# Post-quantum Online Voting Scheme

Guillaume Kaim, Sébastien Canard, Adeline Roux-Langlois, Jacques Traoré

# Post-Quantum Online Voting Scheme

Guillaume Kaim[1,2], Sébastien Canard[1],
Adeline Roux-Langlois[2], and Jacques Traoré[1]

[1] Orange Labs, Applied Crypto Group, Caen, France
[2] Univ Rennes, CNRS, IRISA, France

**Abstract.** We propose a new post-quantum online voting scheme whose security relies on lattice assumptions. Compared to the state-of-the-art, our work does not make use of homomorphic primitives nor mix-nets, that are more traditional ways to build electronic voting protocols. The main reason is that zero-knowledge proofs, mandatory in the two aforementioned frameworks, are far to be as efficient as in "classical"cryptography, leading us to explore other approaches.
We rather base our work on a framework introduced by Fujioka et al. at Auscrypt 1992 that makes use of a blind signature scheme as the main building block. We depart however from this seminal work by allowing threshold issuance of blind signatures (to prevent ballot stuffing by malicious authorities) and by using a threshold post-quantum public key encryption scheme (rather than a commitment scheme) to allow voters to "vote and go"and to prevent "partial results". We instantiate all the required primitives with lattice-based constructions leading to the first online voting scheme that simultaneously provides post-quantum public verifiability and everlasting privacy (information-theoretic ballot anonymity). Another advantage of our protocol is that it can, contrary to recent proposals, efficiently handle elections with multiple candidates or with complex ballots (and not only referendums or single member plurality voting) without weakening the whole voting protocol by increasing the parameters size as with previous post-quantum voting schemes.

## 1 Introduction

The notion of online voting is appealing since the emergence of remote communications. However until now, there is no online voting protocol that fulfills all the properties (security, efficiency...) required for such a sensitive topic. Still there exists some interesting constructions that have been used in real-world elections such as Votopia [KKLA01] or Helios [Adi08] which was trialed during student elections, for example in Princeton and the Catholic University of Louvain. The International Association of Cryptographic Researcher (IACR) also adopted Helios to elect its Board.

In this work we investigate the construction of a post-quantum online voting system built from a framework introduced by Fujioka et al. in Auscrypt 1992, which mainly relies on the well-known cryptographic primitive called a blind signature scheme. This framework contrasts from the current trend that makes

use of homomorphic encryption, or mix-nets system to improve the efficiency of the tallying phase in addition to offer strong verifiability and privacy properties thanks to zero-knowledge proofs. However in post-quantum settings, a lack of efficiency of some of the primitives used in the two frameworks cited above lead us to investigate on new options for a practical online voting protocol. Indeed since the groundbreaking work of Shor [Sho97], we know that the arrival of quantum computers will harm most of the current cryptosystem used in practice, this is why it is important to replace them with quantum resistant constructions, among them lattice-based cryptography seems to be the most promising.

**Post-Quantum Constructions.** To the best of our knowledge, there exists only 2 post-quantum constructions, both built from lattice-based primitives. The first scheme is based on fully-homomorphic encryption by Chillotti et al [CGGI16]. The second one uses zero-knowledge proofs on top of homomorphic commitments by del Pino et al. [dPLNS17].

Concerning the scheme of Chillotti et al. [CGGI16], the key idea is that they get rid of the zero-knowledge proofs that are inefficient in lattice-based settings. Indeed, their work is inspired by the online voting protocol of Helios [Adi08] which however requires zero-knowledge proofs to allow the voters to prove that their ballots are correctly formed, but also to permit the tally authority to prove that the result of the election is correct. In a nutshell, [CGGI16] uses a fully-homomorphic encryption scheme to replace the zero-knowledge proofs on the voter's side, while they use publicly verifiable ciphertext trapdoors to overcome the absence of zero-knowledge proofs on the authority side. However, using fully-homomorphic encryption makes the resulting voting scheme quite inefficient as pointed by del Pino et al. [dPLNS17]. This problem of efficiency may explains why implementations for the [CGGI16] scheme are lacking.

Concerning the construction of del Pino et al. [dPLNS17], the most important difference is that they make use zero-knowledge proofs contrary to [CGGI16]. In fact, the study of lattice-based zero-knowledge proofs has been intensive in the past five years with several advances in particular regarding their efficiency. This allows them to rely on a construction that makes a trade-off between efficiency and security. In short, their construction focuses on the Fiat-Shamir framework of [Lyu12], in order to prove the knowledge of the multiple of a short element instead of the element itself. In addition to the zero-knowledge primitive, they use a commitment scheme that benefits of an additive homomorphic property, which is very appealing in the online voting context. Finally, as said above, they provided an implementation of their voting scheme, that permits to analyze the efficiency of their construction in a real-world scenario. Indeed, generating and casting ballot is about 8.5 ms, and the time needed on the authority side as well as for the verification step takes about 0.15 sec. However, their implementation only considers two candidates, while if we want to add more candidates, the globals parameters get bigger. Indeed, for $2^k$ possible candidates, the number of proofs needed is multiplied by a factor $k$, then the size of the vote increases from a logarithmic factor in the number of candidates.

**Framework of Fujioka et al. and Adaptations.** We base our construction on the framework of Fujioka et al. [FOO92] (FOO). In such framework, the anonymity is granted by a cryptographic primitive called blind signature, while everyone can verify that the outcome of the election is correct since all the elements that are necessary to the tally are made public at the end of the election. Blind signatures allow a user to obtain a signature on a message by interacting with a signing authority. At the end of the protocol, the authority has never seen the message and is not able to link a signature to the interactions that led to this signature. Therefore, the main building blocks of this framework are a blind signature scheme and a commitment scheme. The first one allows to preserve the anonymity of each voter, a requirement that is mandatory for any election, while it forbids voters from voting twice. The commitment scheme prevents any partial result to leak before the end of the election.

Concretely to generate his vote, any voter begins by computing a commitment of its voting choice, in order to conceal it from other voters until the end of the election. Then, he authenticates to the voting authority in order to obtain a blind signature on the commitment of his vote. Both the commitment and the (blind) signature constitute the voter's ballot which is then sent to the Bulletin Board (BB) via here again a perfectly anonymous channel. The later only stores signed ballots and discard invalid ones (i.e. either ballot that are not signed or ballot with an invalid signature). At the end of the election, all the voters have to open their commitment (they have to reveal their voting option and the random value used to generate the commitment), and to send both values (voting choice and random value) to the BB. However thanks to the blidness property and the use of a perfectly anonymous channel, the anonymity is preserved since no one will be able to link a signed ballot to the voter who requested the corresponding signature (and therefore no one will be able to link a voter to his vote). Finally anyone can tally the result of the election, by counting the votes and verifying the validity of the blind signatures associated to the opened commitments.

The FOO voting scheme suffers from several major drawbacks. The main one is that all voters have to participate to the ballot counting process, having to open their commitment at the end of the election: their scheme is not "vote and go" and would be unsuitable for real-life elections. Worse, the blind signature private key is held by a single authority who could easily stuff the BB by generating as many blind signatures (meaning valid but illegitimate ballots) as he wishes.

**Our Contribution.** Our main contribution is a new post-quantum online voting scheme whose security relies on lattice assumptions. Compared to the state-of-the-art, our work does not make use of homomorphic primitives nor zero-knowledge proofs of knowledge, that are more traditional ways to build electronic voting protocols. One interest of our construction is also that its efficiency does not depend on the number of candidates considered. Our construction can in particular handle complex ballots and could be used for example for preference voting or for elections with multiple candidates or voting options (for instance to select the most valuable players of a tournament as Votopia [KKLA01] did).

Our scheme uses of the FOO framework, with two main modifications.

- First, we use an encryption scheme instead of a commitment scheme so that the voting choices are now encrypted. At the end of the election, the decryption key will be made public so that anyone can decrypt the ballots and compute the election's result. With such modification, voters won't have to come back at the end of the election to open their commitment. Moreover, thanks to the indistinguishability property of the encryption scheme, the votes will remain hidden until the end of the election.
- Second, we transform the encryption and the blind signature schemes into threshold variants which allows to share the secret key between several authorities. Indeed the private key of the blind signature scheme is given to a single authority in [FOO92], who could generate as many ballots as he wants and stuff the Bulletin Board with them. The same problem would arise for the encryption scheme if we give the private decryption key to a single authority. It means that if the authority owning this private decryption key is corrupted, then he can get partial results by decrypting the ciphertexts before the end of the election, which is not desirable for most elections.

We then instantiate the needed building blocks in the lattice setting. We first chose to use the ring version of Dual-Regev [GPV08,LPR13] as our encryption scheme. The threshold transformation we considered turns it into a slight version of a threshold encryption scheme. Indeed, we just need to avoid that the secret key is given to a single authority, then only the key generation mechanism is impacted. The idea is that at the end of the election, at least a threshold of $T$ authorities publish their shares, so that anyone can reconstruct the whole private key and decrypt the ciphertexts of the valid ballots included in the bulletin board.

We then use the lattice-based blind signature scheme given in [BCE+20] that we also adapt as a threshold variant. This second transformation is heavier, since the whole blind signature protocol is impacted. We start by using the result of Bendlin et al. [BKP13] that exhibits a generic transformation of a trapdoor based signature scheme [MP12] into a threshold variant[3]. Since the security of this transformation is proven using the universally composable (UC) model [Can01], then by composability our threshold variant remains secure. The two operations on the signer's part (the commitment and the signing step) are finally done in a threshold way by communicating with, at least, $t$ signing authorities. We would like to emphasize that a recent paper [HKLN20] pointed out several issues in the one-more unforgeability proofs of previous lattice-based blind signature schemes. It leads to the fact that one-more unforgeability of [BCE+20] is only conjectured, while it proposes a better efficiency than the construction of [HKLN20].

[3] The basic threshold transformation of [BKP13] makes use of a trusted setup. A variant without such trusted setup is also given but needs the use of non-mature multilinear maps [GGH13]. In practice, it is not suitable to have an authority owning the complete secret keys, as implied by the trusted setup. However we would like to emphasize that the parameters of a voting scheme can be set a long time before election day and then we decided to focus on the operations performed by voters and the authority on the election day.

## 2 Preliminaries

*Notation.* The vectors are written in bold lower-case letters, and matrices in bold upper-case letters. The euclidean norm of a vector is denoted by $\|\mathbf{b}\|$, and the norm of a matrix $\|\mathbf{T}\| = \max_i \|\mathbf{t}_i\|$, where the $\mathbf{t}_i$'s are its column vectors. We denote by $D$ a distribution over some countable support $S$ and $x \hookleftarrow D$ the choice of $x$ following the distribution $D$.

### 2.1 Lattices

We define a $m$-dimensional full rank lattice $\Lambda$ as a discrete additive subgroup of $\mathbb{R}^m$. A lattice is the set of all integer combinations of some linearly independent basis vector $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\} \in \mathbb{R}^{n \times m}$: $\Lambda(\mathbf{B}) = \{\sum_{i=1}^{m} z_i \mathbf{b}_i, z_i \in \mathbb{Z}\}$.
We consider $n$ a power of two, such that the polynomial ring $\mathrm{R} = \mathbb{Z}[x]/(x^n + 1)$ is isomorphic to the integer lattice $\mathbb{Z}^n$. Then a polynomial $f = \sum_{i=0}^{n-1} f_i x^i$ in R corresponds to the integer vector of its coefficients $(f_0, \ldots, f_{n-1})$ in $\mathbb{Z}^n$. The notation norm of a polynomial $\|f\|$ means that we consider the norm of its coefficient vector, and as for the integer, the norm of a vector of polynomial $\|\mathbf{f}\| = \max_i \|f_i\|$. For the rest of the paper we will work with polynomials over R, or $\mathrm{R}_q = \mathrm{R}/q\mathrm{R} = \mathbb{Z}_q[x]/(x^n + 1)$, where $q$ is a prime verifying $q = 1 \pmod{2n}$.

*Computational Problems.* We consider Ring-SIS, a variant of the Short Integer Solution problem (SIS), proven to be at least as hard as the Shortest Independent Vectors Problem (SIVP) problem on ideal lattices [LM06,PR06].

**Definition 1 (Ring-SIS$_{q,m,\beta}$).** *Given $\mathbf{a} = (a_1, \ldots, a_m)^T \in \mathrm{R}_q^m$ a vector of $m$ uniformly random polynomials, find a non-zero vector of small polynomials $\mathbf{x} = (x_1, \ldots, x_m)^T \in \mathrm{R}^m$ such that $f_{\mathbf{a}}(\mathbf{x}) = \sum_{i=1}^{m} a_i \cdot x_i = 0 \bmod q$ and $0 < \|\mathbf{x}\| \leqslant \beta$.*

We also define Ring-LWE that is similar to the Learning With Errors problem (LWE) [Reg05] but on a polynomial ring:

**Definition 2 (Ring-LWE$_{q,D_{\mathbb{R},\alpha q},m}$).** *Given a uniformly chosen vector $\mathbf{a} \in \mathrm{R}_q^m$ and a polynomial $\mathbf{b} = \mathbf{a} \cdot s + \mathbf{e} \mod q$, with $s \leftarrow_{\$} \mathrm{R}_q$ and $\mathbf{e} \leftarrow D_{\mathrm{R}^m,\alpha q}$, the search Ring-*LWE *problem asks to find $s$. The decisional version asks to distinguish if a pair $(\mathbf{a}, \mathbf{b}) \in \mathrm{R}_q^m \times \mathrm{R}_q^m$ has been generated from the uniform distribution on $\mathrm{R}_q^m \times \mathrm{R}_q^m$ or if it has been generated as a Ring-LWE sample $(\mathbf{a}, \mathbf{b} = \mathbf{a} \cdot s + \mathbf{e})$.*

*Gaussian distribution.* The Gaussian function of center $\mathbf{c} \in \mathbb{R}^n$ and width parameter $\sigma$ is defined as $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{\sigma^2})$, for all $\mathbf{x} \in \mathbb{R}^n$. A positive definite covariance matrix is defined as $\mathbf{\Sigma} = \mathbf{B}\mathbf{B}^T : \rho_{\sqrt{\mathbf{\Sigma}},\mathbf{c}} = \exp(-\pi(\mathbf{x}-\mathbf{c})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c}))$. The discrete Gaussian distribution over a lattice $\Lambda$ is defined as $D_{\Lambda,\sigma,\mathbf{c}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{x})}{\rho_{\sigma,\mathbf{c}}(\mathbf{\Lambda})}$ where $\rho_{\sigma,\mathbf{c}}(\mathbf{\Lambda}) = \sum_{x \in \Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$. The vectors sampled from $D_{\Lambda,\sigma}$ are short with overwhelming probability.

**Lemma 1 ([Ban93, lemma 1.5]).** *For any lattice $\Lambda \subseteq \mathbb{R}^n, \sigma > 0$ and $\mathbf{c} \in \mathbb{R}^n$, we have $Pr_{\mathbf{x} \hookleftarrow D_{\Lambda,\sigma,\mathbf{c}}}[\|\mathbf{x} - \mathbf{c}\| \leqslant \sqrt{n}\sigma] \geqslant 1 - 2^{-\Omega(n)}$.*

*Trapdoors.* As introduced in [Ajt96] and widespread in [GPV08], a trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is a short basis of the lattice $\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{v} \in \mathbb{Z}^m \text{ such that } \mathbf{Av} = 0 \bmod q\}$. A trapdoor allows to sample short Gaussian vectors solutions to the Inhomogeneous Small Integer Solution (ISIS) problem: $\mathbf{Av} = \mathbf{x} \bmod q$ with $\mathbf{x} \in \mathbb{Z}_q^n$. This technique is called *Preimage Sampling*. In this work we make use of the trapdoor construction of [MP12]. *Construction.* In [MP12], the construction of the gadget-based trapdoor uses a gadget vector $\mathbf{g} = (1, 2, 4, \ldots, 2^{k-1})^T \in \mathrm{R}_q^k$, with $k = \lceil \log_2 q \rceil$, takes as input the modulus $q$, the Gaussian parameter $\tau$, and an optional $\mathbf{a}' \in \mathrm{R}_q^{m-k}$ and $h \in \mathrm{R}_q$. If no $\mathbf{a}'$ is given it is chosen uniformly in $\mathrm{R}_q^{m-k}$ and if no $h$ is given, $h = 1$. The construction outputs a matrix $\mathbf{a} = (\mathbf{a}'^T \| h\mathbf{g} - \mathbf{a}'^T\mathbf{T})^T$ with $\mathbf{T} \in \mathrm{R}^{(m-k)\times k}$ its trapdoor associated to the tag $h$, generated as a Gaussian of parameter $\tau$.

*Preimage sampling.* The construction given in [MP12] enables the use of the PreSample algorithm. Given $\mathbf{a} \in \mathrm{R}_q^m$, such algorithm computes a short vector solution $\mathbf{v} \in \mathrm{R}^m$ of a Ring-SIS problem $f_{\mathbf{a}}(\mathbf{v}) = \sum_{i=1}^{m} a_i \cdot v_i = 0 \bmod q$, available only thanks to a trapdoor $\mathbf{T} \in \mathrm{R}_q^{(m-k)\times k}$ for $\mathbf{a}$.

*Hash function.* We use the hash function family developed in [LM06], denoted $\mathcal{H}(\mathrm{R}_q, m)$. Let $\mathrm{R}_q$ be a ring and $m \geqslant 1$ a positive integer. The hash function $h_{\mathbf{a}} : \mathrm{R}_q^m \to \mathrm{R}_q$ for $\mathbf{a} \in \mathrm{R}_q^m$ is defined as: $\mathbf{x} \mapsto \langle \mathbf{a}, \mathbf{x} \rangle = \sum_{i=0}^{m-1} a_i \cdot x_i$.

**Definition 3 (inspired by [Rüc10, definition 2.1]).** *Let $D \subset \mathrm{R}$, the collision problem $Col(\mathcal{H}(\mathrm{R}_q, m), D)$ asks to find a distinct pair $(\mathbf{x}, \mathbf{x}') \in D^m \times D^m$ such that $h(\mathbf{x}) = h(\mathbf{x}')$ for $h \leftarrow \mathcal{H}(\mathrm{R}_q, m)$.*

*Rejection sampling.* The construction of blind signature we consider, makes a significant use of the rejection sampling technique from [Lyu12]. Such rejection sampling is used in the case we have a distribution depending on a secret we want to hide. The main idea is to "reject" the elements of this distribution using a distribution probability not depending on the related secret. In case we can not perform any rejection sampling, the following lemma also allows to hide the center of a gaussian distribution.

**Lemma 2 ([GKPV10, Lemma 3]).** *Let $v \in \mathrm{R}$ be arbitrary. The statistical distance between the distributions $D_{\mathrm{R},\sigma}$ and $D_{\mathrm{R},\sigma,v}$ is at most $\frac{\|v\|}{\sigma}$.*

### 2.2 Online Voting Definition and Security Properties

We now give the definition of an online voting system, first talking about entities.

- First we get a set of $N$ eligible voters $\mathcal{V}_i$ for $i \in [N]$.
- We also need a set of $p$ authorities $\mathcal{A}_j$ for $j \in [p]$, that will share the private election keys.
- Finally we need a bulletin board $BB$, that will collect the (valid) ballots cast by the voters. At the end of the election, the valid ballots will be tallied.

We take as a basis the definition of an online voting protocol by Cortier et al. [CGGI14]. However, we voluntarily omit the credential phase, in which eligible voters obtain their voting credentials. We also modify some parts to manage the fact that our online voting framework needs interactiveness between voters and authorities.

*Security.* We discuss the security properties a secure online voting protocol should fulfill. Concerning **correctness**, we fit in with the definition of [CGGI14]. The idea is that a genuinely generated ballot is always accepted into the bulletin board, and for an election where all parties behave honestly, the result of the tally always corresponds to the votes cast by the voters.

The **verifiability** property is a fundamental security property needed in online voting schemes which has been the subject of several papers in the voting literature (see [CGK$^+$16]). However, we also rely on the verifiability property introduced in [CGGI14]. Before describing it, we would like to emphasize that we only consider "partial tallying online voting protocols", which means that the tallying phase is not performed in a single computation, but each ballot is open separately, then the resulting tally is computed step by step. Verifiability asks the tallying result to be consistent with the votes cast by honest voters.

Vote secrecy is another fundamental security property. It asks that the voting choice of a voter remains private during and after the end of the election. In our definition, called **ballot anonymity** we depart from the classical ballot privacy requirement, that has been the subject of an intensive research (summarized in [BCG$^+$15]). Indeed in our online voting protocol, each ballot will be anonymous, that is, it does not identify the voter who casts it. This contrasts with most of other voting protocols, where each ballot is directly linked to the voter who casts it, leading to the fact that in the tally procedure each individual ballot could not be open (or decrypt) otherwise this would leak for whom a voter voted. Our ballot anonymity requirement is very close to the privacy property defined in [KR05]. A voting protocol satisfies our ballot anonymity requirement if an attacker cannot link a ballot to the voter who casts it.

## 3 Our Construction

In this section, we first briefly recall the main tools that we are using: blind signature schemes and the ring version of the dual Regev encryption scheme. Then, we transform them in a distributed variant where the private key is shared among several authorities using the result from Bendlin et al. [BKP13]. Finally, we present our scheme and discuss its security.

### 3.1 Cryptographic Primitives

**Blind Signatures.** We first recall the blind signature described in [BCE$^+$20].

– **Setup.** We consider the polynomial ring $\mathrm{R}_q = \mathbb{Z}_q[X]/(X^n+1)$. Two families of hash functions are necessary in this protocol, first a generic one $\mathrm{H} \leftarrow_{\$}$

$\mathcal{H}(1^n) : \{0,1\}^* \to \mathrm{R}_2$ (modelled as a random oracle), and a second one on the specific ring $\mathrm{R}_q$, typically $h \leftarrow_\$ \mathcal{H}(\mathrm{R}_q, m)$ as defined in the preliminaries. Table 1 shows up the different sizes of the parameters: $n$ is a power of 2, in order to have the polynomial $X^n + 1$ irreducible, $m$ ensures the worst-case to average case reduction of the scheme. The others parameters are set such that the rejection sampling and security arguments work.

- **Key Generation.** The key generation algorithm BS.Keygen$(1^n)$ selects a secret key $\mathbf{s} \in \mathrm{R}_3^m$ and a vector of polynomial $\mathbf{a} = (\mathbf{a}'^T \| h\mathbf{g} - \mathbf{a}'^T\mathbf{T_a})^T \in \mathrm{R}_q^m$, along with a trapdoor $\mathbf{T_a}$ on $\mathbf{a}$, such that the hash function $h_\mathbf{a} \in \mathcal{H}(\mathrm{R}_q, m)$ is built with this polynomial vector $\mathbf{a}$. Finally the public key $\mathrm{p} = h_\mathbf{a}(\mathbf{s})$ is made public. BS.Keygen$(1^n)$ outputs $s_k = (\mathbf{s}, \mathbf{T_a})$, $p_k = (\mathrm{p}, \mathbf{a})$.
- **Signature.** The interactive blind signature protocol BS.Sign(Signer$(\mathbf{s}, \mathbf{T_a})$, User$(\mathrm{p}, M)$) is composed of 3 exchanges:
  - The signer generates $\mathbf{y} \hookleftarrow D^m_{\mathrm{R},\sigma}$ and sends $\mathrm{x} = h_\mathbf{a}(\mathbf{y})$ to the user.
  - The user generates two ephemeral vectors $\mathrm{t}_1 \hookleftarrow D_{\mathrm{R},\alpha}, \mathbf{t}_2 \hookleftarrow D^m_{\mathrm{R},\beta}$, such that if $\|\mathbf{t}_2\| > t\sqrt{n \cdot m} \cdot \beta$ it generates a fresh $\mathbf{t}_2 \hookleftarrow D^m_{\mathrm{R},\beta}$ until the test succeeds. He then generates the hashing values $\mathrm{e} = \mathrm{H}(\mathrm{x} - \mathrm{p} \cdot \mathrm{t}_1 - h_\mathbf{a}(\mathbf{t}_2), M)$ and $\mathrm{e}^* = \mathrm{e} - \mathrm{t}_1$ and applies the rejection sampling test on $\mathrm{e}^*$. If this test passes, it sends $\mathrm{e}^*$ and otherwise, it restarts this whole step.
  - The signer generates the signature $\mathbf{z}^* = \mathrm{e}^* \cdot \mathbf{s} + \mathbf{y}$, it applies the rejection sampling test and sends $\mathbf{z}^*$ if the test passes and uses its trapdoor $\mathbf{T_a}$ to generate a presample on $\mathrm{e}^* \cdot \mathrm{p} + \mathrm{x}$ with parameter $\sigma$ if not. Finally if $\|\mathbf{z}^*\| > t\sqrt{n \cdot m} \cdot \sigma$ it generates fresh $\mathbf{z}^*$ with its trapdoor until this test passes. He sends $\mathbf{z}^*$ to the user.

  The user computes $\mathbf{z} = \mathbf{z}^* - \mathbf{t}_2$ and outputs the blind signature $(M, (\mathbf{z}, \mathrm{e}))$.
- **Verification.** The verification procedure BS.Verif$(\mathrm{p}, M, (\mathbf{z}, \mathrm{e}))$ outputs 1 iff $\|\mathbf{z}\| \leqslant D$ and $\mathrm{H}(h_\mathbf{a}(\mathbf{z}) - \mathrm{p} \cdot \mathrm{e}, M) = \mathrm{e}$.

**Encryption Scheme.** Concerning the encryption scheme, we use the Dual-Regev encryption scheme [GPV08,LPR13] on a polynomial ring $\mathrm{R}_q$. However, any post-quantum encryption scheme would fit into our voting protocol.

- **Setup.** The set-up algorithm PK.Setup chooses integers $n, m, q$ and two real $\alpha, \beta$ such that the dual-Regev encryption scheme on the polynomial rings is secure (see [LPR13]).

| Parameter | Value | Asymptotic |
|---|---|---|
| $n$ | power of 2 | - |
| $m$ | $\lfloor \log q \rfloor + 1$ | $\Omega(\log n)$ |
| $\gamma$ | $n\alpha$ | $\mathrm{O}(n\sqrt{n})$ |
| $\alpha$ | $\omega(k\sqrt{\log n})$ | $\mathrm{O}(\sqrt{n})$ |
| $\beta$ | $2^{\omega(\log n)}\sigma\sqrt{n}$ | $\mathrm{O}(n^3\ 2^{\omega(\log n)})$ |
| $\sigma$ | $\omega((n\sqrt{n}\alpha)\sqrt{\log n})$ | $\mathrm{O}(n^2\sqrt{n})$ |
| $D$ | $t\sqrt{n \cdot m}(\beta + \sigma)$ | $\mathrm{O}(n^3\sqrt{n}\ 2^{\omega(\log n)})$ |
| $q$ | $\geqslant 4mn\sqrt{n}\log(n)D$.prime | $\Theta(n^6\ 2^{\omega(\log n)})$ |

**Table 1.** Parameters of [BCE+20].

- **KeyGen.** The Key generation algorithm PK.KeyGen($1^n$) starts by sampling $\mathbf{s} \hookleftarrow D_{\mathrm{R}^m,\alpha}$, $\mathbf{a} \leftarrow_{\$} \mathrm{R}_q^m$ uniformly at random and computes $u = \mathbf{a}^T\mathbf{s} \in \mathrm{R}_q^m$. It outputs $(e_k, m_k)$ where $e_k = \mathbf{s} \in \mathrm{R}_q^m$ is the secret key and $m_k = (\mathbf{a}, u) \in \mathrm{R}_q^m \times \mathrm{R}_q$ is the public key.
- **Encrypt.** Given a message $m \in \mathrm{R}_2$, and a public key $m_k$, the encryption algorithm PK.Encrypt$(m, m_k)$ chooses a vector $v \in \mathrm{R}_q$ uniformly at random, and outputs the ciphertext $(\mathbf{b} = \mathbf{a}v + \mathbf{e}, c = u \cdot v + e' + \lfloor q/2 \rfloor m) \in \mathrm{R}_q^m \times \mathrm{R}_q$ where $\mathbf{e} \hookleftarrow D_{\mathrm{R}^m,\beta}$, $e' \hookleftarrow D_{\mathrm{R},\beta}$.
- **Decrypt.** Given a ciphertext $(\mathbf{b}, c) \in \mathrm{R}_q^m \times \mathrm{R}_q$ and a private key $e_k = \mathbf{s} \in \mathrm{R}_q^m$, PK.Decrypt$((\mathbf{b}, c), e_k)$ computes $\mu = c - \mathbf{b}^T\mathbf{s} = -\mathbf{e}^T \cdot \mathbf{s} + e' + \lfloor q/2 \rfloor m$. To recover the message $m$, it suffices to look after each coordinate of $\mu$, if the $i$-th coordinate is closer to 0 than to $\lfloor q/2 \rfloor$ then the $i$-th bit of $m$ is equal to 0 and 1 otherwise.

### 3.2 Threshold Functionalities

**Threshold Tools and Variants.** In the original version of a blind signature scheme, there is only one signer who could easily, in the context of online voting, stuff the Bulletin Board by adding as many valid (but illegitimate) ballots as he wishes. We therefore transform it into a threshold one, using the generic transformation of a trapdoor based signature scheme with strong trapdoor of [MP12], into a threshold trapdoor based signature scheme by [BKP13]. The construction of [BKP13] is built on the integer ring $\mathbb{Z}_q$, but the blind signature of [BCE+20] relies on polynomial ring $\mathrm{R}_q$. However the strong trapdoor construction can be adapted to this ring setting [MP12], and the Shamir secret sharing [Sha79] still works on this type of rings. Then the whole construction of [BKP13] can be adapted to the polynomial ring setting.

As our transformation is applied on a blind signature scheme and not on a signature scheme, we have several modifications to provide. The signer's part of the blind signature is composed of two steps. At first, it has to generate a commitment, which one can be transformed in a threshold manner using Shamir secret sharing and a trusted setup to share a Gaussian vector. The second step consists in a classic "Fiat-Shamir with abort" signature, which can easily be transformed into a threshold one by means of homomorphic properties of the Shamir secret sharing. In case of abort, the signer performs a GPV-like signature which is a generic signature scheme and can be transformed using the generic transformation of [BKP13] into a threshold scheme.

The proofs of the various protocols from [BKP13] are realized in the UC model [Can01], so that we just have to plug the threshold functionalities into the blind signature scheme, to obtain, by composability, a secure threshold variant of the blind signature scheme. Below we describe the two main protocols, which are the KeyGen and the SampleZ protocol. Moreover we choose to give an informal description of the functionalities involved for these two protocols. The full construction can be found in the paper of [BKP13].

We consider $p$ authorities, such that a threshold of $t$ authorities is mandatory to execute the various functions developed below. Let $\mathbf{a}' \in \mathrm{R}_q^{m-k}$ be a uniformly

distributed vector of polynomial and $\mathbf{T} \in \mathrm{R}_q^{(m-k)\times k}$ be a Gaussian-distributed matrix. Let $\{[\mathbf{T}]^i\}_{i\in[p]}$ be the shares of the polynomial matrix $\mathbf{T}$. Let us denote by $\mathbf{a}_1 = \mathbf{a}'^T \cdot \mathbf{T} \mod q$ and $\mathbf{a} = [\mathbf{a}'|\mathbf{a}_1]$.

- $\mathcal{F}_{\text{Blind}}$: This functionality takes as input shares of an arbitrary value $x$ and output fresh shares $[x]^i$ of this same value.
- $\mathcal{F}_{\text{SampZ}}$: This functionality takes as input dimensions $h \times d$ and a gaussian variance $z$. It outputs shares $[\mathbf{Z}]^i$ of a gaussian distributed matrix $\mathbf{Z} \leftarrow D_z^{h\times d}$.
- *Threshold KeyGen protocol*: The KeyGen protocol is realised in the $\mathcal{F}_{\text{Blind}}$, $\mathcal{F}_{\text{SampZ}}$ model. On input the tuple $(\mathbf{a}', h^* \in \mathbb{R}_q, z \in \mathbb{Z})$, each party $i$ does:
  1. call $\mathcal{F}_{\text{SampZ}}((m-k)\times k, z)$, then receive $[\mathbf{T}]^i$;
  2. call $\mathcal{F}_{\text{Blind}}(-\mathbf{a}'^T[\mathbf{T}]^i)$ , then receive $[\mathbf{a}_1]^i$;
  3. broadcast $[\mathbf{a}_1]^i$ and reconstruct $\mathbf{a}_1 = \mathbf{a}'^T \cdot \mathbf{T} \mod q$ from other shares;
  4. output $\mathbf{a} = [\mathbf{a}'|h^* \cdot \mathbf{g} + \mathbf{a}_1]$ as the public key and $[\mathbf{T}]^i$ as the private key of the authority $i$.
- $\mathcal{F}_{\text{Gadget}}$: It takes as input a coset value $v \in \mathrm{R}_q$ and outputs shares $[\mathbf{u}]^i \in \mathrm{R}^k$ of a gaussian distributed polynomial vector such that $\mathbf{g}^T \cdot \mathbf{u} = v$.
- $\mathcal{F}_{\text{Correct}}$: This functionality generates for each $j \in [k]$ and $v \in \mathrm{R}_q$ queues $Q_{j,v}$ of at least $B$ values in each queue, that will allow the signer to perform at least $B$ pre-image of each vector $v \in \mathrm{R}_q$. Each queue $Q_{j,v}$ is composed by using the gadget functionality developed above and the shares $[\mathbf{T}]^i$ of the trapdoor such that each authority gets a share of $\mathbf{y}_{j,v} = \begin{bmatrix}\mathbf{T}\\ \mathbf{I}\end{bmatrix} (\mathbf{e}_j \otimes \mathbf{z}_{j,v})$ for $\mathbf{z}_{j,v} \in \Lambda_v^{\perp}(\mathbf{g}^T)$, with $\mathbf{e}_j$ the vector composed of 0 elements except the $j$-th coordinate which is equal to 1.
  Then, in the sampling algorithm of [MP12], when we have to correct a perturbation to get a correct sample for a given syndrom $v \in \mathrm{R}_q$, the authorities recover a value in the corresponding queue $Q_{j,v_1}, \cdots, Q_{j,v_n}$.
- $\mathcal{F}_{\text{Perturb}}$: The perturb algorithm in the threshold setting, is the same as in the standard setting, but the perturbation vector is then shared between the $p$ authorities using the functionality $\mathcal{F}_{\text{SampZ}}$. then it takes as inputs a dimension $h\times d$ and a gaussian parameter $z$, it outputs $[\mathbf{P}]^i$ with $\mathbf{P} \leftarrow D_z^{h\times d}$.

*SampleZ protocol.* Using $\mathcal{F}_{\text{Perturb}}$ and $\mathcal{F}_{\text{Correct}}$ defined above (threshold counterparts of the steps composing the **Preimage sampling** protocol introduced in [MP12]), the SampleZ protocol generates a presample in the same way, but with the threshold variants of the subalgorithms perturb and correct.

**Threshold Variants of our Building Blocks.** Using the above tools, we now give the modifications we need to provide a threshold variant of both the encryption scheme (PK.KeyGen to TBS.KeyGen) and the blind signature primitive of [BCE+20] (BS.KeyGen and BS.Sign to TBS.KeyGen and TBS.Sign resp.).

- **TPK.KeyGen**$(1^n, 1^p)$. It generates $\mathbf{s} \hookleftarrow D_{\mathrm{R}^m,\alpha}$ as the secret key in a distributed way using the $\mathcal{F}_{\text{SampZ}}$ algorithm, such that each authority $\mathcal{A}^i, i \in [p]$ gets a share $[\mathbf{s}]^i$. Concerning the public key, it chooses $\mathbf{a} \leftarrow_{\$} \mathrm{R}_q^m$ uniformly at

random. Finally each authority computes and reveals $[u]^i = \mathbf{a}^T[\mathbf{s}]^i \in \mathrm{R}_q^m, i \in [p]$ such that $u$ can be recovered and output publicly. The secret keys are then the elements $[e_k]^i = [\mathbf{s}]^i \in \mathrm{R}_q^m, i \in [p]$ and the public key is composed of the pair $m_k = (\mathbf{a}, u) \in \mathrm{R}_q^m \times \mathrm{R}_q$, it outputs $(e_k, m_k)$.

- **TBS.KeyGen**$(1^n, 1^p)$. TBS.KeyGen generates a public polynomial vector $\mathbf{a}$ with a trapdoor $\mathbf{T}$ using a trapdoor generation algorithm in a distributed way using the Threshold Keygen protocol described above. It then generates a random polynomial vector $\mathbf{s} \in \mathrm{R}_3^m$ with its image by the hash function such that $\mathrm{p} = h_{\mathbf{a}}(\mathbf{s})$. Concerning $\mathbf{s}$, the algorithm $\mathcal{F}_{\mathrm{SampZ}}$ is executed by each authority, in order to obtain $[\mathbf{s}]^i, i \in [p]$, they each then have to broadcast their public part $\mathbf{a} \cdot [\mathbf{s}]^i, i \in [p]$ to recover and output the public key p. Finally the algorithm outputs the public key $p_k = (\mathrm{p}, \mathbf{a})$ and the private key share $[s_k]^i = ([\mathbf{T}]^i, [\mathbf{s}]^i), i \in [p]$ to each authority $\mathcal{A}^i, i \in [p]$.
- **TBS.Sign**$(\{(\mathcal{A}^i([s_k]^i))\}_{i \in T}, \mathcal{V}(p_k, M))$. Considering a set of $T$ signing authorities $\mathcal{A}^i, i \in [T]$, the signature algorithm is the same as the one in [BCE+20] from the user's side. Concerning the signer's view, firstly the commitment $\mathbf{y}$ is generated in a distributed manner using the algorithm $\mathcal{F}_{\mathrm{SampZ}}$, such that the authorities get a share $[\mathbf{y}]^i$ and distributively output the corresponding element $\mathrm{x} = h_{\mathbf{a}}(\mathbf{y})$ in the same way as it was done in the TBS.KeyGen algorithm for the pair $(\mathbf{s}, \mathrm{p})$. Concerning the signing step, from the signer's view, the first attempt of signature, which is a Fiat-Shamir like signature [Lyu12], is performed between the authorities thanks to the homomorphic property of the Shamir secret sharing, while the GPV-like [GPV08] signature generation is performed in a threshold manner using the SampZ protocol. Finally, the algorithm outputs the signature $\sigma = (M, \mathrm{e}, \mathbf{z})$.

### 3.3 Our Scheme

As explained above, we chose to modify the [FOO92] framework in order to let voters "Vote and go" and to prevent ballot stuffing by a malicious authority. Instead of committing to their voting choices, voter will have to encrypt them using the public election key. At the end of the election, the decryption key will be disclosed so that anyone will be able to decrypt the ballots and compute the result of the election. To avoid fraud by a malicious authority, we transform the underlying encryption and blind signature schemes considered into threshold variants, so that the corresponding private keys would be shared among several authorities and not a single one.

Considering these modifications of the [FOO92] framework, we describe the complete online voting protocol that we build from the above cryptographic primitives. First, a setup phase generates the parameters of the protocol, including the private and public keys of the used cryptographic schemes. Next the voting phase is composed of two steps: the voter first encrypts his voting option (using the public election key) and then interacts with (at least) $t$ voting authorities to obtain a blind signature on his ciphertext. His ballot $b$ is composed of the ciphertext $\mathbf{c}$ of his voting choice $v$ along with a (blind) signature $\sigma$ on $\mathbf{c}$ such that $b = (\mathbf{c}, \sigma)$. The Bulletin Board accepts the ballot if $\sigma$ is a valid signature

on $\mathbf{c}$ and discards it otherwise. In the counting phase, the tallying authorities reveal their share of the private encryption key, so that anyone can recover the corresponding decryption key and decrypt the ballots (the ciphertexts $\mathbf{c}$) to compute the result of the election. Auditing the election is easy. For this purpose, an interested voter first has to check that all the ballots collected by the Bulletin Board are valid (i.e. that the signatures $\sigma$ are valid) and that the decryption key published by the talliers is correct (i.e., corresponds to the public election key). He then has to decrypt all the ballots using the decryption key and computes the result of the election just as the talliers did.

• **Setup**$(1^n, 1^p, 1^N)$**.** This algorithm has to generate two pairs of secret/public keys, one pair for the encryption scheme and another one for the blind signature scheme. Moreover these keys have to be generated in a threshold manner, for a number $p$ of authorities, with a threshold number of $t$. Let us denote by $(s_k, p_k) \leftarrow \mathbf{BS.KeyGen}(1^n)$ and $(e_k, m_k) \leftarrow \mathbf{PK.KeyGen}(1^n)$, and by $[s_k]^i$ (resp $[e_k]^i$) the shares of the private blind signature key (resp encryption key). Then the setup algorithms outputs $\mathbf{pk} = (p_k, m_k)$ and $\mathbf{sk} = ([s_k]^i, [e_k]^i)_{i\in[p]}$.

• **Vote**$(\mathcal{V}_i(v, \mathbf{pk}), \mathcal{A}^j([s_k]^j)_{j\in \mathrm{T}})$**.** The voting phase is split in two steps. First, the voter $\mathcal{V}_i$ encrypts his vote $v \in \{0,1\}^*$ in $\mathbf{c} = \mathbf{PK.encrypt}(v, m_k)$ in an offline phase. Then in an online phase, he is first authenticated (to check whether he is an eligible voter who has not yet requested a blind signature from the voting authorities). The protocol aborts if the authentication failed or if the voter already requested a blind signature. He then interacts with voting authorities $\mathcal{A}^j$ to get a blind signature $\sigma = \mathbf{BS.Sign}(\{\mathcal{A}^j([s_k]^j)\}_{j\in \mathrm{T}}, \mathcal{V}_i(p_k, \mathbf{c}))$, with T a set of authorities of size at least $t$. Finally the voter outputs $(\sigma, \mathbf{c})$ as his ballot and casts it, anonymously, into the bulletin board BB.

• **Validate(**$b$**, pk).** On input a ballot $b = (\sigma, \mathbf{c})$, anyone can check its validity by performing the verification algorithm of the blind signature $\mathbf{BS.Verify}(p_k, \mathbf{c}, \sigma)$, it outputs 0 if the blind signature verification fails and 1 otherwise.

• **Box(**$BB$**,** $b$**).** It takes as input the current state of the bulletin board $BB$, along with a ballot $b$. It first checks the validity of $b$ by performing the algorithm described above: Validate($b$, **pk**). It updates $BB \leftarrow BB \cup \{b\}$ if Validate outputs 1 and remains unchanged if it outputs 0.

• **Tally**$(BB, p_k, \mathcal{E}^j([e_k]^j)_{j\in[p]})$**.** At the end of the election, at least $t$ (the threshold) authorities $(\mathcal{E}^j)_{j\in[t]}$ holding the shares of the decryption key $e_k$ reveal publicly their share $([e_k]^j)_{j\in t}$, such that anyone can rebuild the decryption key $e_k$. Then for each ballot $(\sigma, \mathbf{c}) \in BB$, anyone can decrypt $\mathbf{c}$ and retrieve the vote $v = \mathrm{PK.decrypt}(\mathbf{c}, e_k)$ of each voter, after verifying that $\mathrm{BS.Verify}(\sigma, p_k) = 1$. Then he/she can tally and outputs the result $r$, which corresponds to the outcome of the election $\mathbf{r} = \{v_i\}_{i\in k}$ with $k \leqslant N$ the number of voters that output a valid ballot.

• **Verify**$(BB, \mathbf{r}, e_k)$**.** This algorithm is straightforward. Since the decryption secret key $e_k$ is made public (and since anyone can check that it corresponds to the public election key), anyone can check the validity of the result by decrypting all the ciphertexts $\mathbf{c}$ contained in valid ballots $(\sigma, \mathbf{c}) \in BB$ (i.e., with a valid blind signature $\sigma$) and tally them to compare to the announced result $\mathbf{r}$.

### 3.4 Security of our Scheme

**Theorem 1 (Correctness).** *Since the blind signature and public key encryption schemes are correct, then our online voting scheme is correct.*

*Proof.* According to our definition, our voting protocol is correct if a ballot generated by an honest voter is accepted with overwhelming probability by the BB, and if the result of an election where every party behaves honestly, corresponds to the votes cast by voters. The first condition is fulfilled since the blind signature scheme used to authenticate valid ballots is correct. The second condition is verified since the encryption scheme satisfies the correctness requirement. Since our voting protocol satisfies both conditions it is therefore correct.

**Theorem 2 (Verifiability).** *Using a strong authentication scheme and a one-more unforgeable blind signature scheme, our voting scheme is verifiable.*

*Sketch of proof.* To win the game, the attacker has either to (1) impersonate an honest voter or (2) cast more valid votes (let say $n_C + 1$) than the number $n_C$ of corrupted users. (1) would mean that he has successfully broken the strong authentication scheme used by voters to authenticate to Voting Authorities. (2) would mean that the attacker could generate more valid blind signatures than requested (therefore breaking the one-more unforgeability of the threshold blind signature scheme) or that there exists more dishonest voting authorities than assumed (which could generate as many valid but illegitimate-ballots as they wish). Furthermore, he can not cheat after the end of the election, since the election's tally is made public. We further notice that the tally can not give two different results for two iterations of the Tally algorithm since the decryption mechanism and the blind signature verification algorithm are both deterministic.

**Theorem 3 (Ballot anonymity).** *Our voting scheme provides perfect (information-theoretic) ballot anonymity.*

*Sketch of proof.* In the ballot anonymity game, an attacker $\mathcal{A}^*$ chooses two honest voters $\mathcal{V}_0$ and $\mathcal{V}_1$ and two voting options $v_0$ and $v_1$. It then interacts with $\mathcal{V}_0$ and $\mathcal{V}_1$ who then cast, using a perfectly anonymous channel, two ballots $b_c$ (on $v_0$) and $b_{1-c}$ (on $v_1$). $\mathcal{A}^*$ has to identify which voter outputs which ballot. We then have to prove that the attacker $\mathcal{A}^*$ has a negligible advantage (compared to random guessing) to win. In our protocol, each ballot does not include any information about the identity of each voter, since the encryption and blind signature schemes are performed only on the voting choices $v_0$ and $v_1$. As our blind signature scheme provides perfect blindness and since we assumed that ballots are cast via a perfectly anonymous channel, $\mathcal{A}^*$ cannot find $c$ better than random guessing. Therefore, provided that ballots are cast via a perfectly anonymous channel, our voting protocol provides perfect ballot anonymity.

*Partial results.* The encryption scheme prevents any partial result to leak. Indeed the decryption key is shared among several authorities, which cannot open the ballots without at least $t$ shares of it. Then as long as $p - t + 1$ of them remain honest, the ballots cannot be opened before the end of the election.

## 4 Conclusion

In this paper we presented a new practical lattice-based online voting system. In contrast to traditional schemes, our protocol does not rely on homomorphic aggregation or mix-nets and does not make use of zero-knowledge proofs, which have previously been the main issue in the post-quantum setting. Instead, our scheme extends on an idea first introduced at Auscrypt'92, where the security is (among others) achieved through a blind signature scheme. Compared to the state-of-the art in post-quantum online voting, our system supports complex ballots and provides stronger privacy guarantees (namely everlasting privacy thanks to the perfect blindness provided by the blind signature scheme we used). In a future version of this work, we plan to implement our protocol and present benchmarks of its computational runtime and to develop the intuitive security analysis presented here, using rigorous definitions and formal proofs.

## Acknowledgement

The authors wants to thank the anonymous reviewers for their useful comments. This work has been supported by the European Union H2020 PROMETHEUS Innovation Program Grant 780701.

## References

Adi08. B. Adida. Helios: Web-based open-audit voting. In *USENIX Security Symposium*, pages 335–348. USENIX Association, 2008.

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108. ACM, 1996.

Ban93. W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–636, 1993.

BCE+20. S. Bouaziz-Ermann, S. Canard, G. Eberhart, G. Kaim, A. Roux-Langlois, and J. Traoré. Lattice-based (partially) blind signature without restart. *IACR Cryptol. ePrint Arch.*, 2020:260, 2020.

BCG+15. D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi. Sok: A comprehensive analysis of game-based ballot privacy definitions. In *IEEE Symposium on Security and Privacy*, pages 499–516. IEEE Computer Society, 2015.

BKP13. Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *ACNS*, volume 7954 of *Lecture Notes in Computer Science*, pages 218–236. Springer, 2013.

Can01. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

CGGI14. V. Cortier, D. Galindo, S. Glondu, and M. Izabachène. Election verifiability for helios under weaker trust assumptions. In *ESORICS (2)*, volume 8713 of *Lecture Notes in Computer Science*, pages 327–344. Springer, 2014.

CGGI16. I. Chillotti, N.s Gama, M. Georgieva, and M. Izabachène. A homomorphic LWE based e-voting scheme. In *PQCrypto*, volume 9606 of *LNCS*, pages 245–265. Springer, 2016.

CGK+16. V. Cortier, D. Galindo, R. Küsters, J. Müller, and T. Truderung. Sok: Verifiability notions for e-voting protocols. In *IEEE Symposium on Security and Privacy*, pages 779–798. IEEE Computer Society, 2016.

dPLNS17. Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In *ACM Conference on Computer and Communications Security*, pages 1565–1581. ACM, 2017.

FOO92. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *AUSCRYPT*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer, 1992.

GGH13. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013.

GKPV10. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008.

HKLN20. Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. *IACR Cryptol. ePrint Arch.*, 2020:769, 2020.

KKLA01. K. Kim, J. Kim, B. Lee, and G. Ahn. Experimental design of worldwide internetvoting system using pki. *SSGRR2001*, 2001.

KR05. S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *ESOP*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.

LM06. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2006.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-lwe cryptography. In *EUROCRYPT*, volume 7881 of *LNCS*, pages 35–54. Springer, 2013.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, volume 7237 of *LNCS*, pages 738–755. Springer, 2012.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, 2012.

PR06. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer, 2006.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

Rüc10. Markus Rückert. Lattice-based blind signatures. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 2010.

Sha79. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

Sho97. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.