More Efficient Post-Quantum Electronic Voting from NTRU

Patrick Hough[1⋆], Caroline Sandsbråten[2], and Tjerand Silde[2]

[1] Mathematical Institute
Oxford University
patrick.hough@maths.ox.ac.uk
[2] Department of Information Security and Communication Technology
Norwegian University of Science and Technology
{caroline.sandsbraten, tjerand.silde}@ntnu.no

Abstract. In recent years, there has been much focus on developing core cryptographic primitives based on lattice assumptions, driven by the NIST call for post-quantum key encapsulation and digital signature algorithms. However, more work must be conducted on efficient privacy-preserving protocols with post-quantum security.

Electronic voting is one such privacy-preserving protocol whose adoption is increasing across the democratic world. E-voting offers both a fast and convenient alternative to postal voting whilst further ensuring cryptographic privacy of votes and offering full verifiability of the process. Owing to the sensitivity of voting and the infrastructure challenges it poses, it is important that post-quantum security be baked into e-voting solutions early.

We present a post-quantum e-voting scheme based on the hardness of the RLWE and NTRU lattice problems, providing concrete parameters and an efficient implementation. Our design achieves a factor $5.3\times$ reduction in ciphertext size, $2.5\times$ reduction in total communication cost, and $2\times$ reduction in total computation time compared to the state-of-the-art lattice-based voting scheme by Aranha et al. (ACM CCS 2023). We argue that the efficiency of this scheme makes it suitable for real-world elections.

Our scheme makes use of non-ternary NTRU secrets to achieve optimal parameters. In order to compute the security of our design, we extend the ternary-NTRU work of Ducas and van Woerden (ASIACRYPT 2021) by determining the concrete fatigue point (for general secrets) of NTRU to be $q = 0.0058 \cdot \sigma^2 \cdot d^{\,2.484}$ (above which parameters become *overstretched*) for modulus q, ring dimension d, and secrets drawn from a Gaussian of parameter σ. We consider this relation to be of independent interest and demonstrate its significance by improving the efficiency of the (partially) blind signature scheme by del Pino and Katsumata (CRYPTO 2022).

Keywords: Lattice Cryptography · Electronic Voting · NTRU

⋆ Work done in part while visiting the Norwegian University of Science and Technology.

1 INTRODUCTION

With the advent of quantum computers, all public key primitives based on the hardness of factoring or computing discrete logarithms will be deemed insecure.

To mitigate this, there has been an international effort to replace these primitives with ones based on assumptions conjectured to be secure against quantum adversaries. This process, led by the National Institute of Standards and Technology (NIST), has recently concluded with the selection of standards for key encapsulation and digital signature algorithms. Three of four standards [SAB+20, LDK+20, PFH+20] are built from *structured lattice assumptions*; Ring Short Integer Solution (RSIS) [Ajt96], Ring Learning With Errors (RLWE) [Reg05], and NTRU [HPS98], or the *module* versions of the two former assumptions. Despite this standardization effort, there is still much work to be done in designing post-quantum secure *privacy-preserving* primitives. In this work, we focus on one of these; electronic voting (e-voting). More precisely, we consider internet voting which allows for fully remote ballot casting via a voter's device as opposed to voting machines at a polling station, though our framework could in theory be implemented in this setting also. Herein, 'e-voting' should be read as synonymous with internet voting.

E-voting has become increasingly prevalent with the first experiments for democratic elections beginning around the turn of the millennium. The first binding election to be carried out online was for the Arizona primary in 2000 [CBS00]. In 2005, Estonia offered internet voting nationally [Vin15] and in 2019, over 45% of the votes cast in Estonian parliamentary elections were cast online. Switzerland used its Swiss Post voting system in the 2023 national elections for the first time [Swi23] and continues to be one of the leaders in e-voting uptake. Ontario, Canada increasingly offers online voting with 177 municipalities exclusively using online voting in the 2018 municipal elections [AC19]. In Australia, over 650,000 online voters participated in the 2021 state election in New South Wales [New21] and in the US, Microsoft's ElectionGuard protocol was successfully used for the 2022 midterm elections in Franklin County, Idaho [Mic23].

E-voting has a number of attractive advantages over traditional voting methods. Analysis of Estonian local elections in 2017 showed the per-vote cost of online ballots was a factor $2\times$ to $10\times$ cheaper than election-day paper ballots at just 2.3 Euro with the most expensive paper ballot option used being 20.4 Euro [RK18]. Moreover, the 2023 Estonian parliamentary elections revealed that the environmental impact (CO2 emissions) of paper ballots was 180 times higher than that for online ballots and in many cases, its adoption has resulted in a higher voter satisfaction and turnout rate [Sol01, SMPS16].

In addition to these benefits, e-voting offers the potential for a set of unique features that enhance both the *integrity* and *privacy* of voting. The first is verifiability; both individual and universal. Individual verifiability allows a voter to check that their ballot was recorded correctly in the final count, whilst universal verifiability allows anyone to check that parties involved in the ballot processing carried out their tasks correctly. While this represents a great bolstering to the integrity of the voting process, it can be executed whilst preserving the privacy of voters and their ballots. The second significant property enabled by e-voting is the distribution of the ballot processing. A distributed decryption of the ballots ensures that no single party can alter the election outcome. We emphasise that using a distributed ballot opening process is also a necessary condition for voter privacy. In the framework deployed by Swiss Post, a centralised decryption server could choose to decrypt the set of shuffled ballots passed from the mix-net in addition to decrypt the ones sent by voters to the mix-net (these ciphertexts are public). Such behaviour would de-anonymise voters while the honest behaviour of only a single decryption server in the distributed setting prevents such an attack on voter privacy. Finally, we wish to highlight the inherent *lack of privacy in postal voting*. In the last national elections of the US and UK, postal votes accounted for roughly a quarter [U.S16] and a fifth [Ele17] of all ballots respectively. However, in both cases personal information must be included with the ballot in order to identify the voter. Such a system clearly does not preserve the privacy of voters, a weakness that is eliminated by e-voting.

Moreover, we emphasise the importance of *long-term* privacy of electronically cast ballots. The increasing deployment of e-voting protocols currently outpaces solutions providing post-quantum security. Evidenced by the rapid adoption of the PQC NIST standards, the urgency to defend against a potential quantum adversary is plain to see. It is thus essential that post-quantum security be baked into the designs for e-voting from the outset. Post-quantum security is thus central to the long-term privacy of voters.

From a desire to achieve these privacy enhancements has emerged the 'mix-and-decrypt' paradigm. Here, multiple servers verifiably shuffle encrypted ballots before they are decrypted in a distributed manner. This is commonly achieved by combining a verifiable mix-net [Cha03] with a distributed public-key encryption scheme. Many previous voting designs have used this structure and in 2019, Switzerland (via its national postal service Swiss Post), deployed a national e-voting infrastructure using this paradigm and the Bayer-Groth mix-net [BG12].

Despite much success in developing e-voting protocols, only a few are based on post-quantum assumptions. The most notable works are the schemes by del Pino et al. [dLNS17], Aranha et al. [ABG$^+$21], Farzaliyev et al. [FWK21], and Aranha et al. [ABGS23], the latter being the most efficient scheme based on (Ring) SIS and (Ring) LWE. Of these schemes, only the last one satisfies the golden mix-and-decrypt standard for general ballots. Even then, the communication cost of this scheme is around two orders of magnitude greater than the one employed by Swiss Post, based on classical assumptions.

The inefficiency of the state-of-the-art scheme in [ABGS23] stems from the need to decrypt ballots correctly being hindered by a few key features of their design. Firstly, we note that in order to optimise the computational cost of lattice-based protocols and to rely on reductions to worst-case problems, one would like to use polynomial rings whose dimension is a power of two. This imposes the first constraint on parameters ([ABGS23] uses ring dimension 4096). Next, the mixing and distributed decryption stages both use homomorphic operations on encrypted ballots. This has the effect of increasing the noise within each ciphertext. To accommodate this, one must use a ring with a larger modulus to 'soak up' this extra noise. This larger modulus itself increases the size of objects in the scheme but further still, one might need to use a larger ring dimension in turn to ensure that the underlying assumptions are still secure. Finally, the distributed decryption process requires decryption servers to use so-called 'noise-drowning' to ensure that decryption shares do not leak anything about the server's decryption key. Further, each decryption server must prove, in zero-knowledge, that they have applied this noise drowning operation. So far, proving knowledge of such a large element over lattices, can only be done using 'approximate' proofs where one can give only approximate guarantees about the size of the noise-drowning term. The noise drowning hugely increases the noise in ciphertexts, and the loose zero-knowledge proof further pushes up parameters if correct ballot decryption is to be ensured. Unfortunately, all three of these features appear to be crucial in realising the coveted mix-and-decrypt framework in a quantum-secure fashion; the power-of-two ring dimension allows for a highly optimised implementation, the noise-growing homomorphic operations are fundamental to constructing their mix-net and distributed decryption building blocks, and despite two decades of lattice-based distributed decryption design, no efficient alternative to noise-drowning has been found.

Despite the large efficiency gap between classically secure schemes and the work in [ABGS23], it is hard to see how significant efficiency improvements can be found without a new approach.

In privacy-preserving protocols, zero-knowledge proofs (ZKPs) are deployed to verify the honest actions of parties, and usually dominate the communication cost. Observation of the recent NIST PQC standards reveals that both the RLWE and RSIS problems are used, however, there is a third long-standing problem which appears in the Falcon digital signature [PFH+20]; NTRU. Crucially, NTRU ciphertexts contain only a single ring element with three secret elements give rise to simpler ZKP relations when compared to their two-component RLWE-based counterparts such as the BGV encryption scheme [BGV12], as used in [ABGS23], which contains five secret elements. Thus, NTRU might appear to be an attractive candidate problem from which to design a voting protocol.

However, whilst the hardness of the RLWE and RSIS problems are well understood, the picture for NTRU is less clear. Recall, the NTRU problem [HPS98]. Let R_q be a polynomial ring of dimension d and modulus q and sample polynomials f and g with coefficients from some discrete Gaussian D_{σ}^{d}. Informally, the NTRU problem is to recover f and g given h, where $h = g/f \in R_q$. In recent years, it has been shown that NTRU is vulnerable to a unique attack when defined for so-called 'overstretched' parameters [ABD16, CJL16a] i.e. when the modulus q is very large compared to d. Whilst a line of recent works [KF17, Dv21] has made progress in understanding the parameters for which this attack applies, it is not clear how the size of the secrets f and g (parametrized by σ) influence the feasibility of the attack. Thus, designs using large parameters as found in privacy-preserving lattice constructions tend to use the RLWE and RSIS problems for which such behaviour is well understood.

1.1 Our Contribution

We propose an electronic voting protocol in the mix-and-decrypt paradigm based on the RLWE and NTRU lattice assumptions. We build each building block from the ground up by presenting an NTRU-LWE-based verifiable distributed decryption scheme and an NTRU-based verifiable mix-net. Moreover, via an in-depth analysis of the NTRU problem, we give a concrete relation describing the hardness of the NTRU problem for general secret sizes. We demonstrate the significance of this relation in allowing optimal parameter selection for our scheme and other NTRU-LWE-based works in which large parameters are necessary. Finally, we give an efficient implementation demonstrating significant efficiency gains over the state-of-the-art in both communication and computational cost.

Verifiable Mix-Net from NTRU. We present a verifiable mix-net for NTRU ciphertexts comprising a series of shuffle servers that each apply a secret permutation to the set of input ciphertexts. As long as one shuffle server is honest, the set of input ciphertexts cannot be pair-wise matched to the set of output ciphertexts. Our mix-net is simpler than the one in [ABGS23] owing to the single-element NTRU ciphertexts which yield a cleaner protocol than two-element BGV ciphertexts do. Furthermore, one proves knowledge of fewer secret objects when applying ZKPs for verifiability.

Verifiable Distributed Decryption from NTRU. We present a distributed decryption protocol based on a variant of the NTRUEncrypt scheme of Steinfeld and Stehlé [SS11], proving its security using both the RLWE and NTRU assumptions. This allows for for favourable parameters owing to a computationally secure public NTRU key (vs. a statistically secure one in [SS11]). We then apply an exact zero-knowledge proof (ZKP) in order to prove the well-formedness of decryption shares. In particular, this proof proves knowledge of the large noise drowning term needed to prevent leakage of the decryption key and does so in an *exact* fashion. That is, our ZKP (which is a modification of the one by Bootle et al. [BLNS21]) proves a tight bound on the size of the noise downing term. To our knowledge, this is the first exact ZKP of a 'large' secret vector for lattice relations and may be of independent interest. We note that while this makes the proof of distributed decryption larger, it allows for a less restrictive correctness condition, leading to better global parameters throughout the scheme, more than making up for any additional communication cost incurred by this proof.

NTRU Security Analysis. We build upon the work of Ducas and van Woerden [Dv21], on NTRU hardness, to analyse the so-called 'overstretched attack' against NTRU when the norm of the secrets grows with respect to the dimension and modulus. We stress that [Dv21] does give an asymptotic fatigue point for general NTRU but only a concrete relation for ternary secrets. Employing the scripts provided in [Dv21], our analysis shows that when we increase σ, then q can be increased with the *square* of this increase before reaching the fatigue point. Concretely, given σ and ring dimension d our experiments suggest a fatigue point q given by the following expression

$$q = 0.0058 \cdot \sigma^2 \cdot d^{\,2.484}.$$

Note, by following a similar asymptotic analysis to that in [Dv21], we confirm that the influence of σ on the fatigue point must indeed manifest only in the leading constant and not in the exponent of d.

To demonstrate the importance of this quadratic relationship, we recompute parameters for the recent blind signature by del Pino and Katsumata [dK22], improving its efficiency compared to the original scheme, which uses ternary secrets. Most significantly, for this work, the fatigue relation's quadratic nature allows for parameters reaching the required security level without needing to increase the ring dimension used in our voting protocol (which would significantly impact performance).

A New Lattice-Based E-Voting Design from NTRU. Our main contribution is the presentation of a new lattice-based e-voting protocol following the standard mix-and-decrypt framework which also supports general ballots. Our design combines our NTRU-based verifiable mix net and distributed decryption schemes. Moreover, we call on our analysis of the NTRU problem to choose fine-tuned concrete parameters. Crucially, when choosing the NTRU secret keys, we can drop the ring dimension down to 2048 from 4096 and modulus down to 59 bits from 78 bits as used in [ABGS23] whilst maintaining a 128-bit security level. This would not have been possible without the quadratic nature of the fatigue relation. Furthermore, we provide an efficient C++-implementation of our design.

Overall, we reduce the voting protocol's communication by 2.5× and computation by 2× over [ABGS23], see Section 5 for more details. It is interesting to note that, when compared to their classically-secure counterparts, post-quantum secure replacements typically come with a 30× communication cost (e.g. ECDH vs Kyber). Comparing our voting scheme to ElGamal-based schemes often used in practice, we incur a cost of at most 20× in ciphertext size, suggesting that our design may be approaching what can be optimally achieved.

Scheme	Ciphertexts	Shuffle	Dist. Dec.	Total
[ABGS23] [KB]	80	370	157	2188
Our [KB]	15	130	85	875
[ABGS23] [ms]	0.74	261	138	1182
Our [ms]	0.20	62	328	576

Table 1. Per vote comparison to [ABGS23] of ciphertexts, shuffle proofs, decryption proofs, and overall with four servers. Shuffles are sequential, while decryption is parallel.

1.2 Related Works

Lattice-Based Electronic Voting. In [ABGS23], the authors provide a verifiable mix-net and verifiable distributed decryption protocol based on BGV, showing for the first time that lattice-based electronic voting can be practical for real-world systems. We build directly upon their framework and conduct a more detailed comparison in Section 5. This work utilises the verifiable shuffle of known commitment openings by [ABG$^+$21]; a building block we adopt. del Pino et al. [dLNS17] gives a practical scheme based on homomorphic counting, but it does not scale well for systems with more complex ballots.

A shuffle by [CMM19] was implemented in [FWK21]; however, it is less efficient than [ABGS23]. More theoretical works include [HMS21a], [Str19], and [CGGI16], but none of these are efficient enough to be considered for practical deployment. Moreover, [CMM19, FWK21, HMS21b] do not consider the decryption of ballots, which would heavily impact the parameters of the protocols in practice. Finally, [BHM20] gives a fast decryption mix-net, but it cannot achieve universal verifiability and is thus unsuitable for real-world elections.

NTRU Cryptanalysis. The most relevant work analysing NTRU fatigue is that of Ducas and Van Woerden [Dv21]. It is important to acknowledge that this sits atop a line of work in recent years. The concurrent works [ABD16, CJL16b] showed, for the first time, that NTRU security is more subtle than simply finding a notably short vector in a lattice. These works exploit the specific algebraic structure of the NTRU lattice to gain an advantage on standard lattice reduction for so-called 'overstretched' parameter regimes.

This work was closely followed by Kirchner and Fouque [KF17], who showed that improved attacks were, in fact, only due to the geometric existence of an unusually dense sublattice of large dimension within the NTRU lattice. Moreover, their analysis concludes that q larger than $d^{2.783+o(1)}$ already lies in the overstretched range (for ternary secrets). This bound was improved upon by the work of [Dv21] as discussed in Section 4.

2 PRELIMINARY

Here we detail the essential tools employed in our constructions. We recall standard lattice results and necessary cryptographic building blocks. We begin with some notation.

Notation. For a set S and distribution (or algorithm) D, "$\leftarrow S[\rho]$" and"$\leftarrow D[\rho]$" denote the processes of uniformly sampling from S with randomness ρ and sampling from (or executing) D with randomness ρ, respectively. We denote by $\mathsf{Perm}[i]$ the set of permutations of the integers $\{1, ..., i\}$. For column vectors $\mathbf{a}$ and $\mathbf{b}$, $[\mathbf{a} \| \mathbf{b}]$ denotes the vertical concatenation. with an overload of notation, for two strings s and r over some alphabet, $s \| r$ denotes the concatenated string.

2.1 Lattices

The Ring $\mathbb{Z}[x]/(x^d+1)$. Consider the rings $R = \mathbb{Z}[x]/\phi$ and $R_q = \mathbb{Z}_q[x]/\phi$, where $\phi = (x^d+1)$ for d an integer power of 2 and q a prime. Elements in both rings are polynomials of degree at most $d-1$, with those in the latter ring having coefficients between $-(q-1)/2$ and $(q-1)/2$. We denote elements of $\mathbb{Z}$ and R by lower-case letters, vectors in R^k by bold lower-case letters, and matrices in $R^{(k\times\ell)}$ by bold upper-case letters. For a positive real σ, let $D_{\mathbb{Z}^d,\sigma}$ denote the discrete Gaussian distribution over $\mathbb{Z}^d$. To make the notation simple, we denote $a \leftarrow D_\sigma$ to mean that the coefficient vector of $a \in R_q$ is sampled from $D_{\mathbb{Z}^d,\sigma}$. For $a, b \in R$, we have that $\|ab\|_\infty \le \|a\|_1 \cdot \|b\|_\infty$ and $\|ab\|_\infty \le \|a\|_2 \cdot \|b\|_2$. Let S_ν denote the set of all elements $a \in R$ such that the absolute norm is $\|a\|_\infty \le \nu$.

We use the following standard results for Gaussian vectors:

Lemma 1 (Tail Bounds [MR04, Lyu12]). *For any real $t > 0$ and $t' > 1$, we have*

$$\Pr[x \leftarrow D_{\mathbb{Z}^n,\sigma} : \|x\|_\infty > t\sigma] < 2n \cdot 2^{-\frac{\log e}{2}\cdot t^2},$$
$$\Pr[x \leftarrow D_{\mathbb{Z}^n,\sigma} : \|x\|_2 > t'\sigma\sqrt{n}] < 2^{n\cdot\left(\frac{\log e}{2}(1-t'^2)+\log t'\right)}.$$

Rejection Sampling. In lattice-based cryptography in general, and in our zero-knowledge protocols in particular, we would like to output vectors $\mathbf{z} = \mathbf{y}+\mathbf{v}$ such that $\mathbf{z}$ is independent of $\mathbf{v}$, and hence, $\mathbf{v}$ is masked by the vector $\mathbf{y}$. Here, $\mathbf{y}$ is sampled according to a Gaussian distribution $\mathcal{N}_\sigma^k$ with standard deviation σ and we want the output vector $\mathbf{z}$ to be from the same distribution. The procedure is shown in Figure 1.

Here, $1/M$ is the probability of success, and M is computed as

$$\max \frac{\mathcal{N}_\sigma^k(\mathbf{z})}{\mathcal{N}_{\mathbf{v},\sigma}^k(\mathbf{z})} \le \exp\left[\frac{24\sigma\|\mathbf{v}\|_2 + \|\mathbf{v}\|_2^2}{2\sigma^2}\right] = M \qquad (1)$$

where we use the tail bound from Lemma 1, saying that $|\langle \mathbf{z}, \mathbf{v}\rangle| < 12\sigma\|\mathbf{v}\|_2$ with probability at least $1 - 2^{-100}$. Hence, for $\sigma = 11\|\mathbf{v}\|_2$, we get $M \approx 3$. This is the standard way to choose parameters, see e.g. [BLS19]. However, if the procedure is only done once for the vector $\mathbf{v}$, we can decrease the parameters slightly, to the cost of leaking only one bit of information about $\mathbf{v}$ from given $\mathbf{z}$.

In [LNS21], Lyubashevsky et al. suggest to require that $\langle \mathbf{z}, \mathbf{v}\rangle \ge 0$, and hence, we can set $M = \exp(\|v\|_2/2\sigma^2)$. Then, for $\sigma = 0.675\|\mathbf{v}\|_2$, we get $M \approx 3$. In Figure 1, we use the pre-determined bit b to denote if we only use $\mathbf{v}$ once or not, with the effect of rejecting about half of the vectors before the sampling of uniform value μ in the case $b = 1$ but allowing a smaller standard deviation.

The NTRU Problem. We give the historical presentation of the NTRU problem as it is more convenient for our analysis in Section 4. We note that some works refer to this problem as the 'search/decisional short polynomial ratio' problem. Furthermore, one can consider the so-called 'module' NTRU problem [CPS+20], which considers the ratio of matrices of polynomials $\mathbf{F}$ and $\mathbf{G}$. Our

```
Rej(z, v, b, M, σ)
  1. if $b = 1$ and $\langle \mathbf{z}, \mathbf{v} \rangle < 0$, return 1
  2. $\mu \xleftarrow{\$} [0, 1)$
  3. if $\mu > \frac{1}{M} \cdot \exp\left[\frac{-2\langle \mathbf{z}, \mathbf{v} \rangle + \|\mathbf{v}\|_2^2}{2\sigma^2}\right]$, return 1
  4. return 1
```

Fig. 1. Rejection Sampling.

analysis and applications can naturally be extended to the module setting, so for ease of presentation, we use the basic (polynomial) NTRU formulation [HPS98].

Definition 1 (Search/Decision NTRU). *Let $q > 2$ be a prime, d be the ring dimension, and $D_{\sigma_{\mathsf{NTRU}}}$ be a distribution over R_q. Sampling $(f, g) \leftarrow D^2_{\sigma_{\mathsf{NTRU}}}$ with rejection if f is not invertible in R_q, define $h = g/f \in R_q$. The search-$\mathsf{NTRU}_{q,d,\sigma_{\mathsf{NTRU}}}$ problem is, given h, to recover any rotation $(X^i f, X^i g)$ of the pair (f, g). The decision-$\mathsf{NTRU}_{q,d,\sigma_{\mathsf{NTRU}}}$ problem is, given h, to decide if h is computed as $h = g/f$ for $(f, g) \leftarrow D^2_{\sigma_{\mathsf{NTRU}}}$ or if h is sampled uniformly from R_q.*

The RLWE and RSIS Problems. We define the standard lattice-hardness problems over rings [Ajt96, Reg05, LPR10].

Definition 2 (Ring Learning with Errors). *Let $q > 2$ be a prime, d be the ring dimension, $D_{\sigma_{\mathsf{RLWE}}}$ be a distribution over R_q, and $\mathcal{A}$ a PPT algorithm that makes at most Q oracle queries. Then the advantage of $\mathcal{A}$ in solving the ring learning with errors $\mathsf{RLWE}_{d,q,Q,\sigma_{\mathsf{RLWE}}}$ problem is defined as*

$$\mathsf{Adv}^{\mathsf{RLWE}}_{d,q,Q,\sigma_{\mathsf{RLWE}}}(\mathcal{A}) = \left|\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{RLWE}}}(1^\lambda) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}_\$}(1^\lambda) \to 1]\right|,$$

where oracles $\mathcal{O}_{\mathsf{RLWE}}$ and $\mathcal{O}_\$$ are defined as

- *$\mathcal{O}_{\mathsf{RLWE}}$: Samples $a \leftarrow R_q$, $(s_1, s_2) \leftarrow D^2_{\sigma_{\mathsf{RLWE}}}$, and then output $(a, as_1 + s_2)$;*
- *$\mathcal{O}_\$$: Samples $(a, b) \leftarrow R_q \times R_q$, and then output (a, b).*

Definition 3 (Ring Short Integer Solutions). *Let $q > 2$ be a prime, d be the ring dimension, $\|\cdot\|$ a norm, and $\beta \in \mathbb{R}^+$ a positive integer. The $\mathsf{RSIS}_{d,q,\beta}$ problem is, given a uniformly random $a \in R_q$, find $s_1, s_2 \in R_q$ such that $as_1 + s_2 = 0 \in R_q$ and $\|s_1, s_2\| \leq \beta$.*

2.2 Building Blocks

NTRU Encryption. In this work, we will use the provably secure variant of the NTRU cryptosystem first presented by Steinfeld and Stehlé in [SS11]. This scheme relies on the hardness of both the RLWE and NTRU assumptions. Note we make two minor modifications to ensure perfectly correct decryption: (1) encryption randomness is sampled from a bounded distribution, and (2) the secret keys f and g are rejected unless their ℓ_2 norm is below a given bound. When sampled accordingly, this limitation has only a negligible effect on the completion probability of the key generation algorithm and the entropy of resulting keys.

Setup. Let $p \ll q$ be primes and d a power of two which define the rings R_p and R_q. Messages lie in R_p. Let $\sigma_{\mathsf{NTRU}} \in \mathbb{R}$ and $D_{\sigma_{\mathsf{NTRU}}}$ a discrete Gaussian distribution over R with standard deviation σ_{NTRU}, $t \in (1, 2]$ and $\nu \in \mathbb{N}$. Let the setup parameters be $\mathsf{sp} = (d, p, q, \sigma_{\mathsf{NTRU}}, t, \nu)$. The encryption scheme is described in Figure 2.

Key Generation $\mathsf{KeyGen}_{\mathsf{NTRU}}(\mathsf{sp})$. Given input $\mathsf{sp} = (d, p, q, \sigma_{\mathsf{NTRU}}, t, \nu)$:

1. $f, g \leftarrow D_{\sigma_{\mathsf{NTRU}}}$; if $f \notin R_q^\times$ or $f \not\equiv 1 \in R_p$, resample.
2. If $\|f\|_2, \|g\|_2 > t \cdot \sqrt{d} \cdot \sigma_{\mathsf{NTRU}}$, restart.
3. Return $\mathsf{sk} = f$, $\mathsf{pk} = h := g/f \in R_q$.

Encryption $\mathsf{Enc}_{\mathsf{NTRU}}(m, \mathsf{pk})$. Given message $m \in R_p$ and public key $\mathsf{pk} = h$:

1. Sample encryption randomness $s, e \leftarrow S_\nu$.
2. Return ciphertext $c = p \cdot (hs + e) + m \in R_q$.

Decryption $\mathsf{Dec}_{\mathsf{NTRU}}(c, \mathsf{sk})$. Given ciphertext c and key $\mathsf{sk} = f$:

1. Return message $m = (f \cdot c \mod q) \mod p$.

Fig. 2. Adapted `NTRUEncrypt` [SS11].

Lemma 2 (`NTRUEncrypt` Security). *Let $p \cdot d \cdot t \cdot \sigma_{\mathsf{NTRU}}(2\nu + 1/2) < \lfloor q/2 \rfloor$. Then the encryption scheme in Figure 2 is (perfectly) correct. Moreover, assuming the hardness of the $\mathsf{NTRU}_{q,d,\sigma_{\mathsf{NTRU}}}$ and $\mathsf{RLWE}_{d,q,Q,\chi}$ problems, the scheme is* IND-CPA *secure.*

The BDLOP Commitment Scheme. Here we recall the BDLOP commitment scheme from [BDL+18]. For simplicity, we present the scheme instantiated over rings instead of modules, committing to only one ring element at a time:

$\mathsf{Setup}(1^\lambda)$: On input a security parameter λ, samples uniformly random a_1, a_2, a_3 from R_q and outputs the public commitment key $\mathsf{pk_C}$ defined as:

$$\mathsf{pk_C} = \begin{bmatrix} \boldsymbol{a}_1 & 0 \\ \boldsymbol{a}_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_3 & 1 \end{bmatrix}.$$

$\mathsf{Com}(\mathsf{pk}_C, x)$: On input a public commitment key $\mathsf{pk_C}$ and an element x in R_q, samples a vector $\boldsymbol{r} \in R_q^3$ such that $\|\boldsymbol{r}\|_\infty \le B_{\mathsf{Com}}$, and computes the commitment as:

$$\mathsf{com} = \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{bmatrix} = \begin{bmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_3 & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ x \end{bmatrix} = [\![x]\!].$$

It outputs the commitment com and the opening $d = (x, \boldsymbol{r}, 1)$.

$\mathsf{Open}(\mathsf{pk}_C, \mathsf{com}, d)$: On input a public commitment key pk_C, the commitment com and the opening $d = (x, \boldsymbol{r}, f)$ where $f \in \bar{\mathcal{C}}$. It verifies:

$$f \cdot \mathsf{com} \overset{?}{=} \begin{bmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_3 & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ f \cdot x \end{bmatrix},$$

and $\forall i \in [3]\colon \|r_i\|_2 \overset{?}{\leq} 4 \cdot \sigma_{\mathsf{Com}} \sqrt{d}$. It outputs 1 if the relations hold and 0 otherwise.

The BDLOP commitment scheme is hiding if the RLWE problem is hard for vectors of ℓ_∞ norm B_{Com} over a lattice of dimension $2 \cdot d$. Furthermore, the scheme is binding if the RSIS problem is hard for vectors of ℓ_2 norm $16\sigma_{\mathsf{Com}}\sqrt{\kappa d}$ over a lattice of dimension $2 \cdot d$ [BDL+18].

Amortized Proofs of Boundedness. To prove boundedness on the large noise drowning terms E_i in our voting scheme, we define Π_{BND} to be a slightly adapted version of the Π_{SMALL} protocol used to prove boundedness of very small values (see Section 3.4 for full details). The main idea of Π_{BND} is that we use bit-decomposition of the integers E_{ij} to produce a long vector with small entries. This allows for the application Π_{SMALL} to prove an exact bound on E_i.

The previous work by Aranha et al. [ABGS23] used the amortised relaxed proofs by Baum et al. [BBC+18] to get smaller proof sizes of the cost of slightly increasing the overall parameters of the voting scheme because of the slack inherent in the proof system. In practice this leads to a slightly larger modulus q but have no impact on the ring dimension d. However, in our setting, we get better parameters in practice for the whole scheme when giving exact proofs of boundedness, even though the proofs themselves are larger. The exact relation for the proof system, with batch size ℓ' and secret vectors bounded in the ℓ_∞ norm by B_{Drown}, is:

$$\mathcal{R}_{\text{BND}} := \left\{ (x, w) \middle| \begin{array}{c} x := (\mathsf{pk}_C, \{\mathsf{com}_i\}_{i \in [\ell']}) \ \wedge w := (\{d_i = (u_i, \boldsymbol{r}_i, f_i)\}_{i \in [\ell']}) : \\ \forall i \in [\ell'] : \ \|u_i\|_\infty \leq B_{\mathsf{Drown}} \ \wedge \ \mathsf{Open}(\mathsf{pk}_C, \mathsf{com}_i, d_i) \end{array} \right\}.$$

Since Π_{SMALL} is a proof system that scales with the number of possible values of the secret vectors, we use bit decomposition techniques to limit a blow-up in terms of running time, memory usage and proof size, to the cost of proving knowledge of longer secret vectors.

Any integer E between 0 and q can be represented in base b as $E = [b_0\, b_1\, ...\, b_\zeta] \circ [1\, b\, ...\, b^\zeta]$ for unique coefficients b_i between 0 and $b-1$ and $\zeta = \lceil \log_b q \rceil - 1$ where $\circ$ is the dot product. This can be naturally extended to vectors, matrices and modules, particularly for our commitment matrix A. Since the commitment randomness is already short, we only need to decompose the last element in the secret vector, and we can do so in the following way (note that we abuse notation, where after $(*)$ the elements before $|$ are in R_q and the elements after are in $\mathbb{Z}_q$, but any element in R_q can be represented in $\mathbb{Z}_q^d$):

$$\mathbf{A}_{ij}\mathbf{s}_{ij} = \begin{bmatrix} 1 & a_{1,1} & \mathbf{a}_{1,2} & |0 \\ 0 & 1 & \mathbf{a}_{2,2} & |1 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{E_{ij}} \\ E_{ij} \end{bmatrix}$$

$$\overset{(*)}{=} \begin{bmatrix} 1 & a_{1,1} & \mathbf{a}_{1,2} & |0 \dots 0 \\ 0 & 1 & \mathbf{a}_{2,2} & |1 \dots b^\zeta \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{E_{ij}} \\ E_{0ij} \\ \vdots \\ E_{\zeta ij} \end{bmatrix} = \bar{\mathbf{A}}_{ij}\bar{\mathbf{s}}_{ij}.$$

Here, the ring element E_{ij} is decomposed, and elements $E_{0ij}, \dots, E_{\zeta ij}$ have integer values between 0 and $b-1$. We note that these statements are equivalent to prove, and that the length of $\bar{\mathbf{A}}_{ij}$ is $d(k+\zeta+1)$ over $\mathbb{Z}_q$ instead of $d(k+2)$.

Finally, we use the Π_{SMALL} protocol to prove ternary secret values as above but with a tweak: the public matrix input to the protocol is $\bar{\mathbf{A}}_{ij}$ instead of $\mathbf{A}_{ij}$, and we change the coefficient values that we are checking for in the proof. For the first $d \cdot k$ values we are checking for $(0, 1, -1)$ coefficients but for the next $d(\zeta+1)$ values we are checking for $(0, 1, 2)$ coefficients instead (this is a small tweak of line 3 in [ABGS23, Figure 5] that does not impact the performance of the protocol in any way, these values are initially arbitrary to the proof system). Since the other secret parts are ternary, we have that $\zeta = \lceil \log_3 B_{\mathsf{Drown}} \rceil - 1$.

We refer the reader to Section 3.4, where the actively-secure scheme is presented, for definitions of the remaining zero-knowledge proofs used in our construction.

3 THE VOTING SCHEME

A *cryptographic voting scheme* is usually defined in terms of the algorithms for the tasks of election setup, casting ballots, and counting cast ballots. To accurately model the counting process, we need algorithms for shuffling and distributed decryption. To make such a scheme verifiable (actively secure), we additionally need a mechanism by which to verify that the encryption, shuffling, and decryption algorithms are computed honestly. In this section, we present an NTRU-based voting protocol in the well-established ‘mix-and-decrypt’ paradigm, comprising new verifiable distributed decryption and mix-net protocols.

3.1 Voting Overview

Setup Phase. A trusted party runs the key generation algorithm for the PKE scheme with distributed decryption. In this work, we will assume a trusted key generation and leave the design of a distributed key generation algorithm for NTRU to future work, as this is common for voting schemes. The generated public parameters sp are given to every participant, while the decryption key shares dk_j are distributed amongst the decryption servers.

Casting Phase. Each voter instructs their voting device to cast their chosen ballot. The device encrypts the ballot under the public key pk to create a ciphertext c, and it computes a *ballot proof*. The standard way to do this is to use a verifiable encryption scheme such as the one presented in [LNP22], proving that the submitted ciphertext contains a genuine ballot in zero-knowledge.

Counting Phase. This is divided into three sequential processes. First, encrypted ballots are passed through a series of shuffle servers.

The ξ_1 shuffle servers $\mathcal{S}_1, ..., \mathcal{S}_{\xi_1}$ consecutively run the shuffle algorithm of the set of encrypted ballots $\{c_i^{(k-1)}\}$, passing the shuffled and re-encrypted ballots $\{c_i^{(k)}\}$ to the next shuffle server. They also generate a shuffle proof which anyone can verify. We may refer to this whole shuffle process as the *mix-net*.

Each of the ξ_2 decryption servers $\mathcal{D}_j$ receives the output of each shuffle server and verifies the corresponding shuffle proofs. Only after verifying each proof does a decryption server begin decryption. $\mathcal{D}_j$ then computes a set of partial decryption shares $\{\mathsf{ds}_{ij}\}$, one for each of the ciphertexts. Finally, it creates a proof of decryption to guarantee that it computed its shares correctly. Each decryption server passes its shares to the combining algorithm Comb.

The Comb algorithm performs the task of recovering the ballots. Having received all decryption shares from decryption servers, the Comb algorithm verifies the decryption proofs. If all decryption proofs are verified, it recovers the ballots by combining the decryption shares.

A schematic of these processes, in the malicious setting, is shown in Figure 3. This figure is adapted from [ABGS23] and shows the voting protocol beginning with input of a set of encrypted ballots and finishing with a set of ballots in plaintext. We note that some works consider an auditor whose role is to verify the processes at each step by checking the proofs provided. This is somewhat of a stylistic design choice. For the purposes of this paper, it is useful to think of the proofs as providing verifiability of each phase by any third party and by the component servers before carrying out their roles.

3.2 Passively Secure Scheme

Here we present our passively secure voting scheme. Whilst our ultimate goal is to give a verifiable (actively secure) voting scheme, we first isolate the core, passively secure skeleton for clarity of presentation. We begin by defining the algorithms and syntax of this construction.

Definition 4 (Passively Secure Voting Scheme). *Let τ be the number of voters, ξ_1 the number of shuffle servers, and ξ_2 the number of decryption servers.*

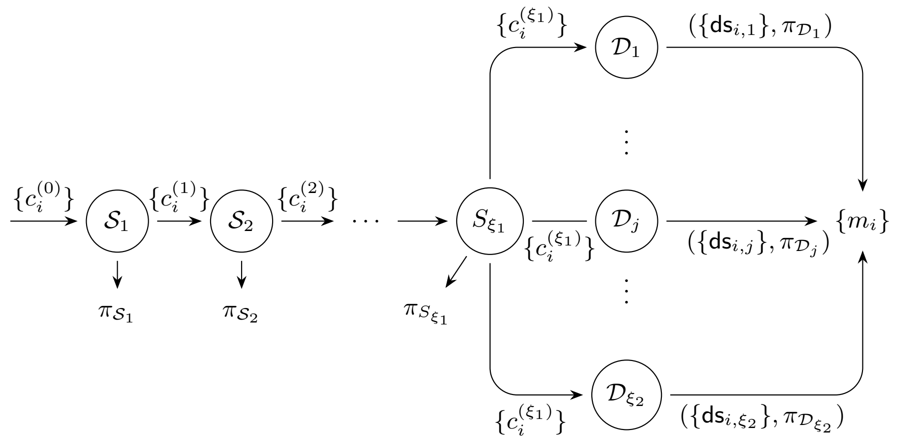

Fig. 3. The voting protocol with verifiable mix-net and distributed decryption, adapted from [ABGS23, Figure 1].

A passively secure cryptographic voting scheme Π_{PVote} *consists of five algorithms* $(\mathsf{KeyGen}, \mathsf{Cast}, \mathsf{Shuffle}, \mathsf{DDec}, \mathsf{Comb})$.

$\mathsf{KeyGen}(\mathsf{sp}) \to (\mathsf{pk}, \mathsf{sk}, \{\mathsf{dk}_j\}_{j\in[\xi_2]})$: *On input setup parameters* sp*, it returns a public encryption key* pk*, a secret key* sk *and a set of* ξ_2 *secret decryption key shares* $\{\mathsf{dk}_j\}_{j\in[\xi_2]}$.

$\mathsf{Cast}(\mathsf{pk}, v) \to c$: *On input a public key* pk *and vote* v *it returns an encrypted ballot* c.

$\mathsf{Shuffle}(\{c_i\}_{i\in[\tau]}) \to \{\hat{c}_i\}_{i\in[\tau]}$: *On input a set of encrypted ballots* $\{c_i\}_{i\in[\tau]}$ *it returns a set of encrypted ballots* $\{\hat{c}_i\}_{i\in[\tau]}$.

$\mathsf{DDec}_j(\{c_i\}_{i\in\tau}, \mathsf{dk}_j) \to \{\mathsf{ds}_{i,j}\}$: *On input a set of encrypted ballots* $\{c_i\}_{i\in[\tau]}$ *and a decryption key* dk_j*, it returns a set of decryption shares* $\mathsf{ds}_j = \{\mathsf{ds}_{i,j}\}_{i\in[\tau]}$.

$\mathsf{Comb}(\{c_i\}_{i\in[\tau]}, \{\mathsf{ds}_j\}_{j\in[\xi_2]}) \to \{v\}_{i\in[\tau]}$: *On input a set of encrypted ballots* $\{c_i\}_{i\in[\tau]}$ *and a set of decryption shares* $\{\mathsf{ds}_j\}_{j\in[\xi_2]}$*, it outputs a set of votes* $\{v\}_{i\in[\tau]}$.

We instantiate the algorithms, present our passively secure voting scheme and give an overview in Figure 4.

Setup. Let $p \ll q$ be primes and d a power of two which define the rings R_p and R_q. Votes lie in R_p. Let $\sigma_{\mathsf{NTRU}}, B_{\mathsf{Dec}}, B_{\mathsf{Drown}} \in \mathbb{R}^+$, $t \in (1, 2]$, and $\nu, \tau, \xi_1, \xi_2 \in \mathbb{N}$. Let $\mathsf{sp} = (d, p, q, \sigma_{\mathsf{NTRU}}, t, \nu, \tau, \xi_1, \xi_2)$.

3.3 Actively secure scheme

We present our actively secure (verifiable) voting scheme. This can be seen as a natural extension of the passive protocol Π_{PVote} by adding verifiability to the

$\mathsf{KeyGen}(\mathsf{sp})$. On input system parameters sp:

1. $(\mathsf{sk} = f, \mathsf{pk} = h) \leftarrow \mathsf{KeyGen}_{\mathsf{NTRU}}(d, p, q, \sigma_{\mathsf{NTRU}}, t)$.
2. For $j \in [\xi_2 - 1]$, $\mathsf{dk}_j \leftarrow \mathcal{U}(R_q)$ and set $\mathsf{dk}_{\xi_2} = \mathsf{sk} - \sum_{j=1}^{\xi_2-1} \mathsf{dk}_j \mod q$.
3. Return $(\mathsf{pk}, \mathsf{sk})$ and key shares $\{\mathsf{dk}_j\}_{j\in[\xi_2]}$.

$\mathsf{Cast}(\mathsf{pk}, v)$. On input the public key pk and a vote $v \in R_p$:

1. Compute $c = \mathsf{Enc}_{\mathsf{NTRU}}(\mathsf{pk}, v)$.
2. Return encrypted ballot c.

$\mathsf{Shuffle}(\{c_i\}_{i\in[\tau]})$. On input encrypted ballots $\{c_i\}_{i\in[\tau]}$:

1. For each $i \in [\tau]$, compute $c'_i = \mathsf{Enc}_{\mathsf{NTRU}}(\mathsf{pk}, 0)$.
2. For each $i \in [\tau]$, compute $\hat{c}_i = c_i + c'_i \mod q$.
3. Sample a random permutation $\pi \leftarrow \mathsf{Perm}[\tau]$.
4. Return re-encrypted ballots $\{\hat{c}_{\pi(i)}\}_{i\in[\tau]}$.

$\mathsf{DDec}_j(\{c_i\}_{i\in\tau}, \mathsf{dk}_j)$. On input a set of encrypted ballots $\{c\}_{i\in\tau}$ and a decryption key share dk_j:

1. For each $i \in [\tau]$, sample $E_{ij} \leftarrow S_{B_{\mathsf{Drown}}}$ and compute share $\mathsf{ds}_{ij} = \mathsf{dk}_j \cdot c_i + p \cdot E_{ij} \mod q$.
2. Return the set of decryption shares $\mathsf{ds}_j = \{\mathsf{ds}_{ij}\}_{i\in[\tau]}$.

$\mathsf{Comb}(\{c_i\}_{i\in[\tau]}, \{\mathsf{ds}_j\}_{j\in[\xi_2]})$. On input encrypted ballots $\{c_i\}_{i\in[\tau]}$ and decryption shares $\{\mathsf{ds}_j = \{\mathsf{ds}_{ij}\}_{i\in[\tau]}\}_{j\in[\xi_2]}$:

1. For each $i \in [\tau]$, $v_i = \left(\sum_{j\in[\xi_2]} \mathsf{ds}_{ij} \mod q\right) \mod p$.
2. Return the set of votes $\{v_i\}_{i\in[\tau]}$.

Fig. 4. The passively-secure voting scheme $\mathit{\Pi}_{\mathsf{PVote}}$.

shuffle and distributed decryption processes. This is done by applying the zero-knowledge proofs of Section 3.4 so that the outputs of $\mathsf{Shuffle}_{\mathsf{A}}$ and $\mathsf{DDec}_{\mathsf{A}}$, now include a proof of shuffle $\pi_{\mathcal{S}}$ and a proof of decryption $\pi_{\mathcal{D}}$ respectively.

Now, any third party can verify that the processes of shuffling and distributed decryption were carried out correctly without compromising the privacy or integrity of the voting system. As usual, we assume a trusted dealer for key generation and leave the construction of an NTRU-based distributed key generation for other applications to future work.

This construction implicitly defines a verifiable mix-net with distributed decryption from NTRU. We consider these of independent interest and give an overview as stand-alone protocols.

Verifiable Shuffle. Our aim here is, given a set of input ciphertexts, to generate a new set of ciphertexts that decrypts to the same set of plaintexts. Crucially, input-output ciphertext correspondence must be obscured. Additionally, we would like any third party to verify that this process has been performed correctly without compromising the privacy of the mix.

For this, we apply the proof of [ABG⁺21], which allows one to prove a shuffle of openings of the lattice commitments in Section 2.2. We denote this proof system $\mathit{\Pi}_{\mathrm{SHUF}}$. Since NTRU ciphertexts only contain a single element, we can import their scheme without modification where the committed messages are ciphertexts. We also employ the $\mathit{\Pi}_{\mathrm{SMALL}}$ proof systems described in Section 3.4 to prove that the new ciphertext noise is sufficiently bounded. At a high level, our verifiable shuffle of NTRU ciphertexts $c_1, ..., c_\tau$ re-randomises the input ciphertexts and then permutes their order where the permutation is only known to the shuffle server:

1. The shuffle server creates encryptions $c'_1, ..., c'_\tau$ of 0 and commits to these as $[\![c'_i]\!]$ for each $i \in [\tau]$. Run the $\mathit{\Pi}_{\mathrm{SMALL}}$ protocol to prove that each committed ciphertext is honestly computed.
2. Adding the original ciphertexts c_i to these commitments homomorphically yields commitments $[\![\hat{c}_i]\!]$ to ciphertexts with the same plaintext as in $c_1, ..., c_\tau$, now with fresh randomness.
3. The server now reveals the openings $\hat{c}_i$ in a randomly permuted order and runs the $\mathit{\Pi}_{\mathrm{SHUF}}$ protocol to prove that these are indeed a permutation of the correct openings of the commitments.

We note that verification of the shuffle proof should be done before any ballot decryption begins. This can be seen as a first step of the DDec algorithm or as part of a global verification process carried out by an auditor. For simplicity of presentation and since this is covered in [ABGS23], we omit this from the full protocol.

Verifiable Distributed Decryption. Our aim here is, given a set of input ciphertexts, to generate a set of decryption shares so that we can extract the encrypted plaintexts when all the shares are combined. Furthermore, each decryptor must prove that they decrypted their decryption share correctly using their secret key share. Therefore, in the active setting, the public key additionally contains a commitment $[\![\mathsf{dk}_j]\!]$ to each secret key share dk_j, and each decryptor holds an opening to exactly one of the commitments. The verifiable distributed decryption protocol works as follows:

1. For each $i \in [\tau]$, the decryptor samples a noise value $E_{ij} \leftarrow S_{B_{\mathsf{Drown}}}$, computes a decryption share $\mathsf{ds}_{ij} = \mathsf{dk}_j \cdot c_i + p \cdot E_{ij}$ and commits to the noise as $[\![E_{ij}]\!]$.
2. For each $i \in [\tau]$, it uses the $\mathit{\Pi}_{\mathrm{Lin}}$ protocol to prove that the linear decryption equation above is computed honestly with respect to $[\![\mathsf{dk}_j]\!]$ and $[\![E_{ij}]\!]$.
3. For each $i \in [\tau]$, it uses the $\mathit{\Pi}_{\mathrm{Bnd}}$ protocol to prove that $[\![E_{ij}]\!]$ is an honestly created commitment and the committed value is bounded by B_{Drown}.

We wish to emphasise the importance of our $\mathit{\Pi}_{\mathrm{Bnd}}$ proof system. $\mathit{\Pi}_{\mathrm{Bnd}}$ is a modification of the ZKP by Bootle et. al in [BLNS21]). Crucially, it proves a tight bound on the size of the noise downing term E_{ij}. To our knowledge, this is the first exact ZKP of a ‘large’ secret vector for lattice relations and may be of independent interest. Though a more costly proof, proving an exact bound on E_{ij} will lead to more efficient global parameters in Section 5.

Setup. Let $p \ll q$ be primes and d a power of two which define the rings R_p and R_q. Votes lie in R_p. Let $\sigma_{\mathsf{NTRU}}, B_{\mathsf{Dec}}, B_{\mathsf{Drown}}, B_{\mathsf{Com}}$, $B_{\mathsf{Small}} \in \mathbb{R}^+$, $t \in (1,2]$, and $\nu, \tau, l, \xi_1, \xi_2 \in \mathbb{N}$. Let $\mathsf{sp} = (d, p, q, \sigma_{\mathsf{NTRU}}, t, \nu, \tau, l, \xi_1, \xi_2, B_{\mathsf{Dec}}, B_{\mathsf{Drown}}, B_{\mathsf{Com}})$.

$\mathsf{KeyGen_A}(\mathsf{sp})$. On input system parameters sp:

1. Run $\big((\mathsf{pk}, \mathsf{sk}), \{\mathsf{dk}_j\}_{j \in [\xi_2]}\big) \leftarrow \mathsf{KeyGen}(\mathsf{sp})$.
2. For $j \in [\xi_2]$, compute the commitments and openings $(\llbracket \mathsf{dk}_j \rrbracket, \boldsymbol{r}_{\mathsf{dk}_j}) \leftarrow \mathsf{Com}(\mathsf{dk}_j)$.
3. Return $\mathsf{pk_A} = (\mathsf{pk}, \llbracket \mathsf{dk}_1 \rrbracket, ..., \llbracket \mathsf{dk}_{\xi_2} \rrbracket)$, $\mathsf{sk_A} = \mathsf{sk}$, and key shares $\{\mathsf{dk}_{A,j} = \big(\mathsf{dk}_j, \boldsymbol{r}_{\mathsf{dk}_j}\big)\}_{j \in [\xi_2]}$.

$\mathsf{Cast_A}(\mathsf{pk_A}, v)$. On input a public key pk_A and a vote v in R_p, retrieving pk from pk_A:

1. Return $c \leftarrow \mathsf{Cast}(\mathsf{pk}, v)$.

$\mathsf{Shuffle_A}(\{c_i\}_{i \in [\tau]})$. On input a set of encrypted ballots $\{c_i\}_{i \in [\tau]}$:

1. For $i \in [\tau]$, compute $c'_i \leftarrow \mathsf{Enc_{NTRU}}(\mathsf{pk}, 0)$ using encryption randomness (s'_i, e'_i).
2. For $i \in [\tau]$, commit to c'_i as $\mathsf{com}'_i := \llbracket c'_i \rrbracket \leftarrow \mathsf{Com}(\mathsf{pk_C}, c'_i)$ where $\boldsymbol{r}_{c'_i}$ is the commitment randomness used. Then denoting
$$\mathbf{A}_{\mathsf{M}} = \begin{bmatrix} 1 & a_{1,1} & a_{1,2} & 0 & 0 \\ 0 & 1 & a_{2,2} & ph & p \end{bmatrix},$$
and $\mathbf{s}_{c'_i} = [\boldsymbol{r}_{c'_i}, s'_i, e'_i]^T$ compute $\pi_{\mathsf{Small}_i} \leftarrow \Pi_{\text{SMALL}}$ for matrix $\mathbf{A}_{\mathsf{M}}$, input vector $\mathbf{s}_{c'_i}$, targets com'_i, and bound B_{Small}. Set $\pi_{\mathsf{Small}} := \{\pi_{\mathsf{Small}_i}\}_{i \in [\tau]}$.
3. For $i \in [\tau]$, compute $\hat{c}_i = c_i + c'_i$. Sample $\pi \leftarrow \mathsf{Perm}([\tau])$, and compute $\pi_{\mathsf{Shuf}} \leftarrow \Pi_{\text{SHUF}}$ with input commitments $\{\llbracket \hat{c}_i \rrbracket\}_{i \in [\tau]}$, randomness $\{\boldsymbol{r}_{c'_i}\}_{i \in [\tau]}$, ciphertexts $\{\hat{c}_i\}_{i \in \tau]}$, and permuted ciphertexts $\{\hat{c}_{\pi(i)}\}_{i \in [\tau]}$.
4. Return $\big(\{\hat{c}_{\pi(i)}\}_{i \in [\tau]}, \pi_{\mathcal{S}}\big)$, where $\pi_{\mathcal{S}} = \big(\{\mathsf{com}'_i\}_{i \in [\tau]}, \pi_{\mathsf{Small},} \pi_{\mathsf{Shuf}}\big)$.

Fig. 5. Π_{AVote} key generation, casting, and shuffle.

3.4 Zero-Knowledge Proofs

We present the proof systems needed in the actively secure voting protocol. We wish to highlight, in particular, the way in which we adapt the amortized proof of boundedness Π_{BND} as compared to previous works [ABGS23]. The crucial observation here is that whilst we get slightly larger proofs there, the *exact* guarantees provided by the proof allow for better parameters to be chosen for the overall voting scheme. This leads to a net reduction in communication cost.

$\mathsf{DDec}_{A,j}(\{c_i\}_{i\in[\tau]}, \mathsf{dk}_{A,j})$. On input a set of ciphertexts $\{c_i\}_{i\in[\tau]}$ and decryption key share $\mathsf{dk}_{A,j} = \big(\mathsf{dk}_j, \boldsymbol{r}_{\mathsf{dk}_j}\big)$:

1. For $i \in [\tau]$, sample $E_{ij} \leftarrow S_{B_{\mathsf{Drown}}}$, and compute $\mathsf{ds}_{ij} = \mathsf{dk}_j \cdot c_i + p \cdot E_{ij}$.
2. For $i \in [\tau]$, compute $\big([\![E_{ij}]\!], \boldsymbol{r}_{E_{ij}}\big) \leftarrow \mathsf{Com}(E_{ij}, \mathsf{pk}_\mathsf{C})$ and use the Π_{LIN} protocol to compute a proof $\pi_{\mathsf{Lin_{ij}}}$ for the linear relation $\mathsf{ds}_{ij} = \mathsf{dk}_j \cdot c_i + p \cdot E_{ij}$.
3. Apply the amortized proof Π_{BND} from Section 3.4, to create a proof π_{Bnd} that, for all $i \in [\tau]$, $\|E_{ij}\|_\infty \leq B_{\mathsf{Drown}}$ and $\|\boldsymbol{r}_{E_{ij}}\|_\infty \leq B_{\mathsf{Com}}$.
4. Return $\mathsf{ds}_j := \big(\{\mathsf{ds}_{ij}\}_{i\in[\tau]}, \pi_{\mathcal{D}}\big)$, where $\pi_{\mathcal{D}} = \big(\{[\![E_{ij}]\!]\}_{i\in[\tau]}, \{\pi_{\mathsf{Lin_{ij}}}\}_{i\in[\tau]}, \pi_{\mathsf{Bnd}}\big)$.

$\mathsf{Comb}_A(\{c_i\}_{i\in[\tau]}, \{\mathsf{ds}_j\}_{j\in[\xi_2]})$. On input encrypted ballots $\{c_i\}_{i\in[\tau]}$ and decryption shares $\{\mathsf{ds}_j\}_{j\in[\xi_2]}$:

1. Parse ds_j as $\big(\{\mathsf{ds}_{ij}\}_{i\in[\tau]}, \pi_{\mathcal{D}_j}\big)$, and verify the proofs $\pi_{\mathsf{Lin_{ij}}}$ and $\pi_{\mathsf{Bnd},ij}$, returning $\perp$ if either fails to verify.
2. Compute
$$v_i = \Big(\sum_{j\in[\xi_2]} \mathsf{ds}_{ij} \mod q\Big) \mod p.$$
3. Return the set of votes $\{v_i\}_{i\in[\tau]}$.

Fig. 6. Π_{AVote} distributed decryption and combining.

Challenge Set. Let κ be such that $\binom{d}{\kappa} \cdot 2^\kappa > 2^\lambda$ and define $\mathcal{C}_\kappa = \{c \in R_q \mid \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\}$ and $\bar{\mathcal{C}}_\kappa = \{c - c' \mid c, c' \in \mathcal{C}_\kappa \wedge c \neq c'\}$.

Proof of Linearity. In Step 2 of DDec, we prove well-formedness of the linear decryption share. The protocol Π_{LIN} produces a proof that a committed value v is a multiple of another committed value u with respect to a public scalar g. In our setting, we will prove $[\![E_{ij} - p^{-1}\mathsf{ds}_{ij}]\!] = -p^{-1}\mathsf{c}_i[\![\mathsf{dk}_j]\!]$.

The exact relation for the proof system is:

$$\mathcal{R}_{\text{LIN}} := \left\{ (x, w) \;\middle|\; \begin{array}{c} x := (\mathsf{pk}_C, \mathsf{com}_u, \mathsf{com}_v, g) \;\wedge \\ w := (d_u = (u, \boldsymbol{r}_u, f_u), d_v = (v, \boldsymbol{r}_v, f_v)) : \\ u = g \cdot v \;\wedge\; \mathsf{Open}(\mathsf{pk}_C, \mathsf{com}_u, d_u) \;\wedge \mathsf{Open}(\mathsf{pk}_C, \mathsf{com}_v, d_v) \end{array} \right\}.$$

The proof of linearity π_{LIN} is computed as follows [BDL+18]:

1. Sample vectors $\boldsymbol{y}_u$ and $\boldsymbol{y}_v$ of length k over R_q according to $D_{\sigma_{\text{LIN}}}$ and compute $\boldsymbol{w}_u = \boldsymbol{a}_1 \cdot \boldsymbol{y}_u$ and $\boldsymbol{w}_v = \boldsymbol{a}_1 \cdot \boldsymbol{y}_v$ and $t = g \cdot \boldsymbol{a}_2 \cdot \boldsymbol{y}_u - \boldsymbol{a}_2 \cdot \boldsymbol{y}_v$.
2. Hash $(\boldsymbol{w}_u, \boldsymbol{w}_v, t)$ to c in $\mathcal{C}_\kappa$, and compute $\boldsymbol{z}_u = \boldsymbol{y}_u + c \cdot \boldsymbol{r}_u, \boldsymbol{z}_v = \boldsymbol{y}_v + c \cdot \boldsymbol{r}_v$.
3. Rejection sample with respect to $(\boldsymbol{y}_u, \boldsymbol{z}_u)$, and $(\boldsymbol{y}_v, \boldsymbol{z}_v)$. If it outputs 1 then output $\pi_{\text{LIN}} = (c, \boldsymbol{z}_u, \boldsymbol{z}_v)$ and otherwise restart by sampling new $(\boldsymbol{y}_u, \boldsymbol{y}_v)$.

The verifier checks if $\|\boldsymbol{z}_u, \boldsymbol{z}_v\|_2 \leq 2\sigma_{\text{LIN}}\sqrt{k \cdot d}$ and if the hash of $(\boldsymbol{a}_1 \cdot \boldsymbol{z}_u - c \cdot c_{u,1}, \boldsymbol{a}_1 \cdot \boldsymbol{z}_v - c \cdot c_{v,1}, g \cdot \boldsymbol{a}_2 \cdot \boldsymbol{z}_u - \boldsymbol{a}_2 \cdot \boldsymbol{z}_v + c_{v,2} + g \cdot c_{u,2})$ equals c. It outputs 1 if all checks verify, and otherwise it outputs 0.

Using the improved rejection sampling techniques from [LNS21], we set $\sigma_{\text{LIN}} = B_{\mathsf{Com}} \cdot \kappa\sqrt{d}$. The size of π_{LIN} is $2kd\log_2(4\sigma_{\text{LIN}})$ bits.

Proof of Shuffle. In Step 3 of the shuffle, we use Π_{SHUF} to prove that a set of committed values is a permutation of public values.

In our context, the committed values will be the $\hat{\mathsf{c}}_i$ values. These can be constructed by the verifier from the c_i and the $[\![\mathsf{c}'_i]\!]$. The public values are the $\hat{\mathsf{c}}_i$. The proof then convinces the verifier that output ciphertexts are a genuine permutation of the set of re-randomized input ciphertexts. The exact relation for the proof system is:

$$\mathcal{R}_{\text{SHUF}} := \left\{ (x,w) \;\middle|\; \begin{array}{l} x := (\{(\mathsf{com}_i, \bar{u}_i)\}_{i\in[\tau]}) \;\wedge w := (\{d_i = (u_i, \boldsymbol{r}_i, f_i)\}_{i\in[\tau]}, \rho) : \\ \rho \in \mathsf{Perm}[\tau] \;\wedge\; \forall i \in [\tau] :\; u_i = \bar{u}_{\rho(i)} \wedge\; \mathsf{Open}(\mathsf{pk}_C, \mathsf{com}_i, d_i) \end{array} \right\}.$$

The proof of shuffle π_{SHUF} is computed as follows [ABG+21, Section 4]:

1. Hash the statement to get a uniform value and then shift all commitments and messages to u'_i and $\bar{u}'_i$ (the commitments are additionally homomorphic).
2. For all $i \in [\tau - 1]$, sample random values θ_i and commit to random linear combinations of the form $[\![D_i]\!] = [\![\theta_{i-1} \cdot u'_i + \theta_i \cdot \bar{u}'_i]\!]$ (where $\theta_0 = \theta_\tau = 0$).
3. Hash the commitments to get a uniform challenge β. Then for all $i \in [\tau]$ compute s_i so that it solves the linear system with respect to β.
4. For all $i \in [\tau]$, compute proofs of linearity for the commitment equations of the form $[\![D_i]\!] = s_{i-1}[\![u'_i]\!] + s_i \cdot \bar{u}'_i$ (where $s_0 = \beta$ and $s_\tau = (-1)^\tau \beta$).

The verifier accepts if all proofs of linearity are valid. This proof π_{SHUF} consists of one ring element, one commitment and one proof of linearity per shuffled element. Using the proof of linearity π_{LIN} from above, the size of π_{SHUF} is $\tau d(2k \log_2(4\sigma_{\text{LIN}}) + 3\log_2 q)$ bits.

Amortized Proof of Shortness. In Step 2 of the shuffle, we use Π_{SMALL} to prove that we have committed to well-formed encryptions of zero. The protocol Π_{SMALL} produces a proof that a batch of equations $\boldsymbol{A}\boldsymbol{s}_i = \boldsymbol{t}_i$ for $i \in [\ell]$ is satisfied for a set of secret vectors $\boldsymbol{s}_i$ with ℓ_∞ norm bounded by ν. The exact relation for the proof system is:

$$\mathcal{R}_{\text{SMALL}} := \left\{ (x,w) \;\middle|\; \begin{array}{c} x := (\mathsf{pk}_C, \{\mathsf{com}_i\}_{i\in[\ell]}) \;\wedge w := (\{d_i = (u_i, \boldsymbol{r}_i, f_i)\}_{i\in[\ell]}) : \\ \forall i \in [\ell] :\; \|u_i\|_\infty \le \nu \;\wedge\; \mathsf{Open}(\mathsf{pk}_C, \mathsf{com}_i, d_i) \end{array} \right\}.$$

The proof of shortness π_{SMALL} is quite involved, combining error-correcting codes, Merkle trees, Lagrange interpolation and proximity testing, and we refer to [ABGS23] for details. The proof size, for batch size ℓ of ternary secret vectors, is given in [ABGS23, Equation (1)] as

$$(3vd + (3\ell + 2)\eta) \log_2 q + 2\lambda\eta(1 + \log_2 \gamma) \text{ bits},$$

using an $[\gamma, \mu, \iota]$ Reed-Solomon Code with code-length γ, message length μ and minimal distance ι where $\mu = d(k+2) + \eta \le \gamma < q$ for encoding randomness of length η. λ is the security parameter. The soundness of the proof is given as

$$2 \cdot \max\left\{ 2\left(\frac{\mu'}{\gamma - \eta}\right)^\eta, \frac{1}{q-\ell} + \left(1 - \frac{\mu' - \mu}{6\gamma}\right)^\eta,\; 2 \cdot \left(1 - \frac{2(\mu' - \mu)}{3\gamma}\right)^\eta, \frac{18\ell}{q-\ell} \right\},$$

for a choice of message length μ' such that $\mu \leq \mu' \leq \gamma < q$.

Amortized Proofs of Boundedness. In Step 3 of DDec, we use Π_{BND} to prove boundedness on noise drowning terms E_{ij}. The main idea is that we use a bit-decomposition E_i to produce a long vector with small entries, which allows for the application Π_{SMALL} to prove an exact bound on E. We define Π_{BND} to be an adapted version of the Π_{SMALL} protocol, see Section 2.2 for details.

3.5 Security Analysis

Shuffle Protocol. We analyse the security of the verifiable shuffle protocol implicitly defined by the tuple of algorithms $\Pi_{\mathsf{AShuf}} := (\mathsf{KeyGen_A}, \mathsf{Cast_A}, \mathsf{Shuffle_A}, \mathsf{Dec_{NTRU}})$. We say that Π_{AShuf} is *secure* if it satisfies the properties of shuffle completeness, shuffle soundness, and shuffle simulatability. We refer the reader to Appendix A for formal definitions of these notions.

Lemma 3 (**Π_{AShuf} Security).** *Suppose Π_{SMALL} and Π_{SHUF} are complete, knowledge sound, and HVZK, and that* Com *is hiding. Furthermore, suppose that* `NTRUEncrypt` *is IND-CPA secure and let the total noise, B_{Mix}, added to ciphertexts in the shuffle be such that $B_{\mathsf{Dec}} + B_{\mathsf{Mix}} \leq \lfloor q/2 \rfloor$. Then the verifiable shuffle Π_{AShuf} is secure.*

We sketch the proof. Since Π_{SMALL} and Π_{SHUF} are complete, the protocol will finish and the shuffle will verify correctly. Since $B_{\mathsf{Mix}} + B_{\mathsf{Dec}} \leq \lfloor q/2 \rfloor$, decryption will be correct. Thus, Π_{AShuf} is complete.

Now assume we have an adversary Adv breaking the knowledge soundness of the shuffle. That is, given a set of input ciphertexts, Adv can produce a new set of ciphertexts and a shuffle proof which verifies but the new ciphertexts do not decrypt to the original plaintexts. Without loss of generality, assume that ciphertext $\hat{c}_i$ decrypts incorrectly. By the knowledge soundness of π_{Small}, one can extract commitment randomness $\mathbf{s}_{c'_i} = [\mathbf{r}c'_i, s'_i, e'_i]^T$ such that $\mathbf{A}_{\mathsf{M}}\mathbf{s}_{c'_i} = \mathbf{t}'_i$. By knowledge soundness of π_{Shuf}, it is easy to check that one can extract a second vector $\tilde{\mathbf{s}}_{c'_i} = [\tilde{\mathbf{r}}c'_i, \tilde{s}'_i, \tilde{e}'_i]^T$ satisfying $\mathbf{A}_{\mathsf{M}}\tilde{\mathbf{s}}_{c'_i} = \mathbf{t}'_i$. Then incorrect decryption of $\hat{c}_i$ corresponds to either $\tilde{s}'_i \neq s'_i$ or $\tilde{e}'_i \neq e'_i$. In either case, we break the binding property of the BDLOP commitment scheme.

Finally, we can argue the shuffle simulatability in the standard way. We construct a simulator that, given a set of input ciphertexts and output ciphertexts from an honest shuffle, simulates the proofs π_{Small} and π_{Shuf} using the HVZK property of those systems and replaces the commitments to ciphertexts with commitments to zero. Simulatability then follows from the hiding property of the commitments and IND-CPA security of `NTRUEncrypt` .

Distributed Decryption Protocol. We now analyse the security of the PKE with distributed decryption implicitly defined by the tuple of algorithms $\Pi_{\mathsf{ADDec}} :=$ $(\mathsf{KeyGen_A}, \mathsf{Cast_A}, \mathsf{Dec_{NTRU}}, \mathsf{DDec_A},,$
$\mathsf{Comb_A})$. We say that Π_{ADDec} is *secure* if it satisfies the properties of IND-CPA

security, threshold correctness, threshold verifiability, and distributed decryption simulatability. For completeness, we give the full definitions of these notions in Appendix B. Since many of these properties rely on building blocks used in previous works, we provide a proof sketch here and refer the reader to [ABGS23] for the full arguments. We will however make parameter constraints explicit to aid in the performance analysis of Section 5.

Lemma 4 (Π_{ADDec} Security). *Let $B_{\mathsf{Drown}} = 2^{\sec}(B_{\mathsf{Dec}}/p\xi_2) < \lfloor q/2 \rfloor$. Suppose that Π_{LIN} and Π_{BND} are complete, knowledge sound, and HVZK, and* `NTRUEncrypt` *is IND-CPA secure. Then the verifiable, distributed decryption protocol Π_{ADDec} is secure.*

We sketch the proof. IND-CPA security of Π_{ADDec} follows trivially from the IND-CPA security of the underlying NTRU encryption scheme and the HVZK property of the proofs Π_{LIN} and Π_{BND}.

Examining the threshold correctness, define the predicate $P_{\mathsf{sk_A}}(\cdot)$ so that $P_{\mathsf{sk_A}}(c) = 1$ if and only if $\|\mathsf{sk_A} \cdot c\|_\infty < B_{\mathsf{Dec}}$. Then given a set of adversarially generated ciphertexts $\{c\}_{i\in[\tau]}$ satisfying $P_{\mathsf{sk_A}}(c_i) = 1$ for all $i \in [\tau]$, we have that $\left\|\sum_{j\in[\xi_2]} \mathsf{ds}_{ij}\right\|_\infty < q/2$ and so the Comb algorithm will return the correct decryption of c_i. The completeness of Π_{LIN} and Π_{BND} ensure that the arguments will be accepted, thus, Π_{ADDec} is threshold correct.

For threshold verifiability we also consider only ciphertexts such that $P_{\mathsf{sk_A}}(c) = 1$. Note that if Comb accepts a ciphertext for which decryption is incorrect then, for some j, no relation $\mathsf{ds}_{ij} = \mathsf{dk}_j \cdot c_i + pE_{ij}$ holds for an E_{ij} of norm at most B_{Drown}. This implies either an adversary against the soundness of Π_{LIN} or of Π_{BND}.

Finally, we describe a simulator for our distributed decryption. Firstly, let us replace commitments to E_{ij} with commitments to zero. This change is indistinguishable from the hiding property of the BDLOP commitment scheme. Next, one can simulate the proofs $\pi_{\mathsf{Lin}_{ij}}$ and π_{Bnd}, otherwise a distinguisher breaks the HVZK of the respective proof systems. Lastly, the simulatability of the ds_{ij} follows from the choice of sec which is chosen so that ds_{ij} is statistically indistinguishable from uniform.

4 NTRU HARDNESS

Before setting out to choose concrete parameters for the implementation of our new voting scheme, it became clear that we needed to better understand the hardness of the NTRU problem. This section contains that in-depth analysis which informs our parameter choices in Section 5.

Research on the security of the NTRU problem has revealed a significant improvement in the performance of lattice reduction algorithms when applied to NTRU lattices for so-called *overstretched* parameters. More precisely, analysis carried out over a series of works [ABD16, KF17, LW20] shows this weakening of NTRU occurs when the modulus q is very large compared to the ring dimension d

and when secrets are small. Naturally, these works seek to determine the turning point at which q becomes large enough for such attacks to apply. We refer to this as the *fatigue point*.

4.1 Extending the NTRU Analysis

Until recently, only an asymptotic result was known about the position of the fatigue point, determined by Kirchner and Fouque as $q = d^{2.783+o(1)}$ [KF17].

Ducas-van Woerden Analysis. In their recent paper [Dv21], Ducas and van-Woerden improve on the asymptotic result of Kirchner and Fouque, narrowing down the fatigue point for ternary secrets to $q = d^{2.484+o(1)}$. Building on this result, the authors perform an average-case analysis (rather than a worst-case bound) based on the volume of the relevant lattices and sublattices to arrive at a concrete prediction of the fatigue point. They identify two lattice reduction events that distinguish standard regimes from their overstretched counterparts to facilitate their analysis.

– Secret Key Recovery (SKR): The event in which a vector as short as the secret key is inserted into the lattice basis.
– Dense Sublattice Discover (DSD): The event in which a vector of the dense sublattice is inserted into the basis.

A DSD event has been shown to shortly precede SKR by a cascading of further DSD events or enabling decryption of fresh ciphertexts.

Through careful observation of the occurrence of these events, Ducas and van Woerden use their predictive model to determine the concrete fatigue point of NTRU with ternary secrets to be $q = 0.004 \cdot d^{2.484}$ for $d > 100$. One can use the scripts provided[3] in their work to estimate the concrete hardness of NTRU. We also affirm their predictive model by running real experiments on low-dimensional instances to confirm this relation.

Beyond Ternary Secrets. The reader may have noticed that the discussion of the fatigue point, thus far, only focuses on the modulus and dimension of the ring. Recalling Definition 1 reminds us that f and g need not always be ternary. Indeed, many NTRU-based constructions use secrets with non-ternary coefficients. Let us consider f and g generated according to a Gaussian distribution D_σ of standard deviation σ. For convenience, the analysis of [Dv21] models the ternary secret case by sampling $f, g \leftarrow D_\sigma$ with $\sigma^2 = 2/3$. Varying σ can model any secret key size, and thus we will herein consider that f and g are always sampled according to some Gaussian D_σ. The natural question arises:

Does the choice of (secret size) σ influence the fatigue point, and if so, what is its impact?

[3] See github.com/WvanWoerden/NTRUFatigue for their code.

To get some intuition on this, we recall the work of Steinfeld and Stehlé [SS11] in which the authors show how selecting σ sufficiently large gives rise to a public key $h = g/f$ that is statistically indistinguishable from uniform when f and g are sampled from D_σ. Moreover, they show that using such parameters allows one to remove the NTRU assumption from a proof of the NTRU cryptosystem. The σ needed for statistical security depends on the size of q and d. This suggests that fixing q and d and increasing σ makes the NTRU problem harder.

This observation goes some way to answering the first part of our question since it is clear that, for a sufficiently large σ, both SKR and DSD become ineffective.

Whilst using statistically uniform public keys provides peace of mind, this practice comes with significant efficiency losses. In addition to much larger key sizes, conditions for a cryptosystem's correctness can become much more constraining. Note that for correct decryption of the `NTRUEncrypt` cryptosystem defined in Figure 2, one needs the relation $\|p(gs + fe) + fm\|_\infty < q/2$ to hold. Clearly, using larger secrets f and g thus leads to less favourable parameters by pushing up the modulus q.

We, therefore, have a balancing act that needs to be performed when setting NTRU parameters; to keep parameters small whilst avoiding the attacks affecting overstretched regimes. Fortunately, the script provided in [Dv21] also allows for NTRU hardness estimations using any choice of σ though no analysis is performed in their work outside the ternary case. Nevertheless, their estimator provides a tool, much like the LWE estimator of Albrecht et al. [APS15], to analyse the concrete hardness of any given NTRU parameter set.

A More General Fatigue Relation. In our analysis, we are interested in answering the second part of the above question. In particular, we would like to know by *how much* an increase in NTRU secret size affects the position of the fatigue point for a given ring dimension.

A simple calculation, following the analysis of [Dv21], Section 3.2, confirms that the asymptotic relation $q = d^{\,2.484+o(1)}$ holds regardless of the value of σ. This suggests that if the value of σ plays a role in the concrete, average case relation, it manifests in the leading constant. We can thus infer that, for some function ψ and constant c, the fatigue point is given by

$$q = c \cdot \psi(\sigma) \cdot d^{\,2.484}.$$

In order to determine the nature of ψ, we consider a range of $\sigma \in [2, 2^2, \ldots, 2^{20}]$. For each σ, we perform a loglog-linear regression on the estimated fatigue points overall prime ring dimensions $199, \ldots, 499$. This mimics the calculations of [Dv21] used in the ternary case. For a full explanation of why this is a sensible range to examine, we refer the reader to Section 5.5 of that work.

Next, we consider the predicted fatigue points as a geometric series. This reveals the predicted average-case fatigue point to be

$$q = 0.0058 \cdot \sigma^2 \cdot d^{\,2.484}. \tag{2}$$

The precision of this relation across all σ considered is very high. Whilst we could extend this part of our analysis to larger σ, it is highly unlikely that, for cryptographic applications, one would need to take σ higher than 2^{20}. We note also that, setting $\sigma^2 = 2/3$, we recover the fatigue point determined for the ternary case [Dv21].

This gives a definitive answer to our question about the impact of σ on the fatigue point. To give more gravity to this prediction, we also run a series of experiments for computable ring dimensions to validate this estimate. The results are displayed in Figure 7.

We also give a second figure (Figure 8) in which we plot q/σ^2 along the vertical axis. This reveals the accuracy of the preceding constant by revealing how closely bunched the estimations and experiments are when normalised across varying σ. As was observed by Ducas and Woerden in the ternary case, the estimator is slightly pessimistic, predicting a fatigue point roughly 15% lower than the one suggested by real experiments. They give potential explanations for this discrepancy, pointing to the slope parameter used in the estimator, which is not well calibrated for such small block sizes. In practice, though, this small error only translates to a difference of 2-3 in the block size needed to run BKZ and thus hardly affects the predicted security. Importantly, our experiments show that this error remains constant even at larger moduli.

The Significance of σ^2. Having determined the impact of σ on the concrete fatigue point for NTRU, we reflect on the structure of Equation (2). As an illustrating example, let us return to the decryption correctness constraint for the NTRU cryptosystem. This can be written as $\sigma \cdot \mathcal{F}(p, d, \nu) < q$ for some function $\mathcal{F}$. Suppose for a given parameter set (d, q, σ), this constraint is satisfied, but the corresponding NTRU instance does not provide adequate security. Let us then increase q by a constant factor δ, say. According to the constraint, this gives room for an increase in σ to $\delta \cdot \sigma$. Then Equation (2) tells us that the new fatigue point for the set $(d, \delta \cdot q, \delta \cdot \sigma)$ increases by a factor of δ^2. The important observation here is that, while increasing q in the first move might weaken the NTRU instance, the same increase permitted for σ actually gives rise to a *net increase in the hardness of the instance.* In summary, Equation (2) tells us that it is possible to 'win' this cat-and-mouse game for NTRU that so often arises when setting lattice parameters.

We now consider how our analysis might be applied to existing works to refine parameter choices.

4.2 Implications for Existing Work

While the authors of [Dv21] note that parameters used in the NTRU-based NIST finalists are still secure to the degrees claimed, many works in the literature use different sets, some of which may fall foul of the dense sublattice attack, and thus, one needs to use the techniques described in the previous section to set parameters.

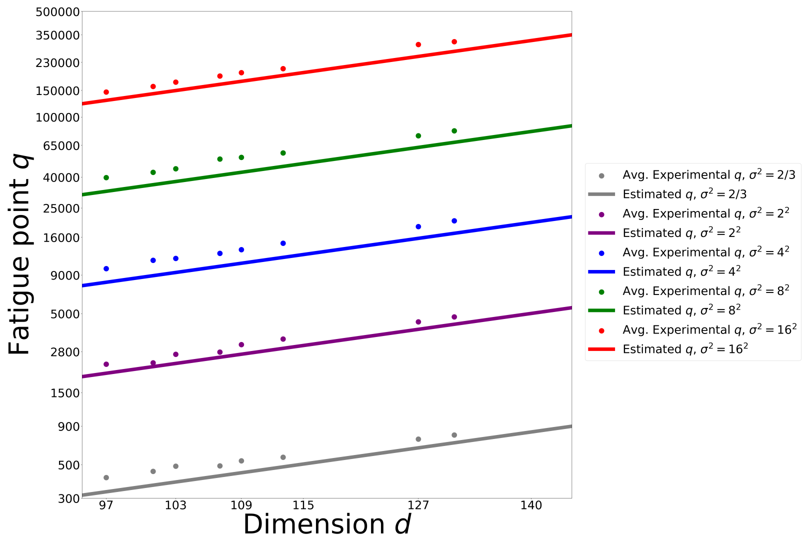

Fig. 7. Experimental fatigue point values for a range of σ, calculated using BKZ with 8 tours on matrix NTRU instances. The straight-colored lines show the estimated values using the (modified) estimator from [Dv21].

We examine an existing primitive in which the parameters fall short of providing claimed security levels. However, as suggested by Equation (2), we can carefully re-select parameters so that a small increase in the secrets yields the security bump-up needed.

Blind Signatures [dK22]. del Pino and Katsumata present a lattice-based (partially) blind signature using trapdoor sampling. In the (round optimal) construction given, a user passes the message to be signed in a *blind* way so that the signer does not learn the message they sign. This is done by committing to the message and then proving the well-formedness of this commitment. We will call this the *first flow*. The signer then creates an output passed back to the user (*second flow*). Finally, the user computes a signature for its original message using this response message from the signer.

In the first flow, Pino and Katsumata employ the NTRU-based linear homomorphic commitment scheme (LinHC) of [Kat21] to ensure the soundness and overall QROM security of the well-formedness proof. One must, therefore, choose parameters so that the relevant NTRU instance is hard. The choice of $d = 2048$, $q = 2^{66}$, and ternary NTRU secrets is informed by the constraint requiring straight-line extractability of the proof system. However, as we have observed, such large moduli run the risk of taking a parameter set into overstretched territory. Moreover, these values give rise to only 63 bits of security when run through the estimator of [Dv21] rather than 128.

To rectify this situation, there are two common strategies; either one can increase the ring dimension used throughout the scheme or use sufficiently large

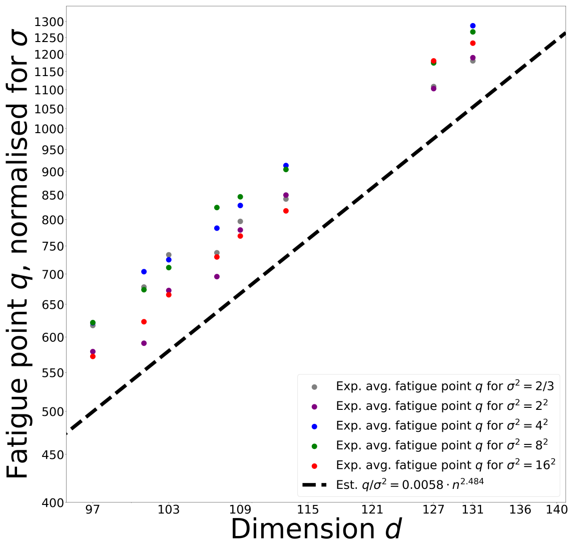

Fig. 8. Experimental values for q/σ^2 illustrate that the fatigue point, when adjusted for σ, is modeled by $q/\sigma^2 = 0.0058 \cdot d^{\,2.484}$.

NTRU secrets that the corresponding public key is *statistically* indistinguishable from uniform.

Doubling the ring dimension in [dK22] from 2048 to 4096 (to retain the implementation benefits of a power-of-two dimension) and computing the other parameters accordingly, 128 bits of security is reached at the cost of doubling the sign-request flow (69.2MB), doubling the returned 'pre-signature', and doubling the user's final signature size to 200 KB.

The alternative method, using a statistically uniform public commitment key turns out to be impossible whilst satisfying all parameter constraints simultaneously.

We now exhibit the benefits of the relation Equation (2), as revealed by our analysis, when applied to the problem of setting NTRU parameters. with the same ring dimension $d = 2048$, we increase σ_{NTRU} (secret size). This has the effect of pushing up the modulus needed to facilitate the straight-line extraction condition. The reader might observe that increasing q reduces the hardness of the problem again. However, Equation (2) reveals that it is possible to 'win' this cat-and-mouse game since the fatigue point increases *quadratically* with the size of the secrets. We thus propose the following parameters to ensure 128 bit security is reached:

$$q \approx 2^{74}, \quad p \approx 2^{41}, \quad \sigma_{\mathsf{NTRU}} = 13,$$

where p is the prime used to commit to the witness in the LinHC protocol. Fortunately, this change only has a small effect on the total communication cost. In the first flow, the user signing query increases from 34 MB to 35.4 MB, and the sizes of the user's pre-signature and final signature output are unchanged. This significantly improves the sizes that arise from changing the ring dimension and avoids doubling the final signature.

Summary. Simply increasing the size of the NTRU secrets may be all that is needed to ensure the correct security threshold is reached. In other settings, this also pushes up the modulus over which a scheme is defined, as in the examples above. However, the scheme may also rely on other hardness assumptions, such as RLWE, as in our voting scheme, defined over the same ring. Now, the RLWE problem may no longer be hard for the adjusted parameters and one may need to increase the ring *dimension* to find parameters for which both problems are hard. This can make what was an efficient scheme into one that cannot be deployed in practice.

Clearly, such balancing acts must be approached with a good understanding of the hardness of NTRU instances. We aim to further demonstrate the advantages of this approach when setting concrete parameters for our voting scheme in Section 5 where our fine-grained analysis allows us to bring down the overall communication cost dramatically.

5 PERFORMANCE

5.1 Setting Parameters

We begin by collecting all parameters of the scheme and noting any constraints applying to them in Table 6.

Next, we closely examine the constraint needed for the correct (perfect) decryption of votes as performed by the Comb algorithm. This turns out to be the most influential constraint on the overall efficiency of the scheme. In particular, this constraint informs our choice of the global ring dimension d and modulus q, which most directly affect the communication sizes.

Decryption Correctness. After passing through the mix-net of ξ_1 shuffle servers, a ciphertext is of the form

$$c = p(h \sum_{k \in [\xi_1]} s_k + \sum_{k \in [\xi_1]} e_k) + m,$$

where the encryption randomness terms s_k and e_k are sampled from S_ν. Next, this ciphertext is passed to a decryption server, which computes a decryption share of the form $\mathsf{ds}_j = f_j \cdot c + pE_j$. Then the Comb algorithm, on collecting $\{\mathsf{ds}_j\}_{j \in [\xi_2]}$, outputs

$$v' = (\sum_{j \in [\xi_2]} \mathsf{ds}_j \mod q) \mod p.$$

Parameter	Explanation	Value
λ	Computational security parameter	128
d	Ring dimension	2048
q	Ciphertext and commitment modulus	$\approx 2^{59}$
sec	Statistical security parameter	40
p	Plaintext modulus	2
t	$\mathsf{KeyGen}_{\mathsf{NTRU}}$ rejection parameter	1.058
ν	Infinity norm of encryption randomness	1
ξ_1, ξ_2	Number of shuffle and decryption servers	4
B_{Com}	Infinity norm of commitment randomness	1
B_{Dec}	Noise in ciphertext	262144
B_{Drown}	Infinity norm of noise drowning term E_{ij}	$\approx 2^{55}$
σ_{NTRU}	Standard deviation for encryption secret key	7.12
η	Reed-Solomon encoding randomness length	325
ℓ_{Small}	Proof batch size in Π_{SMALL}	9830
ℓ_{Bnd}	Proof batch size in Π_{BND}	12288
μ_{Small}	Reed-Solomon message length in Π_{SMALL}	10565
μ_{Bnd}	Reed-Solomon message length in Π_{BND}	8517
μ'_{Small}	Reed-Solomon message dimension in Π_{SMALL}	23988
μ'_{Bnd}	Reed-Solomon message dimension in Π_{BND}	181550
γ_{Small}	Reed-Solomon code length in Π_{SMALL}	26616
γ_{Bnd}	Reed-Solomon code length in Π_{BND}	198668

Table 2. Sample parameter set.

In order for the result of this process to yield the original ballot cast, we require the infinity norm of the sum here to be bounded by $\lfloor q/2 \rfloor$. It follows that a sufficient constraint for this correct decryption is:

$$p \cdot d \cdot t \cdot \sigma_{\mathsf{NTRU}} \cdot (2\xi_1 \cdot \nu + 1/2)(1 + 2^{\text{sec}}) < \lfloor q/2 \rfloor, \tag{3}$$

where t is the rejection parameter in $\mathsf{KeyGen}_{\mathsf{NTRU}}$.

Computational Security. Having chosen parameters satisfying the constraints of Table 6, we must ensure that the underpinning lattice problems are sufficiently hard for these parameters.

For RLWE we follow standard convention by using the estimator [APS15]. This estimates the cost of BKZ conservatively by focusing only on the cost of a single uSVP oracle call, a core operation in BKZ. The number of such calls required has been estimated to be $8d$ for a lattice dimension d, and we follow this estimate.

To determine the NTRU problem's hardness, we use the analysis of Section 4. Having settled on a ring dimension d and modulus q giving sufficient hardness of the RLWE problem, we use (3) to determine the maximum standard deviation permissible for generating the NTRU secrets (f, g). Finally, following the procedure described in Section 4, one can calculate the estimated of NTRU. Again, we employ the conservative formula $0.292\beta + 16.4 + \log_2(8d)$ used in [DTGW17, SPL+17, BIP+22] to compute bit-security from blocksize β.

In order to ensure the binding property of the BDLOP commitment schemes we use, the RSIS problem must be hard. We use the relation due to Micciancio and Regev [MR09], which states that LLL will recover a short vector a vector of 2-norm $2^{(2\sqrt{d \log_2 q \log_2 \delta})}$. δ is the root Hermite factor and $\delta < 1.0045$ gives rise to at least 128 bits of security. Owing to the horizontally long nature of the commitment matrix used, the hardness of the corresponding RSIS instance easily meets this threshold.

5.2 Sample Parameters and Total Size

Table 2 gives a sample set of parameters generated by following the process described in the previous section. In Table 3, we present the total sizes of objects in our voting scheme and compare them with those of [ABGS23]. We denote the output of each shuffle node by $\pi_{\mathcal{S}_i}$, including ciphertexts, commitments, proofs of shortness, and shuffle proofs. Similarly, we denote the total output of each decryption node as $\pi_{\mathcal{D}_j}$, consisting of decryption shares, commitments, proofs of linearity and boundedness.

Our scheme achieves a reduction in ciphertext size by over a factor of five. Moreover, the reduction in commitment sizes and constituent proofs leads to shuffle server outputs of 130 KB per vote, which are three times smaller, and decryption server outputs of 85 KB per vote, which are half of those in [ABGS23]. Overall this represents a 2.5× efficiency gain over [ABGS23] as summarized in Table 1.

Scheme	c_i	$[\![R_q]\!]$	π_{Shuf}	π_{Lin}	π_{Small}	π_{Bnd}
[ABGS23] [KB]	80	80/120	150	35	20	2
Our [KB]	15	30	63	18	22	22

Table 3. Ciphertext, commitment, and proof sizes per voter. Note that the two sizes in [ABGS23] reflect commitments to noise-drowning terms and ciphertexts, respectively.

5.3 Benchmarks

We adapt the proof-of-concept implementation by [ABGS23] to fit our scheme since the framework is the same. Our benchmarks were collected on an Intel Kaby Lake Core i7-7700 CPU machine with 64 GB of RAM running single-threaded at 3.6 GHz, with Turbo Boost disabled to reduce measurement variability. This is a similar machine as in [ABGS23]. Our code is available at `https://github.com/carrosa/ntru_voting_impl`.

Scheme	Com	Open	Enc	Dec	DDec
[ABGS23] [ms]	0.45	2.76	0.74	0.64	1.56
Our [ms]	0.17	0.80	0.20	0.21	0.45

Table 4. Ciphertext and commitment timings. Numbers were obtained averaging over 10^4 executions measured using the cycle counter available on the platform.

We compare the timings in Table 4 and Table 5. Analysing our experiments, each shuffle server takes $(0.20+0.17+17.5+44.2) = 62$ ms and each decryption server takes $(0.17+0.45+16.9+310.5) = 328$ ms. Given four servers, where shuffles are performed sequentially and decryption is performed in parallel, the total time is 576 ms, making our scheme twice as fast as [ABGS23] as summarized in Table 1.

Scheme	π_{Lin}	π_{Shuf}	π_{Small}	π_{Bnd}
[ABGS23] [ms]	$(43.4 + 6.4)$	$(44.9 + 7.9)$	$(214.4 + 10.0)$	$(92.7 + 23.9)$
Our [ms]	$(16.9 + 2.0)$	$(17.5 + 2.1)$	$(44.2 + 4.0)$	$(310.5 + 4.3)$

Table 5. Proving and verification times, obtained by computing the average of 100 executions with $\tau = 1000$.

We finally note that the proofs of linearity in the shuffle and the batched proofs of shortness and boundedness during shuffles and decryption can be computed in parallel, and that powerful servers dedicated to an election with Turbo Boost enabled would most likely outperform our numbers by at least an order of magnitude.

5.4 Future Improvements

We provide some directions for interesting future work:

1. *Return codes.* To extend our scheme and ensure voter verifiability, we need to add return codes to our scheme. This can be done by extending the work of [HS22] from BGV to NTRU. This also includes verifiable encryption.
2. *Improved noise analysis.* Our results can possibly be improved using techniques in [AKSY22,BS23,CSS$^+$22]. We use 40 bits of statistical noise drowning to protect the secret key in the distributed decryption protocol. This can possibly be improved if we choose parameters based on how many ciphertexts we will decrypt or change noise drowning techniques to Gaussian and compute the Rényi divergence to estimate the leakage.
3. *Improved parameters in other schemes.* Our extended NTRU analysis might lead to more efficient FHE parameters in [BIP$^+$22] and [Klu22] using the same methodology that led to a more efficient instantiation of `NTRUEncrypt` .

Acknowledgments. We thank Wessel van Woerden and Leo Ducas for their helpful insights on the design of their NTRU estimator and Ngoc Khanh Nguyen for useful discussions on zero-knowledge arguments. Furthermore, we thank both the Mathematical Institute of Oxford University and The Department of Information Security and Communication Technology at NTNU for the use of their computing clusters when conducting the NTRU hardness experiments. We finally thank Diego F. Aranha for helpful input when updating the code base and running performance experiments of our voting protocol.

References

ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 153–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 4, 7, 21

ABG$^+$21. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Lattice-based proof of shuffle and applications to electronic voting. In Kenneth G. Paterson, editor, *Topics in Cryptology – CT-RSA 2021*, volume 12704 of *Lecture Notes in Computer Science*, pages 227–251, Virtual Event, May 17–20, 2021. Springer, Heidelberg, Germany. 3, 7, 16, 19

ABGS23. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1467–1481, New York, NY, USA, 2023. Association for Computing Machinery. 3, 4, 5, 6, 7, 11, 12, 13, 14, 16, 17, 19, 21, 29, 30, 37, 38, 40

AC19. Aleksander Essex Anthony Cardillo, Nicholas Akinyokun. Online voting in ontario municipal elections: A conflict of legal principles and technology?, 2019. Accessed: 2024-02-27. 2

Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108, Philadephia, PA, USA, May 22–24, 1996. ACM Press. 2, 9

AKSY22. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 39–53, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. 31

APS15. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 23, 29

BBC$^+$18. 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, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 669–699, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. 11

BDL$^+$18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany. 10, 11, 18

BG12. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 263–280, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 3

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery. 4

BHM20. 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: 25th European Symposium on Research in Computer Security, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 336–356, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany. 7

BIP$^+$22. Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart. FINAL: Faster FHE instantiated with NTRU and LWE. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 188–215, Taipei, Taiwan, December 5–9, 2022. Springer, Heidelberg, Germany. 29, 31

BLNS21. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. More efficient amortization of exact zero-knowledge proofs for LWE. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021: 26th European Symposium on Research in Computer Security, Part II*, volume 12973 of *Lecture Notes in Computer Science*, pages

608–627, Darmstadt, Germany, October 4–8, 2021. Springer, Heidelberg, Germany. 5, 16

BLS19. Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 176–202, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. 8

BS23. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In *Advances in Cryptology – ASIACRYPT 2023, Part I*, Lecture Notes in Computer Science, pages 371–404. Springer, Heidelberg, Germany, December 7–11, 2023. 31

CBS00. CBS News. Online first in arizona, 2000. Accessed: 27-02-2024. 2

CGGI16. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A homomorphic LWE based E-voting scheme. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, pages 245–265, Fukuoka, Japan, February 24–26, 2016. Springer, Heidelberg, Germany. 7

Cha03. David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. In Dimitris Gritzalis, editor, *Secure Electronic Voting*, volume 7 of *Advances in Information Security*, pages 211–219. Springer, 2003. 3

CJL16a. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016. 4

CJL16b. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016. 7

CMM19. Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-based proof of a shuffle. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala, editors, *FC 2019 Workshops*, volume 11599 of *Lecture Notes in Computer Science*, pages 330–346, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019. Springer, Heidelberg, Germany. 7

CPS$^+$20. Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. ModFalcon: Compact signatures based on module-NTRU lattices. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security*, pages 853–866, Taipei, Taiwan, October 5–9, 2020. ACM Press. 8

CSS$^+$22. Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient threshold FHE with application to real-time systems. Cryptology ePrint Archive, Report 2022/1625, 2022. https://eprint.iacr.org/2022/1625. 31

dK22. Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances* *in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 306–336, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. 6, 25, 26

dLNS17. 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: 24th Conference on Computer and Communications Security*, pages 1565–1581, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 3, 7

DTGW17. Jintai Ding, Tsuyoshi Takagi, Xinwei Gao, and Yuntao Wang. Ding Key Exchange. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. 29

Dv21. Léo Ducas and Wessel P. J. van Woerden. NTRU fatigue: How stretched is overstretched? In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 3–32, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. 4, 5, 6, 7, 22, 23, 24, 25

Ele17. Electoral Commission. Uk parliamentary general election 2017: Electoral data report, 2017. Accessed: 2024-02-27. 3

FWK21. Valeh Farzaliyev, Jan Willemson, and Jaan Kristjan Kaasik. Improved lattice-based mix-nets for electronic voting. In *Information Security and Cryptology – ICISC 2021*. Springer International Publishing, 2021. 3, 7

HMS21a. Javier Herranz, Ramiro Martínez, and Manuel Sánchez. Shorter lattice-based zero-knowledge proofs for the correctness of a shuffle. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops*, pages 315–329, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg. 7

HMS21b. Javier Herranz, Ramiro Martínez, and Manuel Sánchez. Shorter lattice-based zero-knowledge proofs for the correctness of a shuffle. In *International Conference on Financial Cryptography and Data Security*, pages 315–329. Springer, 2021. 7

HPS98. Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998. 2, 4, 9

HS22. Audhild Høgåsen and Tjerand Silde. Return codes from lattice assumptions. *E-VOTE-ID*, 2022. 31

Kat21. Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 580–610, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 25

KF17. Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 3–26, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. 4, 7, 21, 22

Klu22. Kamil Kluczniak. NTRU-v-um: Secure fully homomorphic encryption from NTRU with small modulus. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 1783–1797, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. 31

LDK+20. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. 2

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 71–101, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. 13

LNS21. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 215–241, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. 8, 18

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. 9

LW20. Changmin Lee and Alexandre Wallet. Lattice analysis on MiNTRU problem. Cryptology ePrint Archive, Report 2020/230, 2020. https://eprint.iacr.org/2020/230. 21

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. 8

Mic23. Microsoft Corporation. Council of state governments election technology initiative: Electionguard, 2023. Accessed: 2024-02-27. 2

MR04. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 8

MR09. Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 29

New21. New South Wales Electoral Commission. ivote and 2021 nsw local government elections, 2021. Accessed: 2024-02-27. 2

PFH+20. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. 2, 4

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 2, 9

RK18. Iuliia Krivonosova Priit Vinkel Arne Koitmae Robert Krimmer, David Duenas-Cid. How much does an e-vote cost? cost comparison per vote in multichannel elections in estonia, 2018. Accessed: 2024-02-27. 2

SAB$^+$20. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. 2

SMPS16. Paolo Spada, Jonathan Mellon, Tiago Peixoto, and Fredrik M. Sjoberg. Effects of the internet on participation: Study of a public policy referendum in brazil. *Journal of Information Technology & Politics*, 13(3):187–207, 2016. 2

Sol01. Frederic I. Solop. Digital democracy comes of age: Internet voting and the 2000 arizona democratic primary election. *PS: Political Science & Politics*, 34(2):289–293, 2001. 2

SPL$^+$17. Minhye Seo, Jong Hwan Park, Dong Hoon Lee, Suhri Kim, and Seung-Joon Lee. EMBLEM and R.EMBLEM. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-1-submissions. 29

SS11. Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany. 5, 9, 10, 23

Str19. Martin Strand. A verifiable shuffle for the GSW cryptosystem. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *Lecture Notes in Computer Science*, pages 165–180, Nieuwpoort, Curaçao, March 2, 2019. Springer, Heidelberg, Germany. 7

Swi23. Swiss Post. Swiss post's e-voting system to be used for the first time in elections this autumn following further development and successful hacker test, 2023. Accessed: 27-02-2024. 2

U.S16. U.S. Election Assistance Commission. 2016 election administration and voting survey (eavs) comprehensive report, 2016. Accessed: 2024-02-27, Page 31. 3

Vin15. Priit Vinkel. Remote electronic voting in estonia: Legality, impact, and confidence. *ResearchGate*, 2015. 2

A Security of Verifiable Mixing

First, we define *completeness*, *soundness* and *simulatability* for a mixing protocol Π_{Mix} executed by a prover Prover, with respect to a generic encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ [ABGS23].

Definition 5 (Mixing Completeness). *We say that the mixing protocol Π_{Mix} is* complete *if for honest PPT parties* Prover *and* $\mathsf{Verifier}$ *that follows the protocol then* Prover *on input a set of honestly generated ciphertexts will output a new set of ciphertexts together with a proof such that* $\mathsf{Verifier}$ *accepts the proof and the output ciphertexts decrypt to the same set of messages as the input ciphertexts. Hence, we want that for all* $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$*,* $\{c_i\}_{i\in[\tau]} \leftarrow \mathsf{Enc}(\mathsf{pk}, \{m_i\}_{i\in[\tau]})$*, and* $(\{\hat{c}_i\}_{i\in[\tau]}, \pi) \leftarrow \mathsf{Prover}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]})$*, we have*

$$\Pr\left[\begin{array}{c}\{m_i\}_{i\in[\tau]} = \mathsf{Dec}(\mathsf{sk}, \{\hat{c}_i\}_{i\in[\tau]} \\ 1 \leftarrow \mathsf{Verifier}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]}, \{\hat{c}_i\}_{i\in[\tau]}, \pi)\end{array}\right] \leq 1 - \epsilon(\lambda),$$

where the probability is taken over KeyGen*,* Enc *and* Prover*.*

Definition 6 (Mixing Soundness). *We say that the mixing protocol Π_{Mix} is* sound *if a dishonest PPT adversary* Adv *that can behave arbitrarily on input a set of honestly generated ciphertexts will not be able to output a new set of ciphertexts together with a proof such that an honest* $\mathsf{Verifier}$ *accepts the proof but the output ciphertexts decrypt to a different set of messages than the input ciphertexts. Hence, we want that for all* $(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$*,* $\{c_i\}_{i\in[\tau]} \leftarrow \mathsf{Enc}(\mathsf{pk}, \{m_i\}_{i\in[\tau]})$*, and* $(\{\hat{c}_i\}_{i\in[\tau]}, \pi) \leftarrow \mathsf{Adv}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]})$*, we have*

$$\Pr\left[\begin{array}{c}\{m_i\}_{i\in[\tau]} \neq \mathsf{Dec}(\mathsf{sk}, \{\hat{c}_i\}_{i\in[\tau]} \\ 1 \leftarrow \mathsf{Verifier}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]}, \{\hat{c}_i\}_{i\in[\tau]}, \pi)\end{array}\right] \leq \epsilon(\lambda),$$

where the probability is taken over KeyGen*,* Enc *and* Adv*.*

Definition 7 (Mixing Simulatability). *We say that the mixing protocol Π_{Mix} is* simulatable *if a PPT adversary $\mathcal{A}$ that on input a set of honestly generated ciphertexts can not distinguish between a real execution of the mixing protocol with accepting output and a protocol execution from a PPT simulator $\mathcal{S}$ (given a set honestly mixed output ciphertexts) producing a simulated mixing proof. Hence, we want that*

$$\left|\Pr\left[b = b' \ : \begin{array}{c}(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa); b \xleftarrow{\$} \{0,1\} \\ \{c_i\}_{i\in[\tau]} \leftarrow \mathsf{Enc}(\mathsf{pk}, \{m_i\}_{i\in[\tau]}) \\ (\{\hat{c}_i\}_{i\in[\tau]}, \pi^{(0)}) \leftarrow \mathsf{Prover}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]}) \\ (\pi^{(1)}) \leftarrow \mathcal{S}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]}, \{\hat{c}_i\}_{i\in[\tau]}) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, \mathsf{pk}, \{c_i\}_{i\in[\tau]}, \{\hat{c}_i\}_{i\in[\tau]}, \pi^{(b)})\end{array}\right] - \frac{1}{2}\right| \leq \epsilon(\lambda),$$

where the probability is taken over KeyGen*,* Enc*,* Prover*,* $\mathcal{S}$ *and* Adv*.*

B Security of Distributed Decryption

Here we define the syntax and security properties for a PKE with distributed decryption [ABGS23].

Definition 8 (PKE with Distributed Decryption). *A PKE scheme with distributed decryption consists of five algorithms: key generation* (KeyGen)*, encryption* (Enc)*, decryption* (Dec)*, distributed decryption* (DDec)*, and combine* (Comb)*, where*

- KeyGen. *On input security parameter 1^λ and number of key-shares ξ_2, outputs public parameters* pp*, a public key* pk*, a secret key* sk*, and key-shares $\{\mathsf{sk}_j\}$,*
- Enc. *On input* pk *and messages $\{m_i\}$, outputs ciphertexts $\{c_i\}$,*
- Dec. *On input* sk *and ciphertexts $\{c_i\}$, outputs messages $\{m_i\}$,*
- DDec. *On input a secret key share sk_{j^*} and ciphertexts $\{c_i\}$, outputs decryption shares $\{\mathsf{ds}_{i,j^*}\}$,*
- Comb. *On input ciphertexts $\{c_i\}$ and decryption shares $\{\mathsf{ds}_{i,j}\}$, outputs either messages $\{m_i\}$ or $\perp$,*

and pp *are implicit inputs to* Enc*,* Dec*,* DDec *and* Comb*.*

Definition 9 (Chosen Plaintext Security). *We say that the public key encryption scheme is secure against* chosen plaintext attacks *if an adversary* Adv*, after choosing two messages m_0 and m_1 and receiving an encryption c of either m_0 or m_1 (chosen at random), cannot distinguish which message c is an encryption of. Hence, we want that*

$$\left| \Pr\left[b = b' \;:\; \begin{array}{c} (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathsf{Adv}(\mathsf{pp}, \mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\}, c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b' \leftarrow \mathsf{Adv}(c, \mathsf{st}) \end{array} \right] - \frac{1}{2} \right| \leq \epsilon(\lambda),$$

where the probability is taken over KeyGen *and* Enc*.*

Definition 10 (Threshold Correctness). *We say that the public key distributed encryption scheme is* threshold correct *with respect to $P_{\mathsf{sk}}(\cdot)$ if the following probability equals 1:*

$$\Pr\left[\begin{array}{c} \mathsf{Comb}(\{c_i\}_{i\in[\tau]}, \{\mathsf{ds}_{i,j}\}_{i\in[\tau]}^{j\in[\xi_2]}) \\ = \\ \mathsf{Dec}(\mathsf{sk}, \{c_i\}_{i\in[\tau]}) \end{array} : \begin{array}{c} (\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, \{\mathsf{sk}_j\}_{j\in[\xi_2]}) \leftarrow \mathsf{KeyGen}(1^\lambda, \xi_2) \\ \{c_1, \ldots, c_\tau\} \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk}) \\ \forall i \in [\tau] :\; P_{\mathsf{sk}}(c_i) = 1, \forall j \in [\xi_2] : \\ \{\mathsf{ds}_{i,j}\}_{i\in[\tau]} \leftarrow \mathsf{DDec}(\mathsf{sk}_j, \{c_i\}_{i\in[\tau]}) \end{array} \right],$$

where the probability is taken over KeyGen *and* DDec*.*

Definition 11 (Threshold Verifiability). *A PKE scheme with distributed decryption is* threshold verifiable *with respect to $P_{\mathsf{sk}}(\cdot)$ if an adversary $\mathcal{A}$ corrupting $J \subseteq [\xi_2]$ secret key shares $\{\mathsf{sk}_j\}_{j\in J}$ cannot convince* Comb *to accept maliciously* *created decryption shares* $\{\mathsf{ds}_{i,j}\}_{i\in[\tau],j\in J}$. *More concretely, the following probability is bounded by a negligible* $\epsilon(\lambda)$*:*

$$\Pr\left[\begin{array}{c} \mathsf{Dec}(\mathsf{sk},\{c_i\}_{i\in[\tau]}) \\ \neq \\ \mathsf{Comb}(\{c_i\}_{i\in[\tau]},\{\mathsf{ds}_{i,j}\}_{i\in[\tau]}^{j\in[\xi_2]}) \\ \neq \\ \perp \end{array} : \begin{array}{c} (\mathsf{pp},\mathsf{pk},\mathsf{sk},\{\mathsf{sk}_j\}_{j\in[\xi_2]}) \leftarrow \mathsf{KeyGen}(1^\lambda,\xi_2) \\ (\{c_1,\ldots,c_\tau\}) \leftarrow \mathcal{A}(\mathsf{pp},\mathsf{pk},\{\mathsf{sk}_j\}_{j\in J}) \\ \forall i\in[\tau]:\ P_{\mathsf{sk}}(c_i)=1, \forall j\notin J: \\ \{\mathsf{ds}_{i,j}\}_{i\in[\tau]} \leftarrow \mathsf{DDec}(\mathsf{sk}_j,\{c_i\}_{i\in[\tau]}) \\ \{\mathsf{ds}_{i,j}\}_{i\in[\tau],j\in J} \leftarrow \mathcal{A}(\{\mathsf{ds}_{i,j}\}_{i\in[\tau],j\notin J}) \end{array}\right],$$

where the probability is taken over KeyGen *and* DDec*.*

Definition 12 (Distributed Decryption Simulatability). *A PKE scheme with distributed decryption is* simulatable *with respect to* $P_{\mathsf{sk}}(\cdot)$ *if an adversary* $\mathcal{A}$ *corrupting* $J\subsetneq[\xi_2]$ *secret key shares* $\{\mathsf{sk}_j\}_{j\in J}$ *cannot distinguish the transcript of the decryption protocol from a simulation by a simulator* Sim *which only gets* $\{\mathsf{sk}_j\}_{j\in J}$ *as well as correct decryptions as input. More concretely, the following probability is bounded by a negligible* $\epsilon(\mathrm{sec})$*:*

$$\left|\Pr\left[b=b' \ :\ \begin{array}{c} (\mathsf{pp},\mathsf{pk},\mathsf{sk},\{\mathsf{sk}\}_{j\in[\xi_2]}) \leftarrow \mathsf{KeyGen}(1^\lambda,\xi_2) \\ (\{c_1,\ldots,c_\tau\}) \leftarrow \mathcal{A}(\mathsf{pp},\mathsf{pk},\{\mathsf{sk}_j\}_{j\in J}) \\ \forall i\in[\tau]:\ P_{\mathsf{sk}}(c_i)=1 \\ \{\mathsf{ds}^0_{i,j}\} \leftarrow \mathsf{DDec}(\{\mathsf{sk}_j\}_{j\in[\xi_2]},\{c_i\}_{i\in[\tau]}) \\ \{\mathsf{ds}^1_{i,j}\} \leftarrow \mathsf{Sim}(\mathsf{pp},\{\mathsf{sk}_j\}_{j\in J},\{c_i,\mathsf{Dec}(\mathsf{sk},c_i)\}_{i\in[\tau]}) \\ b \overset{\$}{\leftarrow} \{0,1\}, b' \leftarrow \mathcal{A}(\{\mathsf{ds}^b_{i,j}\}_{i\in[\tau],j\in[\xi_2]}) \end{array}\right] - \frac{1}{2}\right|,$$

where the probability is taken over KeyGen, DDec, Sim*.*

C Parameter Constraints

Here we give the describe the parameters used in our electronic voting scheme. Table 6 lists these and makes explicit any constraints that apply to them. These constrains inform the choice of concrete values computed in Section 5.2.

Parameter	**Explanation**	**Constraints**
λ	Computational security parameter	≥ 128
sec	Statistical security parameter	≥ 40
d	Ring dimension of R_p and R_q	d a power of two
p	Plaintext modulus	p a small prime
q	Ciphertext and commitment modulus	Prime $q = 1 \mod 2d$ s.t. $\left\|\sum_{j \in \xi_2} \mathsf{ds}_j\right\|_\infty \leq \lfloor q/2 \rfloor$
t	$\mathsf{KeyGen}_{\mathsf{NTRU}}$ rejection parameter	Set for rej. prob. $< 1/1000$ (Lemma 1)
k	Length of binding vector in BDLOP commitment	$k > 2$
$\mathcal{C}$	Challenge space for linear ZK proofs of commitments	$\mathcal{C} = \left\{c \in R_q \mid \|c\|_\infty = 1, \|c\|_1 = \kappa\right\}$
κ	Maximum ℓ_1-norm of elements in $\mathcal{C}$	$2^\kappa \cdot \binom{d}{\kappa} > 2^\lambda$
ξ_1, ξ_2	Number of shuffle and decryption-servers	At least two servers
B_{Com}	Bound on the commitment noise	So that SIS is hard
B_{Dec}	Noise in ciphertext	$B_{\mathsf{Dec}} = p \cdot d \cdot t \cdot \sigma_{\mathsf{NTRU}} \cdot (2\xi_1 \cdot \nu + 1/2)$
B_{Drown}	Infinity norm of noise drowning term E_{ij}	$B_{\mathsf{Drown}} = 2^{\mathrm{sec}}(B_{\mathsf{Dec}}/p\xi_2)$
σ_{NTRU}	Standard deviation for encryption secret key	So that NTRU is hard
ν	Bound on encryption randomness	So that LWE is hard
σ_{Com}	Standard deviation in ZK proofs of linear relations	Chosen to be $\sigma_{\mathsf{Com}} = \kappa \cdot B_{\mathsf{Com}} \cdot \sqrt{kd}$
τ	Total number of messages/number of voters	For soundness we need $(\tau^\delta + 1)/\lvert R_q \rvert < 2^{-\lambda}$
η	Reed-Solomon encoding randomness length	Make soundness $\geq 2^{-\lambda}$ in $\mathit{\Pi}_{\text{SMALL}}$ and $\mathit{\Pi}_{\text{BND}}$
ℓ_{Small}	Proof batch size in $\mathit{\Pi}_{\text{SMALL}}$	Same secret length as in [ABGS23]
ℓ_{Bnd}	Proof batch size in $\mathit{\Pi}_{\text{BND}}$	Same secret length as in [ABGS23]
μ_{Small}	Reed-Solomon message length in $\mathit{\Pi}_{\text{SMALL}}$	$\mu_{\mathsf{Small}} = (k+2) \cdot d + \eta$
μ_{Bnd}	Reed-Solomon message length in $\mathit{\Pi}_{\text{BND}}$	$\mu_{\mathsf{Bnd}} = (k+1) \cdot d + \eta$
μ'_{Small}	Reed-Solomon message dimension in $\mathit{\Pi}_{\text{SMALL}}$	$\mu_{\mathsf{Small}} \leq \mu'_{\mathsf{Small}} \leq \gamma < q$
μ'_{Bnd}	Reed-Solomon message dimension in $\mathit{\Pi}_{\text{SMALL}}$	$\mu_{\mathsf{Bnd}} \leq \mu'_{\mathsf{Bnd}} \leq \gamma < q$
γ_{Small}	Reed-Solomon code length in $\mathit{\Pi}_{\text{BND}}$	$\mu_{\mathsf{Small}} \leq \mu'_{\mathsf{Small}} \leq \gamma < q$
γ_{Bnd}	Reed-Solomon code length in $\mathit{\Pi}_{\text{BND}}$	$\mu_{\mathsf{Bnd}} \leq \mu'_{\mathsf{Bnd}} \leq \gamma < q$

Table 6. System parameters and constraints.