




# Concrete NTRU Security and Advances in Practical Lattice-Based Electronic Voting

Patrick Hough<sup>1\*</sup> , Caroline Sandsbråten<sup>2</sup> , and Tjerand Silde<sup>2</sup> 

<sup>1</sup> Mathematical Institute  
Oxford University

`patrick.hough@maths.ox.ac.uk`

<sup>2</sup> Department of Information Security and Communication Technology  
Norwegian University of Science and Technology  
`caroline.sandsbraten@ntnu.no`, `tjerand.silde@ntnu.no`

**Abstract.** In recent years there has been much focus on the development of core cryptographic primitives based on lattice assumptions. This has been driven by the NIST call for post-quantum key encapsulation and digital signature specifications. However, there has been much less work on efficient privacy-preserving protocols with post-quantum security.

In this work we present an efficient electronic voting scheme from lattice assumptions, ensuring the long-term security of encrypted ballots and voters' privacy. The scheme relies on the NTRU and RLWE assumptions. We begin by conducting an extensive analysis of the concrete hardness of the NTRU problem. Extending the ternary-NTRU analysis of Ducas and van Woerden (ASIACRYPT 2021), we determine the concrete *fatigue point* 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  $\sigma$ . Moreover, we demonstrate that the nature of this relation enables a more fine-grained choice of secret key sizes, leading to more efficient parameters in practice.

Using the above analysis, our second and main contribution is to significantly improve the efficiency of the state-of-the-art lattice-based voting scheme by Aranha et al. (ACM CCS 2023). Replacing the BGV encryption scheme with NTRU we obtain a factor  $\times 5.3$  reduction in ciphertext size and  $\times 2.6$  more efficient system overall, making the scheme suitable for use in real-world elections.

As an additional contribution, we analyse the (partially) blind signature scheme by del Pino and Katsumata (CRYPTO 2022). We note that the NTRU security is much lower than claimed and propose new parameters. This results in only a minor efficiency loss, enabled by our NTRU analysis where previous parameter selection techniques would have been much more detrimental.

**Keywords:** NTRU Security · Lattice Cryptography · Electronic Voting

---

\* 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 is an international effort to replace these primitives with others based on hardness assumptions that are conjectured to be secure against quantum adversaries, and some of the most promising candidates are the lattice-based assumptions (Ring)-Short Integer Solution (RSIS) [Ajt96], (Ring)-Learning With Errors (RLWE) [Reg05], and NTRU [HPS98]. In fact, in the final round before standardization of post-quantum Key Encapsulation Mechanisms (KEMs) and Digital Signatures (DSs), the National Institute of Standards and Technology (NIST) settled upon seven candidate schemes, five of which are based on lattices. In the end, they decided on the RLWE-based KEM Crystals Kyber [SAB+20], the RSIS/RLWE-based DS Crystals Dilithium [LDK+20], and NTRU-based DS Falcon [PFH+20], in addition to the hash-based DS SPHINCS+ [HBD+20].

The security of RSIS and RLWE is relatively well understood today. The RSIS problem is believed to be hard when the logarithm of the  $\ell_2$  norm of the secret vector is less than  $2\sqrt{d \log_2 q \log_2 \delta}$  for lattice dimension  $d$ , modulus  $q$ , and root Hermite factor  $\delta$  [MR09]. Current literature adopts the notion that  $\delta = 1.0045$  or smaller gives rise to 128 bits of security. Furthermore, Albrecht et al. have collated a long line of algorithmic cryptanalysis into a publicly available estimator [APS15] used to estimate the hardness of a given RLWE instance based on the dimension, modulus, and norm of the secrets. However, the hardness of the NTRU problem is less understood and the most up-to-date security estimates are either asymptotic or only computed for very particular instances, for example when the secret vectors are ternary.

We briefly recall the NTRU problem. Let  $R_q$  be a polynomial ring  $R_q = \mathbb{Z}[x]/(x^d + 1)$  of dimension  $d$  and modulus  $q$  and let  $D_\sigma$  be a discrete Gaussian distribution with standard deviation  $\sigma$  over  $\mathbb{Z}$ . Then, informally, the NTRU problem defined by Hoffstein et al. [HPS98] is the following: given  $h \in R_q$ , determine whether  $h$  is constructed as  $h = g/f \in R_q$  for  $g$  and  $f$  sampled from  $D_\sigma^d$  or is in fact sampled from the uniform distribution over  $R_q$ . Existing NTRU encryption and signature schemes use  $h$  as the public key and  $f$  as the secret key.

Intuitively, a key recovery attack (find  $\sigma$ -bounded  $g, f$  given  $h$ ) is by definition to find vectors with small norm in the corresponding *NTRU lattice*, similar to breaking the RSIS problem above. In general, the hardness of the lattice problems grows exponentially with the dimension  $d$ . However, a more in-depth analysis of this problem has led to an algebraic attack against so-called *overstretched* NTRU [ABD16, C JL16a]. This attack does not apply to RSIS/RLWE (where the general strategy is sieving [ADH+19] or enumeration [ABLR21] to find short vectors in these lattices), revealing that the NTRU problem gets substantially easier to solve when  $\sigma$  is small and  $q$  is much larger than  $d$ . Herein, we will refer to the point at which this complexity improvement begins as the *fatigue point*. We note that, in contrast, the RSIS problem gets much harder to solve when  $q$  grows compared to  $\sigma$ . The work of Kirchner and Fouque [KF17] shows that the overstretched attack is noticeably more efficient than the general lattice

reduction attacks when  $g$  and  $f$  are polynomials with ternary coefficients and  $q$  is (asymptotically) larger than  $d^{2.783+o(1)}$ . A recent analysis by Ducas and van Woerden [DvW21] decreases the exponent here to 2.484 in the asymptotic case and fixes the constant in front of  $d$  to be 0.004 in the concrete, ternary setting.

On the other hand, we have provable hardness bounds for NTRU when  $\sigma$  is large. Stehlé and Steinfeld [SS11] proved that when  $\sigma$  is roughly of size  $q^{1/2}$  then  $h$  is statistically close to uniform and the NTRU problem enjoys worst-case hardness. This result is similar to that of RLWE, where the problem gets harder when  $D_\sigma$  is a discrete Gaussian distribution and  $\sigma$  increases towards the square root of the moduli and eventually results in a reduction to worst-case problems over general lattices [Reg05]. While the hardness estimates for RSIS and RLWE secrets of various norms are easily available, an understanding of NTRU’s hardness for secret sizes between these two aforementioned extremes essentially ends here.

Let  $\sigma_1$  be the standard deviation for sampling ternary values and let  $\sigma_{\text{stat}}$  be roughly  $q^{1/2}$ . The natural research question to ask is then:

*What is the concrete hardness of the NTRU problem when secrets are sampled with standard deviation  $\sigma_{\text{NTRU}}$  for  $\sigma_1 < \sigma_{\text{NTRU}} < \sigma_{\text{stat}}$ ?*

Understanding such behaviour for RSIS and RLWE has already received much attention. The norm of the secrets used in Crystals Dilithium [LDK<sup>+</sup>20] are not ternary but are instead sampled with absolute norm 2 (NIST level 2 and 5) or 4 (NIST level 3) as this is needed to give the appropriate security level for the given dimension and modulus of the system. Here, a smaller norm of the secret would lead to larger parameters overall to compensate for the security loss. To further illustrate the importance of secret size when it comes to setting parameters, recall that an element in  $R_q$  takes  $d \cdot \log_2 q$  bits to represent, where  $d$  is a power of two. If the secret NTRU vector is ternary and a given pair  $(d, q)$  does not yield sufficient security, one remedy is to increase  $d$ , leading to a doubling in communication cost. Alternatively, one can increase the norm of the secrets, if possible, without changing  $d$ . This provokes the natural question:

*Can a carefully chosen non-ternary NTRU secret bound lead to more efficient instantiations of lattice-based protocols in practice?*

## 1.1 Our Contribution

In this work, we answer both of the above questions comprehensively; we determine a concrete relation for the fatigue point of general NTRU (parametrised not only by  $d$  and  $q$ , but also by  $\sigma$ ) and show how a careful choice of secret size yields optimal parameter selection in practice.

We begin by building upon the work of Ducas and van Woerden [DvW21], which considers ternary NTRU, to analyse the overstretched attack against NTRU when the norm of the secrets grows with respect to the dimension and modulus. We stress that [DvW21] does give an asymptotic fatigue point for general NTRU but only a concrete relation for ternary secrets. Our aim is to better understand

the point at which parameters are overstretched, meaning that the traditional key-recovery lattice reduction attacks are outperformed by ones that exploit the existence of a dense sublattice of the NTRU lattice. Our analysis shows that when we increase the standard deviation, the modulus can be increased roughly with the square of this increase before reaching the fatigue point again. We analyse the security and efficiency of some recent lattice-based protocols in the literature, and find that the (partial) blind signatures by del Pino and Katsumata [dK22] can be instantiated more efficiently when choosing secure parameters based on our analysis.

Secondly, we analyse the framework of the recent lattice-based cryptographic voting system by Aranha et al. [ABGS23]. The protocol uses the RLWE-based BGV encryption scheme [BGV12], inherently requiring two ring elements per ciphertext, while the NTRU encryption scheme requires only one. Moreover, by using a variant of the NTRU cryptosystem that relies both on RLWE and NTRU, we are able to optimise parameters to reduce the ring dimension by a factor of two and slightly decrease the modulus, reducing ciphertext sizes by a factor of five. Furthermore, this leads to  $\times 2.6$  smaller communication in the voting system overall. This effort is an important step to push more advanced quantum secure primitives closer to practicality and real-world deployment.

We expand upon some of the more technical details of our results:

*NTRU Security Analysis.* Our starting point here is the work of Ducas and Woerden [DvW21]. Through asymptotic analysis, they narrow in on a lower bound for the fatigue point of NTRU. Next, through a combination of predictive modeling of lattice volumes and concrete experiments on low-dimensional NTRU instances, they give an average-case concrete fatigue point for NTRU with ternary secrets.

Using the scripts provided in [DvW21] we analyse the behaviour of the predicted fatigue point for NTRU with *general* secrets. That is, for secrets sampled from a Gaussian of variable parameter  $\sigma$ . This suggests, with strong correlation, that the concrete fatigue point of NTRU is described by

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

Note, by following a similar asymptotic analysis to that in [DvW21], we confirm that the influence of  $\sigma$  on the fatigue point must indeed manifest only in the leading constant and not in the exponent of  $d$ .

We verify this prediction with extensive experiments over a range of  $\sigma$  for computable ring dimensions, mirroring the process of [DvW21] in the ternary case. These experiments confirm the accuracy of the estimator. Moreover, the small error discrepancy in the predictive model remains constant and continues to have an insignificant impact on the predicted hardness of NTRU. Thus, our results also serve to support the use of Ducas and Woerden’s estimator for general NTRU instances.

*Electronic Voting.* We present a new electronic voting scheme based on the RLWE and NTRU lattice assumptions. The first major design choice is to use the NTRU cryptosystem [SS11] in place of BGV encryption [BGV12]. Crucially, NTRU ciphertexts consist of only a single ring element as opposed to two in BGV. In privacy-preserving constructions where zero-knowledge proofs (ZKPs) are deployed to verify the honest actions of parties, one often wants to minimize the number of relations to be proven in zero-knowledge since ZKPs usually dominate the communication cost. Using ciphertexts of a single element thus yields significant savings over their two-component counterparts when combined with ZKPs.

We then employ our NTRU analysis above to make a fine-tuned parameter selection for our voting scheme. Crucially, when choosing the size of NTRU secret keys just right, we are able to drop the ring dimension down to 2048 from 4096 as used in [ABGS23] whilst maintaining a 128-bit security level.

The resulting voting scheme thus represents a significant efficiency improvement over the state-of-the-art. This is displayed by the communications costs shown in Table 1. Whilst we do not implement our scheme in code, we expect the running times to also improve on those given in [ABGS23] due to the reduction in ring dimension and modulus for the ring over which computations are performed.

Finally, we think it interesting to observe that when classical encryption schemes like ECDH are compared to their optimized post-quantum counterparts like Kyber, the lattice-based scheme usually comes with a communication cost increase of roughly  $\times 30$ . Comparing our voting scheme to the classical one based on finite field ElGamal, we incur a cost of  $\times 20$  in ciphertext size suggesting that the penalty of our post-quantum security design is somewhat inherent in the lattice structure and that our design may be approaching what can be optimally achieved.

Scheme	Ciphertexts	Shuffle Output $\pi_{S_i}$	Dist. Dec. Output $\pi_{D_j}$	Total Size
[ABGS23]	$80\tau$ KB	$370\tau$ KB	$157\tau$ KB	$607\tau$ KB
Ours	$15\tau$ KB	$130\tau$ KB	$85\tau$ KB	$230\tau$ KB

**Table 1.** Sizes of ciphertexts, shuffle proofs, decryption proofs, and total size as compared to [ABGS23]. Here,  $\tau$  denotes the number of ballots cast.

## 1.2 Related Works

*NTRU Cryptanalysis.* The most relevant work in the area of NTRU fatigue is that of Ducas and Van Woerden [DvW21] as mentioned in the previous section. 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 lie in the overstretched range (for ternary secrets). This bound was improved upon by the work of [DvW21] as we discuss in Section 3.

*Electronic Voting.* In [ABGS23], the authors provide a verifiable mix-net and verifiable distributed decryption protocol based on RLWE and RSIS, 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<sup>+</sup>21]; a building block we also adopt. The work of del Piño et al. [dLNS17] gives a practical voting protocol based on homomorphic counting but this system does not scale well for voting systems with more complex ballots.

Another work worth mentioning is the shuffle by [CMM19] which was implemented in [FWK21], however, it is less efficient than [ABGS23]. More theoretical works include [HMS21], [Str19], and [CGGI16], but none of these are efficient enough to likely be considered for practical deployment.

Recently, the authors of [HMS21] gave a new proof of correct shuffle based on Beneš networks and sub-linear lattice-based proofs for arithmetic circuit satisfiability. However, the scheme is not implemented and the example parameters do not take the soundness slack of the amortised zero-knowledge proofs into account. Moreover, [CMM19, FWK21, HMS21, HMS21] do not consider the decryption of ballots, which would heavily impact the parameters of the protocols in practice. [BHM20] gives a fast decryption mix-net, but it cannot achieve universal verifiability and is thus not suitable for real-world elections.

### 1.3 Organisation

The paper is organised as follows: We give some mathematical background in Section 2, followed by our analysis of the NTRU problem in Section 3. Next, we present the improved voting scheme in Section 4 and detail the performance improvements for our voting instantiation in Section 5.

## 2 PRELIMINARY

*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  $\rho$  and sampling from (or executing)  $D$  with randomness  $\rho$ , respectively. We denote by  $\text{Perm}[i]$  the set of permutations of the integers  $\{1, \dots, i\}$ . 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.

*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 lying 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.

Throughout this document we will view elements  $a \in R$  interchangeably as both polynomials and vectors in  $\mathbb{Z}^d$  (in the coefficient embedding). For such elements, we can thus consider their  $\ell_1$ ,  $\ell_2$ , and  $\ell_\infty$ -norms, extending their definitions in the natural way for  $k$ -dimensional vectors  $\mathbf{a} \in R^k$ . For an elements  $a \in R_q$ , we may sometimes write  $a \bmod p$  to mean that  $a$ 's coefficients are reduced modulo  $p$  so that  $(a \bmod p) \in R_p$ .

For  $a, b \in R$ , we have that  $\|ab\|_\infty \leq \|a\|_1 \cdot \|b\|_\infty$  and  $\|ab\|_\infty \leq \|a\|_2 \cdot \|b\|_2$ . Furthermore, let  $S_\nu$  denote the set of all elements  $a \in R$  such that  $\|a\|_\infty \leq \nu$ .

## 2.1 Lattice Assumptions

We will need the following three computational problems over lattices. In Section 3 we will take a close look at the hardness of the NTRU problem whilst our voting scheme in Section 4 additionally relies on the ring learning with errors (RLWE) assumption and the ring short integer solutions (RSIS) assumption.

*The NTRU Problem.* We choose to give the historic presentation of the NTRU problem as it is more convenient for our analysis in Section 3. We note that some works refer the search/decisional variants of this problem as the ‘search/decisional short polynomial ratio’ problems, (S/D)SPR. Furthermore, one can consider the so-called ‘module’ NTRU problem [CPS+20] which considers the ratio of matrices of polynomials  $\mathbf{F}$  and  $\mathbf{G}$ , (S/D)SMR. Our 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  $D_{\sigma_{\text{NTRU}}}$  be a distribution over  $R_q$ . Sampling  $(f, g) \leftarrow D_{\sigma_{\text{NTRU}}}^2$  with rejection if  $f$  is not invertible in  $R_q$ , define  $h = g/f \in R_q$ . The search-NTRU $_{q,d,\sigma_{\text{NTRU}}}$  problem is, given  $h$ , to recover any rotation  $(X^i f, X^i g)$  of the pair  $(f, g)$ . The decision-NTRU $_{q,d,\sigma_{\text{NTRU}}}$  problem is, given  $h$ , to decide if  $h$  is computed as  $h = g/f$  where  $(f, g) \leftarrow D_{\sigma_{\text{NTRU}}}^2$  or if  $h$  is sampled uniformly from  $R_q$ .*

*Ring Learning With Errors and Ring Short Integer Solution.* 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,  $D_{\sigma_{\text{RLWE}}}$  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 RLWE $_{d,q,Q,\sigma_{\text{RLWE}}}$  problem is defined as  $\text{Adv}_{d,q,Q,\sigma_{\text{RLWE}}}^{\text{RLWE}}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{O}_{\text{RLWE}}}(1^\lambda) \rightarrow 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathbb{S}}}(1^\lambda) \rightarrow 1]|$ , where oracles  $\mathcal{O}_{\text{RLWE}}$  and  $\mathcal{O}_{\mathbb{S}}$  are defined as

- $\mathcal{O}_{\text{RLWE}}$  : Samples  $a \leftarrow R_q$ ,  $(s_1, s_2) \leftarrow \sigma_{\text{RLWE}}^2$ , and output  $(a, as_1 + s_2)$ ;
- $\mathcal{O}_{\mathbb{S}}$  : Samples  $(a, b) \leftarrow R_q \times R_q$  and 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  $\text{RSIS}_{d,q,\beta}$  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 NTRU Encryption

In this work we will use the provably-secure variant of the NTRU cryptosystem first presented by Steinfeld and Stehlé in [SS13]. This scheme relies on the hardness of both the RLWE and NTRU assumptions. Note we make two minor modifications so as 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  $\ell_2$  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  $\sigma_{\text{NTRU}} \in \mathbb{R}$  and  $D_{\sigma_{\text{NTRU}}}$  a discrete Gaussian distribution over  $R$  with standard deviation  $\sigma_{\text{NTRU}}$ ,  $t \in (1, 2]$  and  $\nu \in \mathbb{N}$ . Let the setup parameters be  $\text{sp} = (d, p, q, \sigma_{\text{NTRU}}, t, \nu)$ . The NTRU encryption scheme is described in Figure 1.

---

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

1. Sample  $f$  from  $D_{\sigma_{\text{NTRU}}}$ ; if  $(f \bmod q) \notin R_q^\times$  or  $f \not\equiv 1 \in R_p$ , resample.
2. Sample  $g$  from  $D_{\sigma_{\text{NTRU}}}$ ; if  $(g \bmod q) \notin R_q^\times$ , resample.
3. If  $\|f\|_2 > t \cdot \sqrt{d} \cdot \sigma_{\text{NTRU}}$  or  $\|g\|_2 > t \cdot \sqrt{d} \cdot \sigma_{\text{NTRU}}$ , restart.
4. Return the secret key  $\text{sk} = f$ ,  $\text{pk} = h := g/f \in R_q$ .

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

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

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

1. Compute  $m = (f \cdot c \bmod q) \bmod p$ .
  2. Return the plaintext message  $m$ .
- 

**Fig. 1.** The encryption scheme  $\text{NTRUEncrypt}$  adapted from [SS13].



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

### 2.3 The BDLOP Commitment Scheme

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

*Commitments.* We can commit to elements in  $R_q$  using the BDLOP commitment scheme [BDL<sup>+</sup>18]. For simplicity, we present the scheme instantiated over rings instead of modules, committing to only one ring element at a time:

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

$$\text{pk}_C = \begin{bmatrix} \mathbf{a}_1 & 0 \\ \mathbf{a}_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_3 & 1 \end{bmatrix}.$$

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

$$\text{com} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} = \begin{bmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_3 & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ x \end{bmatrix} = \llbracket x \rrbracket.$$

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

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

$$f \cdot \text{com} \stackrel{?}{=} \begin{bmatrix} 1 & a_1 & a_2 & 0 \\ 0 & 1 & a_3 & 1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ f \cdot x \end{bmatrix} \text{ and } \forall i \in [3]: \|r_i\|_2 \stackrel{?}{\leq} 4 \cdot \sigma_{\text{Com}} \sqrt{d}.$$

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  $\ell_\infty$  norm  $B_{\text{Com}}$  over a lattice of dimension  $2 \cdot d$ . Furthermore, the scheme is binding if the RSIS problem is hard for vectors of  $\ell_2$  norm  $16\sigma_{\text{Com}}\sqrt{\kappa d}$  over a lattice of dimension  $2 \cdot d$  [BDL<sup>+</sup>18].

### 3 NTRU HARDNESS

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] has shown this weakening of the NTRU problem to occur when the modulus  $q$  is very large compared to the ring dimension  $d$  and the secret vectors are small. Naturally, these works seek to determine the turning point at which  $q$  becomes large enough for these attacks to kick in and so security concerns arise. We herein refer to this point as the *fatigue point*.

#### 3.1 Extending the NTRU Analysis

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

*Ducas-van Woerden Analysis.* In their recent paper [DvW21], 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. To facilitate their analysis, they identify two lattice reduction events used to distinguish standard regimes from their overstretched counterparts.

- 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 lattice basis. Such an event has been shown to shortly precede an SKR event by a cascading of further DSD events or to enable decryption of fresh ciphertexts itself.

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<sup>3</sup> 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 concrete relation<sup>4</sup>.

<sup>3</sup> See [github.com/WvanWoerden/NTRUFatigue](https://github.com/WvanWoerden/NTRUFatigue) for their code and experiments.

<sup>4</sup> For the experiments to run without error for the highest dimensions  $d$ , larger variance  $\sigma^2$ , and modulus  $q$  used in [DvW21] and our paper, a high-precision floating point library such as QD is required, otherwise a vanilla fpyll installation should suffice. Installation instructions for this are found in the fpyll documentation at [fpyll.readthedocs.io/en/latest](https://fpyll.readthedocs.io/en/latest).

*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  $D_\sigma$  of standard deviation  $\sigma$ . For convenience, the analysis of [DvW21] models the ternary secret case by sampling  $f, g \leftarrow D_\sigma$  with  $\sigma^2 = 2/3$ . Observe that by varying  $\sigma$  one can model any secret key size and thus we will herein consider that  $f$  and  $g$  are always sampled according to some Gaussian  $D_\sigma$ . The natural question arises:

*Does the choice of (secret size)  $\sigma$  influence the position of the fatigue point and if so, what is its impact?*

To get some intuition on this, we recall the work of Steinfeld and Stehlé [SS13] in which the authors show how, selecting  $\sigma$  sufficiently large, gives rise to a public key  $h = g/f$  that is statistically indistinguishable from uniform when  $f$  and  $g$  are sampled from  $D_\sigma$ . Moreover, they show that using such parameters allows one to remove the NTRU assumption from a proof of the NTRU cryptosystem altogether. Crucially, the  $\sigma$  needed for statistical security depends on the size of  $q$  and  $d$ . This suggests that fixing  $q$  and  $d$  and increasing  $\sigma$  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  $\sigma$ , both the SKR and DSD events 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 1, one needs the relation  $\|p(gs + fe) + fm\|_\infty < q/2$  to hold. Clearly, using larger secrets  $f$  and  $g$  thus leads to less favorable 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 [DvW21] also allows for NTRU hardness estimations using any choice of  $\sigma$  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], with which to analyze 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 [DvW21], Section 3.2, confirms that the asymptotic relation  $q = d^{2.484+o(1)}$  holds regardless of the value of  $\sigma$ . This suggests that if the value of  $\sigma$  does play a role in the concrete, average case,

relation then it manifests in the leading constant. We can thus infer that, for some function  $\psi$  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  $\psi$ , we consider a range of  $\sigma \in [2, 2^2, \dots, 2^{20}]$ . For each  $\sigma$ , we perform a loglog-linear regression on the estimated fatigue points overall prime ring dimensions  $199, \dots, 499$ . This mimics the calculations of [DvW21] 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.

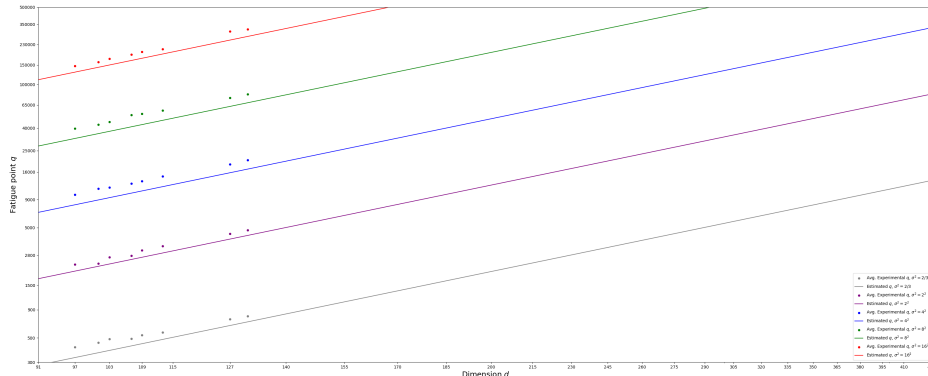
Next, we consider the predicted fatigue points as a geometric series. This reveals the predicted average-case fatigue point to be

$$q = 0.0058 \cdot \sigma^2 \cdot d^{2.484}. \tag{1}$$

Moreover, the precision of this relation across all  $\sigma$  considered is very high. Whilst we could extend this part of our analysis to larger  $\sigma$ , it is highly unlikely that, for cryptographic applications, one would need to take  $\sigma$  higher than  $2^{20}$ . We note also that, setting  $\sigma^2 = 2/3$  we recover the fatigue point determined for the ternary case in [DvW21].

This gives a definitive answer to our question as to the impact of  $\sigma$  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 estimated trend. These results are displayed in Figure 2. We also give a second figure (Figure 3) in which we plot  $q/\sigma^2$  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  $\sigma$ . As was observed by Lucas 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 some potential explanations for this discrepancy, pointing to the slope parameter used in the estimator which is perhaps not well calibrated for such small block sizes. In practice though, this small error only translates to a difference of 2 or 3 in the block size needed to run BKZ and thus hardly affects the predicted bit security at all. Importantly, our experiments show that this error remains constant even at larger moduli.

*The Significance of  $\sigma^2$ .* Having determined the impact of  $\sigma$  on the concrete fatigue point for NTRU we reflect on the structure of Equation (1). 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  $(d, q, \sigma)$ , this constraint is satisfied but the corresponding NTRU instance does not provide adequate security. Let us then increase  $q$  by a constant factor  $\delta$ , say. According to the constraint, this gives room for an increase in  $\sigma$  to  $\delta \cdot \sigma$ . Then Equation (1) tells us that the new fatigue point for the set  $(d, \delta \cdot q, \delta \cdot \sigma)$  increases by a factor of  $\delta^2$ . The important observation here is that, while increasing  $q$  in the first move might weaken the NTRU instance, the same increase permitted for  $\sigma$  actually gives rise to a *net*



**Fig. 2.** Average experimental fatigue point  $q$  values plotted against estimated fatigue point using progressive BKZ with 8 tours on matrix NTRU instances with variance  $\sigma^2 \in \{2/3, 4, 16, 64, 256\}$ . The straight colored lines show the estimated values using the (modified) estimator from [DvW21]. The colored dots show the experimental results, where a DSD event has a 50% chance of triggering before an SKR event. The plot is scaled to  $\log q$  and  $\log d$ .

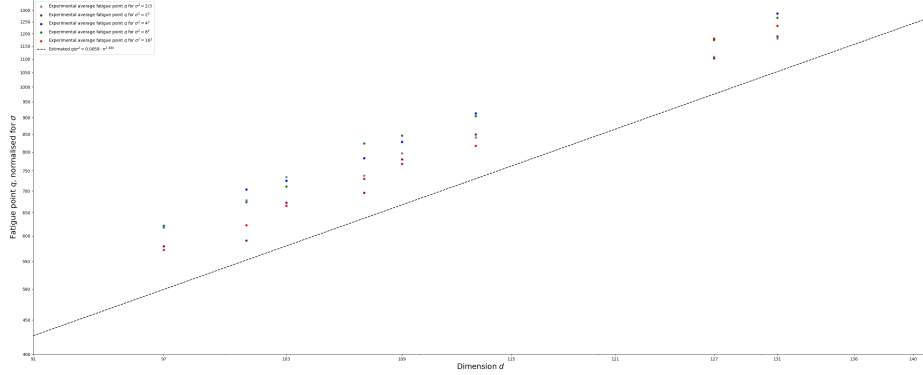
*increase in the hardness of the instance.* In summary, Equation (1) tells us that it is possible to ‘win’ this cat-and-mouse game for NTRU that so often arises when setting lattice parameters.

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

### 3.2 Implications for Existing Work

While the authors of [DvW21] 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. It is our belief that some parameters may fall into the range where the dense sublattice attack begins to kick in and thus one needs to use the techniques described in the previous section to set parameters.

Here, we revisit some NTRU-based schemes in the literature, applying the estimator of [DvW21] to determine their concrete NTRU security. In the cases we examine, the parameters chosen fall short of providing the security levels claimed. However, as suggested by Equation (1), we are able to carefully re-select parameters so that a small increase in the size of NTRU secrets yields the security bump-up needed.



**Fig. 3.** Average experimental values for  $q/\sigma^2$  illustrate that the fatigue point, when adjusted for  $\sigma$ , is modeled by  $q/\sigma^2 = 0.0058 \cdot d^{2.484}$ . The plot is scaled to  $\log_2(q/\sigma^2)$  and  $\log_2 d$ .

*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 that is 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 over-stretched territory. Moreover, these values give rise to only 63 bits of security when run through the estimator of [DvW21] rather than the claimed 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 NTRU secrets that the corresponding public key is *statistically* indistinguishable from uniform. The latter strategy removes the dependence on the NTRU assumption altogether and follows from the regularity lemma in [SS11]. Let us consider the impact of these strategies in turn.

Suppose we begin by increasing the ring dimension in [dK22] from 2048 to 4096 (for security and implementation benefits  $d$  should be a power of 2).

After 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.

Alternatively, one could try creating a statistically uniform public commitment key. This can be done, as shown in [SS11], by using Gaussian NTRU secrets with  $\sigma_{\text{NTRU}} > q^{1/2}$ <sup>5</sup>. Then, as dictated by the parameter constraints in [dK22], one must take (in simplified terms) the decryption parameter  $p > \sigma_{\text{NTRU}}$  and then  $q > p \cdot \sigma_{\text{NTRU}} > q$ . Thus, it is not possible to set parameters (of any size) using statistically uniform NTRU public keys.

The above approaches are the two standard ones taken in the literature for ensuring NTRU hardness. We now exhibit the benefits of the relation Equation (1), as revealed by our analysis, when applied to the problem of setting NTRU parameters. With the same ring dimension  $d = 2048$ , we increase  $\sigma_{\text{NTRU}}$  (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 (1) 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 the 128 bit security threshold is reached:

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

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 the doubling of the final signature altogether. Note that without these changes, the scheme would not satisfy the blinding property of the protocol since the signer could extract the message to be signed from the LinHC commitment.

*Linear Homomorphic Commitments [Kat21].* As demonstrated by the scheme in [dK22], the LinHC protocol of [Kat21] is a useful tool for bootstrapping an existing NIZK to one allowing straight-line extraction. This enables one to avoid rewinding techniques that cause difficulties in settings like the QROM and CCA-security. Therefore, this technique is being adopted by many schemes. However, since the most efficient instantiation of LinHC relies on the NTRU assumption, we wish to highlight the importance of setting parameters here according to the analysis of [DvW21].

*Summary.* We note that 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 might also push up the modulus of the ring over which a scheme is defined as in the examples above. However, in some situations, the scheme may

---

<sup>5</sup> In fact one must take  $\sigma_{\text{NTRU}} > \sqrt{d \log(2d(1+2q))/\pi} \cdot q^{1/2}$  but the simplified bound suffices for the purposes of our argument.

also rely on the hardness of other computational problems such as RLWE, as in our voting scheme, which is defined over the same ring. In this case, the RLWE problem may no longer be hard for the adjusted parameters and one may need to increase the ring *dimension* in order to find a parameter set for which both problems are hard. This could 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 demonstrate the advantages of this approach when presenting our voting scheme in Section 4 where our fine-grained analysis allows us to dramatically bring down the overall communication cost.

## 4 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, to ensure that the election output is correct. In this paper we follow the approach of Aranha et al. [ABGS23] and show how their protocol can be improved in terms of efficiency by replacing the underlying BGV encryption scheme with the NTRU encryption scheme and setting parameters based on our prior NTRU hardness analysis.

To ease the presentation of our construction, we give a passively secure voting scheme in Section 4.2 and then show how this can be lifted to yield a verifiable voting scheme in Section 4.3. We begin with an overview of the voting system structure.

### 4.1 Voting Overview

*Setup Phase.* A trusted set of players run 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. The generated public parameters  $\text{sp}$  are given to every participant, while the decryption key shares  $\text{dk}_j$  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  $\text{pk}$  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] which proves, in zero-knowledge, that the submitted ciphertext contains a genuine ballot.



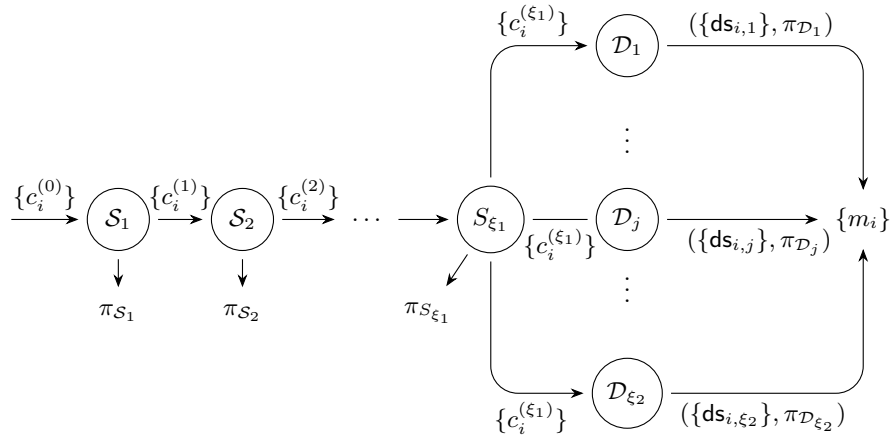
*Counting Phase.* This is divided into three sequential processes. First, the set of encrypted ballots is passed through a series of shuffle servers.

The  $\xi_1$  shuffle servers  $\mathcal{S}_1, \dots, \mathcal{S}_{\xi_1}$  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 can be verified by anyone. We may refer to this whole shuffle process as the *mix-net*.

Each of the  $\xi_2$  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  $\{ds_{ij}\}$ , one for each of the ciphertexts. Finally, it creates a proof of decryption to guarantee that it computed its decryption shares correctly. Each decryption server then 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 verify, it then recovers the ballots originally cast by combining the decryption shares.

A schematic of these processes, in the actively secure setting, is shown in Figure 4. This figure is taken from [ABGS23, Figure 1] and shows the full voting protocol beginning with input 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 to 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.



**Fig. 4.** The voting protocol with verifiable mix-net and distributed decryption, taken from [ABGS23, Figure 1] and adapted with our notation.

## 4.2 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  $\tau$  be the number of voters,  $\xi_1$  the number of shuffle servers, and  $\xi_2$  the number of decryption servers. A passively secure cryptographic voting scheme  $\Pi_{\text{PVote}}$  consists of five PPT algorithms (KeyGen, Cast, Shuffle, DDec, Comb).*

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

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

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

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

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

We now instantiate the above algorithms, present our passively secure voting scheme, and give an overview in Figure 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  $\sigma_{\text{NTRU}}, B_{\text{Dec}}, B_{\text{Drown}} \in \mathbb{R}^+$ ,  $t \in (1, 2]$ , and  $\nu, \tau, \xi_1, \xi_2 \in \mathbb{N}$ . Let  $\text{sp} = (d, p, q, \sigma_{\text{NTRU}}, t, \nu, \tau, \xi_1, \xi_2)$ .

## 4.3 Actively secure scheme

Here we present the actively secure (verifiable) voting scheme. In brief, the construction utilises the zero-knowledge proofs of Section 4.4 and applies them to the passive scheme  $\Pi_{\text{PVote}}$  presented in Figure 5. In particular, any third party can now verify that the processes of shuffling and distributed decryption were carried out correctly without compromising the privacy or integrity of the voting system. We assume a trusted dealer for key generation and leave the construction of an NTRU-based distributed key generation to future work. See Figure 6.

We note that this construction implicitly defines both a verifiable mix-net and PKE with distributed decryption from NTRU. We consider these of independent interest and thus now give an overview of their workings as stand-alone protocols.

**Definition 5 (Actively secure voting scheme).** *Let  $\tau$  be the number of voters,  $\xi_1$  the number of shuffle servers, and  $\xi_2$  the number of decryption servers. An actively secure (verifiable) cryptographic voting scheme  $\Pi_{\text{AVote}}$  consists of five PPT algorithms (KeyGen, Cast, Shuffle, DDec, Comb) with the following syntax:*

---

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

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

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

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

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

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

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

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

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

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

---

**Fig. 5.** The passively-secure voting scheme  $\Pi_{\text{PVote}}$ .

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

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

**Shuffle $_A(\{c_i\}_{i \in [\tau]}) \rightarrow (\{\hat{c}_i\}_{i \in [\tau]}, \pi_S)$**  : On input a set of encrypted ballots  $\{c_i\}_{i \in [\tau]}$  it returns another set of encrypted ballots  $\{\hat{c}_i\}_{i \in [\tau]}$  and a shuffle proof  $\pi_S$ .

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

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

We continue by giving a high-level description of the verifiable shuffle and verifiable distributed decryption procedures.

*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 that any third-party can verify that this process has been performed correctly without compromising the privacy of the mix.

To add this verifiability, we apply the scheme of [ABG<sup>+</sup>21] which allows one to prove a shuffle of openings of the lattice commitments in Section 2.3. We denote this proof system  $\Pi_{\text{SHUF}}$ . Since NTRU ciphertexts only consist of a single element, we can import their scheme without modification where the committed messages are ciphertexts. We also employ the  $\Pi_{\text{SMALL}}$  proof systems described in Section 4.4 to prove that the new ciphertext noise is sufficiently bounded. At a high level, our verifiable shuffle of NTRU ciphertexts  $c_1, \dots, c_\tau$  follows the same framework as [ABGS23]:

1. The shuffle server creates encryptions  $c'_1, \dots, c'_\tau$  of 0 and commits to these as  $\llbracket c'_i \rrbracket$  for each  $i \in [\tau]$ . Run the  $\Pi_{\text{SMALL}}$  protocol to prove that each committed ciphertext is honestly computed.
2. Adding the original ciphertexts  $c_i$  to these commitments homomorphically yields commitments  $\llbracket \hat{c}_i \rrbracket$  to ciphertexts with the same plaintext as in  $c_1, \dots, c_\tau$ , now with fresh randomness.
3. The server now reveals the openings  $\hat{c}_i$  in a randomly permuted order and runs the  $\Pi_{\text{SHUF}}$  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  $\llbracket \text{dk}_j \rrbracket$  to each secret key share  $\text{dk}_j$ , 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  $E_{ij} \leftarrow S_{B_{\text{Drown}}}$ , computes a decryption share  $\text{ds}_{ij} = \text{dk}_j \cdot c_i + p \cdot E_{ij}$  and commits to the noise as  $\llbracket E_{ij} \rrbracket$ .
2. For each  $i \in [\tau]$ , it uses the  $\Pi_{\text{Lin}}$  protocol to prove that the linear decryption equation above is computed honestly with respect to  $\llbracket \text{dk}_j \rrbracket$  and  $\llbracket E_{ij} \rrbracket$ .
3. For each  $i \in [\tau]$ , it uses the  $\Pi_{\text{Bnd}}$  protocol to prove that  $\llbracket E_{ij} \rrbracket$  is an honestly created commitment and the committed value is bounded by  $B_{\text{Drown}}$ .

We note that this framework closely follows the decryption protocol in [ABGS23], but for the decryption protocol we use a proof system that yields an exact upper bound on the noise values, instead of a relaxed bound, to be able to keep the system parameters smaller when ensuring correct decryption. We now detail the complete verifiable voting scheme in Figure 6.

*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  $\sigma_{\text{NTRU}}, B_{\text{Dec}}, B_{\text{Drown}}, B_{\text{Com}}, B_{\text{Small}} \in \mathbb{R}^+$ ,  $t \in (1, 2]$ , and  $\nu, \tau, l, \xi_1, \xi_2 \in \mathbb{N}$ . Let  $\text{sp} = (d, p, q, \sigma_{\text{NTRU}}, t, \nu, \tau, l, \xi_1, \xi_2, B_{\text{Dec}}, B_{\text{Drown}}, B_{\text{Com}})$ .

#### 4.4 Zero-Knowledge Proofs

Here we present the proof systems needed in the actively secure voting protocol. These proofs

*Rejection Sampling.* To ensure that the proofs do not leak any information about the secret values, we use rejection sampling to force the output to be independent of the secrets [Lyu09, Lyu12]. For an overview of rejection sampling techniques, see Appendix A.2.

*Proof of Linearity.* The protocol  $\Pi_{\text{LIN}}$  produces a proof that a committed value  $v$  is a multiple of another committed value  $u$  with respect to a public scalar  $g$ . The exact relation for the proof system is:

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

The proof of linearity  $\pi_{\text{LIN}}$  is computed as follows [BDL+18]:

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

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

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

*Proof of Shuffle.* The protocol  $\Pi_{\text{SHUF}}$  produces a proof that a set of committed values is a permutation of a set of public values. The exact relation for the proof system is:

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

The proof of shuffle  $\pi_{\text{SHUF}}$  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  $\theta_i$  and commit to random linear combinations of the form  $\llbracket D_i \rrbracket = \llbracket \theta_{i-1} \cdot u'_i + \theta_i \cdot \bar{u}'_i \rrbracket$  (where  $\theta_0 = \theta_\tau = 0$ ).

---

**KeyGen<sub>A</sub>(sp).** On input system parameters  $\text{sp}$ :

1. Run  $(\text{sk} = f, \text{pk} = h) \leftarrow \text{KeyGen}_{\text{NTRU}}(d, p, q, \sigma_{\text{NTRU}}, t)$ .
2. For  $j \in [\xi_2 - 1]$ , sample  $\text{dk}_j \leftarrow \mathcal{U}(R_q)$  and set  $\text{dk}_{\xi_2} = \text{sk} - \sum_{j=1}^{\xi_2-1} \text{dk}_j \pmod q$ .
3. For all  $j \in [\xi_2]$ , compute the commitments and openings  $(\llbracket \text{dk}_j \rrbracket, \mathbf{r}_{\text{dk}_j}) \leftarrow \text{Com}(\text{dk}_j)$ .
4. Return public key  $\text{pk}_A = (\text{pk}, \llbracket \text{dk}_1 \rrbracket, \dots, \llbracket \text{dk}_{\xi_2} \rrbracket)$ , secret key  $\text{sk}_A = \text{sk}$ , and decryption key shares  $\{\text{dk}_{A,j} = (\text{dk}_j, \mathbf{r}_{\text{dk}_j})\}_{j \in [\xi_2]}$ .

**Cast<sub>A</sub>(pk<sub>A</sub>, v).** On input a public key  $\text{pk}_A$  and a vote  $v$  in  $R_p$ , retrieving  $\text{pk}$  from  $\text{pk}_A$ :

1. Compute  $c \leftarrow \text{Enc}_{\text{NTRU}}(\text{pk}, v)$ , and return encrypted ballot  $c$ .

**Shuffle<sub>A</sub>({c<sub>i</sub>}<sub>i ∈ [τ]</sub>).** On input a set of encrypted ballots  $\{c_i\}_{i \in [\tau]}$ :

1. For each  $i \in [\tau]$ , compute  $c'_i \leftarrow \text{Enc}_{\text{NTRU}}(\text{pk}, 0)$  using encryption randomness  $(s'_i, e'_i)$ .
2. For each  $i \in [\tau]$ , commit to  $c'_i$  as  $\text{com}'_i := \llbracket c'_i \rrbracket \leftarrow \text{Com}(\text{pk}_C, c'_i)$  where  $\mathbf{r}_{c'_i}$  is the commitment randomness used. Then denoting

$$\mathbf{A}_M = \begin{bmatrix} 1 & a_{1,1} & a_{1,2} & 0 & 0 \\ 0 & 1 & a_{2,2} & ph & p \end{bmatrix},$$

and  $\mathbf{s}_{c'_i} = [\mathbf{r}_{c'_i}, s'_i, e'_i]^T$  compute  $\pi_{\text{Small},i} \leftarrow \Pi_{\text{SMALL}}$ , for matrix  $\mathbf{A}_M$ , input vector  $\mathbf{s}_{c'_i}$ , targets  $\text{com}_i$ , and bound  $B_{\text{Small}}$ . Set  $\pi_{\text{Small}} := \{\pi_{\text{Small},i}\}_{i \in [\tau]}$ .

3. For each  $i \in [\tau]$ , compute  $\hat{c}_i = c_i + c'_i \pmod q$ . Sample  $\pi \leftarrow \text{Perm}([\tau])$ , and compute  $\pi_{\text{Shuf}} \leftarrow \Pi_{\text{SHUF}}$  with input commitments  $\{\llbracket \hat{c}_i \rrbracket\}_{i \in [\tau]}$ , randomness  $\{\mathbf{r}_{c'_i}\}_{i \in [\tau]}$ , ciphertexts  $\{\hat{c}_i\}_{i \in [\tau]}$ , and permuted ciphertexts  $\{\hat{c}_{\pi(i)}\}_{i \in [\tau]}$ .
4. Return  $(\{\hat{c}_{\pi(i)}\}_{i \in [\tau]}, \pi_S)$ , where  $\pi_S = (\{\text{com}'_i\}_{i \in [\tau]}, \pi_{\text{Small}}, \pi_{\text{Shuf}})$ .

**DDec<sub>A,j</sub>({c<sub>i</sub>}<sub>i ∈ [τ]</sub>, dk<sub>A,j</sub>).** On input a set of ciphertexts  $\{c_i\}_{i \in [\tau]}$  and decryption key share  $\text{dk}_{A,j} = (\text{dk}_j, \mathbf{r}_{\text{dk}_j})$ :

1. For each  $i \in [\tau]$ , sample  $E_{ij} \leftarrow S_{B_{\text{Down}}}$ , and compute  $\mathbf{ds}_{ij} = \text{dk}_j \cdot c_i + p \cdot E_{ij}$ .
2. For each  $i \in [\tau]$ , compute  $(\llbracket E_{ij} \rrbracket, \mathbf{r}_{E_{ij}}) \leftarrow \text{Com}(E_{ij}, \text{pk}_C)$  and use the  $\Pi_{\text{LIN}}$  protocol to compute a proof  $\pi_{\text{Lin}_{ij}}$  for the linear relation  $\mathbf{ds}_{ij} = \text{dk}_j \cdot c_i + p \cdot E_{ij}$ .
3. Each commitment is of the form

$$\llbracket E_{ij} \rrbracket = \underbrace{\begin{bmatrix} 1 & a_{1,1} & a_{1,2} & 0 \\ 0 & 1 & a_{2,2} & 1 \end{bmatrix}}_{\mathbf{A}_D} \underbrace{\begin{bmatrix} \mathbf{r}_{E_{ij}} \\ E_{ij} \end{bmatrix}}_{\mathbf{s}_{ij}},$$

where  $\|\mathbf{r}_{E_{ij}}\|_{\infty} \leq B_{\text{Com}}$ . Now, applying the amortized proof  $\Pi_{\text{BND}}$  from Section 4.4, create a proof  $\pi_{\text{Bnd}}$  that, for all  $i \in [\tau]$ , the noise  $\|E_{ij}\|_{\infty} \leq B_{\text{Down}}$ .

4. Return  $\mathbf{ds}_j := (\{\mathbf{ds}_{ij}\}_{i \in [\tau]}, \pi_{\mathcal{D}})$ , where  $\pi_{\mathcal{D}} = (\{\llbracket E_{ij} \rrbracket\}_{i \in [\tau]}, \{\pi_{\text{Lin}_{ij}}\}_{i \in [\tau]}, \pi_{\text{Bnd}})$ .

**Comb<sub>A</sub>({c<sub>i</sub>}<sub>i ∈ [τ]</sub>, {ds<sub>j</sub>}<sub>j ∈ [ξ<sub>2</sub>]</sub>).** On input a set of encrypted ballots  $\{c_i\}_{i \in [\tau]}$  and decryption shares  $\{\mathbf{ds}_j\}_{j \in [\xi_2]}$ :

1. Parse  $\mathbf{ds}_j$  as  $(\{\mathbf{ds}_{ij}\}_{i \in [\tau]}, \pi_{\mathcal{D}_j})$ , and verify the proofs  $\pi_{\text{Lin}_{ij}}$  and  $\pi_{\text{Bnd},ij}$ .
2. If any verification protocol returns 0 then output  $\perp$ . Otherwise, compute

$$v_i = \left( \sum_{j \in [\xi_2]} \mathbf{ds}_{ij} \pmod q \right) \pmod p.$$

3. Return the set of votes  $\{v_i\}_{i \in [\tau]}$ .
- 

**Fig. 6.** The verifiable voting scheme  $\Pi_{\text{AVote}}$ .

3. Hash the commitments to get a uniform challenge  $\beta$ . Then for all  $i \in [\tau]$  compute  $s_i$  so that it solves the linear system with respect to  $\beta$ .
4. For all  $i \in [\tau]$ , compute proofs of linearity for the commitment equations of the form  $\llbracket D_i \rrbracket = s_{i-1} \llbracket u'_i \rrbracket + s_i \cdot \bar{u}'_i$  (where  $s_0 = \beta$  and  $s_\tau = (-1)^\tau \beta$ ).

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

*Amortized Proof of Shortness.* The protocol  $\Pi_{\text{SMALL}}$ , produces a proof that a batch of equations  $\mathbf{A}\mathbf{s}_i = \mathbf{t}_i$  for  $i \in [\ell]$  is satisfied for a set of secret vectors  $\mathbf{s}_i$  with  $\ell_\infty$  norm bounded by  $\nu$ . The exact relation for the proof system is:

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

The proof of shortness  $\pi_{\text{SMALL}}$  is quite involved, combining error-correcting codes, Merkle trees, Lagrange interpolation and proximity testing, and we refer to [ABGS23] for details. The size of the proof, for batch size  $\ell$  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  $\gamma$ , message length  $\mu$  and minimal distance  $\iota$  where  $\mu = d(k + 2) + \eta \leq \gamma < q$  for encoding randomness of length  $\eta$ .  $\lambda$  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 some choice of message length  $\mu'$  such that  $\mu \leq \mu' \leq \gamma < q$ .

*Amortized Proofs of Boundedness.* We define  $\Pi_{\text{BND}}$  to be a slightly adapted version of the  $\Pi_{\text{SMALL}}$ , protocol. The previous work by Aranha et al. [ABGS23] used the amortised relaxed proofs by Baum et al. [BBC<sup>+</sup>18] to get smaller proof sizes of the cost of slightly increasing the general parameters in the voting scheme because of the slack inherent in the proof system. In practice this lead to a slightly larger modulus  $q$  but 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  $\ell'$  and secret vectors bounded in the  $\ell_\infty$  norm by  $B_{\text{Drown}}$ , is:

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

Since  $\Pi_{\text{SMALL}}$ , 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  $E = [b_0 \ b_1 \ \dots \ b_\zeta] \circ [1 \ b \ \dots \ b^\zeta]$  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  $\mathbb{Z}_q^d$ ):

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

Here, the ring element  $E_{ij}$  is decomposed, and all 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  $\Pi_{\text{SMALL}}$  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  $\zeta = \lceil \log_3 q \rceil - 1$ .

#### 4.5 Security Analysis

Here we analyze the security of our (verifiable) voting scheme presented in Figure 6. We begin by examining the implicit shuffle and distributed decryption protocols and prove their security. Then, we will consider the security of the overall voting scheme, showing how the security of the two aforementioned sub-protocols gives rise to the notions of integrity and privacy as required by an electronic voting scheme.

We begin by examining the security of the verifiable shuffle protocol implicitly defined by the tuple of algorithms  $\Pi_{\text{AShuf}} := (\text{KeyGen}_A, \text{Cast}_A, \text{Shuffle}_A, \text{Dec}_{\text{NTRU}})$ . We say that  $\Pi_{\text{AShuf}}$  is *secure* if it satisfies the properties of shuffle completeness, shuffle soundness, and shuffle simulatability. We refer the reader to Appendix A.3 for formal definitions of these notions.

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



*Proof sketch.* Since  $\Pi_{\text{SMALL}}$ , and  $\Pi_{\text{SHUF}}$  are complete, the protocol will finish and the shuffle will verify correctly. Moreover, since  $B_{\text{MIX}} + B_{\text{Dec}} \leq \lfloor q/2 \rfloor$ , decryption will be correct. Thus,  $\Pi_{\text{AShuf}}$  is complete.

Now assume we have an adversary  $\text{Adv}$  breaking the knowledge soundness of the shuffle. That is, given a set of input ciphertexts,  $\text{Adv}$  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  $\pi_{\text{Small}}$ , one can extract commitment randomness  $\mathbf{s}_{c'_i} = [\mathbf{r}c'_i, s'_i, e'_i]^T$  such that  $\mathbf{A}_M \mathbf{s}_{c'_i} = \mathbf{t}'_i$ . By knowledge soundness of  $\pi_{\text{Shuf}}$ , 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  $\mathbf{A}_M \tilde{\mathbf{s}}_{c'_i} = \mathbf{t}'_i$ . 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  $\pi_{\text{Small}}$ , and  $\pi_{\text{Shuf}}$  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 commitment scheme and IND-CPA security of  $\text{NTRUEncrypt}$ .

We now analyse the security of the PKE with distributed decryption implicitly defined by the tuple of algorithms  $\Pi_{\text{ADDec}} := (\text{KeyGen}_A, \text{Cast}_A, \text{Dec}_{\text{NTRU}}, \text{DDec}_A, \text{Comb}_A)$ . We say that  $\Pi_{\text{ADDec}}$  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 A.4. 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 3 ( $\Pi_{\text{ADDec}}$  Security).** *Let  $B_{\text{Drown}} = 2^{\text{sec}}(B_{\text{Dec}}/p\xi_2) < \lfloor q/2 \rfloor$ . Furthermore, assume  $\text{NTRUEncrypt}$  is IND-CPA secure and the protocols  $\Pi_{\text{LIN}}$  and  $\Pi_{\text{BND}}$  are complete, HVZK, and sound. Then the verifiable, distributed decryption protocol  $\Pi_{\text{ADDec}}$  is secure.*

*Proof sketch.* IND-CPA security of  $\Pi_{\text{ADDec}}$  follows trivially from the IND-CPA security of the underlying NTRU encryption scheme and the HVZK property of the proof systems  $\Pi_{\text{LIN}}$  and  $\Pi_{\text{BND}}$ .

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

For threshold verifiability we also consider only ciphertexts such that  $P_{\text{sk}_A}(c) = 1$ . Note that if  $\text{Comb}$  accepts a ciphertext for which decryption is incorrect then,

for some  $j$ , no relation  $\mathbf{ds}_{ij} = \mathbf{dk}_j \cdot c_i + pE_{ij}$  holds for an  $E_{ij}$  of norm at most  $B_{\text{Down}}$ . This implies either an adversary against the soundness of  $\Pi_{\text{LIN}}$  or one against the soundness of  $\Pi_{\text{BND}}$ .

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 by the hiding property of the BDLOP commitment scheme. Next, one can simulate the proofs  $\pi_{\text{Lin}_j}$  and  $\pi_{\text{Bnd}}$ , else a distinguisher breaks the HVZK of the respective proof systems. Lastly, the simulatability of the  $\mathbf{ds}_{ij}$  follows from the choice of  $\text{sec}$  which is chosen so that  $\mathbf{ds}_{ij}$  is statistically indistinguishable from uniform.

We now argue our voting protocol’s overall security by discussing its integrity and privacy properties. We give a very high-level overview of the security properties a voting system requires and how our design achieves these properties. For a detailed overview and the formal proofs, we refer the reader to [ABGS23]. Whilst some building blocks are different in our work, they are combined in much the same way as the aforementioned work.

*Integrity.* This property ensures that no adversary is able to cause inconsistent decryption or non-unique decryption, even if the adversary obtains all of the key material. It can be shown that an adversary succeeding in causing one of these events can be used to construct an adversary against either the threshold verifiability of the distributed decryption protocol  $\Pi_{\text{ADDec}}$  or the soundness of the shuffle protocol  $\Pi_{\text{AShuf}}$ .

*Privacy.* Privacy dictates that an adversary seeing the contents of the ballot box, the intermediate shuffles, and the decrypted ballot shares should not be able to determine who cast which ballot. This should hold even if the adversary learns some decryption key shares, inserts adversarially generated ciphertexts into the ballot box, introduce adversarially generated intermediate shuffles, and publish adversarially chosen decrypted ballot shares. This reduces to the NTRU cryptosystem’s CPA-security, commitments hiding, and zero-knowledge properties of the underlying proofs.

## 5 PERFORMANCE

We analyse the practical performance of our voting scheme. We begin by identifying all system parameters and any constraints that apply to them. These are displayed in Table 2. Next, we compute a sample set of parameters that satisfy the necessary constraints and give rise to a minimum of 128 bits of security. Table 3 displays these values. Finally, using these parameters, we compute the concrete communication cost of our voting system. The resulting sizes are compared to the previous work of [ABGS23], revealing a significant improvement in the state-of-the-art for cryptographic voting from quantum-safe assumptions. These results are displayed in Table 4.

## 5.1 Setting Parameters

We begin by collecting all parameters of the scheme and noting any constraints applying to them. These are displayed in Table 2.

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. Let us examine this relation in more depth.

*Decryption Correctness.* After passing through the mix-net of  $\xi_1$  shuffle servers, a ciphertext is of the form

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

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

$$v' = \left( \sum_{j \in [\xi_2]} \mathbf{ds}_j \pmod{q} \right) \pmod{p}.$$

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  $\lfloor q/2 \rfloor$ . A simple calculation shows that a sufficient constraint for this correct decryption is the following

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

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

*Computational Security* . Having chosen parameters satisfying the constraints of Table 2, 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 3. Having settled on a ring dimension  $d$  and modulus  $q$  giving sufficient hardness of the RLWE problem, we use (2) to determine the maximum standard deviation permissible for generating the NTRU secrets  $(f, g)$ . Finally, following the

Parameter	Explanation	Constraints
$\lambda$	Computational security parameter	$\geq 128$
sec	Statistical security parameter	$\geq 40$
$d$	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 $q = 1 \pmod{2d}$ s.t. $\ \sum_{j \in \xi_2} ds_j\ _\infty \leq \lfloor q/2 \rfloor$
$t$	KeyGen <sub>NTRU</sub> rejection parameter	Set for rej. prob. $< 1/1000$ .
$k$	Length of binding vector in BDLOP commitment	$> 2$
$\mathcal{C}$	Challenge space for linear ZK proofs of commitments	$\mathcal{C} = \{c \in R_q \mid \ c\ _\infty = 1, \ c\ _1 = \kappa\}$
$\kappa$	Maximum $\ell_1$ -norm of elements in $\mathcal{C}$	$2^\kappa \cdot \binom{d}{\kappa} > 2^\lambda$
$B_{\text{Com}}$	Bound on the commitment noise	So that SIS is hard
$B_{\text{Drown}}$	Infinity norm of noise drowning term $E_{ij}$	$B_{\text{Drown}} = 2^{\text{sec}}(B_{\text{Dec}}/p\xi_2)$
$\sigma_{\text{NTRU}}$	Standard deviation for encryption secret key	So that NTRU is hard
$\nu$	Bound on encryption randomness	So that LWE is hard
$\sigma_{\text{Com}}$	Standard deviation in ZK proofs of linear relations	Chosen to be $\sigma_{\text{Com}} = \kappa \cdot B_{\text{Com}} \cdot \sqrt{kd}$
$\xi_1, \xi_2$	Number of shuffle and decryption-servers	At least two servers
$\tau$	Total number of messages/number of voters	For soundness we need $(\tau^\delta + 1)/ R_q  < 2^{-\lambda}$
$\eta$	Reed-Solomon encoding randomness length	Make soundness $\geq 2^{-\lambda}$ in $\Pi_{\text{SMALL}}$ , and $\Pi_{\text{BND}}$
$\ell_{\text{Small}}$	Proof batch size in $\Pi_{\text{SMALL}}$	Same secret length as in [ABGS23]
$\ell_{\text{Bnd}}$	Proof batch size in $\Pi_{\text{BND}}$	Same secret length as in [ABGS23]
$\mu_{\text{Small}}$	Reed-Solomon message length in $\Pi_{\text{SMALL}}$	$\mu_{\text{Small}} = (k+2) \cdot d + \eta$
$\mu_{\text{Bnd}}$	Reed-Solomon message length in $\Pi_{\text{BND}}$	$\mu_{\text{Bnd}} = (k+1) \cdot d + \eta$
$\mu'_{\text{Small}}$	Reed-Solomon message dimension in $\Pi_{\text{SMALL}}$	$\mu_{\text{Small}} \leq \mu'_{\text{Small}} \leq \gamma < q$
$\mu'_{\text{Bnd}}$	Reed-Solomon message dimension in $\Pi_{\text{SMALL}}$	$\mu_{\text{Bnd}} \leq \mu'_{\text{Bnd}} \leq \gamma < q$
$\gamma_{\text{Small}}$	Reed-Solomon code length in $\Pi_{\text{BND}}$	$\mu_{\text{Small}} \leq \mu'_{\text{Small}} \leq \gamma < q$
$\gamma_{\text{Bnd}}$	Reed-Solomon code length in $\Pi_{\text{BND}}$	$\mu_{\text{Bnd}} \leq \mu'_{\text{Bnd}} \leq \gamma < q$

**Table 2.** System parameters and constraints.

procedure described in Section 3, one can calculate the estimated computational complexity of the given NTRU instance. Again, we employ the conservative formula  $0.292\beta + 16.4 + \log_2(8d)$  used in works such as [DTGW17, SPL<sup>+</sup>17, BIP<sup>+</sup>22] to compute bit-security from an estimated blocksize  $\beta$ .

Finally, 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  $2^{(2\sqrt{d \log_2 q \log_2 \delta})}$ . Here,  $\delta$  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 meets this threshold by far.

## 5.2 Sample Parameter Set and Total Size

Table 3 gives a sample set of parameters generated by following the process described in the previous section.

Parameter	Explanation	Value
$\lambda$	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
$t$	KeyGen <sub>NTRU</sub> rejection parameter	1.058
$\nu$	Infinity norm of encryption randomness	1
$B_{\text{Com}}$	Infinity norm of commitment randomness	1
$\xi_1, \xi_2$	Number of shuffle and decryption servers	4
$\sigma_{\text{NTRU}}$	Standard deviation for encryption secret key	7.12
$\eta$	Reed-Solomon encoding randomness length	325
$\ell_{\text{Small}}$	Proof batch size in $\Pi_{\text{SMALL}}$ ,	9830
$\ell_{\text{Bnd}}$	Proof batch size in $\Pi_{\text{BND}}$	12288
$\mu_{\text{Small}}$	Reed-Solomon message length in $\Pi_{\text{SMALL}}$ ,	10565
$\mu_{\text{Bnd}}$	Reed-Solomon message length in $\Pi_{\text{BND}}$	8517
$\mu'_{\text{Small}}$	Reed-Solomon message dimension in $\Pi_{\text{SMALL}}$ ,	23988
$\mu'_{\text{Bnd}}$	Reed-Solomon message dimension in $\Pi_{\text{BND}}$	181550
$\gamma_{\text{Small}}$	Reed-Solomon code length in $\Pi_{\text{SMALL}}$ ,	26616
$\gamma_{\text{Bnd}}$	Reed-Solomon code length in $\Pi_{\text{BND}}$	198668

**Table 3.** Sample parameter set.

In Table 4 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  $\pi_{\mathcal{D}_j}$  which comprises decryption shares, commitments, linearity proofs, and proofs of 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 which are three times smaller and decryption server outputs which are half of those in [ABGS23]. Overall this represents a significant efficiency gain over previous works as summarized in Table 1.

### 5.3 Future Improvements

We provide some directions for future work to potentially improve our results:

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.

Scheme	$c_i$	$\llbracket R_q \rrbracket$	$\pi_{\text{Shuf}}$	$\pi_{\text{Lin}_{ij}}$	$\pi_{\text{Small}}$	$\pi_{\text{Bnd}}$	$\pi_{S_i}$	$\pi_{D_j}$
[ABGS23]	80 KB	80/120 KB	$150\tau$ KB	35 KB	$20\tau$ KB	$2\tau$ KB	$370\tau$ KB	$157\tau$ KB
Ours	15 KB	30 KB	$63\tau$ KB	18 KB	$22\tau$ KB	$22\tau$ KB	$130\tau$ KB	$85\tau$ KB

**Table 4.** Sizes of ciphertexts, commitments, and proof for our scheme and the previous work in [ABGS23]. Note, the two commitment sizes in [ABGS23] reflect the commitments to the noise-drowning terms and ciphertexts respectively.

2. *Improved noise analysis.* Our results can possibly be improved using techniques in [AKSY22, BS23, CSS<sup>+</sup>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. *Implementation of our scheme.* We have not implemented the scheme with our improved parameters but we expect an improvement of more than  $\times 2$  in performance because of the halved ring dimension as well as the modulus that decreases from two words to only one compared to [ABGS23].
4. *Improved parameters in other schemes.* Our extended NTRU analysis might lead to more efficient FHE parameters in [BIP<sup>+</sup>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. We also thank Ngoc Khanh Nguyen for useful discussions on zero-knowledge arguments. Finally, 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 experiments of Section 3.

## 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. 2, 5, 10
- ABG<sup>+</sup>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. 6, 20, 21
- ABGS23. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. ACM CCS, 2023. <https://eprint.iacr.org/2022/422>. 4, 5, 6, 16, 17, 20, 23, 24, 25, 26, 28, 29, 30, 36, 37

- ABLR21. Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice reduction with approximate enumeration oracles - practical algorithms and concrete performance. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 732–759, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. [2](#)
- ADH<sup>+</sup>19. Martin R. Albrecht, Léo Ducas, Gottfried Herold, Elena Kirshanova, Eamonn W. Postlethwaite, and Marc Stevens. The general sieve kernel and new records in lattice reduction. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 717–746, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. [2](#)
- Ajt96. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108, Philadelphia, PA, USA, May 22–24, 1996. ACM Press. [2](#), [7](#)
- 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. [30](#)
- 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. [2](#), [11](#), [27](#)
- BBC<sup>+</sup>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. [23](#)
- BDL<sup>+</sup>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. [9](#), [21](#)
- 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](#), [5](#)
- 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. [6](#)
- BIP<sup>+</sup>22. Charlotte Bonte, Iliia 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. [28](#), [30](#)
- 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. [36](#)
- BS23. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. Cryptology ePrint Archive, Report 2023/016, 2023. <https://eprint.iacr.org/2023/016>. [30](#)
- 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. [6](#)
- 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. [2](#)
- 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. Cryptology ePrint Archive, Paper 2016/139, 2016. <https://eprint.iacr.org/2016/139>. [5](#)
- 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. [6](#)
- CPS<sup>+</sup>20. Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. ModFalcon: Compact signatures based on module-NTRU lattices. In Hung-Min Sun, Shih-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. [7](#)
- CSS<sup>+</sup>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>. [30](#)
- 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. [4](#), [14](#), [15](#)
- 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. [6](#)
- 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>. [28](#)
- DvW21. 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. [3](#), [4](#), [5](#), [6](#), [10](#), [11](#), [12](#), [13](#), [14](#), [15](#)
- FWK21. Valeh Farzaliyev, Jan Willemsen, and Jaan Kristjan Kaasik. Improved lattice-based mix-nets for electronic voting. In *Information Security and Cryptology – ICISC 2021*. Springer International Publishing, 2021. [6](#)
- HBD<sup>+</sup>20. Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS<sup>+</sup>. 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](#)
- HMS21. 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. [6](#)
- 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](#), [7](#)
- HS22. Audhild Høgåsen and Tjerand Silde. Return codes from lattice assumptions. *E-VOTE-ID*, 2022. [29](#)
- 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. [14](#), [15](#)
- 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. [2](#), [6](#), [10](#)
- Klu22. Kamil Klucznik. 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. [30](#)

- LDK<sup>+</sup>20. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrede 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, 3
- 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. 16
- 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. 21, 36
- 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. 7
- LW20. Changmin Lee and Alexandre Wallet. Lattice analysis on MiNTRU problem. Cryptology ePrint Archive, Report 2020/230, 2020. <https://eprint.iacr.org/2020/230>. 10
- Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. 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. 21, 35
- 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. 35
- MR09. Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 2, 28
- PFH<sup>+</sup>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

- 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](#), [3](#), [7](#)
- SAB<sup>+</sup>20. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancreède 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](#)
- SPL<sup>+</sup>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>. [28](#)
- 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. [3](#), [5](#), [14](#), [15](#)
- SS13. Damien Stehlé and Ron Steinfeld. Making NTRUEncrypt and NTRUSign as secure as standard worst-case problems over ideal lattices. Cryptology ePrint Archive, Report 2013/004, 2013. <https://eprint.iacr.org/2013/004>. [8](#), [11](#)
- 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. [6](#)

## A Omitted Preliminary

In this section, we provide the omitted preliminaries.

### A.1 Tail Bounds.

We use the following standard results for Gaussian-sampled vectors.

**Lemma 4** ( [[MR04](#), [Lyu12](#)] ). *For any real  $t > 0$  and  $t' > 1$ , we have*

$$\Pr[x \leftarrow D_{\mathbb{Z}^n, \sigma} : \|x\|_{\infty} > t\sigma] < 2n \cdot 2^{-\frac{\log e}{2} \cdot t^2},$$

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

## A.2 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  $\sigma$  and we want the output vector  $\mathbf{z}$  to be from the same distribution. The procedure is shown in Figure 7.

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

$$\max \frac{\mathcal{N}_\sigma^k(\mathbf{z})}{\mathcal{N}_{\mathbf{v},\sigma}^k(\mathbf{z})} \leq \exp \left[ \frac{24\sigma\|\mathbf{v}\|_2 + \|\mathbf{v}\|_2^2}{2\sigma^2} \right] = M \quad (3)$$

where we use the tail bound from Appendix A.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  $\langle \mathbf{z}, \mathbf{v} \rangle \geq 0$ , and hence, we can set  $M = \exp(\|\mathbf{v}\|_2/2\sigma^2)$ . Then, for  $\sigma = 0.675\|\mathbf{v}\|_2$ , we get  $M \approx 3$ . In Figure 7, 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  $\mu$  in the case  $b = 1$  but allowing a smaller standard deviation.

Rej( $\mathbf{z}, \mathbf{v}, b, M, \sigma$ )

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. 7. Rejection Sampling.

## A.3 Security of Mixing (Shuffle)

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

**Definition 6 (Mixing Completeness).** *We say that the mixing protocol  $\Pi_{\text{MIX}}$  is complete if for honest PPT parties **Prover** and **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 **Verifier** accepts the proof and the output ciphertexts decrypt to the same set of messages as the input ciphertexts. Hence, we want that*

$$\Pr \left[ 1 \leftarrow \text{Verifier}(\text{pp}, \text{pk}, \{c_i\}_{i \in [\tau]}, \{\hat{c}_i\}_{i \in [\tau]}, \pi) : \begin{array}{l} (\text{pp}, \text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\kappa) \\ \{c_i\}_{i \in [\tau]} \leftarrow \text{Enc}(\text{pk}, \{m_i\}_{i \in [\tau]}) \\ (\{\hat{c}_i\}_{i \in [\tau]}, \pi) \leftarrow \text{Prover}(\text{pp}, \text{pk}, \{c_i\}_{i \in [\tau]}) \end{array} \right] \leq 1 - \epsilon(\lambda),$$

where the probability is taken over  $\text{KeyGen}$ ,  $\text{Enc}$  and  $\text{Prover}$ .

**Definition 7 (Mixing Soundness).** We say that the mixing protocol  $\Pi_{\text{MIX}}$  is sound if a dishonest PPT adversary  $\text{Adv}$  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  $\text{Verifier}$  accepts the proof but the output ciphertexts decrypt to a different set of messages than the input ciphertexts. Hence, we want that

$$\Pr \left[ 1 \leftarrow \text{Verifier}(\text{pp}, \text{pk}, \{c_i\}_{i \in [\tau]}, \{\hat{c}_i\}_{i \in [\tau]}, \pi) : \begin{array}{l} (\text{pp}, \text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\kappa) \\ \{c_i\}_{i \in [\tau]} \leftarrow \text{Enc}(\text{pk}, \{m_i\}_{i \in [\tau]}) \\ (\{\hat{c}_i\}_{i \in [\tau]}, \pi) \leftarrow \text{Adv}(\text{pp}, \text{pk}, \{c_i\}_{i \in [\tau]}) \end{array} \right] \leq \epsilon(\lambda),$$

where the probability is taken over  $\text{KeyGen}$ ,  $\text{Enc}$  and  $\text{Adv}$ .

**Definition 8 (Mixing Simulatability).** We say that the mixing protocol  $\Pi_{\text{MIX}}$  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

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

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

#### A.4 Security of Distributed Decryption

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

**Definition 9 (PKE with Distributed Decryption).** A PKE scheme with distributed decryption consists of five algorithms: key generation ( $\text{KeyGen}$ ), encryption ( $\text{Enc}$ ), decryption ( $\text{Dec}$ ), distributed decryption ( $\text{DDec}$ ), and combine ( $\text{Comb}$ ), where

- $\text{KeyGen}$  On input security parameter  $1^\lambda$  and number of key-shares  $\xi_2$ , outputs public parameters  $\text{pp}$ , a public key  $\text{pk}$ , a secret key  $\text{sk}$ , and key-shares  $\{\text{sk}_j\}$ ,
- $\text{Enc}$  On input  $\text{pk}$  and messages  $\{m_i\}$ , outputs ciphertexts  $\{c_i\}$ ,
- $\text{Dec}$  On input  $\text{sk}$  and ciphertexts  $\{c_i\}$ , outputs messages  $\{m_i\}$ ,
- $\text{DDec}$  On input a secret key share  $\text{sk}_{j^*}$  and ciphertexts  $\{c_i\}$ , outputs decryption shares  $\{\text{ds}_{i,j^*}\}$ ,
- $\text{Comb}$  On input ciphertexts  $\{c_i\}$  and decryption shares  $\{\text{ds}_{i,j}\}$ , outputs either messages  $\{m_i\}$  or  $\perp$ ,

and  $\text{pp}$  are implicit inputs to  $\text{Enc}$ ,  $\text{Dec}$ ,  $\text{DDec}$  and  $\text{Comb}$ .

**Definition 10 (Chosen Plaintext Security).** We say that the public key encryption scheme is secure against chosen plaintext attacks if an adversary  $\text{Adv}$ , 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

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

where the probability is taken over  $\text{KeyGen}$  and  $\text{Enc}$ .

**Definition 11 (Threshold Correctness).** We say that the public key distributed encryption scheme is threshold correct with respect to  $P_{\text{sk}}(\cdot)$  if the following probability equals 1:

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

where the probability is taken over  $\text{KeyGen}$  and  $\text{DDec}$ .

**Definition 12 (Threshold Verifiability).** A PKE scheme with distributed decryption is threshold verifiable with respect to  $P_{\text{sk}}(\cdot)$  if an adversary  $\mathcal{A}$  corrupting  $J \subseteq [\xi_2]$  secret key shares  $\{\text{sk}_j\}_{j \in J}$  cannot convince  $\text{Comb}$  to accept maliciously created decryption shares  $\{\text{ds}_{i,j}\}_{i \in [\tau], j \in J}$ . More concretely, the following probability is bounded by a negligible  $\epsilon(\lambda)$ :

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

where the probability is taken over  $\text{KeyGen}$  and  $\text{DDec}$ .

**Definition 13 (Distributed Decryption Simulatability).** A PKE scheme with distributed decryption is simulatable with respect to  $P_{\text{sk}}(\cdot)$  if an adversary  $\mathcal{A}$  corrupting  $J \subseteq [\xi_2]$  secret key shares  $\{\text{sk}_j\}_{j \in J}$  cannot distinguish the transcript of the decryption protocol from a simulation by a simulator  $\text{Sim}$  which only gets  $\{\text{sk}_j\}_{j \in J}$  as well as correct decryptions as input. More concretely, the following probability is bounded by a negligible  $\epsilon(\text{sec})$ :

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

where the probability is taken over  $\text{KeyGen}$ ,  $\text{DDec}$ ,  $\text{Sim}$ .