of the committed permutation.
2. M_j chooses randomly the re-encryption parameters: $\mathbf{R}'_j \in \mathbb{Z}_q^{N\times n}, \mathbf{E'_1}_j \in \mathbb{Z}_q^{N\times n}$ and $\mathbf{E'_2}_j \in \mathbb{Z}_q^{N\times n}$. It computes the corresponding matrix commitments, publishes them and prove that the committed elements are *small*.

Online phase

1. Given a list of N input ciphertexts, the mix-node M_j permutes and re-encrypts the list using equation 5.
2. In order to prove that the committed matrices have been used to perform the mixing, M_j computes the commitment to its output using those commitments calculated during the online phase, and finally reveals its opening.

3.2 Proof of Knowledge of Permutation Matrix

The permutation matrix is characterized by the following theorem.

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

We refer the reader to [38] for the details about the theorem's proof.

Given a commitment to a matrix $\mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha_m}) = (c_{\mathbf{m_1}}, \ldots, c_{\mathbf{m_N}})$ and a vector $\mathbf{x} = (x_1, \ldots, x_N)$, we can compute a commitment to the product of the matrix by a vector $\mathsf{Com}(\mathbf{Mx}, k)$ using equation 4, where $k = \langle \boldsymbol{\alpha_m}, \mathbf{x} \rangle$. In the special case where the vector $\mathbf{x} = \mathbf{1}$ the identity above is $\mathsf{Com}(\mathbf{1}, t)$ where $t = \sum_{i=1}^N \alpha_{m_j}$. Another important observation is that given a vector $\hat{\mathbf{r}} = (\hat{r}_1, \ldots, \hat{r}_N)$ we can express a commitment to the product of the elements of $\mathbf{x}'$ in a recursive way $\hat{c}_i = g^{\hat{r}_i} \hat{c}_{i-1}^{x'_i}$ for $i = 1, \ldots, N$ and $\hat{c}_0 = g_1$.

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

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

Given these preliminaries we can construct a Σ-proof to prove that the mix-net node knows an opening for the commitment and that the element committed is a permutation matrix. Since this proof follows the approach given by Wikström, we left the details for the Appendix A.

3.3 Proof of knowledge of small exponents

The second step in the offline part will be to prove that the random values used to re-encrypt are small. Remember that in order to re-encrypt a message, the following randomness is used: $\mathbf{r}'^{(i)} = \left(r_1'^{(i)}, \ldots, r_n'^{(i)}\right)$, $\mathbf{e}_1'^{(i)} = \left(e_{1,1}'^{(i)}, \ldots, e_{1,n}'^{(i)}\right)$ and $\mathbf{e}_2'^{(i)} = \left(e_{2,1}'^{(i)}, \ldots, e_{2,n}'^{(i)}\right)$ for $i \in [1, \ldots, N]$. In our case, we would require that the coefficients of these vectors belong to $[-\beta + 1, \beta - 1]$ where $\beta = 2^k$. In order to prove this we are going to use the strategy proposed in [21] by Ling *et al.* As it is explained in [5] the probability of obtaining an element from the error distribution with norm larger than β is negligible (notice that β will depend on the parameters of the encryption). Even when this restriction on the re-encryption elements norm is applied, the RLWE samples remain pseudorandom. This prevents a corrupted node from modifiying the plaintext of the ciphertexts, while an honest node can still use the pseudorandomness to hide the relation between its input an output.

We decompose $r_j'^{(i)} = \sum_{l=0}^{k-1} r_{j,l}'^{(i)} 2^l$, $e_{1,j}'^{(i)} = \sum_{l=0}^{k-1} e_{1,j,l}'^{(i)} 2^l$ and $e_{2,j}'^{(i)} = \sum_{l=0}^{k-1} e_{2,j,l}'^{(i)} 2^l$, with $r_j'^{(i)}, e_{1,j,l}'^{(i)}, e_{2,j,l}'^{(i)} \in \{-1, 0, 1\}$ and we prove that these elements have one of the possible values in the set $\{-1, 0, 1\}$ using an OR-proof. Afterwards, using the commitment to every bit of the decomposition we obtain a commitment to the coefficients, and consequently a commitment to each of the corresponding matrix columns. The protocol used to demonstrate that a value belongs to a specific set, $x \in \{-1, 0, 1\}$, is based on a zero knowledge proof that proves that the element x has one of the values in the set without revealing which one it is.

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

Informally, the proof consists of computing three proofs simultaneously, for $x = -1, x = 0$ and $x = 1$, where two of them will be simulated and only that which corresponds to the real value of x will be the *real* proof. As this is a standard proof [11] the details are omitted here and both the proof and the demonstration of its properties are given in Appendix B.

3.4 Opening the commitments

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

Given the commitments $c_{\boldsymbol{m_j}} = \mathsf{Com}\left(\boldsymbol{m_j}, \alpha_{\boldsymbol{m_j}}\right)$, $c_{\boldsymbol{r'_j}} = \mathsf{Com}\left(\boldsymbol{r'_j}, \alpha_{\boldsymbol{r'_j}}\right)$, $c_{\boldsymbol{e'_{1,j}}} = \mathsf{Com}\left(\boldsymbol{e'_{1,j}}, \alpha_{\boldsymbol{e'_{1,j}}}\right)$ and $c_{\boldsymbol{e'_{2,j}}} = \mathsf{Com}\left(\boldsymbol{e'_{2,j}}, \alpha_{\boldsymbol{e'_{2,j}}}\right)$ and equation 5, we can compute the following commitments to matrix products and sums,

$$c_{\mathbf{e}'_{1,k}} \left(\prod_{j=1}^{N} c_{\mathbf{m}_j}^{u_k^{(j)}}\right) \left(\prod_{j=1}^{n} c_{\mathbf{r}'_j}^{a_{k,j}}\right) = \mathsf{Com}\left(\hat{\mathbf{u}}''_{\mathbf{k}}, \alpha_{e'_{1,k}} + \langle \boldsymbol{\alpha}_M, \hat{\mathbf{u}}_{\mathbf{k}} \rangle + \langle \boldsymbol{\alpha}_{r'}, (a_{k,1}, \ldots, a_{k,n}) \rangle\right)$$

$$c_{\mathbf{e}'_{2,k}} \left(\prod_{j=1}^{N} c_{\mathbf{m}_j}^{v_k^{(j)}}\right) \left(\prod_{j=1}^{n} c_{\mathbf{r}'_j}^{b_{k,j}}\right) = \mathsf{Com}\left(\hat{\mathbf{v}}''_{\mathbf{k}}, \alpha_{e'_{2,k}} + \langle \boldsymbol{\alpha}_M, \hat{\mathbf{v}}_{\mathbf{k}} \rangle + \langle \boldsymbol{\alpha}_{r'}, (b_{k,1}, \ldots, b_{k,n}) \rangle\right)$$

denoting $\hat{\mathbf{u}}_{\mathbf{k}}, \hat{\mathbf{v}}_{\mathbf{k}}, \hat{\mathbf{u}}''_{\mathbf{k}}, \hat{\mathbf{v}}''_{\mathbf{k}}$ the corresponding k-column of each matrix $\mathbf{U}, \mathbf{V}, \mathbf{U}'', \mathbf{V}''$. The only thing that the mix node should do in order to prove that it has used the appropiate values during the shuffling, is to open the commitments above revealing the openings.

$$\left(\alpha_{e'_{1,k}} + \left\langle \boldsymbol{\alpha}_M, \left(u_k^{(1)}, \ldots, u_k^{(N)}\right)\right\rangle + \langle \boldsymbol{\alpha}_{r'}, (a_{k,1}, \ldots, a_{k,n}) \rangle\right) \quad \forall k \in [1, \ldots, n]$$
$$\left(\alpha_{e'_{2,k}} + \left\langle \boldsymbol{\alpha}_M, \left(v_k^{(1)}, \ldots, v_k^{(N)}\right)\right\rangle + \langle \boldsymbol{\alpha}_{r'}, (b_{k,1}, \ldots, b_{k,n}) \rangle\right) \quad \forall k \in [1, \ldots, n]$$

The verifier has to check that these values are appropiate openings of the commitments in order to verify the node has used the committed matrices $\mathbf{M}, \mathbf{R}', \mathbf{E}'_1$ and $\mathbf{E}'_2$ to shuffle the encrypted messages (at its input).

As we have seen above, given the commitments to $\mathbf{M}, \mathbf{R}', \mathbf{E}'_1$ and $\mathbf{E}'_2$ we can compute the commitment to the matrix of permuted votes $\mathbf{M}\left(\mathbf{U}\ \mathbf{V}\right)$ and the re-encryption matrix $\left(\mathbf{R}'\left(\mathbf{A}^T\ \mathbf{B}^T\right) + \left(\mathbf{E}'_1\ \mathbf{E}'_2\right)\right)$. Notice that the $2n$ linear combinations of the values $\alpha_{m_j}, \alpha_{r'_j}, \alpha_{e'_{1,j}}, \alpha_{e'_{2,j}}$ that the mix node reveals, allow us to open the commitments to the sum of these matrices, but not to each matrix separately. Given that $\boldsymbol{\alpha}_M$, and $\boldsymbol{\alpha}_r$ appear on all the openings that we reveal we have to double check if they could leak any information about any relations between the α's that (in a post-quantum scenario) may reveal information about the permutation and the re-encryption elements. This is not the case because all the $\alpha_{e_{1',j}}$ and $\alpha_{e'_{2,j}}$ are uniformly and independently chosen from $\mathbb{Z}_q$. All the linear combinations that we reveal have a different $\alpha_{e'_{i,j}}$, and this implies that the combinations are also uniformly and independently distributed, and thereby it is impossible to isolate any of the α. The full protocol and a discussion about its properties are given in Appendix C.

4 Conclusions

We have proposed the first universally verifiable proof of a shuffle for a lattice-based cryptosystem. The messages at the input of the mix-net are encrypted using an RLWE encryption system and then they are shuffled by the mixing nodes. In order to prove the correctness of this shuffle each node must provide a proof of a shuffle, demonstrating that the protocol has been executed correctly without leaking any secret information. Our proposal follows the idea presented in [42] but introduces two significant differences: during the offline part the random elements used to re-encrypt the ciphertexts are committed using the generalized version of Pedersen commitment and it is proved that these elements belong to a certain interval using OR-proofs. On the other hand, during the online part each node computes a commitment to its output using the homomorphic properties of both the commitment scheme and the encryption scheme. Opening this commitment the mix node proves that it has used the values committed during the offline part to compute its output. Revealing this opening does not give any information about the secret information required to do the shuffling.

It is worth noticing that shuffling the votes is not enough to guarantee the voters' privacy, as the system can be insecure, for instance, due to malleability attacks [39]. To avoid this kind of attack additional security proofs might be provided before the mixing process starts.

Regarding efficiency, the number of OR-proofs to be computed by each mix node is proportional to knN, where N is the number of encrypted messages received by the node, n is the dimension of the lattice and k is the number of bits of each element of the re-encryption matrices. There are some techniques that allow to reduce the computational cost of these proofs and we leave for a future work to explore these improvements. We refer the reader to [42] for the details about the efficiency of the ZKP for a permutation matrix.

References

1. M. Abe. Mix-networks on permutation networks. In *Proc. of the Int. Conf. on the Theory and Applications of Cryptology and Information Security*, ASIACRYPT '99, pages 258–273, London, UK, 1999. Springer-Verlag.
2. B. Adida and D. Wikström. How to shuffle in public. In *Proc. of the 4th Conf. on Theory of Cryptography*, TCC'07, pages 555–574, Berlin, Heidelberg, 2007. Springer-Verlag.
3. B. Adida and D. Wikström. Offline/online mixing. In *Proc. of the 34th Int. Conf. on Automata, Languages and Programming*, ICALP'07, pages 484–495, Berlin, Heidelberg, 2007. Springer-Verlag.
4. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Proc. of the 29th Annual Int. Cryptology Conf. on Advances in Cryptology*, CRYPTO '09, pages 595–618, Berlin, Heidelberg, 2009. Springer-Verlag.
5. F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *Proc., Part I, of the 20th European Symposium on Computer Security – ESORICS 2015 - Volume 9326*, pages 305–325, NY, USA, 2015. Springer-Verlag.
6. J. Buchmann, D. Demirel, and J. Van de Graaf. Towards a publicly-verifiable mix-net providing everlasting privacy. In A.R Sadeghi, editor, *17th Int. Conf. on Financial Cryptography*, volume LNCS 7859 of *FC'13*, pages 197–204, 2013.
7. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 136–, Washington, USA, 2001. IEEE Comp. Soc.
8. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

9. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. A homomorphic LWE based e-voting scheme. In *Proc. of the 7th Int. Workshop on PQ Crypto*, volume 9606 of *PQCrypto 2016*, pages 245–265, NY, USA, 2016. Springer-Verlag.
10. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proc. of the 16th Annual Int. Conf. on Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, pages 103–118, Berlin, Heidelberg, 1997. Springer-Verlag.
11. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proc. of the 16th Annual Int. Conf. on Theory and Application of Cryptographic Techniques*, EUROCRYPT'97, pages 103–118, Berlin, Heidelberg, 1997. Springer-Verlag.
12. I. Damgard. On σ-protocols. Lecture on Cryptologic Protocol Theory; Faculty of Science, University of Aarhus, 2010.
13. D. Demirel, M. Henning, J. Van de Graaf, P. Y. A. Ryan, and J. Buchmann. Prêt à voter providing everlasting privacy. In *Proc. of the 4th Int. Conf. on E-Voting and Identity*, Vote-ID'13, pages 156–175, Berlin, Heidelberg, 2013. Springer-Verlag.
14. P. Fauzi and H. Lipmaa. Efficient culpably sound NIZK shuffle argument without random oracles. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conf. 2016*, pages 200–216, San Francisco, CA, USA, 2016.
15. P. Fauzi, H. Lipmaa, and M. Zajac. A shuffle argument secure in the generic model. *IACR Cryptology ePrint Archive*, 2016:866, 2016.
16. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of the 21st Annual Int. Cryptology Conf. on Advances in Cryptology*, CRYPTO '01, pages 368–387, London, UK, 2001. Springer-Verlag.
17. P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In T. Okamoto, editor, *Topics in Cryptology – CT-RSA 2004: The Cryptographers' Track, San Francisco, USA*, pages 163–178, Berlin, Heidelberg, 2004. Springer-Verlag.
18. J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Proc. of the 6th Int. Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, PKC '03, pages 145–160, London, UK, UK, 2003. Springer-Verlag.
19. J. Groth and S. Lu. A non-interactive shuffle with pairing based verifiability. In *Proc. of the Advances in Crypotology 13th Int. Conf. on Theory and Application of Cryptology and Information Security*, ASIACRYPT'07, pages 51–67, Berlin, Heidelberg, 2007. Springer-Verlag.
20. R. Lindner and C. Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Proc. of the 11th Int. Conf. on Topics in Cryptology: CT-RSA 2011*, pages 319–339, Berlin, Heidelberg, 2011. Springer-Verlag.
21. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In K. Kurosawa and G. Hanaoka, editors, *PKC 2013: 16th Int. Conf. on Practice and Theory in Public-Key Cryptography*, pages 107–124, Berlin, Heidelberg, 2013. Springer-Verlag.
22. H. Lipmaa and B. Zhang. A more efficient computationally sound non-interactive zero-knowledge shuffle argument. In *Security and Cryptography for Networks - 8th Int. Conf., SCN 2012*, pages 477–502, Amalfi, Italy, 2012.
23. P. Locher and R. Haenni. Verifiable internet elections with everlasting privacy and minimal trust. In *Proc. of the 5th Int. Conf. on E-Voting and Identity - Volume 9269*, VoteID 2015, pages 74–91. Springer-Verlag New York, Inc., 2015.
24. P. Locher and R. E. Haenni, R.and Koenig. *Coercion-Resistant Internet Voting with Everlasting Privacy*, pages 161–175. Springer Berlin Heidelberg, 2016.

25. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, 2013.
26. J. Markus and J. Ari. Millimix: Mixing in small batches. Technical report, 1999.
27. D. Micciancio and O. Regev. Lattice-based cryptography. In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer-Verlag, Berlin, Heidelberg, 2009.
28. T. Moran and M. Naor. Split-ballot voting: Everlasting privacy with distributed trust. In *Proc. of the 14th ACM Conf. on Computer and Communications Security*, CCS '07, pages 246–255. ACM, 2007.
29. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. of the 8th ACM Conf. on Comp. and Comm. Security*, CCS '01, pages 116–125, NY, USA, 2001.
30. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *In Advances in Cryptology*, EUROCRYPT '93, pages 248–259, Secaucus, NJ, USA, 1994. Springer-Verlag.
31. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of the 11th Annual Int. Cryptology Conf. on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, 1992. Springer-Verlag.
32. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.
33. O. Regev. The learning with errors problem. In *IEEE 25th Annual Conf. on Computational Complexity (CCC)*, pages 191–204, 2010.
34. M. Ruckert and M. Schneider. Estimating the security of lattice-based cryptosystems. *IACR Cryptology ePrint Archive*, 2010:137, 2010.
35. K. Sako and J. Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *Proc. of the 14th Annual Int. Conf.*, EUROCRYPT'95, pages 393–403, Berlin, Heidelberg, 1995. Springer-Verlag.
36. K. Singh, C. Pandu Rangan, and A. K. Banerjee. Lattice based universal re-encryption for mixnet. *Journal of Internet Services and Information Security (JISIS)*, 4(1):1–11, 2014.
37. K. Singh, C. Pandu Rangan, and A. K. Banerjee. Lattice based mix network for location privacy in mobile system. *Mobile Information Systems*, 2015:1–9, 2015.
38. B. Terelius and D. Wikström. Proofs of restricted shuffles. In D. J. Bernstein and T. Lange, editors, *AFRICACRYPT '10: Third Int. Conf. on Cryptology in Africa, Stellenbosch, South Africa*, pages 100–113. Springer Berlin Heidelberg, 2010.
39. D. Wikström. The security of a mix-center based on a semantically secure cryptosystem. In *INDOCRYPT*, pages 368–381. Springer, 2002.
40. D. Wikström. A universally composable mix-net. In M. Naor, editor, *First Theory of Cryptography Conf., TCC 2004, Cambridge, USA*, pages 317–335, Berlin, Heidelberg, 2004. Springer-Verlag.
41. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *Proc. of the 11th Int. Conf. on Theory and Application of Cryptology and Information Security*, ASIACRYPT'05, pages 273–292, Berlin, Heidelberg, 2005. Springer-Verlag.
42. D. Wikström. A commitment-consistent proof of a shuffle. In *Proc. of the 14th Australasian Conf. on Information Security and Privacy*, ACISP '09, pages 407–421, Berlin, Heidelberg, 2009. Springer-Verlag.

A Proof of a committed permutation matrix

As Wikström defines in his article, we let n_v, n_c and n_r denote the bitsize of components in random vectors, challenges, and random paddings respectively. The security parameters 2^{-n_v}, 2^{-n_c} and 2^{-n_r} must be negligible in n.

This protocol meets the requirements of completeness, soundess and zero knowledge defined in section 2[38]. We can construct a simulator selecting $B_1, \ldots, B_N \in G_q$, $\mathbf{d}, \mathbf{d'} \in \mathbb{Z}_q^N$ and $d_\alpha, d_\gamma, d_\delta \in \mathbb{Z}_q$ randomly, and computing $\alpha, \beta_i, \gamma, \delta$ using the verification equations. In order to prove the consistency we have to undo the built recurrences in the same way that Wikström explains in his article.

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

The protocol is shown in the next page.

$$\mathbf{r}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^N$$

$$s_\alpha \xleftarrow{\$} \mathbb{Z}_q$$

$$s_\gamma \xleftarrow{\$} \mathbb{Z}_q$$

$$s_\delta \xleftarrow{\$} \mathbb{Z}_q$$

$$\hat{n} = n_v + n_r + n_c$$

$$s' \xleftarrow{\$} \left[0, 2^{\hat{n}} - 1\right]$$

$$B_0 = g_1$$

$$B_i = g^{r_i} B_{i-1}^{\lambda'_i}$$

$$\alpha = g^{s_\alpha} \prod_{i=1}^{N} g_i^{s'_i}$$

$$\beta_i = g^{s_i} B_{i-1}^{s'_i}$$

$$\gamma = g^{s_\gamma}$$

$$\delta = g^{s_\delta}$$

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

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

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

$$\lambda''_1 = s_1$$

$$\lambda''_i = \lambda''_{i-1} \lambda'_i + s_i$$

$$d_\alpha = ck + s_\alpha$$

$$d'_i = c\lambda'_i + s'_i$$

$$d_i = cr_i + s_i$$

$$d_\gamma = c \left\langle \mathbf{s}, \mathbf{1} \right\rangle + s_\gamma$$

$$d_\delta = c\lambda''_N + s_\delta$$

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

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

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

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

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

B OR-proof for re-encryption parameters

In this appendix we give the details of the proof that an element x is in the set $\{-1, 0, 1\}$.

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

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

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

$$k \overset{\$}{\leftarrow} \mathbb{Z}_q$$

$$\mathcal{P} \overset{k}{\leftarrow} \mathcal{V}$$

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

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

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

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

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

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

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

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

In order to prove the *consistency*, we define two accepted transcriptions of the protocol:

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

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

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

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

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

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

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

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

C Full protocol and its properties

The mix node output is defined by the following equation:

$$\left(\mathbf{U}''\ \mathbf{V}''\right) = \mathbf{M}\left(\mathbf{U}\ \mathbf{V}\right) + \mathbf{R}'\left(\mathbf{A}^T\ \mathbf{B}^T\right) + \left(\mathbf{E_1'}\ ,\mathbf{E_2'}\right)$$

where matrices $\mathbf{M}, \mathbf{R}', \mathbf{E_1'}, \mathbf{E_2'}$ are selected by the mix node and kept secret and $\left(\mathbf{U}''\ \mathbf{V}''\right), \left(\mathbf{U}\ \mathbf{V}\right), \left(\mathbf{A}\ \mathbf{B}\right)$ are public values since they are the output and the input of the mix node, and the public key of the encryption scheme, respectively. The full protocol run by each node consists of committing the secret matrices, demonstrating that the committed matrix $\mathbf{M}$ is a permutation matrix (see Appendix A), that the committed re-encryption parameters $\mathbf{R}', \mathbf{E_1'}, \mathbf{E_2'}$ are *small* (see Appendix B) and finally on verifying that the committed values have been used to perform the mixing (see Section 3.4). Briefly, the proof computed by the mix node can be expressed as:

$$\Sigma\text{-proof}\left[\begin{array}{c|c} \begin{array}{c} \pi \\ \boldsymbol{r}'^{(1)}, \ldots, \boldsymbol{r}'^{(N)} \\ \boldsymbol{e}_1'^{(1)}, \ldots, \boldsymbol{e}_1'^{(N)} \\ \boldsymbol{e}_2'^{(1)}, \ldots, \boldsymbol{e}_2'^{(N)} \end{array} & \begin{array}{c} \left(\left(\boldsymbol{u}''^{(1)}, \boldsymbol{v}''^{(1)}\right), \ldots, \left(\boldsymbol{u}''^{(N)}, \boldsymbol{v}''^{(N)}\right)\right)^T \\ = \\ \begin{pmatrix} \mathsf{Re\text{-}encrypt}\left(\left(\boldsymbol{u}^{\pi(1)}, \boldsymbol{v}^{\pi(1)}\right), \boldsymbol{r}'^{(1)}, \boldsymbol{e}_1'^{(1)}, \boldsymbol{e}_2'^{(1)}\right)^T \\ \cdots \\ \mathsf{Re\text{-}encrypt}\left(\left(\boldsymbol{u}^{\pi(N)}, \boldsymbol{v}^{\pi(N)}\right), \boldsymbol{r}'^{(N)}, \boldsymbol{e}_1'^{(N)}, \boldsymbol{e}_2'^{(N)}\right)^T \end{pmatrix} \end{array} \end{array}\right]$$

Recall that the commitments are calculated in the following way:

$$\begin{aligned} \mathsf{Com}(\mathbf{M}, \boldsymbol{\alpha_m}) &= (c_{\mathbf{m_1}}, \ldots, c_{\mathbf{m_N}}) \\ \mathsf{Com}(\mathbf{R}', \boldsymbol{\alpha_r'}) &= (c_{\mathbf{r_1'}}, \ldots, c_{\mathbf{r_n'}}) \\ \mathsf{Com}(\mathbf{E_1'}, \boldsymbol{\alpha_{e_1'}}) &= (c_{\mathbf{e_{1,1}'}}, \ldots, c_{\mathbf{e_{1,n}'}}) \\ \mathsf{Com}(\mathbf{E_2'}, \boldsymbol{\alpha_{e_2'}}) &= (c_{\mathbf{e_{2,1}'}}, \ldots, c_{\mathbf{e_{2,n}'}}) \end{aligned}$$

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

$$\begin{aligned} c_{\mathbf{m}_j} &= \mathsf{Com}(\mathbf{m_j}, \alpha_{m_j}) = g^{\alpha_{m_j}} \prod_{i=1}^{N} g_i^{m_{i,j}} && \forall j \in [1, \ldots, N] \\ c_{\mathbf{r}_j'} &= \mathsf{Com}(\mathbf{r_j'}, \alpha_{r_j'}) = g^{\alpha_{r_j'}} \prod_{i=1}^{N} g_i^{r_j'^{(i)}} && \forall j \in [1, \ldots, n] \\ c_{\mathbf{e}_{1,j}'} &= \mathsf{Com}(\mathbf{e_{1,j}'}, \alpha_{e_{1,j}'}) = g^{\alpha_{e_{1,j}'}} \prod_{i=1}^{N} g_i^{e_{1,j}'^{(i)}} && \forall j \in [1, \ldots, n] \\ c_{\mathbf{e}_{2,j}'} &= \mathsf{Com}(\mathbf{e_{2,j}'}, \alpha_{e_{2,j}'}) = g^{\alpha_{e_{2,j}'}} \prod_{i=1}^{N} g_i^{e_{2,j}'^{(i)}} && \forall j \in [1, \ldots, n] \end{aligned}$$

Moreover, in order to prove that the re-encryption parameters are *small* they are represented using their bit decomposition:

$$\underbrace{\begin{pmatrix} r_1'^{(1)} \\ r_2'^{(1)} \\ \vdots \\ r_n'^{(1)} \\ r_1'^{(2)} \\ \vdots \\ r_n'^{(N)} \end{pmatrix}}_{nN\times 1} = \underbrace{\begin{pmatrix} r_{1,0}'^{(1)} & r_{1,1}'^{(1)} & \cdots & r_{1,k-1}'^{(1)} \\ r_{2,0}'^{(1)} & r_{2,1}'^{(1)} & \cdots & r_{2,k-1}'^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,0}'^{(1)} & r_{n,1}'^{(1)} & \cdots & r_{n,k-1}'^{(1)} \\ r_{1,0}'^{(2)} & r_{1,1}'^{(2)} & \cdots & r_{1,k-1}'^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,0}'^{(N)} & r_{n,1}'^{(N)} & \cdots & r_{n,k-1}'^{(N)} \end{pmatrix}}_{nN\times k} \underbrace{\begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{k-1} \end{pmatrix}}_{k\times 1}$$

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

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

$$c_{r_{j,l}'^{(i)}} = g^{\alpha_{r_{j,l}'^{(i)}}} g_i^{r_{j,l}'^{(i)}}$$
$$c_{e_{1,j,l}'^{(i)}} = g^{\alpha_{e_{1,j,l}'^{(i)}}} g_i^{e_{1,j,l}'^{(i)}}$$
$$c_{e_{2,j,l}'^{(i)}} = g^{\alpha_{e_{2,j,l}'^{(i)}}} g_i^{e_{2,j,l}'^{(i)}}$$

Using that $r_j'^{(i)} = \sum_{l=0}^{k-1} r_{j,l}'^{(i)} 2^l$, $e_{1,j}'^{(i)} = \sum_{l=0}^{k-1} e_{1,j,l}'^{(i)} 2^l$ and $e_{2,j}'^{(i)} = \sum_{l=0}^{k-1} e_{2,j,l}'^{(i)} 2^l$, is easy to compute the commitment to $c_{\mathbf{r}_j'}$, $c_{\mathbf{e}_{1,j}'}$, $c_{\mathbf{e}_{2,j}'}$ using the commitments mentioned above.

The full protocol is detailed in the following diagram. We discuss later the completeness, soundness and zero knowledge properties of the protocol.

Protocol 1.1: Mix-net protocol

$\mathcal{P}\left(\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}, \boldsymbol{u}''^{(i)}, \boldsymbol{v}''^{(i)}; \pi, \boldsymbol{r}'^{(i)}, \boldsymbol{e}_1'^{(i)}, \boldsymbol{e}_2'^{(i)}\right)$ $\qquad$ $\mathcal{V}\left(\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}, \boldsymbol{u}''^{(i)}, \boldsymbol{v}''^{(i)}\right)$

$\forall i \in [1, \ldots, N],\ \forall j \in [1, \ldots, n],$
$\forall l \in [0, \ldots, k-1]$

$\alpha_{r'^{(i)}_{j,l}}, \alpha_{e'^{(i)}_{1,j,l}}, \alpha_{e'^{(i)}_{2,j,l}} \xleftarrow{\$} \mathbb{Z}_q$

$\boldsymbol{c}_{r'^{(i)}_{j,l}} = \mathsf{Com}_{g,g_i}\left(r'^{(i)}_{j,l}, \alpha_{r'^{(i)}_{j,l}}\right)$

$\boldsymbol{c}_{e'^{(i)}_{1,j,l}} = \mathsf{Com}_{g,g_i}\left(e'^{(i)}_{1,j,l}, \alpha_{e'^{(i)}_{1,j,l}}\right)$

$\boldsymbol{c}_{e'^{(i)}_{2,j,l}} = \mathsf{Com}_{g,g_i}\left(e'^{(i)}_{2,j,l}, \alpha_{e'^{(i)}_{2,j,l}}\right)$

$$\xrightarrow{\boldsymbol{c}_{r'^{(i)}_{j,l}}, \boldsymbol{c}_{e'^{(i)}_{1,j,l}}, \boldsymbol{c}_{e'^{(i)}_{2,j,l}}}$$

$$\Sigma\text{-proof}\left[r'^{(i)}_{j,l} \middle| \left(\boldsymbol{c}_{r'^{(i)}_{j,l}} = \mathsf{Com}\left(r'^{(i)}_{j,l}\right)\right) \wedge \left(r'^{(i)}_{j,l} = -1 \vee r'^{(i)}_{j,l} = 0 \vee r'^{(i)}_{j,l} = 1\right) \right]$$

$$\Sigma\text{-proof}\left[e'^{(i)}_{1,j,l} \middle| \left(\boldsymbol{c}_{e'^{(i)}_{1,j,l}} = \mathsf{Com}\left(e'^{(i)}_{1,j,l}\right)\right) \wedge \left(e'^{(i)}_{1,j,l} = -1 \vee e'^{(i)}_{1,j,l} = 0 \vee e'^{(i)}_{1,j,l} = 1\right) \right]$$

$$\Sigma\text{-proof}\left[e'^{(i)}_{2,j,l} \middle| \left(\boldsymbol{c}_{e'^{(i)}_{2,j,l}} = \mathsf{Com}\left(e'^{(i)}_{2,j,l}\right)\right) \wedge \left(e'^{(i)}_{2,j,l} = -1 \vee e'^{(i)}_{2,j,l} = 0 \vee e'^{(i)}_{2,j,l} = 1\right) \right]$$

$\boldsymbol{\alpha_M} \xleftarrow{\$} \mathbb{Z}_q^N$
$\boldsymbol{c_M} = \mathsf{Com}(\boldsymbol{M}, \boldsymbol{\alpha_M})$

$$\xrightarrow{\boldsymbol{c_M}}$$

$$\boldsymbol{\lambda} \xleftarrow{\$} \mathbb{Z}_q^N$$

$$\xleftarrow{\boldsymbol{\lambda}}$$

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

$\alpha_{r'^{(i)}_j} = \sum_{l=0}^{k-1} 2^l \alpha_{r'^{(i)}_{j,l}}, \quad \alpha_{\boldsymbol{r}'_j} = \sum_{i=1}^N \alpha_{r'^{(i)}_j}, \quad \boldsymbol{\alpha_{r'}} = \left(\alpha_{\boldsymbol{r}'_1}, \ldots, \alpha_{\boldsymbol{r}'_n}\right)$

$\alpha_{e'^{(i)}_{1,j}} = \sum_{l=0}^{k-1} 2^l \alpha_{e'^{(i)}_{1,j,l}}, \quad \alpha_{\boldsymbol{e}'_{1,j}} = \sum_{i=1}^N \alpha_{e'^{(i)}_{1,j}}$

$\alpha_{e'^{(i)}_{2,j}} = \sum_{l=0}^{k-1} 2^l \alpha_{e'^{(i)}_{2,j,l}}, \quad \alpha_{\boldsymbol{e}'_{2,j}} = \sum_{i=1}^N \alpha_{e'^{(i)}_{2,j}}$

$\alpha_{1,j} = \left(\alpha_{e'_{1,j}} + \langle \boldsymbol{\alpha}_M, \boldsymbol{U}_j \rangle + \langle \boldsymbol{\alpha}_{r'}, (a_{j,1}, \ldots, a_{j,n}) \rangle\right)$

$\alpha_{2,j} = \left(\alpha_{e'_{2,j}} + \langle \boldsymbol{\alpha}_M, \boldsymbol{V}_j \rangle + \langle \boldsymbol{\alpha}_{r'}, (b_{j,1}, \ldots, b_{j,n}) \rangle\right)$

$$\xrightarrow{\alpha_{1,j}, \alpha_{2,j}}$$

$$\boldsymbol{c}_{r'^{(i)}_j} = \prod_{l=0}^{k-1} \boldsymbol{c}^{2^l}_{r'^{(i)}_{j,l}}, \quad \boldsymbol{c}_{\boldsymbol{r}'_j} = \prod_{i=1}^N \boldsymbol{c}_{r'^{(i)}_j}$$

$$\boldsymbol{c}_{e'^{(i)}_{1,j}} = \prod_{l=0}^{k-1} \boldsymbol{c}^{2^l}_{e'^{(i)}_{1,j,l}}, \quad \boldsymbol{c}_{\boldsymbol{e}'_{1,j}} = \prod_{i=1}^N \boldsymbol{c}_{e'^{(i)}_{1,j}}$$

$$\boldsymbol{c}_{e'^{(i)}_{2,j}} = \prod_{l=0}^{k-1} \boldsymbol{c}^{2^l}_{e'^{(i)}_{2,j,l}}, \quad \boldsymbol{c}_{\boldsymbol{e}'_{2,j}} = \prod_{i=1}^N \boldsymbol{c}_{e'^{(i)}_{2,j}}$$

$$c_{\mathbf{e}'_{1,k}} \left(\prod_{j=1}^N c_{\mathbf{m}_j}^{u_k^{(j)}}\right) \left(\prod_{j=1}^n c_{\mathbf{r}'_j}^{a_{k,j}}\right) \overset{?}{=} \mathsf{Com}\left(\mathbf{U}''_{\mathbf{k}}, \alpha_{1,k}\right)$$

$$c_{\mathbf{e}'_{2,k}} \left(\prod_{j=1}^N c_{\mathbf{m}_j}^{v_k^{(j)}}\right) \left(\prod_{j=1}^n c_{\mathbf{r}'_j}^{b_{k,j}}\right) \overset{?}{=} \mathsf{Com}\left(\mathbf{V}''_{\mathbf{k}}, \alpha_{2,k}\right)$$

We restate here the full protocol with its properties for completeness.

C.1 Completeness

Completeness follows from the homomorphic property of the Pedersen commitment and the completeness of the Σ-protocols for the *small* elements and the permutation matrix. The prover computes $\alpha_{1,j}$ and $\alpha_{2,j}$ for all $j \in [1, \dots, n]$ using the random elements from the initial commitments. Then, if the prover has been honest, the verifier builds the commitments to the output using the published commitments and applying equation 5, and check that $\alpha_{1,j}$ and $\alpha_{2,j}$ are valid openings for the output commitments.

C.2 Soundness

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

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

C.3 Zero Knowledge

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

Given $\boldsymbol{\lambda}$ and the responses of the Σ-protocols we choose π and $r'^{(i)}_{j,l}, e'^{(i)}_{1,j,l}, e'^{(i)}_{2,j,l}$ uniformly at random except for $e'^{(1)}_{1,j,0}, e'^{(1)}_{2,j,0}$. We compute its commitments, publish them and answer the challenges from the Σ-protocols as usual. Then we choose $\alpha_{1,k}$ and $\alpha_{2,k}$ uniformly at random and we define:

$$\widehat{c_{e'^{(1)}_{1,j,0}}} = \frac{\mathsf{Com}\left(\mathbf{U}''_{\mathbf{k}}, \alpha_{1,k}\right)}{\left(\prod_{l=1}^{k-1} c^{2^l}_{e'^{(1)}_{1,j,l}}\right)\left(\prod_{i=2}^{N} c_{e'^{(i)}_{1,j}}\right)\left(\prod_{j=1}^{N} c^{u_k^{(j)}}_{\mathbf{m}_j}\right)\left(\prod_{j=1}^{n} c^{a_{k,j}}_{\mathbf{r}'_j}\right)}$$

$$\widehat{c_{e'^{(1)}_{2,j,0}}} = \frac{\mathsf{Com}\left(\mathbf{V}''_{\mathbf{k}}, \alpha_{2,k}\right)}{\left(\prod_{l=1}^{k-1} c^{2^l}_{e'^{(1)}_{2,j,l}}\right)\left(\prod_{i=2}^{N} c_{e'^{(i)}_{2,j}}\right)\left(\prod_{j=1}^{N} c^{v_k^{(j)}}_{\mathbf{m}_j}\right)\left(\prod_{j=1}^{n} c^{b_{k,j}}_{\mathbf{r}'_j}\right)}$$

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

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

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