Shorter Lattice-based Zero-Knowledge Proofs for the Correctness of a Shuffle

Javier Herranz, Ramiro Martínez, Manuel Sánchez

Dept. Matemàtiques, Universitat Politècnica de Catalunya
Barcelona, Spain
javier.herranz@upc.edu,ramiro.martinez@upc.edu

Abstract. In an electronic voting procedure, mixing networks are used to ensure anonymity of the casted votes. Each node of the network re-encrypts the input list of ciphertexts and randomly permutes it in a process named shuffle, and must prove (in zero-knowledge) that the process was applied honestly. To maintain security of such a process in a post-quantum scenario, new proofs are based on different mathematical assumptions, such as lattice-based problems. Nonetheless, the best lattice-based protocols to ensure verifiable shuffling have linear communication complexity on N, the number of shuffled ciphertexts.
In this paper we propose the first sub-linear (on N) post-quantum zero-knowledge argument for the correctness of a shuffle, for which we have mainly used two ideas: arithmetic circuit satisfiability results from [6] and Beneš networks to model a permutation of N elements. The achieved communication complexity of our protocol with respect to N is $\mathcal{O}(\sqrt{N}\log^2(N))$, but we will also highlight its dependency on other important parameters of the underlying lattice ingredients.

Keywords: electronic voting, verifiable shuffle, lattice-based cryptography, zero-knowledge

1 Introduction

E-voting has already been used in real political elections in Norway, Estonia, Switzerland and Australia, among other countries. It could provide voters with the ability to cast votes from anywhere, aid voters with disabilities to cast their votes autonomously, reduce the logistic costs of an election, obtain accurate vote counts faster and in general improve the flexibility of democratic processes. However we can only take advantage of these benefits if the election system is publicly trusted, for which it has to satisfy strong security requirements.

Two key requirements of e-voting are privacy and verifiability. On the one hand each individual voting option has to remain secret, and only the final tally should be revealed. This is usually addressed encrypting votes with an election public key, whose associated secret key is only known by the electoral board and used for the tally. On the other hand, verifiability ensures the integrity of the

election. It should be guaranteed that the final result has not been manipulated and corresponds to the options chosen by eligible voters. Whenever a voter wants to remotely cast an encrypted vote she digitally signs it before sending it to the voting server, that verifies the electronic signature before adding it to a virtual ballot box that will be published in a so called *Bulletin Board*, enabling everyone to verify that tallied votes come, with a one to one correspondence, from eligible voters. Systems that allow anyone to verify the integrity of the election using only public information without requiring additional interaction are called universally verifiable.

At this point we have an apparent contradiction, as the link established by the signature between the voter and its encrypted vote seems to prevent the desired level of privacy. A solution called mixing networks (or *mix-nets*) was presented by Chaum in its seminal paper [16] and is currently adopted by all previously mentioned actual elections. A mix-net is composed of a set of mixing nodes (or *mix-nodes*) that consecutively permute and re-encrypt/decrypt the output of the previous mix-node. This operation is called a shuffle. As long as one of these nodes is honest and keeps its permutation secret it should be infeasible to link the identity of the voter that signed one of the input encrypted votes with its value decrypted from the output of the mix-net, thus achieving privacy again. Verifiability can be enforced asking the mix-nodes to publish a zero-knowledge proof of well behaviour, in this case proving that they know a permutation and the randomness used such that their respective output is just a permuted re-encryption of its input, without leaking any additional information.

Since the first universally verifiable mix-net was presented by Sako and Kilian in 1995 [32] many proposals have been published with different kind of improvements, that will be discussed in detail in the following subsection 1.1. However there is still one important issue that has to be addressed, all schemes that guarantee universal verifiability publishing proofs of a shuffle in a Bulletin Board need to ensure the long term security of the information that is being published. This is particularly important as many constructions base their privacy on hardness assumptions about problems such as the Discrete Logarithm problem, that is known to be efficiently solvable by a quantum computer using Shor's algorithm [33]. Even if powerful enough quantum computers are not available now, an adversary could keep this public information until he has the ability to break the security with a quantum computer in the near future. Voting data is specially sensitive information that should remain secret in the long term, while it might still have political and personal implications. Therefore post-quantum hardness assumptions that are believed to hold even against a quantum computer should be used, such as the ones employed by lattice-based, code-based, multivariate polynomials or hash-based cryptography.

The main goal of this article is to present the first post-quantum proof of a shuffle with sub-linear size in the number of inputs, that could be used to build secure mix-nets, guaranteeing long term privacy even in a quantum computing era.

1.1 State of the Art

The structure of a proof of a shuffle heavily depends on the choice of a way of representing a permutation. A great variety of approaches appear in the literature, from applying permutation matrices [22, 21, 26, 37, 35], permutation networks [2, 3], showing two sets are equal if they are both roots of the same polynomial [31, 24, 7, 25] or using general arithmetic circuits [13]. Most of the work, from the very beginning [1, 30], focuses on reducing the size of the proofs for different scenarios. A comprehensive study of mix-nets and proofs of shuffles can be found in [27].

However only a handful of post-quantum e-voting proposals have been recently published. Del Pino *et al.* presented EVOLVE in [19], which uses a somewhat homomorphic encryption scheme to add together several ballots before decrypting them. This alternative can only work with elections where the result can be represented as the addition of individual votes, but it can not implement *write-ins*, that are easily handled by mix-nets. The same limitation applies to the recent work [10], which proposes an elegant way of solving some security issues in [19]. Gjøsteen and Strand also propose the use of fully homomorphic encryption to construct a decryption circuit in [23], but while theoretically interesting it is still far from efficient. The recent work in [5] proposes a practical post-quantum e-voting protocol, but under a very strong trust (perhaps unrealistic) assumption: the shuffle entity has no access to the channels used by voters to cast their bots in the ballot box.

Regarding post-quantum mixnets, the universally verifiable mix-nets of Costa *et al.* and Strand [17, 34, 18] are both quite impractical, either because of the use of fully homomorphic encryption or because correctness proofs have linear (in N) size, with large constants. The only quantum-safe practical mix-net we are aware of is [11] by Boyen *et al.*, based on a different model that only allows verification by a (temporarily trusted) auditor, making it not universally verifiable.

The construction of an efficient post-quantum universally verifiable mix-net is still an open problem. In this paper we provide a significant step presenting the first such protocol, to shuffle N ciphertexts, with proofs of sub-linear size in N.

A key ingredient for our protocol is the zero-knowledge proof of satisfiability of an arithmetic circuit presented in [6] (and recently improved/generalized in [9]). These proofs achieve post-quantum security properties by using techniques from lattice-based cryptography, and the size of the proofs is sub-linear in the number of gates M of the arithmetic circuit, since it is $\mathcal{O}(\sqrt{M \log^3(M)})$ (in the protocol in [6]). When the soundness property of a zero-knowledge system is satisfied computationally (assuming the hardness of some underlying mathematical problem) then people often refer to such proofs as *arguments* of knowledge. The proofs in [6, 9] and consequently the proof for the correctness of a shuffle that we present in this paper are indeed arguments of knowledge, but we use both proofs and arguments to refer to them.

1.2 Arithmetic Circuits for Shuffles

The idea is to use the powerful result of [6], to prove in zero-knowledge that a shuffle (re-encryption and permutation) has been correctly performed. Let $L = \{C_1, \ldots, C_N\}$ be the input list of N ciphertexts for the shuffling node; he is assumed to re-encrypt each ciphertexts, which leads to $L' = \{C'_1, \ldots, C'_N\}$ and then to apply a permutation ρ to the list L', which leads to $L'' = \{D_1, \ldots, D_N\}$, where $D_i = C'_{\rho(i)}$, for each $i = 1, \ldots, N$. The list L'' is made public, so the two lists L and L'' are available to the verifier of the zero-knowledge proof of a correct shuffle.

In the case of RLWE-based ciphertexts, the re-encryption step $L \to L'$ can be easily expressed as an arithmetic circuit, where some secret input wires correspond to the (small) random elements used to re-encrypt each ciphertext. The number of gates of this first sub-circuit is $\mathcal{O}(N)$. The challenge is now to express the permutation step, that is the statement that list L'' is a permutation of list L', as an arithmetic circuit with a small enough number of gates. Our solution is to consider the Beneš network that corresponds to that permutation; the circuit that expresses such Beneš network takes as input the N ciphertexts in L' along with a bit $b \in \{0, 1\}$ for each internal 2-in 2-out gate of the Beneš network, indicating if the two input wires must be switched or not, in the output of that gate. The final output of the circuit must be the list of N ciphertexts in L''. The number of gates of such a circuit is $\mathcal{O}(N \log(N))$.

1.3 Our Results

We detail how RLWE ciphertexts must be produced and re-encrypted so that shuffling nodes must prove, in particular, that the noise introduced when re-encrypting each ciphertext is small enough (and thus, in the tally phase, there will not be errors in the decryption of the final ciphertexts). This fact, along with the correct execution of the re-encryption algorithm and the correct execution of a permutation expressed by a Beneš network, constitute the arithmetic circuit to which we apply the results in [6, 9]. Since the number of gates of the circuit is $M \in \mathcal{O}(N \log(N))$, the result is a zero-knowledge proof that a shuffle of N ciphertexts has been correctly applied, with post-quantum security based on the hardness of well-known lattice problems, and with size sub-linear on N.

1.4 Organization

In Section 2 we review some ingredients of our protocol: RLWE encryption, lattice-based zero-knowledge proofs of satisfiability of arithmetic circuits and Beneš networks. Then in Section 3 we propose our protocol, by first describing the arithmetic circuit that represents a shuffle of N RLWE ciphertexts, and then by applying to this circuit the construction in [6]. We analyze our protocol in Section 4, in terms of efficiency and security.

2 Preliminaries

2.1 Ideal Lattices: RLWE Problems and Public Key Encryption

Ideal lattices can be seen as ideals in the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$, where the polynomial $f(X) = X^n + f_n X^{n-1} + \cdots + f_2 X + f_1 \in \mathbb{Z}[X]$. Usually, for real cryptographic applications, we will set n a power of 2 and $f(X) = X^n + 1$ and we will consider the quotient ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ since this setting provides several advantages from an implementation point of view.

Let n and q be integers, $\mathcal{R} = \mathbb{Z}[X]/\langle f(X)\rangle$ with $\deg(f) = n$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let χ_σ be a discrete probability distribution over $\mathcal{R}$ (usually a Gaussian distribution) with parameter σ and a secret polynomial $s \in \mathcal{R}_q$.

Definition 1 (Ring Learning With Errors Distribution). *The RLWE distribution $\mathcal{L}_{s,\chi}$ over $\mathcal{R}_q \times \mathcal{R}_q$ is sampled by choosing $a \xleftarrow{\mathcal{R}} \mathcal{R}_q$, $e \xleftarrow{\mathcal{R}} \chi_\sigma$ and outputting $(a, b = a \cdot s + e \bmod q)$.*

Definition 2 (Search-RLWE Problem). *Given m independent samples $(a_i, b_i) \xleftarrow{\mathcal{R}} \mathcal{L}_{s,\chi}$ for a fixed uniformly random s, find s.*

Definition 3 (Decision-RLWE Problem). *Given m independent samples (a_i, b_i), decide whether this samples are distributed according to $\mathcal{L}_{s,\chi_\sigma}$ for a fixed uniformly random s; or according to a uniform distribution over $\mathcal{R}_q \times \mathcal{R}_q$.*

Hardness of RLWE comes for large enough choices of q. Solving certain instantiations of Search-RLWE is as hard as quantumly solving an approximate Shortest Vector Problem on an ideal lattice.

The problem remains to be hard when the secret s is chosen from the error distribution instead of uniformly at random (see [4] for the reduction).

RLWE encryption scheme. This scheme, first proposed by Lyubashevsky, Peikert and Regev in [29], works as follows:

Definition 4 (RLWE encryption scheme). *We consider the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ with n a power of 2 and q a prime. Messages are strings of n bits encoded as a polynomial in $\mathcal{R}_q$. An error distribution χ must be chosen, producing "small" elements of $\mathcal{R}_q$.*

- ***Gen***(1^λ)*: Compute suitable n and q according to λ. Choose $a \xleftarrow{\mathcal{R}} \mathcal{R}_q$ and small $s, e \xleftarrow{\mathcal{R}} \chi$. Output $sk = s$ and $pk = (a, b = a \cdot s + e) \in \mathcal{R}_q \times \mathcal{R}_q$.*
- ***Enc***(pk, z, r, e_1, e_2)*: To encrypt a message $z \in \{0,1\}^n$ we view it as an element in $\mathcal{R}_q$ by using its bits as the 0-1 coefficients of a polynomial. Then we choose small elements $r, e_1, e_2 \xleftarrow{\mathcal{R}} \chi$ and output $(u, v) = (r \cdot a + e_1, b \cdot r + e_2 + \lfloor \frac{q}{2} \rceil z) \in \mathcal{R}_q \times \mathcal{R}_q$*

– ***Dec***$(sk,(u,v))$*: Compute:*

$$\begin{aligned} v - u \cdot s &= b \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil z - (r \cdot a + e_1) \cdot s \\ &= (a \cdot s + e) \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil z - r \cdot a \cdot s - e_1 \cdot s \\ &= a \cdot s \cdot r + e \cdot r + e_2 + \left\lfloor \frac{q}{2} \right\rceil z - r \cdot a \cdot s - e_1 \cdot s \\ &= (e \cdot r - e_1 \cdot s + e_2) + \left\lfloor \frac{q}{2} \right\rceil z \\ &\approx \left\lfloor \frac{q}{2} \right\rceil z \end{aligned}$$

It has been proved that the RLWE encryption scheme is IND-CPA secure, assuming the hardness of RLWE problems [29]. The usual choice for the error distribution χ consists in running n independent instances (one for each component, if we see elements of $\mathcal{R}_q$ as vectors in $(\mathbb{Z}_q)^n$) of a discrete Gaussian distribution, centered at 0 and with parameter σ, in $\mathbb{Z}_q$.

In this paper we consider *truncated* Gaussian distributions: we fix a positive integer $\hat{k}$ and we check that the output of the Gaussian falls in the set $\{-\hat{k}\sigma, \ldots, -1, 0, 1, \ldots, \hat{k}\sigma\}$; if this is not the case, we reject this sample and do a new one. The statistical distance between the resulting truncated distribution over $\mathcal{R}_q$ and the non-truncated Gaussian distribution over $\mathcal{R}_q$ can be bounded by $n \cdot e^{-\hat{k}^2/2}$ (see for instance [28]). If we take $n, \hat{k}$ such that this value is negligible in the security parameter, then we can safely use truncated Gaussian distributions with the same parameters (q, n, σ) which are considered secure when discrete Gaussian are employed.

The choices of q and σ determine if the encryption scheme works properly: they must be chosen to ensure that all the coefficients of $e \cdot r - e_1 \cdot s + e_2$ can be upper-bounded by less than $\frac{q}{4}$, in this way the message z is recovered by rounding each coefficient of $v - u \cdot s$ to 0 or $\lfloor \frac{q}{2} \rceil$, whichever is closest modulo q. Also, this scheme allows to define a new algorithm **Re-Enc** to re-encrypt previously encrypted data. This algorithm works as follows:

– **Re-Enc**$(pk,(u,v),r',e_1',e_2')$: To re-encrypt a message z encrypted as (u,v) we choose small $r', e_1', e_2' \xleftarrow{\mathcal{R}} \chi_\sigma$ and output the pair

$$(u', v') = (u, v) + Enc(pk, 0, r', e_1', e_2') \in \mathcal{R}_q \times \mathcal{R}_q.$$

Notice that every time we re-encrypt a ciphertext the norm of its noise might grow, and therefore only a limited number of re-encryptions are allowed. Parameters have to be chosen so that the encryption scheme supports at least as many re-encryptions as the number of mix-nodes of the mix-net (which is known in advance). Given that this number is typically a fixed small quantity this requirement is usually already satisfied and has no real impact on the parameter selection.

2.2 Zero-Knowledge Arguments for the Satisfiability of Arithmetic Circuits

An arithmetic circuit over a field $\mathbb{Z}_q$ is a directed acyclic graph whose vertices are called gates and edges are called wires. Gates of in-degree 0 are called input gates and usually are associated to variables or constants. The remaining gates are either multiplication gates or addition gates.

The general idea of the protocol presented by Baum et al. [6] to prove the satisfiability of an arithmetic circuit over Z_q is summarized below:

1. the first idea is to arrange the $\mathcal{O}(M)$ wire values of the circuit into a (more or less square) matrix with $\mathcal{O}(\sqrt{M})$ rows and $\mathcal{O}(\sqrt{M})$ columns;
2. using an appropriate lattice-based homomorphic commitment scheme (with outputs being vectors of elements in $\mathbb{Z}_Q$ for some prime number $Q >> q$), one commit to each row of the above-mentioned matrix;
3. using techniques from [8], one reduces the satisfiability of the arithmetic circuit to the satisfiability of linear-algebraic statements over committed matrices;
4. the last step consists in using a new zero-knowledge proof, designed by themselves, to prove the satisfiability of such algebraic statements (products and additions of matrices) in an efficient way, with the proofs being as short as possible.

Authors of [6] show a possible way of choosing the parameters of the lattice, the commitment scheme and the dimensions of the matrix so that the global protocol to prove satisfiability of the arithmetic circuit has communication complexity $\mathcal{O}(\sqrt{M \log^3(M)} \log(Q))$. The protocol involves 9 rounds of interaction between the prover and the verifier. The security (including computational soundness) is based on the hardness of both the Short Integer Solution (SIS) and the Learning With Errors (LWE) problems. For the security proof to be valid, they need $Q \approx q^5$.

2.3 Beneš Networks

We will use as a model a permutation network called Beneš network proposed by Abraham Waksman in [36]. The use of Beneš networks is not new in cryptography, as early results from Masayuki Abe [2] already considered these constructions to apply them to mix-nets. Nevertheless, the asymptotic cost of these solutions were usually worse than others, and they were considered inefficient. In this paper we see that the recent advances in the area of zero-knowledge proofs/arguments for satisfiability of arithmetic circuits may give a new opportunity to this kind of constructions.

Formally, a permutation network is an acyclic graph with N inputs and N outputs where vertices have in-degree and out-degree equal to 2. These vertices are called *switch gates* and each of them has a special input $b \in \{0, 1\}$, which indicates if the two inputs are switched or if they remain in the same order (see figure 1).

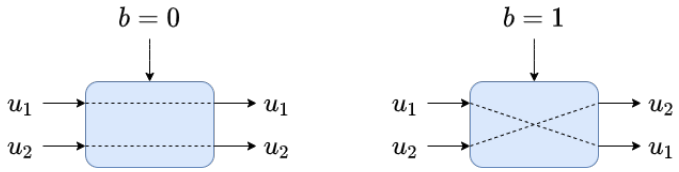

Fig. 1. Switch gate

Beneš networks are constructed recursively. A 2×2 Beneš network is just a switch gate, and it is trivial that a switch gate models every permutation of 2 elements, namely the identity if $b = 0$ and a switch if $b = 1$. Now we can construct a $2^k \times 2^k$ network using two $2^{k-1} \times 2^{k-1}$ Beneš sub-networks and 2^k switch gates, that will be able to perform whichever permutation of 2^k elements.

An easy induction yields that for $N = 2^k$, to craft a $N \times N$ network we will need $2\log_2(N) - 1$ stages of $N/2$ switch gates each, therefore an $\mathcal{O}(N\log(N))$ amount of switch gates. Beneš networks easily model any of the $N!$ permutations without deadlocks (i.e. for each wire only travels one value). Besides, Beneš networks can be extended to arbitrary sizes, and not just powers of 2 [15].

One could imagine that to perform the permutation the prover could just choose the switch bit of each gate uniformly at random from $\{0, 1\}$, and let the circuit apply the resulting permutation. This will not be correct, since in a shuffle every permutation of N elements must have the same probability to appear. The random choosing of the bits implies that some permutation will appear more often than others, so the choice is not uniform, as shown in [3]. Therefore, if we denote by $\mathcal{S}_N$ the set of permutations of N elements, the prover must first choose $\pi \xleftarrow{\mathcal{R}} \mathcal{S}_N$, and then run an algorithm to set the bits accordingly. These algorithms are called *routing algorithms* and have a best known complexity of $\mathcal{O}(N\log(N))$, such as [14], so it does not affect the asymptotic complexity of the prover.

3 The Proposed Protocol

3.1 The Circuit that Encodes a Shuffle

A shuffling node receives as input a list of N ciphertexts $L = \{C_1, \ldots, C_N\}$. He first re-encrypts each ciphertext $C_i = (u_i, v_i)$ using the protocol $C'_i = (u'_i, v'_i) \leftarrow \mathsf{Re\text{-}Enc}(pk, (u_i, v_i), r'_i, e'_{1,i}, e'_{2,i})$, which leads to an intermediate list $L' = \{C'_1, \ldots, C'_N\}$, and then he applies a random permutation $\rho \in \mathcal{S}_N$ to L', which leads to $L'' = \{D_1, \ldots, D_N\}$, where $D_i = C'_{\rho(i)}$, for each $i = 1, \ldots, N$. The list L'' is the output of the whole shuffling process (along with the correctness proof that we will describe in this section).

The shuffling node computes the Beneš network that represents the secret permutation ρ, that is, an assignment $b_1, b_2, \ldots, b_K$ of bits for each of the K switch gates of the network, where $b_\ell \in \{0, 1\}$, for all $\ell = 1, \ldots, K$.

Note that we separate the shuffling in two well-differentiated parts / circuits, one for the re-encryption and one for the permutation. An alternative solution (used for instance in [12]) would be to add a re-randomization step in each switch gate of the Beneš network. However, the size of the global circuit for the shuffle is smaller with our solution: it contains less re-randomization / re-encryption gates, $\mathcal{O}(N)$ than the $\mathcal{O}(N \log N)$ re-randomization gates that would be required in this alternative solution.

All the elements involved in the encryption scheme are polynomials in the ring $\mathcal{R}_q$ of degree at most $n-1$, so we see each element as a tuple of n elements in the field $\mathbb{Z}_q$ underlying the arithmetic circuit that we will consider. For instance, $u_i \leftrightarrow (u_{i,0}, \ldots, u_{i,n-1})$.

The public inputs of the arithmetic circuit are (the $\mathbb{Z}_q$ components of) the N ciphertexts in $L = \{C_1, \ldots, C_N\}$ and the N ciphertexts in $L'' = \{D_1, \ldots, D_N\}$. So, counting elements in the field $\mathbb{Z}_q$ of the circuit, we have $4nN$ public inputs.

The secret inputs of the arithmetic circuit are:

(i) (The $\mathbb{Z}_q$ components of) the N triples of noise $(r'_i, e'_{1,i}, e'_{2,i})$ used to re-encrypt each ciphertext C_i, which means $3nN$ elements in $\mathbb{Z}_q$,
(ii) The K bits $b_1, \ldots, b_K$ for the K switching gates of the Beneš network; we recall that $K \in \mathcal{O}(N \log(N))$.

What does our circuit C_{shuffle} do? Essentially, three different things:

1. Check that the components of the noise are small; for instance, if $r'_i \leftrightarrow (r'_{i,0}, \ldots, r'_{i,n-1})$, then the circuit needs to check that $r'_{i,j} \in \{-\hat{k}\sigma, \ldots, -1, 0, 1, \ldots, \hat{k}\sigma\} \subset \mathbb{Z}_q$, for all $j = 0, \ldots, n-1$. This checking is encoded as the arithmetic circuit equality $(r'_{i,j} + \hat{k}\sigma) \cdot \ldots \cdot (r'_{i,j} + 1) \cdot r'_{i,j} \cdot (r'_{i,j} - 1) \cdot \ldots \cdot (r'_{i,j} - \hat{k}\sigma) = 0$, in $\mathbb{Z}_q$. Since this has to be done for each $i = 1, \ldots, N$, each $j = 0, \ldots, n-1$ and each element in the noise triple, we have a circuit with $6\hat{k}\sigma nN$ gates. If some checking fails, the arithmetic circuit is not satisfied.
2. Check that each secret input b_ℓ is a bit, that is, check that $b_\ell \cdot (1 - b_\ell) = 0$ in $\mathbb{Z}_q$, for all $\ell = 1, \ldots, K$. These circuits consist of $2K \in \mathcal{O}(N \log(N))$ gates. If some checking fails, the arithmetic circuit is not satisfied.
3. Check that the Beneš network, when applied to the result of re-encrypting the ciphertexts in L, produces the ciphertexts in L''. This consists in two phases, one for re-encryption and one for permutation:
   (a) re-encryption circuit: each re-encryption $C'_i = (u'_i, v'_i) \leftarrow \textbf{Re-Enc}(pk, (u_i, v_i), r'_i, e'_{1,i}, e'_{2,i})$ essentially consists in doing two polynomial multiplications and two polynomial additions, in the ring $\mathcal{R}_q$. If we implement these operations with the classical method, this means $\mathcal{O}(n^2)$ gates for each re-encryption, as an arithmetic circuit over $\mathbb{Z}_q$. The complexity of this circuit can be slightly improved when multiplying polynomials if we use Karatsuba's algorithm. Notice that this would imply that the output is the product of the polynomials over $\mathbb{Z}_q[X]$, which has (at most) $2n-1$ monomials. We can reduce this resulting polynomial into $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1 \rangle$ again with $n-1$ subtractions, since $X^n = -1$.

In this case, if the polynomials have degree a power of 2, the number of integer multiplications becomes $\mathcal{O}(n^{\log_2 3})$.

(b) permutation circuit: the same permutation must be applied to all the n components of each of the two elements (u'_i, v'_i) of a re-encrypted ciphertext C'_i. Inside the Beneš network, each switch gate, with bit input $b \in \{0,1\}$, couple of inputs $(u_1, u_2) \in (\mathbb{Z}_q)^2$ and couple of outputs $(v_1, v_2) \in (\mathbb{Z}_q)^2$, just consists in applying the following operation

$$v_1 = (1-b) \cdot u_1 + b \cdot u_2 \bmod q$$
$$v_2 = (1-b) \cdot u_2 + b \cdot u_1 \bmod q$$

Therefore, the number of gates of the arithmetic circuit for the global Beneš network is $12nK \in \mathcal{O}(nN\log(N))$.

The outputs of these two phases are then compared with the list L''. If all the elements are equal, then the arithmetic circuit is satisfied.

The number M of gates of the complete arithmetic circuit C_{shuffle} is thus

$$M \ \in \ \mathcal{O}\ \left(\ N \cdot \left(n\hat{k}\sigma + n^{\log_2 3} + n\log(N)\right)\ \right)$$

3.2 Non-Interactive Proof of Circuit Satisfiability, with Fiat-Shamir

The proof of satisfiability of an arithmetic circuit in [6] is an interactive protocol between the prover (the shuffling node, in our case) and a verifier. The protocol consists in 9 rounds of communication. The amount of information exchanged between the prover and the verifier is $\mathcal{O}(\sqrt{M\log(M)})$ elements of $\mathbb{Z}_Q$, where $Q \approx q^5$, for an arithmetic circuit with M gates operating in the field $\mathbb{Z}_q$.

In our setting of producing a publicly verifiable proof of correctness of a shuffle, interaction is not permitted: the shuffling node must produce a proof π without interacting with the (possibly unknown, yet) verifier.

The standard way of transforming such an interactive protocol into a non-interactive one is to use the Fiat-Shamir paradigm: the challenges sent from the verifier to the prover are replaced with values that are computed by the own prover, applying a secure hash function to the statement of the proof and the values exchanged in the previous rounds of communication. The length of the resulting proof π is thus equivalent to the amount of information exchanged in the interactive version; in our case, π contains $\mathcal{O}(\sqrt{M\log(M)})$ elements of $\mathbb{Z}_Q$.

We emphasize here that relaying in the Fiat-Shamir transformation makes the protocol secure in the ROM, but not in the QROM (that allows oracle queries to be in quantum superposition). This is a common choice in the literature as, at the moment, there has been no natural scheme proven secure in the ROM based on a quantum-safe problem that has later been proven insecure in the QROM. Very recent results have shown how, provided some additional constrains are satisfied, some generic schemes proven secure in the ROM are also secure in the QROM (see [20] and references therein).

3.3 The Resulting Protocol

All in all, our proposed protocol to prove the correctness of a shuffle works as follows. The public input of the shuffling node is a list $L = \{(u_i, v_i)\}$ of N ciphertexts. The node then

1. Chooses noise elements $(r'_i, e'_{1,i}, e'_{2,i})$ using the truncated Gaussian distribution over $\mathcal{R}_q$, and computes re-encryptions $C'_i = (u'_i, v'_i) \leftarrow \mathsf{Re\text{-}Enc}(pk, (u_i, v_i), r'_i, e'_{1,i}, e'_{2,i})$, for $i = 1, \ldots, N$,
2. Chooses at random a permutation ρ for the set $\{1, 2, \ldots, N\}$ and defines $D_i = C'_{\rho(i)}$, for each $i = 1, \ldots, N$,
3. Finds the bit assignment, $\{b_\ell\}_{1 \leq \ell \leq K}$, for the switch gates of the Beneš network that correspond to permutation ρ,
4. Uses the non-interactive (Fiat-Shamir) version of the satisfiability proof of an arithmetic circuit to compute a proof π for circuit C_{shuffle}.

The shuffling node publishes the list $L'' = \{D_1, \ldots, D_N\}$ of shuffled ciphertexts along with the proof π. Any verifier can take the lists L and L'' and verify the correctness of the non-interactive zero-knowledge proof π.

4 Analysis: Efficiency, Security and Possible Improvements

4.1 Complexity Analysis and Possible Choices of Parameters

The goal of this paper was to design a protocol to prove the correctness of a shuffle of N ciphertexts, with post-quantum security and communication complexity lower than $\mathcal{O}(N)$. The protocol proposed in the previous section achieves this goal.

Regarding communication complexity, the size of the resulting proofs is $\mathcal{O}(\sqrt{M \log^3(M)} \log(Q))$, where Q is a prime number for the commitments needed in the construction of [6], roughly $Q \approx q^5$ and M is the number of gates of the shuffle circuit:

$$M \in \mathcal{O}\left(N \cdot \left(n\hat{k}\sigma + n^{\log_2 3} + n\log(N) \right) \right)$$

Therefore, the dependency of the size of a proof on the number N of shuffled ciphertexts is sub-linear, in the order $\mathcal{O}(\log^2(N)\sqrt{N})$. On the other hand, there are other parameters to be taken into account: the dimension n of the underlying ideal lattice and also the values $\hat{k}$ and σ related to the truncated Gaussian distribution used to produce the (re-encrypted) ciphertexts have an impact on the size of a proof π.

Some specific values for these parameters, for a security level of 128 bits, could be: $n = 128 = 2^7$, $q = 2^{20}$ (and thus $Q \approx 2^{100}$), $\hat{k} = 12$, $\sigma = 6$.

The length of the proofs in the new protocol starts improving over previous proposals (with linear dependency on N) once N gets big numbers, e.g. one million ciphertexts. For smaller numbers of ciphertexts, i.e. in small elections, the result of our new protocol does not significantly improve over existing solutions.

Possible Improvements. The results in [6] for the zero-knowledge argument of satisfiability of an arithmetic circuit have been improved and generalized in [9]. Authors of this last work give a protocol, for each positive integer $d \geq 1$, where the communication complexity is essentially $\mathcal{O}\left(\log Q \cdot M^{\frac{1}{d+1}} \cdot (d^3 \lambda \log^2(M) d\lambda^2)\right)$. The result in [6] can be seen as the particular case $d = 1$ of the result in [9].

The improvement on the communication complexity with respect to M comes at the cost of a worse (bigger) value needed for the big prime number Q. Roughly speaking, $Q = \mathcal{O}(q^{4+d})$ is required.

Our protocol for the proof of a shuffle can be instantiated by using this zero-knowledge argument, to prove the satisfiability of the shuffle circuit described in Section 3.1. The result are shorter proofs: each time we increase d by one unit, we increase the size of the proof by a factor of $\log(q)$, but we decrease the size of the proof by a more significant factor: from $d = 1$ to $d = 2$, by a factor $\mathcal{O}(M^{1/6})$, from $d = 1$ to $d = 3$ by a factor $\mathcal{O}(M^{1/4})$, etc.

The idea is thus to take the number of votes in the election (i.e. the number N of ciphertexts to be shuffled) and then find a secure configuration for the lattice parameters $(q, n, \sigma, \hat{k})$ and a suitable value for d in order to minimize the size of the produced proofs of a correct shuffle.

4.2 Security Analysis

The security of our proof of correctness of a shuffle follows from the security properties of the zero-knowledge proof systems for arithmetic circuits in [6, 9]. On the one hand, the zero-knowledge property of the proof systems implies that an execution of the proof of a shuffle protocol does not leak any information about the secret witness, that is about the randomness employed to re-encrypt each input ciphertext and about the permutation that has been applied (through the corresponding Beneš network) to the list of re-encrypted ciphertexts. Therefore, the proof of a shuffle enjoys privacy.

On the other hand, the (computational) soundness property of the proof systems implies that each accepted execution of the proof of a shuffle protocol must have been produced by a prover (a shuffling node) who knows a valid witness for the considered circuit. In the circuit that we have considered in our protocol, this ensures that:

(i) The noise used for re-encryption is small enough, satisfying the same bound as the truncated Gaussian distribution, because each coefficient of the noise polynomials belongs to the set $\{-\hat{k}\sigma, \ldots, -1, 0, 1, \ldots, \hat{k}\sigma\}$;
(ii) The bits for the Beneš network are actually bits, $b_i \in \{0, 1\}$, and so a permutation is applied to the list of re-encrypted ciphertexts.

Therefore, if a dishonest shuffling node wants to fool the election by including false ciphertexts (not coming from a real shuffling of the input ciphertexts) or by adding too much noise to the ciphertexts (in such a way that the final decryption of the encrypted votes, in the tally phase, leads to decryption errors), the proof of

correctness of his shuffling will not be accepted, and so this dishonest behaviour will be detected.

Of course, a dishonest shuffling node can use a non-random permutation (for instance, the identity) and can impose all the re-encryption noise to be 0, in such a way that his input list of ciphertexts is exactly equal to the output list of ciphertexts. Such a behaviour can never be avoided, but once again, we insist that the general assumption is that at least one of the shuffling nodes is honest and performs a correct shuffling (re-encryption and permutation), which is enough to provide anonymity to the election.

Putting together the assumptions for the security of the encryption scheme and the assumptions for the security of the proof systems in [6, 9], we conclude that our proposed protocol to prove the correctness of a shuffle is secure under the assumption that the SIS and the RLWE problems are hard.

Acknowledgements

The work is partially supported by the Spanish *Ministerio de Ciencia e Innovación (MICINN)*, under Project PID2019-109379RB-I00 and by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701).

References

1. Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 437–447, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.
2. Masayuki Abe. Mix-networks on permutation networks. In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273, Singapore, November 14–18, 1999. Springer, Heidelberg, Germany.
3. Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 317–324, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany.
4. Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.
5. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Lattice-based proof of shuffle and applications to electronic voting. To appear in the proceedings of CT-RSA'2021. Cryptology ePrint Archive, Report 2021/338, 2021. `https://eprint.iacr.org/2021/338`.
6. Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

7. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.
8. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
9. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. A non-PCP approach to succinct quantum-safe zero-knowledge. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 441–469, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.
10. Xavier Boyen, Thomas Haines, and Johannes Mueller. Epoque: Practical end-to-end verifiable post-quantum-secure e-voting. To appear in the proceedings of IEEE EuroS&P 2021 Cryptology ePrint Archive, Report 2021/304, 2021. `https://eprint.iacr.org/2021/304`.
11. Xavier Boyen, Thomas Haines, and Johannes Müller. A verifiable and practical lattice-based decryption mix net with external auditing. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 336–356, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany.
12. Elette Boyle, Saleet Klein, Alon Rosen, and Gil Segev. Securing abe's mix-net against malicious verifiers via witness indistinguishability. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 274–291, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.
13. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.
14. A. Chakrabarty, M. Collier, and S. Mukhopadhyay. Matrix-based nonblocking routing algorithm for bene**v**s networks. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pages 551–556, 2009.
15. Chihming Chang and Rami Melhem. Arbitrary size benes networks. *Parallel Processing Letters*, 07, 05 1997.
16. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.
17. Nuria Costa, Ramiro Martínez, and Paz Morillo. Proof of a shuffle for lattice-based cryptography. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevi**v**cius, editors, *Secure IT Systems*, pages 280–296, Cham, 2017. Springer International Publishing.
18. Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-based proof of a shuffle. In Andrea Bracciali et al., editor, *FC 2019 Workshops*, volume 11599 of *LNCS*, pages 330–346, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Heidelberg, Germany.
19. Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1565–1581, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

20. Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.
21. Jun Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 88(1):172–188, 2005.
22. Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.
23. Kristian Gjøsteen and Martin Strand. A roadmap to fully homomorphic elections: Stronger security, better verifiability. In Michael Brenner and et al., editors, *Financial Cryptography and Data Security*, pages 404–418, Cham, 2017. Springer International Publishing.
24. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 145–160, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.
25. Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 379–396, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.
26. Jens Groth and Steve Lu. Verifiable shuffle of large size ciphertexts. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 377–392, Beijing, China, April 16–20, 2007. Springer, Heidelberg, Germany.
27. T. Haines and J. Müller. Sok: Techniques for verifiable mix nets. In *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, pages 49–64, 2020.
28. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 738–755, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
29. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6), November 2013.
30. Jakobsson Markus and Juels Ari. Millimix: Mixing in small batches. Technical report, 1999.
31. C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.
32. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 393–403, Saint-Malo, France, May 21–25, 1995. Springer, Heidelberg, Germany.
33. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
34. Martin Strand. A verifiable shuffle for the GSW cryptosystem. In Aviv Zohar et al., editor, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 165–180, Nieuwpoort, Curaçao, March 2, 2019. Springer, Heidelberg, Germany.
35. Björn Terelius and Douglas Wikström. Proofs of restricted shuffles. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology – AFRICACRYPT 2010*, pages 100–113, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
36. Abraham Waksman. A permutation network. *J. ACM*, 15(1):159–163, January 1968.

37. Douglas Wikström. A commitment-consistent proof of a shuffle. In Colin Boyd and Juan González Nieto, editors, *Information Security and Privacy*, pages 407–421, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.