# A Toolkit for Succinct Lattice-Based Zero Knowledge Proofs

Beatrice Biasioli[1,2], Madalina Bolboceanu[1,2], Vadim Lyubashevsky[1], Antonio Merino-Gallardo[1,2], Michał Osadnik[3], Gregor Seiler[1], and Patrick Steuer[1]

[1] IBM Research Europe, Zurich, Switzerland
beatrice.biasioli@ibm.com, madalina.bolboceanu@ibm.com, vad@zurich.ibm.com, antonio@m-g.es, gseiler@posteo.net, ick@zurich.ibm.com
[2] University of Potsdam, Potsdam, Germany
[3] Aalto University, Espoo, Finland michal.osadnik@aalto.fi

**Abstract.** The development of proof systems whose security relies on the hardness of lattice problems has been a fruitful research area in recent years. By leveraging the techniques introduced in LaBRADOR (Beullens, Seiler, Crypto 2023), the state-of-the-art lattice-based schemes have very fast provers and have output sizes under 100KB for arbitrarily large statements. These proofs are in fact the smallest, and often have the fastest provers, out of all post-quantum schemes.
In addition to succinctness, many applications also require witness privacy. Achieving this can, in theory, be done by combining LaBRADOR with a linear-size zero-knowledge proof. While such a combination has already been described in the LaBRADOR paper itself, as well as in the works of Albrecht et al. (Eurocrypt 2024) and del Pino et al. (Crypto 2025), its concrete costs remained unexplored.
In this work, we provide the first concrete construction and implementation that adds zero-knowledge proofs to LaBRADOR by integrating the linear-size zero-knowledge proof from (Lyubashevsky, Nguyen, Plançon, Crypto 2022) into the protocol. We describe the non-trivial challenges that this entails and show practicality of the construction by benchmarking several use-cases. We make the proof system and primitives accessible by extending the LaZer library (Lyubashevsky, Seiler, Steuer, CCS 2024) in a way that they can easily be used in other applications.

**Keywords:** Lattice Cryptography, Zero-Knowledge, Succinct Proofs, Implementation, Privacy, Quantum-Safe

## 1 Introduction

Succinct lattice-based proof systems have been shown to be competitive with proof systems based on other hardness assumptions. In recent times, there has been a lot of progress, and many concretely efficient lattice-based constructions have been proposed, e.g. [13, 15, 17, 18, 31, 40].

The two other main categories of succinct proof systems are pairing-based SNARKs, e.g. [21, 27, 29], and hash-based STARKs, e.g. [10, 11, 43]. SNARKs offer extremely small constant-size proofs but are not secure against quantum computers and have relatively slow provers. STARKs are quantum safe, do not need any structured computational assumptions, and offer faster runtimes, but produce quite large proofs. In comparison, lattice-based proof systems have the potential to combine quantum-safety and fast runtimes with proof sizes that are smaller than those of the STARKs.

Indeed, the computationally friendly mathematical structure of lattice commitments that are simple high-dimensional linear functions over polynomial rings has been shown to allow for lattice-based polynomial commitment schemes that have leading prover runtimes [31,40] and competitive verifier runtimes in the single-digit millisecond range [31]. Polynomial commitment schemes (PCSs) are the core cryptographic component in modern proof system constructions. At the same time, the proof sizes of these lattice-based PCSs are leading among quantum-safe PCSs.

An important but somewhat neglected feature in current succinct lattice-based proof systems and their implementations is the zero-knowledge property. Only the linear-size `LNP` proof system [34] and its implementation in the LaZer library [38] come with zero-knowledge.

For many proof system applications, the succinctness property is the main enabling feature. A typical example is signature aggregation, where a large number of signatures are compressed into a small proof that can be verified in very little time. Here, the signatures do not need to be hidden as there is no privacy requirement, and hence the proof system does not need to have the zero-knowledge property.

The situation is completely different in privacy-preserving applications where zero-knowledge is the central feature provided by the proof system. Typical application examples are blind signature schemes and anonymous credential systems constructed with the help of zero-knowledge proofs. In such protocols, users possess signatures from some underlying basic signature scheme, but they do not want to reveal these signatures in the clear, as this would make them traceable. Instead, the users only provide zero-knowledge proofs of possession of the signatures and hence keep the signatures private.

In light of the ongoing and pressing transition to quantum-safe cryptography, it is also important to transition advanced privacy-preserving cryptographic protocols such as Privacy Pass, and hence, there is a strong need for practical quantum-safe zero-knowledge proof systems and implementations thereof. Furthermore, given that the main quantum-safe signature scheme Dilithium [25], which is standardized as ML-DSA by NIST, is a lattice scheme, it is often necessary to prove statements about lattice schemes. In this case there are significant synergies when the proof system is also based on lattices. In particular, the proof system then shares a lot of the mathematical structure with the scheme to-be-proven, and hence can be implemented using similar implementation techniques. Furthermore, a large fraction of the operations in the scheme can be phrased as polynomial relations that are native to the proof system. Thus, the proof statement can often be constructed manually, and one does not need to resort to circuit representations and (trusted) compiler tool chains.

This provides further reasons to consider lattice-based zero-knowledge proofs beyond small proof sizes and computational performance. In the present paper we propose a lattice-based proof system that offers both succinct proof sizes as well as zero-knowledge. Concretely, our proof system is a combination of a variant of the `LaBRADOR` proof system with a modification of the `LNP` proof system that we call `LNP-Lite`. While such constructions have been suggested before, we provide the first detailed instantiation and the first implementation of such a proof system.

In more detail, prior works [3,41] have suggested constructing a lattice-based zero-knowledge proof system with succinct proof sizes by proving the input relation with `LNP` in a first step. In `LNP` proofs, only the last prover message (the so-called masked opening $\mathbf{z}$) scales linearly with the witness size. The remaining inner prover messages are sublinear in size, so they can be revealed in the clear. This allows the verifier to compute all Fiat-Shamir hashes. Then, what remains is to prove knowledge of $\mathbf{z}$ fulfilling all verifier checks performed on it in `LNP`. These are all simple polynomial relations of the type supported by `LaBRADOR`. So instead of also outputting the masked opening, we can prove knowledge of it using `LaBRADOR`, resulting in a succinctly-sized zero-knowledge proof.

There are two drawbacks to this approach. First, as also realized by [41], the inner `LNP` prover messages, while sublinear in size, are still concretely quite large, and hence blow up the proof size of the combined proof system. Second, and more importantly, the masking of the secret in $\mathbf{z}$ blows up the size of the witness for `LaBRADOR`, and hence results in poor runtimes. Therefore, a better approach is to first run several iterations of a sublinear-size proof protocol that reduces the witness size. This was already suggested in [13] but without a concrete instantiation. The sublinear-size protocol does not need to be fully zero-knowledge. It suffices if the inner prover messages, which are output, are simulatable. On the contrary, the last prover messages are not output and do not need to be simulatable. They are proven by the next iteration of the protocol, or, in case of the last protocol iteration, proven in zero-knowledge by `LNP`. Finally, one can proceed as before and prove the last prover message of `LNP` with further iterations of `LaBRADOR` where no zero-knowledge requirements exist anymore. In this way, the masking in `LNP` only increases the size of a witness that is much smaller than the initial input witness.

For the sublinear-size proof protocol, we use the improved variant of `LaBRADOR` from [15], which we refer to as `LaBRADOS`. In this protocol all the inner prover messages are wrapped in binding lattice commitments to minimize their size. These commitments can be made simulatable easily by adding sufficient randomness so that the commitments become hiding under LWE. Our `LNP` variant `LNP-Lite` contains commitment

wrappers similarly to [41] in order to reduce the impact on the final proof size. Moreover, we further modify LNP in several ways. Most importantly, we propose a new technique to mask the random projection that allows us to instantiate the protocol with a modulus $q$ small enough for the LaBRADOR implementation. The naive way of masking and committing to the random projection requires moduli that are too big for LaBRADOR. Finally, we implement LNP-Lite on top of the same arithmetic implementation that is used by the LaBRADOR implementation in order to obtain a unified implementation of the complete proof system. Also, we added support for the quadratic input constraints required for proving LaBRADOS. Those constraints are not supported by the LaZer implementation of LNP.

We showcase the performance and practicality of our construction and implementation by using it for proving several useful lattice statements that require sublinear proof sizes as well as zero-knowledge. Concretely, our benchmarks center around proving symmetric-key primitives such as collision-resistant hash functions and PRG evaluation. Such proofs are useful in the construction of numerous high-level protocols. For example, proving PRG evaluation can be used for constructing a verifiable encryption scheme.

## 2 Preliminaries

*Notation* Let $q$ be an odd prime. We denote by $\mathbb{Z}_q$ the ring of integers mod $q$. We write $[n] = \{1, 2, \ldots, n\}$. We say $x \leftarrow S$ when $x \in S$ is sampled uniformly at random from a finite set $S$ and similarly $x \leftarrow \chi$ when $x$ is sampled according to a distribution $\chi$. We denote by $\mathcal{U}(S)$ the uniform distribution over the finite set $S$. Given two moduli $q \geq p \geq 2$, we define the rounding function from $\mathbb{Z}_q$ to $\mathbb{Z}_p$ that maps $x \in \mathbb{Z}_q$ to $\lceil x \rfloor_{q \to p} \coloneqq \lceil \frac{p}{q} \cdot x \rfloor \mod p$ and we extend this function to integer vectors coordinate-wise. Let $d$ be a power of two, and let $\mathcal{R}, \mathcal{R}_q$ be the rings $\mathbb{Z}[X]/(X^d + 1)$ and $\mathbb{Z}_q[X]/(X^d + 1)$, for any integer $q$. Most of the time, $q$ is chosen such that $X^d + 1$ splits in two irreducible factors mod $q$. We work with $d = 256$ or 512. For a polynomial $f = a_0 + a_1 X + \ldots + a_{d-1} X^{d-1} \in \mathcal{R}$, we denote by $\mathtt{ct}(f)$ its constant-term, i.e. $\mathtt{ct}(f) = a_0$. We use $\sigma_{-1}(\cdot)$ to denote the "conjugation automorphism", i.e. $\sigma_{-1} \in \mathsf{Aut}(\mathcal{R})$ such that $\sigma_{-1}(X) = X^{-1}$. Vectors over $\mathbb{Z}$, $\mathcal{R}$, $\mathcal{R}_q$ are denoted by italic-bold lower-case and matrices by italic-bold capital letters. For any polynomial vector $\mathbf{a} \in \mathcal{R}^n$, we denote by $\dim(\cdot)$ their dimension as integer vector, i.e. $\dim(\mathbf{a}) = nd$. For two vectors $\mathbf{a}, \mathbf{b} \in \mathcal{R}^n$, we mean by $\langle \mathbf{a}, \mathbf{b} \rangle$ the $\mathbb{Z}$ inner product of their corresponding coefficient vectors from $\mathbb{Z}^{nd}$. We denote by $\mathbf{1}_r$ the column vector containing all entries equal to (the constant polynomial) 1, and by $\mathbf{0}_r$ the one containing all entries equal to (the constant polynomial) 0. If $\mathbf{a} \in \mathcal{R}_q^{n_a}$ and $\mathbf{b} \in \mathcal{R}_q^{n_b}$ are vectors, we denote by $(\mathbf{a}, \mathbf{b}) \in \mathcal{R}_q^{n_a + n_b}$ their concatenation. A function $f : \mathbb{N} \to \mathbb{R}$ is said to be negligible if for all $c \in \mathbb{N}$ there exists some $n_c \in \mathbb{N}$ such that for all $n > n_c$, $|f(n)| < \frac{1}{n^c}$. In the context of probabilities, we say $a \approx b$ if $a - b$ is negligible. Further notation and additional preliminaries can be found in Section A.

### 2.1 The Challenge Space

The challenge space $\mathcal{C} \subseteq \mathcal{R}_q$ is chosen such that $c - c'$ is invertible for any pair of distinct $c, c' \in \mathcal{C}$ and with bounded $\ell_2$ norm and *operator norm*, i.e. $\|c\| \leq \tau$ and $\|c\|_{\mathrm{op}} \leq T$, for any $c \in \mathcal{C}$, for some constants $\tau, T \in \mathbb{R}$, where $\|c\|_{\mathrm{op}} \coloneqq \sup_{r \in \mathcal{R} \setminus \{0\}} \frac{\|cr\|}{\|r\|}$. A popular choice [13] is to sample $c$ with small $\|c\| = \tau$ by taking a fixed number of short non-zero entries (e.g. $\pm 1, \pm 2$) that ensures a large-enough size of $\mathcal{C}$, while the invertibility is guaranteed by their shortness as proved in [37, Cor.1.2].

### 2.2 Lattice Assumptions

Here we recall the computational lattice problems Module-Short Integer Solution (M-SIS), Module-Learning With Errors (M-LWE), and Module-Learning with Rounding (M-LWR) [25, 32].

**Definition 1 (M-SIS).** *For integers $M, N, q > 0$ and real number $\beta > 0$, we define M-SIS$_{q,M,N,\beta}$ problem as follows: Given $\mathbf{A} \leftarrow \mathcal{R}_q^{M \times N}$, find $\mathbf{v} \in \mathcal{R}_q^N$ such that $\mathbf{A}\mathbf{v} = 0$ and $0 < \|\mathbf{v}\| \leq \beta$.*

**Definition 2 (M-SIS with $\ell_\infty$ norm).** *We define M-SIS$^\infty_{q,M,N,\beta}$ analogously to Definition 1 with $\|\cdot\|$ replaced with $\|\cdot\|_\infty$.*

**Definition 3 (Decision M-LWE).** *For integers $n, m, q > 0$ and $\chi$ a distribution over $\mathcal{R}$, we define the decision M-LWE$_{q,m,n,\chi}$ problem as follows: for $\mathbf{s} \leftarrow \chi^n$, distinguish between the following cases*

- $(\mathbf{A}, \mathbf{As} + \mathbf{e})$, *for* $\mathbf{A} \leftarrow \mathcal{R}_q^{m\times n}, \mathbf{e} \leftarrow \chi^m$,
- $(\mathbf{A}, \mathbf{b})$, *for* $\mathbf{A} \leftarrow \mathcal{R}_q^{m\times n}, \mathbf{b} \leftarrow \mathcal{R}_q^m$.

The Module-Learning with Rounding problem deals with two integer moduli, $q \geq p \geq 2$.

**Definition 4 (Decision M-LWR).** *For integers $n', m', q \geq p > 0$ and $\chi$ be a distribution over $\mathcal{R}$, then the decision M-LWR$_{q,p,m',n',\chi}$ problem as follows: for $\mathbf{s} \leftarrow \chi^{n'}$, distinguish between the following two cases:*

- $(\mathbf{A}, \lceil \mathbf{As} \rfloor_{q\to p})$, *for* $\mathbf{A} \leftarrow \mathcal{R}_q^{m'\times n'}$,
- $(\mathbf{A}, \mathbf{u})$, *for* $\mathbf{A} \leftarrow \mathcal{R}_q^{m'\times n'}, \mathbf{u} \leftarrow \mathcal{R}_p^{m'}$.

We introduce a new version of the Extended M-LWE problem [36], involving two secrets, which is used in proving zero-knowledge of our proofs of knowledge. We show in Section C that the version of this problem without polynomial structure reduces from LWE in knapsack form.

**Definition 5 (Two-Secrets-Extended M-LWE).** *For positive integers $n, m, q$, $\chi_i$ a distribution over $\mathcal{R}$, for $i \in [2]$ a positive real number $\sigma$, a challenge space $\mathcal{C} \subseteq \mathcal{R}_q$ and a distribution $\zeta$ over $\mathcal{C} \times \mathcal{C}$, we define the Extended M-LWE$_{q,m,n,\chi_1,\chi_2,\zeta}$ problem as follows:*

- $(\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_1\mathbf{r}_1, \mathbf{B}_2\mathbf{r}_2, c_1, c_2, \mathbf{z}_1, \mathbf{z}_2, \mathtt{sign}(\langle \mathbf{z}_1, c_1\mathbf{r}_1\rangle + \langle \mathbf{z}_2, c_2\mathbf{r}_2\rangle))$, *for* $\mathbf{B}_i \leftarrow \mathcal{R}_q^{m_i\times n_i}$, $\mathbf{r}_i \leftarrow \chi_i^{n_i}$, $\mathbf{z}_i \leftarrow D_\sigma^{n_i}$, *for any* $i \in [2]$, $(c_1, c_2) \leftarrow \zeta$,
- $(\mathbf{B}_1, \mathbf{B}_2, \mathbf{u}_1, \mathbf{u}_2, c_1, c_2, \mathbf{z}_1, \mathbf{z}_2, \mathtt{sign}(\langle \mathbf{z}_1, c_1\mathbf{r}_1\rangle + \langle \mathbf{z}_2, c_2\mathbf{r}_2\rangle))$, *for* $\mathbf{B}_i \leftarrow \mathcal{R}_q^{m_i\times n_i}$, $\mathbf{u}_i \leftarrow \mathcal{R}_q^{m_i}$, $\mathbf{r}_i \leftarrow \chi_i^{n_i}$, $\mathbf{z}_i \leftarrow D_\sigma^{n_i}$, *for any* $i \in [2]$, $(c_1, c_2) \leftarrow \zeta$.

*where $\mathtt{sign}(a) = 1$ if $a \geq 0$, and 0 otherwise, and each product $c_i\mathbf{r}_i$ is computed without reduction mod $q$.*

### 2.3 LaBRADOR

LaBRADOR is a proof system introduced by Beullens and Seiler [13] whose security is based on the hardness of the M-SIS problem. This proof system enjoys a concretely small communication size, and is tailored for proving lattice-related relations. LaBRADOR has a core base protocol and becomes *succinct* by leveraging recursion: the last message of one iteration of the base protocol becomes the witness for the next one, and the new statement corresponds to the verifier's checks from the previous iteration. In this way, the size of the witness is reduced with each iteration, until no further progress is made, and the prover finally sends the last witness for the verifier to check directly.

More concretely, the LaBRADOR proof system proves knowledge of short witness vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \mathcal{R}^n$, satisfying a *global* norm bound $\sum_{i\in[r]} \|\mathbf{s}_i\|^2 \leq \beta^2$ and dot-product constraints $f : \mathcal{R}_q^n \times \ldots \times \mathcal{R}_q^n \to \mathcal{R}_q$ of the form

$$f(\mathbf{s}_1, \ldots, \mathbf{s}_r) \triangleq \sum_{i,j\in[r]} a_{ij}\langle \mathbf{s}_i, \mathbf{s}_j\rangle + \sum_{i\in[r]} \langle \varphi_i, \mathbf{s}_i\rangle - b, \tag{1}$$

where $a_{ij}, b \in \mathcal{R}_q$, $\varphi_i \in \mathcal{R}_q^n$, such that $a_{ij} = a_{ji}$. For some constraints $f'$, we are only interested in the constant term $\mathtt{ct}(\cdot)$ of their output, so the other coefficients of $b$ are not included in the statement. We collect these *constant term* constraints in a family $\mathcal{F}'$, separated from the *full* constraints, that we collect in a family $\mathcal{F}$. It is worth noting that the witness vectors will actually be proven to satisfy *looser* norm bounds, by a slack factor of $c_{\mathtt{LaBRADOR}} \coloneqq \sqrt{128/30}$, due to the use of random projections.

This proof system is not *zero-knowledge*, i.e., the prover messages leak information about the witness.

**Commitment Layers.** LaBRADOR works with two layers of commitments, that their authors refer to as *inner* and *outer* layers. The inner commitments are Ajtai commitments of the witness vectors $\mathbf{s}_i$, of rank $k_{\mathtt{in}}$,

whereas the outer ones are Ajtai commitments, of rank $k_{\mathtt{mid}}$, of the decompositions of the inner commitments. In anticipation of the additional layer of commitments that we consider in the next section, we refer to the outer commitment layer of LaBRADOR instead as *middle* commitments.

**LaBRADOS.** A recent work [15] introduced a variant of LaBRADOR that further compresses the size of the prover's messages sent in each iteration of the protocol (see [15, Sec. 7.3]), and that serves as an important building block in our construction. We refer to this variant of LaBRADOR as LaBRADOS.

LaBRADOS works by binary-decomposing the original messages of LaBRADOR, and sending *outer* commitments of them. This means that the decompositions need to be included in the last prover message (i.e. the witness for the next iteration) and the verifier has to check that they are indeed binary and the correct openings to the outer commitments. One of the main advantages of this outer layer is that the rank of these commitments can be very small by relying on the hardness of M-SIS for infinity norm bound 1 (following the analysis in [15, Sec. 5.1]).

Another advantage is that, since the projections are no longer sent in the clear, one can prove individual norm bounds on the witness vectors, rather than a global norm bound. The downside is that, rather than checking the exact $\ell_2$-norm of the projections, the verifier only checks that they have been decomposed in binary with a certain number of bits. This leads to a slack factor of $c_{\mathtt{LaBRADOS}} \coloneqq 9.75/0.49$. [16, Thm. 6].

Lastly, the main reason why we use LaBRADOS is that everything the prover sends (except for the last message) are Ajtai (outer) commitments. These can be easily made *hiding* by appending a long enough uniform binary vector before committing and relying on the hardness of the M-LWE problem.

**Outline of LaBRADOR and LaBRADOS** Both protocols share the same blueprint when proving knowledge of witness vectors $\mathbf{s}_i$ satisfying constant-term and full dot-product constraints, as well as $\ell_2$-norm bounds. In Figs. 5 to 8, we provide the LaBRADOR and LaBRADOS protocols. We describe here a shared outline for both protocols, and refer to the discussion above and the figures for the differences between them. At a very high level, the base protocols of LaBRADOR and LaBRADOS perform the following steps:

1. **Commit.** The prover starts by sending commitments to the witness vectors, such that each of them is committed using the same Ajtai matrix. The number of witness vectors in which the witness is split is an important parameter that determines how effective one round of the base protocol is at decreasing the witness size.

2. **Approximate Norm Proof.** The norm of the witness is proven approximately by following the standard approach from [13, 15, 28, 34], which involves projecting the coefficient vectors of the witness using random matrices $\mathbf{\Pi}$ of 256 rows of entries sampled from the $\mathtt{Bin}_1$ distribution (i.e. they take values equal to $\pm 1$ with probability 1/4 each and 0 with probability 1/2). The verifier will check that the norm of these projections is upper bounded by some bound related to the claimed bound on the witness. The correctness of these projections is encoded as additional constant-term constraints to be proven in the protocol.

3. **Aggregate the Constraints.** In this step, the goal of the prover is to derive a single constraint to prove, from the many constant-term and full constraints, following standard techniques [13, 15, 34]. This is realized first by linearly combining (or aggregating) the constant-term constraints with random integer challenges from the verifier. The resulting constraints are lifted into full-domain constraints by having the prover send their evaluation over the full-domain and having the verifier check that their constant coefficient is 0. Together with the input full-domain constraints, these lifted constraints are further aggregated with polynomial challenges into a single dot-product constraint.

4. **Prove the Final Constraint.** To prove this final constraint, the prover sends some auxiliary information (the so-called *garbage terms*) that allows the verifier to check that the final constraint indeed holds. Finally, the prover sends a linear combination of the input witness vectors $\mathbf{s}_i$ with random challenges $c_i \in \mathcal{C}$ from the verifier, namely $\mathbf{z} \coloneqq \sum_{i \in [r]} c_i \mathbf{s}_i$. With this linear combination, the verifier can be convinced of the correctness of the commitments and the garbage terms. In particular, the verifier can check whether multiplying the Ajtai matrix by this vector $\mathbf{z}$ results in the linear combination of the commitments with the challenges $c_i$, and can similarly involve $\mathbf{z}$ and the garbage terms in checks to ensure their consistency.

### 2.4 LNP

Lyubashevsky, Nguyen, and Plançon introduced in [34] a proof system for lattice-related relations that achieves zero-knowledge, and that we refer to as LNP. Unlike LaBRADOR, the proof size of LNP is *linear*, since the size of the last prover message scales linearly with the size of the witness. Its security also relies on the assumed hardness of the M-SIS problem. This proof system proves knowledge of short witness vectors $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_r) \in \mathcal{R}^N$, satisfying a global norm bound $\|\mathbf{s}\|^2 = \sum_{i\in[r]} \|\mathbf{s}_i\|^2 \leq \beta^2$ and constant-term and full domain constraints of $f : \mathcal{R}_q^N \to \mathcal{R}_q, f(\mathbf{s}) = \mathbf{s}^T\mathbf{R}_2\mathbf{s} + \langle \mathbf{r}_1, \mathbf{s}\rangle + r_0$, for a public matrix $\mathbf{R}_2 \in \mathcal{R}_q^{N\times N}$, a public vector $\mathbf{r}_1 \in \mathcal{R}_q^N$ and public $r_0 \in \mathcal{R}_q$. These constraints look more general, compared to the ones LaBRADOR can prove (1). For simplicity of this work, we focus only on LNP proving constraints of type (1), both constant-term and full-domain kinds.

LNP shares the same blueprint with LaBRADOR and LaBRADOS. We provide a short description of LNP in Fig. 3. Here we focus on explaining only the differences between LaBRADOR and LNP at each step, mostly related to ensuring zero-knowledge:

1. **Commit.** The prover also samples additional masks used for ensuring zero-knowledge in the following steps, which are committed using BDLOP [8]. One could also commit instead to their binary decomposition in the Ajtai commitment, along with the witness vectors. Further, this commitment needs to be hiding, for zero-knowledge. For this, a uniform binary vector is sampled, called $\mathbf{r}$, to make the commitment indistinguishable from random under M-LWE.

2. **Approximate Norm Proof.** For zero-knowledge, the projection sent by the prover is masked using a Gaussian mask and rejection sampling [33, 34] is applied, thus making the masked projections follow closely Gaussian distributions. The verifier can check that the norm of this projection is upper bounded by some bound involving the claimed bound on the witness.

3. **Aggregate the Constraints.** In this step, the lifted constraints to be sent are further masked by polynomials with zero constant coefficient and uniform non-constant coefficients, so as not to leak any information about the witness vectors. The verifier can check that their constant coefficient indeed vanishes.

4. **Prove the Final Constraint.** The last LNP prover message consists of $\mathbf{z}_s = \mathbf{y}_s + c\mathbf{s}$, where $\mathbf{s}$ is the input witness, and $\mathbf{z}_r = \mathbf{y}_r + c\mathbf{r}$, where the masks $\mathbf{y}_s$ and $\mathbf{y}_r$ are sampled from public (Gaussian) distributions (for zero-knowledge) and $c \in \mathcal{C}$ is a random challenge from the verifier. One applies rejection sampling [34] on these masked openings $\mathbf{z}_s, \mathbf{z}_r$ to make sure there is no leakage about $\mathbf{s}, \mathbf{r}$. Moreover, the challenge space $\mathcal{C}$ used in LNP is *stable under the automorphism* $\sigma_{-1}$, i.e. $\sigma_{-1}(c) \in \mathcal{C}$, for any $c \in \mathcal{C}$, which allows one to use LNP to prove constraints involving $\sigma_{-1}$, (e.g., the binary check), without committing to the automorphism $\sigma_{-1}$ applied on the witness vector. A commitment $\mathbf{w}$ to the masks $\mathbf{y}_s, \mathbf{y}_r$ is also provided to the verifier, which helps the verifier check, along with the two masked openings, $\mathbf{z}_s$ and $\mathbf{z}_r$, the correctness of the commitment. Further, $\mathbf{z}_s$ and $\mathbf{z}_r$ also help the verifier check that the final constraint indeed holds.

## 3 A Succinct Zero-Knowledge Proof System

In this section, we present our succinct zero-knowledge proof system by combining iterations of LaBRADOS and LNP in the following way. We start by proving knowledge of the input witness with LaBRADOS equipped with hiding outer commitments. The last message of LaBRADOS, i.e., the output witness, is then recursively fed as input to the same protocol, and several iterations are run until the size of the witness is sufficiently small. We then proceed to feed the resulting small witness to LNP-Lite, a compressed version of the linear-size protocol LNP that is tailored to the output of LaBRADOS and that we describe in Section 3.3. The output of LNP-Lite could safely be made public, but we further compress it by running several iterations of LaBRADOS (now with non-hiding outer commitments) (see Section 3.4), and finishing with several iterations of the original LaBRADOR, as done in [15]. Once the size of the final witness cannot be decreased any more, the witness is finally sent to the verifier. We give an overview of our succinct zero-knowledge proof in Fig. 1.

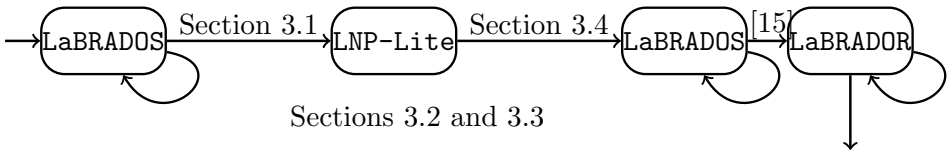


**Fig. 1.** Overview of the succinct zero-knowledge proof system from Section 3.

### 3.1 Understanding the Output of LaBRADOS

The variant of LNP that we describe in the subsequent sections is thus tailored to proving the output of LaBRADOS. In this section, we take a closer look at the latter, with an emphasis on what the output witness looks like and what the output statement to prove is.

LaBRADOS works with the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(X^d+1)$ of degree $d = 256$ and the challenge space $\mathcal{C}$ (Section 2.1). The input of each of the LaBRADOS iterations is made of short witness vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \mathcal{R}^n$, satisfying norm bounds $\|\mathbf{s}_i\| \leq \beta_i$ and constant-term and full-domain dot-product constraints of type (1).

**Output Witness.** As in LaBRADOR, the last message of each of the LaBRADOS iterations contains a linear combination of witness vectors, $\mathbf{z} \coloneqq \sum_{i\in[r]} c_i\mathbf{s}_i$, with random challenges $c_i \leftarrow \mathcal{C}$ from the verifier, along with openings of middle and outer commitments and the random binary vectors used to make the outer commitments hiding. To ensure that the size of the coefficients of $\mathbf{z}$ does not blow up when recursing this protocol, [13] considers decomposing $\mathbf{z}$ in two parts, $\mathbf{z}_0$ and $\mathbf{z}_1$, with respect to a basis $b$ as $\mathbf{z} = \mathbf{z}_0 + b\mathbf{z}_1$. The openings of the middle commitments are collected in $\mathbf{x}_{\mathtt{mid}}$, while the openings to the outer commitments and the random binary vectors for hiding, all being binary vectors, are collected in $\mathbf{x}_{\mathtt{bin}}$. Overall, the output witness is $(\mathbf{z}_0, \mathbf{z}_1, \mathbf{x}_{\mathtt{mid}}, \mathbf{x}_{\mathtt{bin}})$ (computed as in Fig. 7), along with the additional witness vector $\sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$, required to prove $\mathbf{x}_{\mathtt{bin}}$ is binary as we justify below.

**Output Statement.** In order to ensure the binding property of the commitments, all the output witness vectors need to satisfy certain $\ell_2$-norm bounds, which are derived from the bases used to decompose their elements. Additionally, the output vectors satisfy constant-term and full dot-product constraints of type (1), see Fig. 8. They include the final aggregated dot-product constraint, the correctness of all commitment layers, and of the so-called *garbage terms* needed for checking the final aggregated constraint. Finally, we need to perform a binary check for $\mathbf{x}_{\mathtt{bin}}$. For this check, we follow the standard technique [34] of including the additional vector $\sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$ in the witness, and proving its well-formedness together with the appropriate quadratic binary equation (see Section A.3 for more details). Note that we do not need to provide an approximate norm proof for $\sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$, as its norm bound trivially follows from that of $\mathbf{x}_{\mathtt{bin}}$. Overall, the only quadratic checks are the correctness of the quadratic garbage terms, which contains the dot-products $\langle \mathbf{z}_i, \mathbf{z}_j \rangle$ for $i, j \in \{0, 1\}$, and the binary equation, which contains the term $\langle \mathbf{x}_{\mathtt{bin}}, \sigma_{-1}(\mathbf{x}_{\mathtt{bin}}) \rangle$.

### 3.2 Overview of Improvements to LNP

The linear-size zero-knowledge protocol LNP [34] lies at the heart of our construction. The iterations of LaBRADOS run at the beginning of this proof system ensure that the input to LNP is small. Similarly, the iterations of LaBRADOS and LaBRADOR run on the output witness of LNP ensure that the final witness sent in the end to the verifier will be succinct. This section deals with reducing the communication of the LNP protocol itself, used in this composition. We describe the main challenges we overcome and techniques we employ for obtaining a compressed version of LNP. The resulting protocol that we propose is LNP-Lite, and we provide a full description in Section 3.3. We note that since this protocol is meant to be run on the output of LaBRADOS, we make some optimizations tailored to it.

**New Challenge Space and Witness.** For an easier composition of LaBRADOS with our zero-knowledge protocol LNP-Lite, the challenge space $\mathcal{C}$ that we use for this protocol is the same as that used by LaBRADOS (Section 2.1), which is not stable under the automorphism $\sigma_{-1}$, as opposed to that of LNP [34], see Section 2.4.

Thus, when proving the LaBRADOS output vector $\mathbf{x}_{\mathtt{bin}}$ is binary, it is crucial to include $\sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$ in the witness of the LNP-Lite protocol, since the binary check involves both vectors.

**Additional Commitment Layer.** We compress the prover messages by adding an outer commitment layer to all but the last prover message, in a similar way that LaBRADOS is a compressed version of LaBRADOR (Section 2.3). The original LNP commitments are then called *inner* commitments, which in our case are just hiding Ajtai commitments. The outer ones are also Ajtai commitments, but these do not need to be hiding. It is easy to check that extending our protocol with outer commitments preserves its zero-knowledge (see Lemma 6 for a proof).

**Increase of the Projection Slack.** Proving binary with LNP would become impractical if one simply extends [34] with outer commitments. The problem lies in the fact that the masked projections, designed to obtain approximate $\ell_2$ norms on the input witness vectors, are no longer sent directly to the verifier, but only a commitment to their binary decompositions. To provide an approximate $\ell_2$ norm, one would project the coefficients vector of the input using random matrices, mask it using a Gaussian vector and by applying rejection sampling, make it closely follow Gaussian distributions (see Section 2.4). In particular, the masked projection of the LaBRADOS's output binary vector $\mathbf{x}_{\mathtt{bin}} \in \mathcal{R}^n$ would follow, due to rejection sampling, a distribution close to $D_\sigma$, for some $\sigma$, so we could write it using $\log(14\sigma)$ bits. As $\mathbf{x}_{\mathtt{bin}}$ is binary, then $\|\mathbf{x}_{\mathtt{bin}}\| \leq \sqrt{nd}$, so one could set $\sigma \coloneqq \gamma\sqrt{337}\sqrt{nd}$, for a constant $\gamma$, as the projection of this vector does not have $\ell_2$ norm greater than $\sqrt{337} \cdot \sqrt{nd}$, unless with small probability by [34, Lem. 2.8]. Suppose that the prover proves that the decomposed projection is indeed binary and written in that many bits. Then, as $\|\mathbf{x}\| \leq \sqrt{256} \cdot \|\mathbf{x}\|_\infty$, for any $\mathbf{x} \in \mathbb{Z}^{256}$, we can derive an $\ell_2$ norm bound on the masked projection, $\sqrt{256} \cdot 14\sigma$, with high probability. Using [34, Lem.2.9], we could prove that $\|\mathbf{x}_{\mathtt{bin}}\| \leq 2 \cdot 14 \cdot \sqrt{256}/\sqrt{26}\sigma = \alpha\sqrt{nd}$ with high probability, leading to a slack factor $\alpha \coloneqq 2 \cdot 14 \cdot \sqrt{256}/\sqrt{26} \cdot \gamma\sqrt{337} \approx 25806$, for a repetition constant $\gamma$ taken for e.g. 16. To prove that $\mathbf{x}_{\mathtt{bin}}$ contains, for example, $n = 1024$ binary ring elements, we would need to set $2\|\mathbf{x}_{\mathtt{bin}}\|^2 \leq 2\alpha^2 nd \leq q$, to ensure that the binary equation does not wrap around mod $q$ (see for e.g. [34, Sec. 5.2]), leading to a modulus $q \approx 2^{48.31}$ for $d = 256$. For comparison, [34, Prop. 5.1.] could provide an approximate proof for $\mathbf{x}_{\mathtt{bin}}$ with a slack factor $\alpha \coloneqq 2\sqrt{256/26} \cdot \sqrt{2} \cdot \gamma\sqrt{337} \approx 2606.8$, for the same $\gamma$, which would lead to setting a modulus $q \approx 2^{41.7}$.

**Two-Step Projection.** We employ a technique for proving approximate norms that allows us to obtain a smaller slack than the naive approach outlined above. In fact, we get a smaller slack than the original LNP. In our two-step projection approach, we delay using the expensive Gaussian masking and instead prove binary for vectors of *small fixed* dimension, regardless of the dimension of the input binary vectors, allowing for a smaller modulus $q$. Thus, the standard deviation of the Gaussian masking becomes a fixed parameter, independent of the input norm bounds. More precisely, we could write the (non-masked) projection of $\mathbf{x}_{\mathtt{bin}}$, a vector in $\mathbb{Z}^{256}$, on $k+1$ bits, as $k \coloneqq \lceil \log(9.75\sqrt{nd}) \rceil$ ( [28, Cor. 3.2]). In fact, it could be written with high probability on just $k$ bits, so one could think of the high-order bit as an extra bit equal to 0. Then we could mask the lower $k$ bits of the projection of $\mathbf{x}_{\mathtt{bin}}$ with a mask $\mathbf{u}$ uniformly sampled over $[-2^{k-1}, 2^{k-1} - 1]^{256}$ and reject if there are more than $N = 83$ nonzero high-order values (called *carries*), being -1 or 1, across the 256 high-order values (in $\{-1, 0, 1\}$) of each of the projection entries. One could safely send the low-order bits of the projection, collected as $\mathbf{v}_0 \in \mathbb{Z}^{256}$, as they are uniform, but not the high-order values, collected as $\mathbf{v}_1 \in \{-1, 0, 1\}^{256}$, since they leak possible carries. This is where the second step-projection comes into play: we prove that this vector $\mathbf{v}_1$ is ternary by writing it as $\mathbf{v}_1 \coloneqq \mathbf{w} - \mathbf{t}$ for $\mathbf{w}, \mathbf{t} \in \{0, 1\}^{256}$ and by providing an approximate proof for $(\mathbf{w}, \mathbf{t})$ as in [34], whose norm is exactly the number of carries, less than $N$. Its projection needs to be masked for assuring zero-knowledge and could be sent in the clear, without computing its outer commitment, as this projection is an 256-integer vector representing one ring element, when using $d = 256$. Note that, compared to $\mathbf{x}_{\mathtt{bin}}$, this vector has only $n = 2$ binary ring elements. Using the tighter bound $\sqrt{N} = \sqrt{83}$ for $\|(\mathbf{w}, \mathbf{t})\|$ and applying the rationale of [34, Prop. 5.1], one needs to set a modulus $q \approx 2^{30.07}$ to prove $(\mathbf{w}, \mathbf{t})$ is binary. Projecting individually each of the four LaBRADOS output vectors, $\mathbf{z}_0, \mathbf{z}_1, \mathbf{x}_{\mathtt{mid}}, \mathbf{x}_{\mathtt{bin}}$, leads to proving binary for a vector of $n = 8$ binary ring elements, which increases $q$ to being $\approx 2^{32.07}$. Note that the infinity norm of the (non-masked) projection of our vector $\mathbf{x}_{\mathtt{bin}}$ is at most $\|\mathbf{u}\|_\infty + \|2^k\mathbf{v}_1\|_\infty + \|\mathbf{v}_0\|_\infty \leq 2^{k-1} + 2^k + 2^{k-1} = 2^{k+1} \leq 4 \cdot 9.75\sqrt{nd}$, which, applying [16, Thm.6], leads to

$\|\mathbf{x}_{\mathtt{bin}}\| \leq 4\frac{9.75}{0.49}\sqrt{nd}$, giving us a slack factor $\alpha := 4 \cdot \frac{9.75}{0.49} \approx 79.59$. Therefore, using the same requirement on $q$ to prove $\mathbf{x}_{\mathtt{bin}}$ is binary leads us to a modulus $q \approx 2^{31.6}$.

**No Bimodal Rejection Sampling.** The *bimodal* rejection sampling [24] used in approximate proofs in [34] (`BimodRej` in Fig. 3) involves masking a *signed* projection of a witness vector $\mathbf{s}$, as $\mathbf{z} = b\Pi\mathbf{s} + \mathbf{y}$, for a (Gaussian) mask $\mathbf{y}$ and a sign $b \in \{-1, 1\}$, both being considered as part of the witness, thus making the projection constraints quadratic in the witness. In our case, for simplicity, we instead opted for the standard rejection sampling [33].

### 3.3 `LNP-Lite`

We present the linear-size protocol that effectively provides zero-knowledge to our whole construction. It is based on the `LNP` [34] protocol, upgraded with the modifications described in Section 3.2. Therefore, our zero-knowledge protocol, called `LNP-Lite`, follows the same blueprint as `LNP` (Section 2.4), except that the approximate norm proof is now split in two steps. This protocol proves knowledge of witness vectors $\mathbf{s}_i \in \mathcal{R}^{n_i}$, for $i \in [r]$, of not necessarily same dimensions $n_i$, which satisfy constant-term and full-domain constraints of type (1). Its setup consists of the same polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$, of degree $d = 256$, modulus $q$ and challenge space $\mathcal{C}$ (Section 2.1) as `LaBRADOS`, for easier composing `LNP-Lite` with `LaBRADOS`.

We present the `LNP-Lite` protocol tailored directly to `LaBRADOS`'s output, namely proving knowledge of the `LaBRADOS` output vectors $(\mathbf{z}_0, \mathbf{z}_1, \mathbf{x}_{\mathtt{mid}}, \mathbf{x}_{\mathtt{bin}}, \sigma_{-1}(\mathbf{x}_{\mathtt{bin}}))$, denoted for simplicity by $\mathbf{s}_i \in \mathcal{R}^{n_i}$, $i \in [5]$, with bounded norms $\beta_i$, and satisfying the verification checks of `LaBRADOS` (see Fig. 8), which are indeed of type (1). In addition, the dimensions $n_i$ are different and, in fact, we do not require them to be the same, as in `LaBRADOS`. We apply the outer commitment layer as mentioned in Section 2.3. See Fig. 2 for an algorithmic description of `LNP-Lite` with no outer commitments. (and Figs. 11 and 12 for a complete description of the `LNP-Lite` prover and verifier algorithms.)

**Comparing with `LNP` [34].** The differences between `LNP-Lite` (Fig. 2) and `LNP` (Fig. 3) include the two-step approximate norm proof using the standard rejection sampling $\mathtt{Rej}_1$ ( [33]) instead of the bimodal rejection sampling `BimodRej` ( [34]), and the use of a different challenge space, to make composition easier with `LaBRADOS`. Throughout `LNP-Lite`, new witness vectors are created, compared to `LNP`, including $\mathbf{s}_5 := \sigma_{-1}(\mathbf{s}_4)$ needed for proving binary, and $\tilde{\mathbf{v}}$ generated in the approximate norm proof. Moreover, thanks to the outer commitment layer used in `LNP-Lite`, individual (masked) projections on the witness vectors are possible, whereas in `LNP` a single (masked) projection of all witness vectors is provided.

We proceed to describe each step of the `LNP-Lite` protocol below, for simplicity of exposition without adding the outer commitment layer. Hence, unless otherwise specified, all the prover's messages are decomposed in binary and the prover sends instead an outer commitment of them.

**1. Commit.** Before committing, we sample additional masks to be used to ensure zero-knowledge and whose binary decompositions are collected in a new witness vector $\mathbf{s}_6 \in \mathcal{R}^{n_6}$. The purpose of each of these masks will be described in the following steps:

- $\mathbf{u}_i \leftarrow [-2^{k_i-1}, 2^{k_i-1} - 1]^{256}$, $i \leq 4$, for $k_i := \lceil \log(9.75\beta_i) \rceil$, each identified as a polynomial $u_i \in \mathcal{R}^{256/d}$,
- $\mathbf{y} \leftarrow D_\sigma^{256/d}$ over $\mathcal{R}^{256/d}$,
- $g_i \leftarrow \{x \in \mathcal{R}_q : \mathtt{ct}(x) = 0\}$, $i \leq \lceil 128/\log q \rceil$.

Now the prover commits to these masks, along to the `LaBRADOS` output vectors $\mathbf{s}_i$, $i \in [5]$. Namely, he computes and sends the hiding Ajtai commitment of $\tilde{\mathbf{s}} := (\mathbf{s}_1, \ldots, \mathbf{s}_6)$. For this, he samples a random binary vector $\mathbf{r}_s \in \mathcal{R}^{n_s}$, to make the commitment indistinguishable from random under M-LWE:

$$\mathbf{t}_s := \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}'_1\mathbf{r}_s \in \mathcal{R}_q^{k_s}, \tag{2}$$

where $\mathbf{A}_1 \in \mathcal{R}_q^{k_s \times \sum_{i\in[6]} n_i}$, $\mathbf{A}'_1 \in \mathcal{R}_q^{k_s \times n_s}$ are public matrices.

**2. Approximate Norm Proof in Two Steps.** As explained in Section 3.2, the approximate norm proof is split in two steps.

**Step 1.** Here, the prover replaces the norm statements of the vectors $\mathbf{s}_i$, $i \in [4]$, with modular Johnson-Lindenstrauss projections [28] of their coefficient vectors. Upon receiving $\mathbf{\Pi}_i \leftarrow \mathtt{Bin}_1^{256 \times dn_i}$ from the verifier,

Instance: Verifier's checks from `LaBRADOS`
Witness: Output vectors from `LaBRADOS`
$\mathbf{s}_i \in \mathcal{R}^{n_i}, i \in [4]$

**1. Commit**

$\mathbf{s}_5 \coloneqq \sigma_{-1}(\mathbf{s}_4)$
$\mathbf{u}_i \leftarrow [-2^{k_i-1}, 2^{k_i-1}-1]^{256}, i \in [4]$
$\mathbf{y} \leftarrow D_\sigma^{256/d}$
$g_i \leftarrow \{x \in \mathcal{R}_q : \mathtt{ct}(x) = 0\},\ i \in [\lambda],\ \lambda \coloneqq \lceil 128/\log q \rceil$
$\mathbf{s}_6 \in \mathcal{R}^{n_6}$ : bin. decomposition of $(\mathbf{u}_i), y, (g_i)$
$\tilde{\mathbf{s}} \coloneqq (\mathbf{s}_1, \ldots, \mathbf{s}_6)$
$\mathbf{t}_s \coloneqq \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}_1'\mathbf{r}_s \in \mathcal{R}_q^{k_s},\ \mathbf{r}_s \leftarrow \mathcal{R}_2^{n_s}$
Send $\mathbf{t}_s$

**2. Approximate Norm Proof**

Receive $\mathbf{\Pi}_i \leftarrow \mathtt{Bin}_1^{256 \times dn_i}, i \in [4]$
$\mathbf{\Pi}_i\mathbf{s}_i + \mathbf{u}_i = 2^{k_i}\mathbf{v}_i^{(1)} + \mathbf{v}_i^{(0)} \in \mathbb{Z}_q^{256}$
if $\|\mathbf{v}_i^{(1)}\|^2 > N$ abort
$\mathbf{v}_i^{(1)} \coloneqq \mathbf{w}_i - \mathbf{t}_i,\ i \in [4]$
$\tilde{\mathbf{v}}_1 \coloneqq (w_1, t_1, \ldots, w_4, t_4) \in \mathcal{R}^8,\ \tilde{\mathbf{v}}_2 \coloneqq \sigma_{-1}(\tilde{\mathbf{v}}_1)$
$\tilde{\mathbf{v}} \coloneqq (\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2)$
$\mathbf{t}_v \coloneqq \mathbf{A}_2\tilde{\mathbf{v}} + \mathbf{A}_2', \mathbf{r}_v \in \mathcal{R}_q^{k_v},\ \mathbf{r}_v \leftarrow \mathcal{R}_2^{n_v}$
Send $\mathbf{t}_v, (v_i^{(0)})_{i \le 4}$
Receive $\bar{\mathbf{\Pi}} \leftarrow \mathtt{Bin}_1^{256 \times 8d}$
$\mathbf{z} \coloneqq \bar{\mathbf{\Pi}}\tilde{\mathbf{v}}_1 + \mathbf{y}$
if $(\mathtt{Rej}_1(\mathbf{z}) = \perp)$ abort
Send $\mathbf{z}$

**3. Aggregate the Constraints**

Receive $\mu_j^{(i)} \leftarrow \mathbb{Z}_q$
$h_i \coloneqq g_i + \sum_j \mu_j^{(i)} f_j',\ i \in [\lambda]$
Send $(h_i)_{i \le \lambda}$
Receive $\nu_j \leftarrow \mathcal{R}_q$
$f \coloneqq \sum_j \nu_j f_j$

**4. Prove the Final Constraint**

$\mathbf{y}_1 \coloneqq (\mathbf{y}_1^s, \mathbf{y}_1^v) \leftarrow D_{\sigma_1}^{\sum_{i \in [6]} n_i + 16}$
$\mathbf{y}_2 \coloneqq (\mathbf{y}_2^s, \mathbf{y}_2^v) \leftarrow D_{\sigma_2}^{n_s + n_v}$
$\mathbf{w}_s \coloneqq \mathbf{A}_1\mathbf{y}_1^s + \mathbf{A}_1'\mathbf{y}_2^s \in \mathcal{R}_q^{k_s},\ \mathbf{w}_v \coloneqq \mathbf{A}_2\mathbf{y}_1^v + \mathbf{A}_2'\mathbf{y}_2^v \in \mathcal{R}_q^{k_v}$
Compute $u', v' \in \mathcal{R}_q$ to make Equation (7) hold
$u \coloneqq u' + \langle \varphi_{rand}, \mathbf{r}_s \rangle,\ v \coloneqq v' + \langle (\varphi_{rand}, 0), \mathbf{y}_2 \rangle$.
Send $(\mathbf{w}_s, \mathbf{w}_v, u, v)$
Receive $c \leftarrow \mathcal{C}$
$\mathbf{z}_1 \coloneqq \mathbf{y}_1 + c \cdot (\tilde{\mathbf{s}}, \tilde{\mathbf{v}})$
$\mathbf{z}_2 \coloneqq \mathbf{y}_2 + c \cdot (\mathbf{r}_s, \mathbf{r}_v)$
if $(\mathtt{Rej}_1(\mathbf{z}_1) = \perp) \vee (\mathtt{Rej}_2(\mathbf{z}_2) = \perp)$ abort
Send $(\mathbf{z}_1, \mathbf{z}_2)$

**Fig. 2.** The `LNP-Lite` protocol.

Instance: Verifier's checks from `LaBRADOS`
Witness: Output vectors from `LaBRADOS`
$\mathbf{s}_i \in \mathcal{R}^{n_i}, i \in [4]$

**Commit**

$\mathbf{y} \leftarrow D_\sigma^{256/d}$
$b \leftarrow \{-1, 1\}$
$g_i \leftarrow \{x \in \mathcal{R}_q : \mathtt{ct}(x) = 0\},$
$\quad i \le \lambda \coloneqq \lceil 128/\log q \rceil$
$\mathbf{s}_5 \in \mathcal{R}^{n_5}$ : bin. decomposition of $y, (g_i)$ and $b$
$\tilde{\mathbf{s}} \coloneqq (\mathbf{s}_1, \ldots, \mathbf{s}_5)$
$\mathbf{t}_s \coloneqq \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}_1'\mathbf{r} \in \mathcal{R}_q^{k_s},\ \mathbf{r} \leftarrow \mathcal{R}_2^{n_s}$
Send $\mathbf{t}_s$

**2. Approximate Norm Proof**

Receive $\mathbf{\Pi} \leftarrow \mathtt{Bin}_1^{256 \times d\sum_{i \in [4]} n_i}$
$\mathbf{z} \coloneqq \mathbf{\Pi}(\mathbf{s}_1, \ldots, \mathbf{s}_4) + \mathbf{y} \in \mathbb{Z}_q^{256}$
if $(\mathtt{BimodRej}(\mathbf{z}, b) = \perp)$ abort
Send $\mathbf{z}$

**3. Aggregate the Constraints**

Receive $\mu_j^{(i)} \leftarrow \mathbb{Z}_q$
$h_i \coloneqq g_i + \sum_j \mu_j^{(i)} f_j',\ i \in [\lambda]$
Send $(h_i)_{i \le \lambda}$
Receive $\nu_j \leftarrow \mathcal{R}_q$
$f \coloneqq \sum_j \nu_j f_j$

**Prove the Final Constraint**

$\mathbf{y}_1 \leftarrow D_{\sigma_1}^{\sum_{i \in [5]} n_i}$
$\mathbf{y}_2 \leftarrow D_{\sigma_2}^{n_s}$
$\mathbf{w}_s \coloneqq \mathbf{A}_1\mathbf{y}_1 + \mathbf{A}_1'\mathbf{y}_2 \in \mathcal{R}_q^{k_s}$
Compute $u', v' \in \mathcal{R}_q$ to make Equation (32) hold
$u \coloneqq u' + \langle \varphi_{rand}, \mathbf{r}_s \rangle,\ v \coloneqq v' + \langle \varphi_{rand}, \mathbf{y}_2 \rangle$.
Send $(\mathbf{w}_s, u, v)$
Receive $c \leftarrow \mathcal{C}$
$\mathbf{z}_1 \coloneqq \mathbf{y}_1 + c \cdot \tilde{\mathbf{s}}$
$\mathbf{z}_2 \coloneqq \mathbf{y}_2 + c \cdot \mathbf{r}$
if $(\mathtt{Rej}_1(\mathbf{z}_1) = \perp) \vee (\mathtt{Rej}_2(\mathbf{z}_2) = \perp)$ abort
Send $(\mathbf{z}_1, \mathbf{z}_2)$

**Fig. 3.** The `LNP` protocol [34], adapted to `LaBRADOS` output witness vectors and checks.

the prover computes the projections $\mathbf{\Pi}_i\mathbf{s}_i \in \mathbb{Z}_q^{256}$. Note that the prover does not need to compute the projections of $\mathbf{s}_5$ or of $\mathbf{s}_6$, as, since we give an approximate norm proof for $\mathbf{s}_4 = \mathbf{x}_{\mathtt{bin}}$, there is no need in the case of $\mathbf{s}_5 = \sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$, whereas for $\mathbf{s}_6$, it is constructed of binary decompositions of masks that would appear in the linear-part of the dot-product constraints to be proven. We can assume that the maps $\mathbf{\Pi}_i$ come from the same projection matrix, by properly removing some columns to compute the respective projections. For $k_i = \lceil \log(9.75\beta_i) \rceil$, $i \in [4]$, we get $\|\mathbf{\Pi}_i\mathbf{s}_i\|_\infty \leq 2^{k_i}$ with high probability. ( [28, Cor. 3.2]). For zero-knowledge, we mask each projection using $\mathbf{u}_i \leftarrow [-2^{k_i-1}, 2^{k_i-1}-1]^{256}$ sampled before, and decompose it as $\mathbf{\Pi}_i\mathbf{s}_i + \mathbf{u}_i = 2^{k_i}\mathbf{v}_i^{(1)} + \mathbf{v}_i^{(0)} \in \mathbb{Z}_q^{256}$, with $\mathbf{v}_i^{(0)} \in [-2^{k_i-1}, 2^{k_i-1}-1]^{256}$, $\mathbf{v}_i^{(1)} \in \{-1, 0, 1\}^{256}$. The role of $\mathbf{u}_i$ here is to uniformize the low-order values, $\mathbf{v}_i^{(0)}$, of the projection $\mathbf{\Pi}_i\mathbf{s}_i$. Sending them as ring elements $v_i^{(0)} \in \mathcal{R}$ to the verifier would then not leak any information. The high-order values, $\mathbf{v}_i^{(1)}$, would leak possible carries that took place when padding, so they need to be committed. We also control the number of carries, i.e., the number of nonzero entries that the ternary vector $\mathbf{v}_i^{(1)}$ has, which is why we reject whenever $\|\mathbf{v}_i^{(1)}\|^2 > N$, for a fixed threshold $N$. The threshold $N$ is set in Section D, such that all $\mathbf{v}_i^{(0)}$ are still statistically close (less than $1/2^{128}$) from being uniform over $[-2^{k_i-1}, 2^{k_i-1}-1]^{256}$. To prove that $\mathbf{v}_i^{(1)}$ is ternary, we write it as $\mathbf{v}_i^{(1)} \coloneqq \mathbf{w}_i - \mathbf{t}_i$, for $\mathbf{w}_i, \mathbf{t}_i \in \{0,1\}^{256}$, such that $\mathbf{w}_i$ encodes the 1's of $\mathbf{v}_i^{(1)}$ and $\mathbf{t}_i$ encodes the -1's of $\mathbf{v}_i^{(1)}$, i.e., for any $j \in [256]$,

$$\mathbf{w}_i[j] = \begin{cases} 1 & \text{if } \mathbf{v}_i^{(1)}[j] = 1 \\ 0 & \text{else} \end{cases}, \mathbf{t}_i[j] = \begin{cases} 1 & \text{if } \mathbf{v}_i^{(1)}[j] = -1 \\ 0 & \text{else} \end{cases}.$$

Therefore, $\|\mathbf{v}_i^{(1)}\|^2 = \|(\mathbf{w}_i, \mathbf{t}_i)\|^2 \leq N$. As $d = 256$, each binary vector $\mathbf{w}_i$ or $\mathbf{t}_i$ corresponds to a binary ring element $w_i$ or $t_i$. Thus, from the four projections, we collect them all in $\tilde{\mathbf{v}}_1 \in \mathcal{R}^8$. As the challenge space $\mathcal{C}$ (Section 2.1) is not stable under $\sigma_{-1}$, we also need to commit to $\tilde{\mathbf{v}}_2 \coloneqq \sigma_{-1}(\tilde{\mathbf{v}}_1) \in \mathcal{R}^8$. Note that $\|\tilde{\mathbf{v}}_1\|^2 = \|\tilde{\mathbf{v}}_2\|^2 \leq 4N$.

The prover computes and sends the hiding commitment of $\tilde{\mathbf{v}} \coloneqq (\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2) \in \mathcal{R}^{16}$. For this, he samples a random binary vector $\mathbf{r}_v \in \mathcal{R}^{n_v}$, to make the commitment indistinguishable from random under M-LWE:

$$\mathbf{t}_v \coloneqq \mathbf{A}_2\tilde{\mathbf{v}} + \mathbf{A}_2'\mathbf{r}_v \in \mathcal{R}_q^{k_v}, \tag{3}$$

where $\mathbf{A}_2 \in \mathcal{R}_q^{k_v \times 16}, \mathbf{A}_2' \in \mathcal{R}_q^{k_v \times n_v}$ are public matrices. Along with $\mathbf{t}_v$, the prover also sends the low-order bits, $v_i^{(0)}$, $i \in [4]$.

**Step 2** Starting from this step, we follow the lines of [34], for proving knowledge of two witness vectors, $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$ satisfying dot-product constraints and norm bounds. As for $\tilde{\mathbf{s}}$ we already provided an approximate norm proof before, it is time now for $\tilde{\mathbf{v}} = (\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2)$. As $\tilde{\mathbf{v}}_2 = \sigma_{-1}(\tilde{\mathbf{v}}_1)$, its shortness trivially follows from proving its well-formedness and from proving $\tilde{\mathbf{v}}_1$ being short. Thus, it suffices to provide such proof only for $\tilde{\mathbf{v}}_1 \in \mathcal{R}^8$. More precisely, upon receiving the projection map $\bar{\mathbf{\Pi}} \leftarrow \mathtt{Bin}_1^{256\times 8d}$ from the verifier, the prover creates its masked projection, using the Gaussian mask $\mathbf{y} \leftarrow D_\sigma$ from $\mathbf{s}_6$ as $\mathbf{z} \coloneqq \bar{\mathbf{\Pi}}\tilde{\mathbf{v}}_1 + \mathbf{y} \in \mathbb{Z}_q^{256}$. For zero-knowledge, the prover applies rejection sampling $\mathtt{Rej}_1$ ( [34, Lem. 2.14 1)]) and sends $\mathbf{z}$, upon acceptance. Then, the verifier checks if $\|\mathbf{z}\| \leq \sigma\sqrt{2 \cdot 256}$. By rejection sampling, $\mathbf{z}$ follows a distribution close to $D_\sigma^{256/d}$, so this check holds with high probability. The standard deviation $\sigma$ is set so that $\mathbf{y}$ fully hides the vector $\tilde{\mathbf{v}}_1$ (see Section D.2). For this particular step, we do not apply the outer commitment, as here $\mathbf{z}$ is just one ring element, since the degree of the polynomial ring $\mathcal{R}$ is $d = 256$. In addition, this norm check will not be part of the input statement of the upcoming `LaBRADOS`.

**3. Aggregate the Constraints.** In this step, the prover linearly combines (or aggregates) the constraints satisfied by $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$, following standard techniques [13, 15, 34], in order to get a single constraint to prove. This is done in two steps. First, the input constant-term constraints, the ones from the `LaBRADOS` checks and some additional ones, are aggregated with random integer challenges $\mu_j$ from the verifier. The additional constant-term constraints are needed for proving the correctness of:

- the masked projections of $\mathbf{s}_i$, $i \leq 4$, all linear in $\tilde{\mathbf{s}}, \tilde{\mathbf{v}}$,
- $\tilde{\mathbf{v}}_1$ being binary, containing the dot-product $\langle \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2 \rangle$,

– $\tilde{\mathbf{v}}_2 = \sigma_{-1}(\tilde{\mathbf{v}}_1) \in \mathcal{R}^8$, all linear in $\tilde{\mathbf{v}}$,
– the projection of $\tilde{\mathbf{v}}_1$, all linear in $\tilde{\mathbf{v}}$.

Aggregating all of these constant-term constraints, called $f'_j$, preserves the zero constant coefficient. This is performed $\lambda$ many times, for $\lambda \coloneqq \lceil 128/\log q \rceil$, as the soundness of one aggregation is just $1/q$. Thus, the prover receives samples $\lambda$ many sets of random integer challenges $\{(\mu_j^{(i)})_j\}_{i\in[\lambda]}$ for performing these aggregations. As these aggregations, called $h_i$, for $i \in [\lambda]$, could leak information on $\tilde{\mathbf{s}}, \tilde{\mathbf{v}}$ when being sent, the prover masks them with the ring elements $g_i$ previously sampled as part of $\mathbf{s}_6$, and sends the final aggregations: $h_i \coloneqq g_i + \sum_j \mu_j^{(i)} f'_j$. By the definition of $g_i$, the coefficients of $h_i$ are all uniform, except the constant coefficient, which is 0, and thus no information about the witness vectors is leaked. The verifier can then check if $\mathtt{ct}(h_i) = 0$ for any such $h_i$.

In the second aggregation step, the prover first has to prove that the prior aggregations $h_i$'s are correctly formed. This is why they are converted into new quadratic constraints, as $F_i \coloneqq g_i + \sum_j \mu_j^{(i)} f'_j - h_i$. The prover then aggregates these new constraints, along with the full constraints from LaBRADOS checks, all called $f_j$, with random polynomial challenges $\nu_i$ from the verifier: $f \coloneqq \sum_j \nu_j f_j$. If one of the full constraints is not satisfied, then the final constraint $f$ would be satisfied with probability $1/q^{d/2}$, assuming $q$ splits into two factors. As all the constraints we have aggregated so far are quadratic in $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$, the final constraint is also quadratic, as follows:

$$f \triangleq \sum_{i,j\in[5]} a_{ij}\langle \mathbf{s}_i, \mathbf{s}_j\rangle + a\langle \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2\rangle + \langle \varphi, (\tilde{\mathbf{s}}, \tilde{\mathbf{v}})\rangle + f_0 = 0, \tag{4}$$

where $a_{ij}, a \in \mathcal{R}_q$, $\varphi \in \mathcal{R}_q^{\sum_{i\in[6]} n_i+16}$, and $f_0$ a linear function in $h_i$'s and $v_i^{(0)}$'s. Here we explain why $f$ looks like above. Note that the vector $\mathbf{s}_3$ appears only in the linear part of LaBRADOS verification checks (Fig. 8). The vector $\mathbf{s}_6$ also appears only in the linear part of the constraints, as it contains the additional masks used in previous steps. Thanks to aggregation, they both appear only in the linear part of $f$, hence the first sum of $f$ does not include $\mathbf{s}_6$. Moreover, the binary check (see Section A.3) on $\tilde{\mathbf{v}}_1$ is the only one quadratic in $\tilde{\mathbf{v}}_1$ and $\tilde{\mathbf{v}}_2$, containing only the dot-product $\langle \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2\rangle$. Further, The coefficients that appear in Eq. (4), $a_{ij}, a, \varphi$ and of $f_0$, can be computed by the verifier, as they depend on the integer and polynomial challenges sent by him and on all the coefficients of all the constraints, all being public.

**4. Prove the Final Constraint.** In this last step, the prover wants to convince the verifier that the final quadratic (Eq. (4)) holds in terms of the *masked openings* of $(\tilde{\mathbf{s}}, \tilde{\mathbf{v}})$ and $(\mathbf{r}_s, \mathbf{r}_v)$. These openings are defined as $\mathbf{z}_1 = (\mathbf{z}_1^s, \mathbf{z}_1^v) \coloneqq \mathbf{y}_1 + c \cdot (\tilde{\mathbf{s}}, \tilde{\mathbf{v}}) \in \mathcal{R}_q^{\sum_{i\in[6]} n_i+16}$ and $\mathbf{z}_2 = (\mathbf{z}_2^s, \mathbf{z}_2^v) \coloneqq \mathbf{y}_2 + c \cdot (\mathbf{r}_s, \mathbf{r}_v) \in \mathcal{R}_q^{n_s+n_v}$, for some challenge $c \in \mathcal{C}$ received from the verifier and masks $\mathbf{y}_1 \in \mathcal{R}^{\sum_{i\in[6]} n_i+16}$ and $\mathbf{y}_2 \in \mathcal{R}^{n_s+n_v}$. One has to apply rejection sampling [34] before outputting these openings, so that they do not leak any information on $\tilde{\mathbf{s}}, \tilde{\mathbf{v}}, \mathbf{r}_s$ or $\mathbf{r}_v$. However, computing separate masked openings of $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$ and applying rejection sampling on each would be too costly, which is why $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$ are treated together. We also apply the same reasoning for $\mathbf{r}_s$ and $\mathbf{r}_v$. Before showing how we can prove the final quadratic using these $\mathbf{z}_i$'s, we first split them (and the masks $\mathbf{y}_i$'s), according to the dimensions of the vectors they are hiding:

$$\mathbf{z}_1 \coloneqq (\mathbf{z}_{1,1}^s, \ldots, \mathbf{z}_{1,6}^s, \mathbf{z}_{1,1}^v, \mathbf{z}_{1,2}^v),\ \mathbf{z}_{1,i}^s \in \mathcal{R}_q^{n_i},\ \mathbf{z}_{1,i}^v \in \mathcal{R}_q^8, \tag{5}$$

$$\mathbf{z}_2 \coloneqq (\mathbf{z}_2^s, \mathbf{z}_2^v),\ \mathbf{z}_2^s \in \mathcal{R}_q^{n_s},\ \mathbf{z}_2^v \in \mathcal{R}_q^{n_v}, \tag{6}$$

$$\mathbf{y}_1 \coloneqq (\mathbf{y}_{1,1}^s, \ldots, \mathbf{y}_{1,6}^s, \mathbf{y}_{1,1}^v, \mathbf{y}_{1,2}^v),\ \mathbf{y}_{1,i}^s \in \mathcal{R}_q^{n_i},\ \mathbf{y}_{1,i}^v \in \mathcal{R}_q^8,$$
$$\mathbf{y}_2 \coloneqq (\mathbf{y}_2^s, \mathbf{y}_2^v),\ \mathbf{y}_2^s \in \mathcal{R}_q^{n_s},\ \mathbf{y}_2^v \in \mathcal{R}_q^{n_v}.$$

For proving Eq. (4), the prover provides the verifier some *garbage terms* $u', v' \in \mathcal{R}_q$, to make the following hold

$$\begin{aligned}&\sum_{i,j\in[5]} a_{ij}\langle \mathbf{z}_{1,i}^s, \mathbf{z}_{1,j}^s\rangle + a\langle \mathbf{z}_{1,1}^v, \mathbf{z}_{1,2}^v\rangle + c \cdot \langle \varphi, \mathbf{z}_1\rangle + c^2 \cdot f_0 \\ &= c^2\Big(\sum_{i,j\in[5]} a_{ij}\langle \mathbf{s}_i, \mathbf{s}_j\rangle + a\langle \tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2\rangle + \langle \varphi, (\tilde{\mathbf{s}}, \tilde{\mathbf{v}})\rangle + f_0\Big) + cu' + v'.\end{aligned} \tag{7}$$

To prove that the coefficient of $c^2$ in Eq. (7) vanishes, we send a commitment $u$ to $u'$ and a randomized version $v$ of $v'$ as

$$u \coloneqq u' + \langle \varphi_{rand}, \mathbf{r}_s \rangle, \ v \coloneqq v' + \langle (\varphi_{rand}, 0), \mathbf{y}_2 \rangle. \tag{8}$$

If the verifier is given $u, v$ and the masked openings $\mathbf{z}_i$, he can check

$$\begin{aligned} &\sum_{i,j \in [5]} a_{ij} \langle \mathbf{z}^s_{1,i}, \mathbf{z}^s_{1,j} \rangle + a \langle \mathbf{z}^v_{1,1}, \mathbf{z}^v_{1,2} \rangle + c \cdot \langle \varphi, \mathbf{z}_1 \rangle + c^2 \cdot f_0 \\ &= cu + v - \langle (\varphi_{rand}, 0), \mathbf{z}_2 \rangle, \end{aligned} \tag{9}$$

which is exactly Eq. (7) when the coefficient of $c^2$ is 0.

Now we explain the remaining part of the protocol, for actually proving the constraint $f$ using the masked openings. The prover samples the masking vectors $\mathbf{y}_1 \coloneqq (\mathbf{y}^s_1, \mathbf{y}^v_1) \leftarrow D_{\sigma_1}^{\sum_{i \in [6]} n_i + 16}$ and $\mathbf{y}_2 \coloneqq (\mathbf{y}^s_2, \mathbf{y}^v_2) \leftarrow D_{\sigma_2}^{n_s + n_v}$ and computes their Ajtai commitments as

$$\mathbf{w}_s \coloneqq \mathbf{A}_1 \mathbf{y}^s_1 + \mathbf{A}'_1 \mathbf{y}^s_2 \in \mathcal{R}_q^{k_s}, \ \mathbf{w}_v \coloneqq \mathbf{A}_2 \mathbf{y}^v_1 + \mathbf{A}'_2 \mathbf{y}^v_2 \in \mathcal{R}_q^{k_v}. \tag{10}$$

Then he computes $u, v$ as in Eqs. (7) and (8). At this point, the prover could send $\mathbf{w}_s$, $\mathbf{w}_v$, $u$, and $v$.

Next, after receiving the challenge $c \leftarrow \mathcal{C}$ from the verifier, the prover computes $\mathbf{z}_1 \in \mathcal{R}_q^{\sum_{i \in [6]} n_i + 16}$ and $\mathbf{z}_2 \in \mathcal{R}_q^{n_s + n_v}$ as $\mathbf{z}_1 \coloneqq \mathbf{y}_1 + c \cdot (\tilde{\mathbf{s}}, \tilde{\mathbf{v}})$ and $\mathbf{z}_2 \coloneqq \mathbf{y}_2 + c \cdot (\mathbf{r}_s, \mathbf{r}_v)$ and applies the rejection sampling $\mathtt{Rej}_i$ from Lemma 3 (i), for each $\mathbf{z}_i$, $i \in [2]$. If both algorithms do not abort, the prover outputs $\mathbf{z}_1, \mathbf{z}_2$. Note that $(\mathbf{r}_s, \mathbf{r}_v)$ is used only once, so a bit of leakage is allowed as when applying Lemma 3 (2). The standard deviations $\sigma_1$ and $\sigma_2$ are set in Section D.2. By rejection sampling, each $\mathbf{z}_i$ (and its splits) follows a distribution close to $D_{\sigma_i}$, so that the verifier can check if it is small enough, i.e. $\|\mathbf{z}_i\| \leq \sigma_i \sqrt{2\dim(\mathbf{z}_i)}$, with high probability, where $\dim(\mathbf{z}_i)$ is the dimension of $\mathbf{z}_i$ as an integer vector (see Lemma 2).

**Output and Checks.** The outer commitment layer is applied to any prover message (binary decomposed) previously described, except the masked openings $\mathbf{z}_1, \mathbf{z}_2$ and the second projection $\mathbf{z}$. Thus, the last message of the prover is, in fact, made by $\mathbf{z}_1, \mathbf{z}_2$ and the openings of the outer commitments, which are all binary polynomials collected in the vector $\mathbf{x}_{\mathtt{bin}} \in \mathcal{R}^{n_{\mathtt{bin}}}$, see Fig. 2. When receiving $\mathbf{x}_{\mathtt{bin}}$, the verifier recomposes them into $\mathbf{t}_s, \mathbf{t}_v, (v_i^{(0)}), (h_i), \mathbf{w}_s, \mathbf{w}_v, u, v$, and when receiving $\mathbf{z}_1, \mathbf{z}_2$, he splits them as in Eqs. (5) and (6). He also computes $a, a_{ij} \in \mathcal{R}_q, \varphi \in \mathcal{R}_q^{\sum_{i \in [6]} n_i + 16}, f_0$ from Eq. (4). We provide below the verifier checks (see Fig. 12 for a complete verifier algorithm), which include correctness of the commitments $\mathbf{t}_s, \mathbf{t}_v$ and outer commitments, of the final constraint (Eq. (9)), of $\mathbf{x}_{\mathtt{bin}}$ being binary, the vanishing constant coefficient of the $h_i$, the separate norm bounds on the splits of $\mathbf{z}_1$ (Eq. (5)) and a global norm bound on $\mathbf{z}_2$. The reason why we do not consider a global norm bound on $\mathbf{z}_1$ is because we want to further feed this `LNP-Lite` output into `LaBRADOS` and it is its splits that appear in the dot-products of Eq. (9). In the case of $\mathbf{z}_2$, we can provide a global norm bound of it as a verifier check, as $\mathbf{z}_2$ appears only in the linear part of Eq. (9). Note that all

these checks are of type (1):

$$\mathbf{A}_1\mathbf{z}_1^s + \mathbf{A}_1'\mathbf{z}_2^s = \mathbf{w}_s + c\mathbf{t}_s \tag{11}$$

$$\mathbf{A}_2\mathbf{z}_1^v + \mathbf{A}_2'\mathbf{z}_2^v = \mathbf{w}_v + c\mathbf{t}_v \tag{12}$$

$$\mathtt{ct}(h_i) = 0,\ i \in [\lambda] \tag{13}$$

$$\sum_{i,j\in[5]} a_{ij}\langle \mathbf{z}_{1,i}^s, \mathbf{z}_{1,j}^s\rangle + a\langle \mathbf{z}_{1,1}^v, \mathbf{z}_{1,2}^v\rangle + c\cdot\langle\varphi, \mathbf{z}_1\rangle + c^2\cdot f_0$$
$$= cu + v - \langle((\varphi_{rand}, 0), \mathbf{z}_2\rangle \tag{14}$$

$$\|\mathbf{z}_{1,i}^s\| \le \sigma_1\sqrt{2\cdot dn_i},\ i \in [6] \tag{15}$$

$$\|\mathbf{z}_{1,1}^v\|, \|\mathbf{z}_{1,2}^v\| \le \sigma_1\sqrt{2\cdot 8d} \tag{16}$$

$$\|\mathbf{z}_2\| \le \sigma_2\sqrt{2\cdot d(n_s+n_v)} \tag{17}$$

$$\mathbf{x}_{\mathtt{bin}} \text{ is binary} \tag{18}$$

$$\text{correctness of outer commits} \tag{19}$$

*Security.* Here we analyze the security of `LNP-Lite`, at a high level, and refer the reader to Section B.3 for more details, such as completeness error or soundness proof.

**Completeness.** Assume that the prover and the verifier behave honestly. Eqs. (11) and (12) hold thanks to Eqs. (2), (3) and (10). Eq. (14) holds thanks to Eq. (7). Then the verifier is convinced of the well-formedness of $h_i$'s, $i \in [\lambda]$, whose check Eq. (13) holds, because of the masks $g_i$. The norm bounds hold with overwhelming probability, according to [6, Lem. 2.5], as, thanks to rejection sampling ( [34, Lem. 2.14]), $\mathbf{z}_1$ and $\mathbf{z}_2$ follow (closely) Gaussian distributions.

**Zero-Knowledge.** It is enough to check that the protocol with no outer commitment has zero-knowledge, thanks to Lemma 6. For this, we follow the same lines as [34, Thm. 4.2], except that here we deal with the Two-Secrets-Extended M-LWE problem (Definition 5) whose secrets correspond to the random vectors $\mathbf{r}_s$ and $\mathbf{r}_v$ used in hiding the (inner) commitments.

**Soundness.** Again, we prove knowledge soundness only for the protocol without outer commitments, thanks to Lemma 5. For simplicity, let us consider the norm bounds as $\|\mathbf{z}_1^s\| \le B_s$, $\|\mathbf{z}_1^v\| \le B_v$, $\|\mathbf{z}_2^s\|, \|\mathbf{z}_2^v\| \le \|\mathbf{z}_2\| \le B_r$. Here we state knowledge soundness. At a high level, we derive the proof by viewing the first step of the projection as a reduction of knowledge from the input relation to a new relation, which could be proven by `LNP` [34, Prop. 5.1.].

**Theorem 1.** *Let $\mathcal{C} \subset \mathcal{R}_q$ be the challenge space from Section 2.1, of polynomials with operator norm bounded by $T$. Suppose M-SIS is hard for $\ell_2$ norm bound $8T\sqrt{B_s^2+B_r^2}$ and rank $k_s$, and for $\ell_2$ norm bound $8T\sqrt{B_v^2+B_r^2}$ and rank $k_v$. Moreover, assume* $\max\{2\sigma\sqrt{512/26}\cdot 2048\cdot 41, \{61\cdot 2^{k_i+1}/0.49\}_{i\le 4}, 2\cdot d\cdot n_4\cdot\left(\frac{4\cdot 9.75}{0.49}\right)^2, 2\cdot(2\sigma\sqrt{512/26})^2\} \le q$. *Then `LNP-Lite` is a knowledge-sound proof with knowledge soundness $\kappa := 5/2^{128} + (1/q)^\lambda + 1/q^{d/2} + 2/|\mathcal{C}|$ and norm slack $c_{\texttt{LNP-Lite}} := \frac{4\cdot 9.75}{0.49} \simeq 79.59$, i.e., the extractor is only guaranteed to extract witness vectors $\bar{\mathbf{s}_i}$ of norm bounds $\|\bar{\mathbf{s}}_i\| \le 4\cdot 9.75/0.49\beta_i$.*

### 3.4 Proving the `LNP-Lite` Output with `LaBRADOS`

In our succinct zero-knowledge proof system, we continue to perform `LaBRADOS` and `LaBRADOR` iterations after the zero-knowledge protocol `LNP-Lite`, to further compress the `LNP-Lite` output, until it cannot be decreased anymore. Now we take the `LNP-Lite` output, $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{x}_{\mathtt{bin}})$ (Fig. 2), and the checks it satisfies (Eqs. (11) to (19)), which are all of type (1), and feed them into `LaBRADOS`. Note that, thanks to the zero-knowledge of `LNP-Lite`, hiding the outer commitments of the next `LaBRADOS` iterations is not needed.

**Decompose.** We decompose each $\mathbf{z}_i$ into two parts, with respect to some integer basis $b_i$, as $\mathbf{z}_i = \mathbf{z}_i^{(0)} + b_i\mathbf{z}_i^{(1)}$, following a rationale similar to that in [13].

**Additional Witness.** Note that $\sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$ has to be part of the LaBRADOS witness vector, as the binary check (18) considers both $\mathbf{x}_{\mathtt{bin}}$ and $\sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$ and further, the challenge space from LaBRADOS (Section 2.1) is not stable under $\sigma_{-1}$.

**Minimize the Number of Witness Parts.** As the input vectors of LNP $\mathbf{s}_3$ and $\mathbf{s}_6$ appear only in the linear part of the final constraint $f$ (Eq. (4)), their masked openings $\mathbf{z}^s_{1,3}$ and $\mathbf{z}^s_{1,6}$ (Eq. (5)) also appear in the linear part of Eq. (14), as well as $\mathbf{z}_2$. Thus, we could think of concatenating (the decomposition parts of) these three vectors. Moreover, Eq. (14) considers the dot-products between $\mathbf{z}^s_{1,i}$ for $i \in \{1,2,4,5\}$, $\mathbf{z}^v_{1,1}, \mathbf{z}^v_{1,2}$, so all of these should be considered as separate witness vectors for LaBRADOS. Thus, we feed LaBRADOS with the witness made of the decomposition parts of each vector $\mathbf{z}^s_{1,1}, \mathbf{z}^s_{1,2}, \mathbf{z}^s_{1,4}, \mathbf{z}^s_{1,5}, \mathbf{z}^v_{1,1}, \mathbf{z}^v_{1,2}, (\mathbf{z}^s_{1,3}, \mathbf{z}^s_{1,6}, \mathbf{z}_2)$, along with $\mathbf{x}_{\mathtt{bin}}, \sigma_{-1}(\mathbf{x}_{\mathtt{bin}})$. The statement that this witness satisfies is created by adapting Eqs. (11), (12) and (14), when replacing the $\mathbf{z}_i$'s with their decomposition parts $\mathbf{z}_i^{(0)}, \mathbf{z}_i^{(1)}$. For this, we use the fact that $\langle \mathbf{z}_i^{(0)} + b_i \mathbf{z}_i^{(1)}, \mathbf{z}_i^{(0)} + b_i \mathbf{z}_i^{(1)} \rangle = \langle \mathbf{z}_i^{(0)}, \mathbf{z}_i^{(0)} \rangle + b_i (\langle \mathbf{z}_i^{(0)}, \mathbf{z}_i^{(1)} \rangle + \langle \mathbf{z}_i^{(1)}, \mathbf{z}_i^{(0)} \rangle) + b_i^2 \langle \mathbf{z}_i^{(1)}, \mathbf{z}_i^{(1)} \rangle$, for a basis $b_i$. The norm checks on (the splits of) $\mathbf{z}_i^{(0)}$, $\mathbf{z}_i^{(1)}$ are set simply by dividing the bounds appearing in Eqs. (15) to (17) by the respective bases $b_i$, see [13, Sec. 5.3].

### 3.5 The Final Composed Protocol

Our succinct zero-knowledge protocol (Fig. 1) is a combination of LaBRADOS, LNP-Lite and LaBRADOR. It takes as input short witness vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \mathcal{R}^n$, satisfying norm bounds $\|\mathbf{s}_i\| \leq \beta_i$ and constant-term and full-domain dot-product constraints of type (1).

**From LaBRADOS to LNP-Lite.** We feed the input above into several iterations of LaBRADOS with hiding outer commitments (Section 3.1), until the witness shortens to be fed into the zero-knowledge part. We prove this witness with LNP-Lite. Then we transform the output of LaBRADOS $(\mathbf{z}_0, \mathbf{z}_1, \mathbf{x}_{\mathtt{mid}}, \mathbf{x}_{\mathtt{bin}})$ into LNP-Lite input as $(\mathbf{s}_1 \coloneqq \mathbf{z}_0, \mathbf{s}_2 \coloneqq \mathbf{z}_1, \mathbf{s}_3 \coloneqq \mathbf{x}_{\mathtt{mid}}, \mathbf{s}_4 \coloneqq \mathbf{x}_{\mathtt{bin}}, \mathbf{s}_5 \coloneqq \sigma_{-1}(\mathbf{s}_4), \mathbf{s}_6)$, where $\mathbf{s}_5$ is an auxiliary vector used to prove that $\mathbf{x}_{\mathtt{bin}}$ is binary, and $\mathbf{s}_6$ is made of additional masks required for zero-knowledge (Section 3.3). The checks of LaBRADOS (see Section 3.1 and Fig. 8) include the correctness of the outer and middle commitments, the dot-product for proving $\mathbf{x}_{\mathtt{bin}}$ is binary, the shortness of $\mathbf{z}, \mathbf{x}_{\mathtt{mid}}$, and

(check commits.) $\mathbf{Az} = \sum_{i \in [r]} c_i \mathbf{t}_i$,

(check garbage) $\langle \mathbf{z}, \mathbf{z} \rangle = \sum_{i,j \in [r]} c_i c_j \mathbf{g}_{ij}$, $\sum_{i \in [r]} \langle \varphi_i, \mathbf{z} \rangle c_i = \sum_{i,j \in [r]} c_i c_j \mathbf{h}_{ij}$,

(check quadratic) $\sum_{i,j \in [r]} a_{ij} \mathbf{g}_{ij} + \sum_{i \in [r]} \mathbf{h}_{ii} - b = 0$,

(check ct. coeff.) $\mathtt{ct}(h_i) = 0$.

These checks, together with the correctness of $\mathbf{s}_5$ with respect to $\mathbf{s}_4$, are dot-product constraints that LNP-Lite can support and thus become input of LNP-Lite.

**From LNP-Lite to LaBRADOS.** Now we run LNP-Lite protocol just once and we feed its output into LaBRADOS, run several times. The LNP-Lite output $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{x}_{\mathtt{bin}})$ could be treated as LaBRADOS input (see Section 3.4) as $(\mathbf{z}^s_{1,1}, \mathbf{z}^s_{1,2}, \mathbf{z}^s_{1,4}, \mathbf{z}^s_{1,5}, \mathbf{z}^v_{1,1}, \mathbf{z}^v_{1,2}, (\mathbf{z}^s_{1,3}, \mathbf{z}^s_{1,6}, \mathbf{z}_2), \mathbf{x}_{\mathtt{bin}}, \sigma_{-1}(\mathbf{x}_{\mathtt{bin}}))$, and the LNP-Lite verifier checks (Eqs. (11) to (19)), adapted as in Section 3.4, become indeed compatible with the LaBRADOS constraint system (1). This time, the LaBRADOS iterations have no longer hiding outer commitments, thanks to the zero-knowledge of LNP-Lite.

**From LaBRADOS to LaBRADOR.** Lastly, we complete our final composed protocol by further running iterations of LaBRADOR after LaBRADOS, which was already considered in [15].

### 3.6 The Security of the Pack

We analyze the security of our succinct zero-knowledge composition. Completeness follows easily as each iteration of it, LaBRADOS, LNP-Lite and LaBRADOR, satisfies completeness. Soundness follows by coupling the

soundness of each iteration, via [13, Lem.3.7.], and the soundness of each iteration follows by Theorem 4 for LaBRADOS, Theorem 1 for LNP-Lite and by [13, Thm.5.1] for LaBRADOR, each combined with Lemma 5 (when extending with outer commitments). Zero-knowledge is straightforward. To simulate a proof of the entire composition, the simulator can first simulate the prover messages from the iterations of LaBRADOS before LNP-Lite as commitments of 0, which, by hiding the outer commitments, are indistinguishable from the real ones. When getting to LNP-Lite, he can first simulate the last message of LNP-Lite and compute the outer commitments of everything from this message, except of the masked openings, which become the remaining LNP prover's messages. Then, when entering the next iterations of LaBRADOS and LaBRADOR, he can simply honestly compute the remaining transcript of the composition, feeding them with the simulated LNP-Lite output. We discuss the security of the non-interactive version in

## 4 Zero-Knowledge in Practice: Case Studies

We evaluate the performance of our succinct zero-knowledge proof system from Section 3 implementing relevant benchmark examples. To this end, we describe our instantiations of some known primitives that we tailor to the constraints our proof system supports.

As a first example, detailed in Section 4.1, we describe a construction of *verifiable symmetric encryption* built upon a pseudorandom generator, obtained from iterative *learning with rounding*. We note that one can use such a scheme in practice with a public key verifiable encryption scheme that encrypts the PRG seed to construct an efficient public key verifiable encryption scheme for long messages. In Section 4.2, we employ a variant of Ajtai's collision-resistant hash that allows to commit to large inputs (in a streaming manner), and for membership proofs (via Merkle trees). In Section 4.3, we describe a verifiable *mixing subroutine* that, combined with the previous building blocks, leads to a potential candidate of a *full-fledged cryptographic hash function*.[4] We choose to benchmark *blind signatures* as a useful example that the verifiable hash enables. We also describe, for theoretical interest, how one can use such a hash to construct *Incrementally Verifiable Computation* (IVC), which is not yet practically achieved by *any* lattice-based constructions.

For all the use cases (except IVC), we provide implementations on top of the LaZer library [38] and report the prover and verifier timings, both when zero-knowledge is enabled (labelled as "zk") and when it is not (labelled as "not zk"). The secure version is the zero-knowledge-enabled one. The reason why we also include the timings without zero-knowledge is to assess the overhead of adding zero-knowledge to the proof. All the timings come from running the benchmarks on a single core of an Intel Core i7-11850H.

We remark that the proof size remains quite constant across use cases and input sizes, since the proof size is dominated by the last round of LaBRADOR, therefore we do not report them explicitly. Excluding some outliers for the smallest input sizes, the proof size for non-zero-knowledge examples is around $95 - 105$KB, whereas the zero-knowledge enabled ones result in around $115-125$KB. The difference of around 20KB comes from the commitments sent in LNP-Lite and in the extra rounds of LaBRADOS that come after LNP-Lite. This relatively small constant overhead in the proof size is achieved thanks to the strategy of running LNP-Lite only once the input is small enough and compressing its messages.

In this section, we work with the ring $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$ with $d = 512$. All operations in this ring can be translated over the proof system ring (with degree $d = 256$), according to [35, Sec. 2.8].

### 4.1 Pseudorandom Generator

The goal of the pseudorandom generator (PRG) is to produce many pseudorandom polynomials (output) from a few random ones (seed). The idea is to apply an expansion function and use part of its output as a new seed and part to compose the PRG output. In this construction, the pseudorandomness of the computed values is based on the hardness of the M-LWR problem.

In detail, the expansion function takes as input two binary polynomials $x_0, x_1 \in \mathcal{R}_2$, it computes $\lceil \mathbf{A}(x_0, x_1)^T \rfloor_{256\to 64}$ for $\mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 2}$, and binary decomposes the result. As the rounding drops the two least

[4] We have not performed any cryptanalysis of this function, so it should be viewed as a possible line of research for further study orthogonal to the main results of the paper.

significant bits, we write abusing notation $\lceil \mathbf{A}\ (x_0, x_1)^T \rfloor_{256\to 64} = \mathbf{G}_2(y^{(2)}, \ldots, y^{(7)})$ with $\mathbf{G}_2 = (1, 2, \ldots, 2^5)$. Let $s_0^{(6)}, s_0^{(7)} \in \mathcal{R}_2$ be the seed. The pseudorandom generator works as follows:

$$\begin{aligned} \lceil \mathbf{A}\ (s_0^{(6)}, s_0^{(7)})^T \rfloor_{256\to 64} &= \mathbf{G}_2\ (s_1^{(2)}, \ldots, s_1^{(7)})^T \\ \lceil \mathbf{A}\ (s_1^{(6)}, s_1^{(7)})^T \rfloor_{256\to 64} &= \mathbf{G}_2\ (s_2^{(2)}, \ldots, s_2^{(7)})^T \\ &\ldots \end{aligned} \tag{20}$$

The output is $\mathbf{s} \coloneqq (s_1^{(2)}, s_1^{(3)}, s_1^{(4)}, s_1^{(5)}, s_2^{(2)}, \ldots)^T$.

**Proof-friendly.** Note that the expansion equation can be equivalently written as $\mathbf{A}(s_i^{(6)}, s_i^{(7)})^T = \mathbf{G}_2'(s_{i+1}^{(0)}, \ldots, s_{i+1}^{(7)})^T \bmod 256$ with $\mathbf{G}_2' = (1, -2, 2^2, \ldots, 2^7)$, where -2 is required to write the product coefficients in [-2, 253] and correctly drop the two least significant bits. To prove correct execution of the PRG, one proves knowledge of the input $s_0^{(6)}, s_0^{(7)}$ and the intermediate witnesses $s_i^{(0)}, \ldots, s_i^{(7)}$ that are binary and satisfy $\mathbf{A}(s_i^{(6)}, s_i^{(7)})^T - 256 l_i = \mathbf{G}_2'(s_{i+1}^{(0)}, \ldots, s_{i+1}^{(7)})^T$ for some $l_i \in \mathcal{R}$ over a ring $\mathcal{R}_q$ with sufficiently large modulo $q$. This is natively proven by the proof system in Section 3.

**Security.** This function is a pseudorandom generator under the hardness assumption of some Module-LWR problem. Informally, an adversary distinguishing a PRG output with advantage $\epsilon$ distinguishes with advantage $\epsilon/k$ a pair of successive hybrids, hence an instance of the Module-LWR problem with parameters $q = 256$, $p = 64$, $m' = 1$, $n' = 2$, $\chi = \mathcal{U}(\mathcal{R}_2)$, whose concrete hardness we analyze with [4]. The full proof is deferred to Section E.2.

**Output transformations.** The PRG output is a binary vector over the ring $\mathcal{R}$ of degree $d = 512$. However, one may target uniformly random vectors over the quotient ring $\mathcal{R}_q$ for $q > 2$, or of a different degree $d'$.

To obtain a uniformly random vector $\mathbf{t} \in \mathcal{R}_q^l$, one needs a binary vector $\mathbf{s} \in \mathcal{R}_2^{l\cdot\ell}$ with $\ell \coloneqq \lceil \log_2 q \rceil + 1$. $t_i$ is obtained by multiplying a block of $\mathbf{s}$, $\tilde{\mathbf{s}}_i \in \mathcal{R}_2^\ell$, by a matrix $\mathbf{B} = (\mathbf{B}', \mathbf{I}_1)$ where $\mathbf{B}' \leftarrow \mathcal{R}_q^{1\times\lceil \log_2 q \rceil}$:

$$\mathbf{t}^T = (t_0, t_1, \ldots) = \mathbf{B} \cdot (\tilde{\mathbf{s}}_0, \tilde{\mathbf{s}}_1, \ldots) \bmod q.$$

The security follows from the hardness of some M-LWE problem with the hybrid argument. We defer its proof to Section E.3.

To obtain an element in a ring of different degree, one can use the isomorphism described in [35, Sec. 2.8].

**Verifiable Symmetric Encryption** Pseudorandom generators are often used in stream ciphers, where, given a binary plaintext $m$ and a PRG-expanded key, the ciphertext $c$ is obtained as $c = m \oplus \mathrm{PRG}(k)$. Noticing that this equation can be equivalently written as $c = m + \mathrm{PRG}(k) \bmod 2$, one can achieve verifiable symmetric encryption by proving knowledge of binary plaintext $m$, key $k$, and lifting $l$ satisfying $c - 2l = m + \mathrm{PRG}(k)$ over the proof system ring.

In Table 1, we present the results of benchmarking the expansion component of that construction. In particular, we expand a small seed of fixed-length (1024 bits) into outputs of increasing length, and use our proof system to prove the correctness of that expansion.

### 4.2 Collision Resistant Hash Function

The hash function described in the following is based on the Merkle-Damgård construction to build collision-resistant hash functions [23,39]. Informally, the arbitrary-length input is divided in blocks, and a compression function is applied iteratively to them to obtain a fixed-length output. If the compression function is collision resistant, also the hash function is. In this construction, collision resistance is based on the hardness of the M-SIS problem.

| Output Dimension | Prover | | Verifier | |
|---|---|---|---|---|
| | zk | not zk | zk | not zk |
| $2^6$ | 0.86 | 0.11 | 0.44 | 0.09 |
| $2^8$ | 1.22 | 0.23 | 0.54 | 0.18 |
| $2^{10}$ | 1.67 | 0.66 | 0.76 | 0.38 |
| $2^{12}$ | 2.97 | 1.94 | 1.3 | 0.93 |
| $2^{14}$ | 6.71 | 5.88 | 2.72 | 2.37 |
| $2^{16}$ | 25.47 | 25.04 | 9.14 | 8.6 |

**Table 1.** Timing results in seconds for proving the **expansion** of a seed into vectors of binary polynomials. The dimension refers to the number of output polynomials of degree 512.

In detail, let $\mathbf{x}$ be an arbitrary-length binary input, written as a polynomial vector over the ring $\mathcal{R}$ [5]. Firstly, a padding is applied to make the input divisible into blocks. It consists of the minimum number of zero polynomials, followed by one polynomial encoding the dimension of the input in binary:

$$\text{padded}(\mathbf{x}) = (\mathbf{x}, \mathbf{0}, \text{poly}(\dim(\mathbf{x}))).$$

The compression function is composed by matrix multiplication over $\mathcal{R}_{256}$, followed by binary decomposition. Let $\mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 16}$, we define the compression function $f : \mathcal{R}_2^{16} \to \mathcal{R}_2^8$ as

$$\mathbf{x} \mapsto \mathbf{y} := \text{BinDecomp}(\mathbf{A}\mathbf{x} \bmod 256).$$

With abuse of notation, we denote it as $\mathbf{A}\mathbf{x} = \mathbf{G}_2\mathbf{y} \bmod 256$ with $\mathbf{G}_2 := (1, 2, \ldots, 2^7)$. Finally, let $\mathbf{x}_0, \mathbf{x}_1, \ldots \in \mathcal{R}_2^8$ be the blocks of 8 polynomials composing the padded input, and let $\text{IV} \in \mathcal{R}_2^8$ be a fixed initial vector. We compute as follows:

$$\begin{aligned} \mathbf{A}(\text{IV},\mathbf{x}_0) &= \mathbf{G}_2\mathbf{y}_0 \bmod 256 \\ \mathbf{A}(\mathbf{y}_0,\mathbf{x}_1) &= \mathbf{G}_2\mathbf{y}_1 \bmod 256 \\ &\ldots \end{aligned} \tag{21}$$

The output is the last value of $\mathbf{y}_i$.

**Proof-friendly.** To prove knowledge of a preimage of the hash function, one proves knowledge of the input blocks $\mathbf{x}_i$ and the intermediate witnesses $\mathbf{y}_i$ that are binary and satisfy over the proof system ring $\mathbf{A}(\mathbf{y}_{i-1}, \mathbf{x}_i) - 256l_i = \mathbf{G}_2\mathbf{y}_i$ for some $l_i \in \mathcal{R}$.

**Security.** The hash function is collision resistant under the hardness assumption of some Module-SIS problem, whose concrete hardness is more than 256-bits [4]. At a high level, this is because from an adversary that finds collisions in the hash function, one constructs an adversary that finds collisions in the compression function, namely pairs $(\mathbf{y}_{i-1}, \mathbf{x}_i) \neq (\mathbf{y}'_{i-1}, \mathbf{x}'_i)$ such that $\mathbf{A}(\mathbf{y}_{i-1}, \mathbf{x}_i) = \mathbf{A}(\mathbf{y}'_{i-1}, \mathbf{x}'_i) \bmod 256$, and obtains a solution to the M-SIS problem. The full proof is deferred to Section E.1.

**Committing to Large Inputs** The hash function presented above can be used as an alternative to the Ajtai commitment for inputs of large dimension. The advantage is that the public key and modulus are smaller and fixed. We show results in Table 2 using our proof system to prove correctness of such commitments to inputs of increasing length.

**Batched Proof of Membership** Suppose that we want to prove knowledge of a private element contained in a public set of $2^N$ elements (e.g. users public keys). We can do so by proving knowledge of a path in a Merkle tree of height $N$ computed as follows:

[5] If the input is originally not in this ring, a padding may be required.

| Input Dimension | Prover | | Verifier | |
|---|---|---|---|---|
| | zk | not zk | zk | not zk |
| $2^6$ | 1.3 | 0.12 | 0.6 | 0.09 |
| $2^8$ | 1.69 | 0.37 | 0.67 | 0.24 |
| $2^{10}$ | 1.77 | 0.73 | 0.78 | 0.41 |
| $2^{12}$ | 3.14 | 2.12 | 1.38 | 0.98 |
| $2^{14}$ | 9.2 | 7.4 | 3.5 | 2.93 |
| $2^{16}$ | 29.59 | 29.6 | 9.71 | 9.36 |

**Table 2.** Timing results in seconds for proving the **compression** of vectors of polynomials. The dimension refers to the number of input polynomials of degree 512.

- The leaves are the hashes of the elements in the set.
- The parent node $\mathbf{y} \in \mathcal{R}_2^8$ is computed from the children $\mathbf{x}_l, \mathbf{x}_r \in \mathcal{R}_2^8$ using the compression function $f$ as $\mathbf{A}(\mathbf{x}_l, \mathbf{x}_r) = \mathbf{G}_2\mathbf{y} \bmod 256$.

To prove knowledge of a path in the tree from the root to the leaf corresponding to the private element in zero-knowledge, one proves knowledge of the nodes $\mathbf{v}_i \in \mathcal{R}_2^8$, their siblings $\mathbf{w}_i \in \mathcal{R}_2^8$, and the path indicators $x_i \in \{0, 1\}$ (encoding 0 if $\mathbf{v}_i$ is the left child, 1 otherwise) satisfying, for any level index $i$,

$$(1 - x_i) \cdot \mathbf{A}(\mathbf{v}_i, \mathbf{w}_i) + x_i \cdot \mathbf{A}(\mathbf{w}_i, \mathbf{v}_i) = \mathbf{G}_2\mathbf{v}_{i-1} \bmod 256.$$

All witnesses must be binary, the path indicators also integers.

In our results in Table 3, we consider a tree of height 64 and prove knowledge of several private elements belonging to the tree.

| Number of Paths | Prover | | Verifier | |
|---|---|---|---|---|
| | zk | not zk | zk | not zk |
| 1 | 1.8 | 0.71 | 0.9 | 0.4 |
| 2 | 1.94 | 0.87 | 0.88 | 0.5 |
| 4 | 2.36 | 1.44 | 1.16 | 0.76 |
| 8 | 3.74 | 2.53 | 1.57 | 1.2 |
| 16 | 5.85 | 4.8 | 2.43 | 2.1 |
| 32 | 10.41 | 9.44 | 4.36 | 3.91 |
| 64 | 22.57 | 20.11 | 8.67 | 8.36 |
| 128 | 38.97 | 36.82 | 13.3 | 12.42 |

**Table 3.** Timing results in seconds for proving **membership** of increasing numbers of elements in a tree of height 64.

### 4.3 Beyond Collision Resistance

In this section, we attempt to construct a cryptographic hash function candidate with the intention of instantiating a random oracle with it. This hash function relies on the *mixing* subroutine, a function aiming to behave as much as possible like a truly random one. We use this instantiation in the construction of a blind signature.

**Mixing** The goal of the mixing is to obtain a function that behaves as much as possible like a truly random one, and can be used in practice to instantiate it (eventually combined with other components).

Let the input vector be $\mathbf{z}_0 \in \mathcal{R}_2^9$. The mixing consists of applying a fixed number of times two functions, described by public parameters $\mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 9}$, $\mathbf{A}' \leftarrow \mathcal{R}_{257}^{1\times 8}$, and $c_0, \dots c_4 \leftarrow \mathcal{R}_{256}$. Let $\mathbf{G}_2 = (1, 2, \dots, 2^7)$, $\mathbf{G}_2' = (1, 2, \dots, 2^8)$. Then $\mathbf{A}\mathbf{z}_0 + c_0 = \mathbf{G}_2\mathbf{z}_1' \mod 256$ and for $i \in [4]$:

$$\begin{aligned} \mathbf{A}'\mathbf{z}_i' &= \mathbf{G}_2'\mathbf{z}_i \mod 257 \\ \mathbf{A}\mathbf{z}_i + c_i &= \mathbf{G}_2\mathbf{z}_{i+1}' \mod 256 \end{aligned} \tag{22}$$

If the mixing achieves its purpose, in practice, it can only be validated by cryptanalysis; however, in the following, we give the intuition behind its components. The matrix multiplication provides collision resistance and *confusion*, namely, makes the relation between input and output as complex as possible. The binary decomposition yields *diffusion*, spreading the influence of an input polynomial across many output ones; additionally, it introduces some non-linearity. The alternating moduli aim at further avoiding patterns that an adversary could exploit in distinguishing the output from random. This idea also appeared in [19]. Finally, the public constants $c_i$ ensure that, at any iteration, the zero input is not assigned to the zero output.

**Proof-friendly.** Due to the operations over the ring $\mathcal{R}_{257}$, proving the mixing relation additionally requires ensuring that the element $\mathbf{z}_i$ is the binary representation of a polynomial modulo 257, i.e. that either the most significant bit is 0, or all the others are. This can be proven by imposing that $z_i^{(0)} + z_i^{(8)}$, ..., $z_i^{(7)} + z_i^{(8)}$ are binary.

**Cryptographic Hash Function Candidate** Many protocols involving hash functions prove security in the random oracle model (ROM), [9], namely, treating a hash function as a truly random function. One could attempt to construct such a candidate hash function by combining the previous building blocks. First, the collision resistant hash function in Section 4.2 is applied to compress the arbitrary-length input. Second, we apply a "mixing component" described in Section 4.3; however, one could use any alternative instantiation of a random function. Finally, we can apply the pseudo-random generator in Section 4.1. It can be proved that if the mixing is simulated by a random oracle, the candidate full-fledged hash function is pseudorandom. Our "compress-mix-expand" cryptographic hash function construction cannot be proven secure in the indifferentiability framework (c.f. [20]) by just viewing the mixing part as a random oracle. Nevertheless, we believe that because the lattice-based compression and expansion components are well-understood cryptographic constructions, one could try to cryptanalyze the function as a whole. We leave this line of research to future work.

**Blind signatures** Blind signatures are protocols in which the user obtains signatures of their messages without revealing to the signer any information about the message. In [12], the authors propose a blind signature scheme from lattice assumptions, whose security is proven in the random oracle model. At a high level, in this protocol the user commits to the message $\mu$ computing $\mathbf{c} = \mathbf{B}\mathbf{r} + H(\mu, H(\mathbf{r})) \in \mathcal{R}_q$ with commitment matrix $\mathbf{B} \leftarrow \mathcal{R}_q^{1\times 2}$ and randomness $\mathbf{r} \leftarrow \chi^2$, and sends to the signer the commitment $\mathbf{c}$ with a zero-knowledge proof of knowledge for its correctness. The issuer signs the commitment providing a short preimage $\mathbf{s}$, i.e. such that $\mathbf{A}\mathbf{s} = \mathbf{c}$. The user proves that they have a valid signature for $\mu$, with a zero-knowledge proof of knowledge for $\mathbf{s}$, $\mathbf{r}$ such that $\mathbf{A}\mathbf{s} = \mathbf{B}\mathbf{r} + h$. While this linear proof can be efficiently done with the original `LNP`, proving the correctness of the commitment $\mathbf{c}$ benefits from the proof system in Section 3. In the following, we benchmark this proof, when instantiating the hash function with the hash function candidate presented in Section 4.3.

| Message Dimension | Prover | | Verifier | |
|---|---|---|---|---|
| | zk | not zk | zk | not zk |
| 10 | 1.22 | 0.34 | 0.62 | 0.23 |

**Table 4.** Timing results in seconds for proving the knowledge of a commitment in the blind signature protocol. The dimension refers to the number of input polynomials of degree 512.

**Incrementally Verifiable Computation from Folding Scheme** Folding schemes fold witness-statement pairs for a relation into a single pair. They are used as building blocks for incrementally verifiable computation protocols (IVC) [42] and proof-carrying data (PCD) [22]. Both primitives follow a general blueprint: the verifier circuit (including hash functions after the Fiat–Shamir transformation) is expressed as a set of constraints for an input relation, and the folding-scheme prover recursively folds the witness-statement pairs of that verifier circuit at each computation step. A key advantage is that the proof size and verification time of IVC/PCD are independent of the number of computation steps, and the memory required by the prover is also step-independent, unlike "monolithic" SNARKs. A crucial component, and the main reason current constructions remain impractical, is the hash function embedded in the verifier circuit, which must be efficiently verifiable.

## 5 Implementation

We have implemented the benchmarks reported in Section 4 on top of the LaZer library [38]. They can be reproduced at

`https://github.com/lazer-crypto/lazer`.

Those benchmarks depend on two core modules that we have implemented: the proof system of this work, accessible at `https://github.com/lazer-crypto/labrador`; and a C library for the hash function of Section 4.3, located in the `src/lattice-hash` folder of the `lazer-crypto/lazer` GitHub repository.

### 5.1 Proof System

We implement the proof system of this work on top of the C arithmetic and protocols of [15], which in turn builds on top of [13, 38, 40]. We describe here our main contributions.

`Dachshundv2`**.** In practice, the statement that a user wants to prove may not be directly compatible with the statements that `LaBRADOR` can prove. The reason for this is that this protocol expects a small number of witness vectors for efficiency, and there is some pre-processing required to prove exact $\ell_2$-norms and binary constraints. This was addressed by the LaZer library [38] with the `Dachshund` frontend. We re-implement it to make it compatible with `LaBRADOS` and extend it in several fronts, resulting in the protocol that we refer to here as `Dachshundv2`. Therefore, one can see `Dachshundv2` as the frontend protocol whose output is fed to the first iteration of `LaBRADOS` in Figure 1.

First, in addition to exact $\ell_2$-norms and binary, the user can indicate instead that the $\ell_2$-norm of a vector is to be proven approximately, giving both the public bound it satisfies and the public bound that the proof system should guarantee. `Dachshundv2` will then merge many witness vectors, as long as the norm guarantee resulting from projecting all of them together in `LaBRADOS` is small enough for all the vectors involved (also accounting for the slack from the projection). This makes the proof more efficient, since proving exact norms requires proving a quadratic over the integers, and thus a larger modulus to ensure there is no wrap-around in the quadratic.

Second, we support arbitrary quadratics between witness vectors. We do so by committing to the witness and obtaining polynomial challenges that can aggregate the quadratic equations, in a similar fashion to how `Dachshund` deals with exact $\ell_2$-norms (see [38, Sec. 4]). We then extend the witness with two witness vectors that contain the concatenation of the left and right parts of all the quadratic terms, multiplying one of the sides by the polynomial challenges. This means that, after proving the consistency of these new parts with respect to the original witness, the original quadratics can be proven by simply proving the dot-product between these two new witness vectors. As a consequence, the original witness vectors appearing in quadratics can safely be merged together, as long as their norm requirements allow it. Additionally, we support quadratics taking place over rings with a higher degree than that of the proof system by expanding one of the sides of the dot product into the rotation matrix and aggregating its rows with challenges before adding it to the new witness vectors (see [35, Sec. 2.8.] for expressing multiplication over higher-degree rings in terms of a rotation matrix).

`LNP-Lite`. Both the original `LNP` construction and its implementation are not suitable for use as a round in our composed protocol. The conceptual changes and new techniques are described in Sections 3.2 and 3.3, with the two-step projection being the core improvement. Those improvements allow `LNP-Lite` to use the same small modulus as `LaBRADOR`, allowing us to base our implementation on `LaBRADOR`'s AVX-512 optimized single-precision coefficient arithmetic. To enable protocol composition, the verifier was expressed in dot-product constraints using `LaBRADOR`'s constraint library. Both `LNP-Lite`'s input and output are `LaBRADOR`-compatible constraint systems (plus prover messages). The generation of `LNP-Lite` parameters plays a crucial role in determining the round structure of the composed protocol: if no secure `LNP-Lite` parameters are found, more preceding `LaBRADOS` rounds will be added.

`Pack`. We implement a `Pack` logic that orchestrates the composition of the different protocols. Its parameter generation decides how many iterations to perform of each protocol and in which order, according to the flow of Figure 1. By default, iterations of `LaBRADOR` and `LaBRADOS` are added as long as the improvement in witness size is more than 10%. Internally, the parameter generations of those two protocols look for the split of the witness that leads to the smallest output witness. At the end of the protocol, the last iteration of `LaBRADOR` is the special last round described in [13] where no middle (nor outer) commitments are computed. When `LaBRADOS` is run before `LNP-Lite`, then the former is indicated to use randomness in the outer commitments, and further iterations are run until `LNP-Lite` successfully finds secure parameters. Additional complexity comes from the fact that the parametrization of `LaBRADOS` changes depending on whether the protocol that comes after is `LaBRADOS`, `LNP-Lite` or `LaBRADOR`, since each of them proves the $\ell_2$-norms with different slacks. The inclusion of the `LNP-Lite` round in between `LaBRADOS` rounds is optional, and determined by a flag of whether the proof requires zero-knowledge.

`Python Wrapper`. LaZer provides a convenient way of defining the statement and populating the witness that is then fed to `LaBRADOR` or to `LNP`. We extend LaZer so that the proof system used underneath is the composition of protocols described in Section 3, with an optional zero-knowledge flag that determines whether a round of `LNP-Lite` is performed or not. We additionally extend the Python API to support the definition of quadratics and approximate $\ell_2$-norms, powered by our new `Dachshundv2`.

### 5.2 Lattice Hash

The proof-friendly hash of Section 4.3 is realised as a stand-alone C library exposing compression, mixing, and expansion as separate entry points, with padding left to the caller. The library is independent of the rest of the toolkit and can be used externally; a companion Python module ships inside LaZer, integrating the construction with the proof-system frontend.

The hash is defined over $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$ with $d = 512$. Multiplications are evaluated in $\mathcal{R}_q$ for an auxiliary prime $q$ chosen large enough that no reduction modulo $q$ ever occurs on a reused result, and small enough to admit a length-$d$ negacyclic NTT and Intel HEXL's AVX-512 modular kernels [14]. The choice of $q$ is a coding artefact and does not affect any output. Non-NTT stages, namely splitting modulo 256 or 257 and bit decomposition, are vectorised with AVX-512.

Compression (described in Section 4.2) applies a fixed public matrix $\mathbf{A} \in \mathcal{R}^{1\times 16}$ to the concatenation of two binary inputs $\mathbf{x} \in \{0,1\}^{16d}$, with $\mathbf{A}$ precomputed in NTT form. Each output coefficient is split into a remainder $\mathbf{r}$ and quotient $\mathbf{q}$; the surrounding proof system has no native primitive for reducing modulo 256, so the rounding is tracked by committing to both, allowing $\mathbf{Ax} = 256 \cdot \mathbf{q} + \mathbf{r}$ to be enforced as a constraint in the ambient ring. Mixing (described in Section 4.3) alternates two variants of the same primitive differing only in rounding modulus: one reduces modulo 256, producing eight binary output layers (with a public shift selected by round index), the other modulo 257, producing nine. The latter is implemented with a precomputed Shoup multiplier for $257^{-1} \bmod q$, avoiding per-coefficient division. Expansion (done via PRG from Section 4.1) reuses the same NTT and splitting machinery: each iteration computes $\mathbf{a}_1\mathbf{s}_1 + \mathbf{a}_2\mathbf{s}_2$, rounds modulo 256, and bit-decomposes the result, feeding two bit-planes into the next iteration and emitting the rest as digest material.

All public matrices and shifts are derived deterministically from a single fixed published seed via SHAKE128, isolating all randomness in one string and enabling independent audit. All primitives are data-oblivious: HEXL's NTT and elementwise kernels, the custom AVX-512 helpers for splitting and bit decomposition, and

the Shoup-based reduction modulo 257 all run in time independent of operands, with no input-dependent branches or memory accesses.

A SageMath transcript checker recomputes each stage symbolically in $\mathbb{Z}[X]/(X^d + 1)$ as an independent correctness oracle. The C library, checker, and parameter generator are self-contained, and the construction can be used as a Fiat–Shamir or Merkle-tree hash outside of LaZer.

## Acknowledgements

This work was supported by the ERC Consolidator Grant PLAZA (101002845), the ERC Starting Grant GLAZE (101222273), the SNSF Grant PretZel (10008425), Research Council of Finland (358951) and Technology Industries of Finland Centennial Foundation's Future Makers 2025 grant (project SPICED). We thank Shubham Khurana (Motilal Nehru National Institute of Technology Allahabad, India) for spotting a gap in a previous version of the simulatability proof of `LNP-Lite`.

## References

1. Aardal, M.A., Aranha, D.F., Boudgoust, K., Kolby, S., Takahashi, A.: Aggregating falcon signatures with LaBRADOR. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part I. LNCS, vol. 14920, pp. 71–106. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68376-3_3
2. Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the LWE, NTRU schemes! In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 351–367. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_19
3. Albrecht, M.R., Davidson, A., Deo, A., Gardham, D.: Crypto dark matter on the torus - oblivious PRFs from shallow PRFs and TFHE. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 447–476. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1_16
4. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml
5. Alperin-Sheriff, J., Peikert, C.: Circular and KDM security for identity-based encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 334–352. Springer, Berlin, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_20
6. Attema, T., Lyubashevsky, V., Seiler, G.: Practical product proofs for lattice commitments. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 470–499. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_17
7. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. In: Mathematische Annalen. vol. 296, p. 625–635 (1993)
8. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 368–385. Springer, Cham (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_20
9. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596
10. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), https://eprint.iacr.org/2018/046
11. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 103–128. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17653-2_4
12. Beullens, W., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Lattice-based blind signatures: Short, efficient, and round-optimal. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 16–29. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3616613
13. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from module-SIS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 518–548. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_17
14. Boemer, F., Kim, S., Seifu, G., de Souza, F.D.M., Gopal, V.: Intel HEXL: Accelerating homomorphic encryption with intel AVX512-IFMA52. Cryptology ePrint Archive, Report 2021/420 (2021), https://eprint.iacr.org/2021/420

15. Bolboceanu, M., Bootle, J., Lyubashevsky, V., Merino-Gallardo, A., Seiler, G.: Orthus: Practical sublinear batch-verification of lattice relations from standard assumptions. In: Heninger, N., Rosulek, M. (eds.) CRYPTO 2026, Part III. pp. 100–132. LNCS, Springer (2026). https://doi.org/10.1007/978-3-032-35377-1_4, https://doi.org/10.1007/978-3-032-35377-1_4
16. Bolboceanu, M., Bootle, J., Lyubashevsky, V., Merino-Gallardo, A., Seiler, G.: Orthus: Practical sublinear batch-verification of lattice relations from standard assumptions (2026), https://eprint.iacr.org/2026/398
17. Boneh, D., Chen, B.: LatticeFold: A lattice-based folding scheme and its applications to succinct proof systems. In: ASIACRYPT 2025, Part III. pp. 330–362. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5099-9_11
18. Boneh, D., Chen, B.: LatticeFold+: Faster, simpler, shorter lattice-based folding for succinct proof systems. In: CRYPTO 2025, Part VII. pp. 327–361. LNCS, Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01907-3_11
19. Boneh, D., Ishai, Y., Passelègue, A., Sahai, A., Wu, D.J.: Exploring crypto dark matter: New simple PRF candidates and their applications. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 699–729. Springer, Cham (Nov 2018). https://doi.org/10.1007/978-3-030-03810-6_25
20. Canteaut, A., Fuhr, T., Naya-Plasencia, M., Paillier, P., Reinhard, J.R., Videau, M.: A unified indifferentiability proof for permutation- or block cipher-based hash functions. Cryptology ePrint Archive, Paper 2012/363 (2012), https://eprint.iacr.org/2012/363
21. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_26
22. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: Yao, A.C.C. (ed.) ICS 2010. pp. 310–331. Tsinghua University Press (Jan 2010)
23. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 416–427. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_39
24. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 40–56. Springer, Berlin, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_3
25. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES **2018**(1), 238–268 (2018). https://doi.org/10.13154/tches.v2018.i1.238-268, https://tches.iacr.org/index.php/TCHES/article/view/839
26. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
27. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical non-interactive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
28. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 458–487. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-06944-4_16
29. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
30. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. Journal of Cryptology **9**(4), 199–216 (Sep 1996). https://doi.org/10.1007/BF00189260
31. Klooss, M., Lai, R.W.F., Nguyen, N.K., Osadnik, M., Tucci, L.: RoKoko: Lattice-based succinct arguments, a committed refinement. Cryptology ePrint Archive, Paper 2026/575 (2026), https://eprint.iacr.org/2026/575
32. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. DCC **75**(3), 565–599 (2015). https://doi.org/10.1007/s10623-014-9938-4
33. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Berlin, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
34. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 71–101. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_3
35. Lyubashevsky, V., Nguyen, N.K., Plançon, M., Seiler, G.: Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 218–248. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_8

36. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Shorter lattice-based zero-knowledge proofs via one-time commitments. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 215–241. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75245-3_9
37. Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 204–224. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78381-9_8
38. Lyubashevsky, V., Seiler, G., Steuer, P.: The LaZer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In: Luo, B., Liao, X., Xu, J., Kirda, E., Lie, D. (eds.) ACM CCS 2024. pp. 3125–3137. ACM Press (Oct 2024). https://doi.org/10.1145/3658644.3690330
39. Merkle, R.C.: One way hash functions and des. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, New York, NY (1990). https://doi.org/10.1007/0-387-34805-0_40
40. Nguyen, N.K., Seiler, G.: Greyhound: Fast polynomial commitments from lattices. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 243–275. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68403-6_8
41. Pino, R.D., Katsumata, S., Niot, G., Reichle, M., Takemure, K.: Unmasking traccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part VI. pp. 423–456. LNCS, Springer (2025). https://doi.org/10.1007/978-3-032-01887-8_14, https://doi.org/10.1007/978-3-032-01887-8_14
42. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Berlin, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8_1
43. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00060

# Appendix

## A Deferred Preliminaries

We consider the norm of elements in $\mathcal{R}$ to be $\|a\|_p = |a|$ if $a \in \mathbb{Z}$ and $\|a\|_p = \sqrt[p]{\sum |a_i|^p}$ if $a = \sum a_i X^i \in \mathcal{R}$, for $p > 1$. We extend the notation to vectors $\|\mathbf{a}\|_p = \sqrt{\sum_i \|\mathbf{a}_i\|_p^2}$. We remove the subscript of $\|\cdot\|_p$ when $p = 2$. In the quotient ring, the norm of an element in $\mathcal{R}_q$ is the norm of its unique representative $\mathcal{R}$ with coefficients in $[-\frac{q-1}{2}, \frac{q-1}{2}]$.

In power-of-two cyclotomic rings, working over subrings $\mathcal{S}$ of $\mathcal{R}$ is possible, as detailed in [35, Section 2.8.]. Namely, if $\mathcal{S} = \mathbb{Z}[X]/(X^{d'} + 1)$ a subring of $\mathcal{R}$, for $d'|d$, then, there is a norm-preserving bijection $\Phi : \mathcal{R} \to \mathcal{S}^{d/d'}$, using the embedding of $\mathcal{S}$ in $\mathcal{R}$ via $X \to X^{d/d'}$. This bijection can be naturally extended to vectors over $\mathcal{S}$ and $\mathcal{R}$ and over their quotients mod $q$. Using this embedding, one could translate the multiplication by a ring element $a \in \mathcal{R}$ as a linear map over $\mathcal{S}$, described by the so-called *rotation matrix* $\mathtt{Rot}(a) \in \mathcal{S}^{d/d' \times d/d'}$ as $\Phi(ab) = \mathtt{Rot}_a \cdot \Phi(b) \in \mathcal{S}^{d/d'}$. The rotation matrix $\mathtt{Rot}(a)$ is defined as follows

$$\mathtt{Rot}(a) \coloneqq \left(\Phi(a)\ \Phi(aX)\ \ldots\ \Phi(aX^{d/d'-1})\right).$$

We recall the result of [37] which says that short elements in $\mathcal{R}_q$ are invertible.

**Lemma 1 (Cor.1.2, [37]).** *Let $q \equiv 5 \mod 8$ be a prime. Then, any $f \in \mathcal{R}_q$ which satisfies either $0 < \|f\|_\infty < \frac{1}{2}q^{1/2}$ or $0 < \|f\| < q^{1/2}$ is invertible in $\mathcal{R}_q$.*

### A.1 Rejection Sampling and the Gaussian Distribution

In lattice-based zero-knowledge proofs, the prover may want to produce $\mathbf{z}$ whose distribution is independent of a witness vector $\mathbf{v}$, so the verifier would not obtain any secret information. The achievement of this independence is done by *rejection sampling* [33] to ensure a target distribution. We use Gaussian distributions to sample the *mask* $\mathbf{y}$ in our proofs of knowledge.

**Definition 6.** *The discrete Gaussian distribution over $\mathcal{R}^n$ centered at $\mathbf{v} \in \mathcal{R}^n$, with standard deviation $\sigma > 0$ is defined by $D^n_{\mathbf{v},\sigma}(x) \coloneqq \frac{\exp\left(\frac{-\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2}\right)}{\sum_{\mathbf{x}' \in \mathcal{R}^n} \exp\left(\frac{-\|\mathbf{x}'\|^2}{2\sigma^2}\right)}$. When it is centered around $\mathbf{0} \in \mathcal{R}^n$, we simply write $D^n_\sigma = D^n_{\mathbf{0},\sigma}$.*

We also use the following Gaussian tail bound, following from [7, Lem 1.5(i)] and adapted in [6, Lem 2.5].

**Lemma 2.** $\Pr_{\mathbf{z} \leftarrow D^n_\sigma}[\|\mathbf{z}\| < \sigma\sqrt{2nd}] > 1 - 2^{-nd/2 \cdot \log(e/2)} > 1 - 2^{-nd/8}$.

**Lemma 3 ( [33, 36]).** *Let $V \subseteq \mathcal{R}^n$ be a set of vectors with norm at most $B$ and $\rho : V \to [0,1]$ be a probability distribution. Let $\sigma = \gamma B$ a standard deviation, for some constant $\gamma > 0$.*

1. *Let $M \coloneqq \exp(14/\gamma + 1/(2\gamma^2))$. Sample $\mathbf{v} \leftarrow \rho$ and $\mathbf{y} \leftarrow D^n_\sigma$ and set $\mathbf{z} = \mathbf{y} + \mathbf{v}$. Run $b \leftarrow \mathtt{Rej}_1(\mathbf{z}, \mathbf{v}, \sigma)$ from Fig. 4. Then the probability that $b = 0$ is at least $(1 - 2^{-128})/M$ and the distribution of $(\mathbf{v}, \mathbf{z})$, conditioned on $b = 0$, is within statistical distance of $2^{-128}$ of $\rho \times D^n_\sigma$.*
2. *Let $M \coloneqq \exp(1/(2\gamma^2))$. Sample $\mathbf{v} \leftarrow \rho$ and $\mathbf{y} \leftarrow D^n_\sigma$ and set $\mathbf{z} = \mathbf{y} + \mathbf{v}$. Run $b \leftarrow \mathtt{Rej}_2(\mathbf{z}, \mathbf{v}, \sigma)$ from Fig. 4. Then the probability that $b = 0$ is at least $1/(2M)$ and the distribution of $(\mathbf{v}, \mathbf{z})$, conditioned on $b = 0$, is identical to the one obtained as follows: sample $\mathbf{s} \leftarrow \rho$, $\mathbf{z} \leftarrow D^n_\sigma$, conditioned on $\langle \mathbf{z}, \mathbf{v} \rangle \geq 0$ output $(\mathbf{v}, \mathbf{z})$ with probability $1/M$.*

In our zero-knowledge proof, we also deal with rejecting uniformly distributed variables, based on some conditions.

**Lemma 4.** *Let $X$ be a finite set and $A$ a subset of it. Consider $U$ the uniform distribution over $X$ and $V$ the distribution obtained when sampling $x \leftarrow U$ but reject whether $x \notin A$. Let $p = \Pr_{X \leftarrow U}[X \in A] (= |A|/|X|)$. Then $\Delta(U, V) = 1 - p$.*

$\underline{\texttt{Rej}_1(\mathbf{z}, \mathbf{v}, \sigma)}$
01 $u \leftarrow [0, 1)$
02 if $u > \frac{1}{M} \cdot \exp\left(\frac{-2\langle \mathbf{z}, \mathbf{v}\rangle + \|\mathbf{v}\|^2}{2\sigma^2}\right)$
03 return 1 (i.e. *reject*)
04 else
05 return 0 (i.e. *accept*)

$\underline{\texttt{Rej}_2(\mathbf{z}, \mathbf{v}, \sigma)}$
01 if $\langle \mathbf{z}, \mathbf{v}\rangle < 0$
02 return 1 (i.e. *reject*)
03 $u \leftarrow [0, 1)$
04 if $u > \frac{1}{M} \cdot \exp\left(\frac{-2\langle \mathbf{z}, \mathbf{v}\rangle + \|\mathbf{v}\|^2}{2\sigma^2}\right)$
05 return 1 (i.e. *reject*)
06 else
07 return 0 (i.e. *accept*)

**Fig. 4.** Two rejection sampling algorithms: [33] (left) and [36] (right).

*Proof.* Note that the support of the distribution of $V$ is exactly the subset $A$ and moreover, for any $x \in A$, $V(x) = \frac{U(x)}{\Pr[x \in A]}$. By definition of the statistical distance,

$$\begin{aligned}
\Delta(V, U) &= \frac{1}{2} \sum_{x \in X} |V(x) - U(x)| \\
&= \frac{1}{2} \sum_{x \in A} |V(x) - U(x)| + \frac{1}{2} \sum_{x \notin A} |V(x) - U(x)| \\
&= \frac{1}{2} \sum_{x \in A} \left|\frac{U(x)}{\Pr[x \in A]} - U(x)\right| + \frac{1}{2} \sum_{x \notin A} U(x) \\
&= \frac{1}{2} \sum_{x \in A} U(x) \cdot \left(\frac{1}{\Pr[x \in A]} - 1\right) + \frac{1}{2}(\Pr[x \notin A]) \\
&= \frac{1}{2} \sum_{x \in A} U(x) \cdot \left(\frac{1}{p} - 1\right) + \frac{1}{2}(1 - p) \\
&= \frac{1}{2} \cdot \frac{1-p}{p} \cdot p + \frac{1}{2}(1 - p) = 1 - p.
\end{aligned}$$

### A.2 The Johnson-Lindenstrauss Lemmas

These results use Gaussian tail bounds derived under the heuristic of [28] of the projection matrices with entries from the binomial distribution $\texttt{Bin}_1$, having instead normally distributed entries.

**Theorem 2.** *Under the heuristic substitution of $\texttt{Bin}_1$ with the normal distribution of standard deviation $\sqrt{\frac{1}{2}}$, from [28, Cor 3.2], we have for $\mathbf{w} \in \mathbb{Z}^n$,*

$$\Pr_{\mathbf{r} \leftarrow \texttt{Bin}_1^n}[|\langle \mathbf{r}, \mathbf{w}\rangle| > 9.75\|\mathbf{w}\|] < \frac{1}{2^{141}},$$
$$\Pr_{\mathbf{\Pi} \leftarrow \texttt{Bin}_1^{256 \times n}}[\|\mathbf{\Pi}\mathbf{w}\| > \sqrt{337}\|\mathbf{w}\|] < \frac{1}{2^{128}}.$$

We recall the Johnson-Lindenstrauss lemma from [16] that relates the infinity norm of the projection with the $\ell_2$ norm of the witness.

**Theorem 3.** *For every vector $\mathbf{w} \in [\pm q/2]^n$ with $\|\mathbf{w}\|_2 \geq b$, for some bound $b \leq q/61$, we have*

$$\Pr_{\mathbf{\Pi} \leftarrow \texttt{Bin}_1^{256 \times n}}[\|\mathbf{\Pi}\mathbf{w} \bmod q\|_\infty < 0.49b] < \frac{1}{2^{128}}.$$

### A.3 Particular Constraints

Here we recall how binary proofs or exact norm proofs can be given, using lattice techniques from previous work [13, 34].

*Binary proof.* For proving a vector $\tilde{\mathbf{x}} \in \mathcal{R}_q^m$ is made only of ring elements with binary coefficients, or equivalently, the coefficients of any entry of this vector are binary vectors, we make use of the observation [34, Lem 5.2] saying that an integer vector $\mathbf{x} \in \mathbb{Z}^{md}$ is binary if and only if $\langle \mathbf{x}, \mathbf{x} - \mathbf{1}_{md} \rangle = 0$ over the integers, where $\mathbf{1}_n$ is the integer vector with all entries equal to 1. This translates into showing that the corresponding polynomial vector $\tilde{\mathbf{x}}$ over $\mathcal{R}$ satisfies

$$\mathtt{ct}(\langle \sigma_{-1}(\tilde{\mathbf{x}} - \mathbf{1}), \tilde{\mathbf{x}} \rangle) = 0. \tag{23}$$

For proving this, we need to add an auxiliary witness vector, $\sigma_{-1}(\tilde{\mathbf{x}})$. However, one could prove Equation (23) only mod $q$. In order to prove this holds over $\mathbb{Z}$, we would need to also prove that the vector $\tilde{\mathbf{x}}$ is small, i.e. $\|\tilde{\mathbf{x}}\| \leq \sqrt{md}$, by projecting it as before, and thus prove its norm, up to some slack factor $\alpha$. As the left hand side of Equation (23) is $|\langle \mathbf{x}, \mathbf{x} - \mathbf{1}_{md} \rangle| \leq 2\|\mathbf{x}\|^2$, for any $\mathbf{x} \in \mathbb{Z}^{md}$ integer vector, by letting $2 \cdot \alpha^2 \cdot md < q$, we guarantee that Equation (23) holds over the integers.

For a binary vector polynomial input $\mathbf{v} \in \mathcal{R}^n$, the binary check works as follows:

- append $\sigma_{-1}(\mathbf{v})$ to $\mathbf{v}$,
- include constraints on correctness of $\sigma_{-1}(\mathbf{v})$,
- include $\mathtt{ct}(\langle \mathbf{v}, \sigma_{-1}(\mathbf{v} - \mathbf{1}) \rangle) = 0$,
- include approximate norm proof for $\|\mathbf{v}\| \leq \sqrt{d \cdot \dim(\mathbf{v})}$.

*Automorphism proof.* As particular constraints, such as binary or exact norm proofs, involve adding new witness vectors when applying the automorphism $\sigma_{-1}$, one needs to prove their correctness. More precisely, as in [1, 13], we have to prove that the $k$-th coefficient of some polynomial $a \in \mathcal{R}$ matches the $(d-k)$-th coefficient of $\tilde{a} \coloneqq \sigma_{-1}(a)$, up to a sign, as in

$$\mathtt{ct}(X^{d-k}a - X^k\tilde{a}) = 0. \tag{24}$$

## B Deferred proofs

### B.1 On the Outer Commitment Layer

Here we show that applying the outer commitment layer preserves the knowledge soundness and zero-knowledge. For achieving zero-knowledge, one needs to use the hiding version of the outer commitment.

**Lemma 5.** *Let $\Pi$ be a knowledge sound proof system and an extractor $\mathcal{E}$ that extracts with knowledge error $\kappa$ a valid witness for $\Pi$. Let $\Pi'$ be a derived protocol from $\Pi$ by applying the outer commitment layer with hiding, i.e. any prover's message is postponed to the last message and sent instead in a committed form. Then there exists an extractor $\mathcal{E}'$ that either produces a valid witness for $\Pi'$ or breaks the binding of the outer commitment layer.*

*Proof.* This extractor $\mathcal{E}'$ will use as subroutine an extractor $\mathcal{E}$ for the protocol $\Pi$. Indeed, let $\mathcal{E}'$ run the prover of $\Pi'$ and look at his last message. By construction, the last message contains the openings of the commitments the prover sent earlier in $\Pi'$. The binding property of the outer commitment layer ensures that these openings are exactly the prover's messages in $\Pi$. Then $\mathcal{E}$ can reconstruct the transcript of $\Pi$. He uses $\mathcal{E}$ to extract a valid witness for $\Pi$, which, by construction, turns out to be a valid witness for $\Pi'$.

**Lemma 6.** *Let $\Pi$ be a zero-knowledge proof system and $\Pi'$ be a derived protocol from $\Pi$ by applying the outer commitment layer with hiding, i.e. any prover's message is postponed to the last message and sent instead in a committed form. Then $\Pi'$ also satisfies the zero-knowledge property.*

*Proof.* We extend in a straightforward way the simulator of $\Pi$ to the additional commitment layer involved in $\Pi'$. Namely, we construct a simulator $\mathcal{S}'$ for the protocol $\Pi'$ by using as subroutine a simulator $\mathcal{S}$ for the protocol $\Pi$. As $\mathcal{S}$ simulates a valid transcript for $\Pi$, indistinguishable from the real distribution, thanks to zero-knowledge, $\mathcal{S}'$ simply takes all the messages from this transcript and creates commitments of them, in order to create a valid transcript for the protocol $\Pi'$. By the hiding of the commitment layer, these commitments look indistinguishable from the ones of the messages from the real transcript of $\Pi$.

### B.2 Security of LaBRADOS

**Soundness.** The proof of knowledge soundness follows, via Lemma 5, the same lines as the proof of [13, Thm 5.1] and assumes the binding of the outer commitment layer. The only difference is that, as LaBRADOS proves individual projections of the witness vectors as binary vectors using $\log(9.75\beta_i)$ bits (Theorem 2), it leads to proving approximate norm bounds with a slack factor $c_{\texttt{LaBRADOS}} \coloneqq 9.75/0.49$, by Theorem 3. Recall the output of LaBRADOS from Section 3.1 made of $\mathbf{x}_{\texttt{mid}}, \mathbf{x}_{\texttt{bin}}$ and the decomposition parts of $\mathbf{z}$, $\mathbf{z}_0$ and $\mathbf{z}_1$. The norm bounds $\gamma$ and $\gamma'$ are set such that $\|\mathbf{x}_{\texttt{mid}}\| \leq \gamma$ and $\|\mathbf{z}\| \leq \gamma'$, where here $\mathbf{z}$ is recomposed. These bounds are derived based on the decomposition bases being used, as well as on the number of coefficients and their size. We refer to Figures 5 to 8 and [13, Sec. 5.4], [16, Sec. E] for more details. The statement of knowledge soundness is as follows.

**Theorem 4.** *Let $\mathcal{C} \subset \mathcal{R}_q$ be the challenge space from Section 2.1, of polynomials with operator norm bounded by $T$. Suppose M-SIS is hard for $\ell_2$ norm bound $2\gamma$ and rank $k_{mid}$, and for $\ell_2$ norm bound $\max\{8T\gamma', 2\gamma' + 4T \cdot \frac{9.75}{0.49} \cdot \max\beta_i\}$, rank $k_{in}$, and $\frac{9.75}{0.49}\beta_i < \frac{q}{61}$. Then LaBRADOS is a knowledge-sound proof with knowledge soundness $\kappa \coloneqq 1/2^{128} + (1/q)^{\lambda} + 1/q^{d/2} + 2/|\mathcal{C}|$ and norm slack $c_{\texttt{LaBRADOS}} \coloneqq \frac{9.75}{0.49} \approx 19.89$, i.e. the extractor is only guaranteed to extract witness vectors $\bar{\mathbf{s}}_i$ with norm bounds $\|\bar{\mathbf{s}}_i\| \leq 9.75/0.49\beta_i$.*

### B.3 Security of LNP-Lite

**Completeness error.** In the first projection step, the threshold $N$ is set (Section D) such that each of the four masked projections has less than $N$ carries with probability $p_1$, where $1 - p_1 \leq \frac{1}{2^{130}}$. Note that the projections are not independent since they are derived from the same projection matrix. Then, by union bound, we can upper bound the probability that one of them is rejected by $4(1-p_1)$, which leads to the probability that they all are accepted to be at least $1 - 4(1-p_1) \geq 1 - \frac{1}{2^{128}}$. In the second projection step, by rejection sampling, $\mathbf{z}$ follows a distribution close to $D_\sigma$ and accepted with probability $p_2 \coloneqq 1/\exp(14/\gamma + 1/(2\gamma^2))$ (Lemma 3 (1)), where $\sigma = \gamma\sqrt{337}\sqrt{4N}$. By the Gaussian tail bound (Lemma 2), the verifier checks $\|\mathbf{z}\| \leq \sigma\sqrt{2 \cdot 256}$ with probability at least $1 - 2^{-128}$. In the last step, the prover obtains by rejection sampling $\mathbf{z}_i$ following a close distribution to $D_{\sigma_i}$ and passes both the rejection sampling algorithms with probability $1/\exp(14/\gamma_1 + 1/(2\gamma_1^2)) \cdot 1/(2\exp(1/(2\gamma_2^2)))$, where $\sigma_1 = \gamma_1 \cdot T \cdot \|(\tilde{\mathbf{s}}, \tilde{\mathbf{v}})\|$ and $\sigma_2 = \gamma_2 \cdot T \cdot \|(\mathbf{r}_s, \mathbf{r}_v)\|$. Again, by tail bound (Lemma 2), for $n_i, n_s + n_v \geq 128 \cdot 8/d = 4$, the verifier checks Eqs. (15) to (17) with probability at least $1 - 2^{-128}$ each. Then the honest prover convinces the verifier with probability

$$\simeq (1 - 4(1-p_1))p_2 \cdot 1/(2\exp(14/\gamma_1 + 1/(2\gamma_1^2) + 1/(2\gamma_2^2))).$$

**Zero-knowledge.** We follow the lines of [34, Thm 4.2]. We recall the prover messages in LNP-Lite (Section 3.3, Figure 2): $\mathbf{t}_s$, $\mathbf{t}_v$, $\{v_i^{(0)}\}_i$, $\mathbf{z}$, $\{h_i\}_i$, $\mathbf{w}_s$, $\mathbf{w}_v$, $u$, $v$, $\mathbf{z}_1$, $\mathbf{z}_2$. We show that we can simulate a non-aborting transcript between honest prover and verifier through a series of simulators. We are interested in simulating only non-aborting transcripts because, when converting into a non-interactive protocol using the Fiat-Shamir heuristics [26], parts of the protocol can be repeated until non-abort occurs, and thus the verifier sees only non-aborting transcripts.

- The first simulator $\mathcal{S}_0$ knows the witness vector $\tilde{\mathbf{s}} = (\mathbf{s}_i)_{i\in[6]}$. Given verifier challenges, it honestly generates the commitment $(\mathbf{t}_s, u)$ of $(\tilde{\mathbf{s}}, u')$ under fresh randomness $\mathbf{r}_s$ and the commitment $\mathbf{t}_v$ of $\tilde{\mathbf{v}}$, under fresh randomness $\mathbf{r}_v$, both randomness being sampled from the uniform binary distribution. (Recall

that $\tilde{\mathbf{v}}$ is made of the binary vectors reconstructing the high-order values of the masked projections of $\mathbf{s}_i$, $i \in [4]$ and $u'$ is one of the garbage terms needed for the final quadratic check, see Eq. (7), computable from the implied mask $\mathbf{y}_1 := \mathbf{z}_1 - c\,(\tilde{\mathbf{s}}, \tilde{\mathbf{v}})$, and the witness, with no need of commitment randomness). $\mathcal{S}_0$ also computes the lower bits $v_i^{(0)}$ of the masked projections. $\mathcal{S}_0$ further samples $\mathbf{z} \leftarrow D_\sigma$, $h_i \leftarrow \{x \in \mathcal{R}_q |\ \mathtt{ct}(x) = 0\}$, for $i \in [\lambda]$, and fresh masked openings $\mathbf{z}_1 := (\mathbf{z}_1^s, \mathbf{z}_1^v) \leftarrow D_{\sigma_1}$ and $\mathbf{z}_2 := (\mathbf{z}_2^s, \mathbf{z}_2^v) \leftarrow D_{\sigma_2}$, conditioned on $\langle \mathbf{z}_2, c(\mathbf{r}_s, \mathbf{r}_v)\rangle \geq 0$. Finally, it sets $\mathbf{w}_s := \mathbf{A}_1\mathbf{z}_1^s + \mathbf{A}_1'\mathbf{z}_2^s - c\mathbf{t}_s \in \mathcal{R}_q^{k_s}$, $\mathbf{w}_v := \mathbf{A}_2\mathbf{z}_1^v + \mathbf{A}_2'\mathbf{z}_2^v - c\mathbf{t}_v \in \mathcal{R}_q^{k_v}$ and, by writing $\mathbf{z}_1 = (\mathbf{z}_{1,1}^s, \ldots, \mathbf{z}_{1,6}^s, \mathbf{z}_{1,1}^v, \mathbf{z}_{1,2}^v)$, $v := \sum_{i,j\in[5]} a_{ij}\langle \mathbf{z}_{1,i}^s, \mathbf{z}_{1,j}^s\rangle + a\langle \mathbf{z}_{1,1}^v, \mathbf{z}_{1,2}^v\rangle + c \cdot \langle \varphi, \mathbf{z}_1\rangle + c^2 \cdot f_0 - cu + \langle (\varphi_{rand}, 0), \mathbf{z}_2\rangle$. Then, by Lemma 3, the transcript output of $\mathcal{S}_0$ is statistically close to the one of the non-aborting protocol.

– The second simulator $\mathcal{S}_1$ still knows $\tilde{\mathbf{s}}$ (and thus $\tilde{\mathbf{v}}$) and runs identically to $\mathcal{S}_0$, but it computes the commitments $(\mathbf{t}_s, u, \mathbf{t}_v)$ in the following way: it samples $(\mathbf{u}_1, \mathbf{u}_2) \leftarrow \mathcal{R}_q^{k_s+1} \times \mathcal{R}_q^{k_v}$ and sets

$$\begin{pmatrix} \mathbf{t}_s \\ u \end{pmatrix} := \mathbf{u}_1 + \begin{pmatrix} \mathbf{A}_1\tilde{\mathbf{s}} \\ u' \end{pmatrix} \in \mathcal{R}_q^{k_s+1} \tag{25}$$

$$\mathbf{t}_v := \mathbf{u}_2 + \mathbf{A}_2\tilde{\mathbf{v}} \in \mathcal{R}_q^{k_v}. \tag{26}$$

Note that in the real transcript and in the output of $\mathcal{S}_0$

$$\begin{pmatrix} \mathbf{t}_s \\ u \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1' \\ \varphi_{rand}^T \end{pmatrix} \mathbf{r}_s + \begin{pmatrix} \mathbf{A}_1\tilde{\mathbf{s}} \\ u' \end{pmatrix} \in \mathcal{R}_q^{k_s+1}$$
$$\mathbf{t}_v = \mathbf{A}_2'\mathbf{r}_v + \mathbf{A}_2\tilde{\mathbf{v}} \in \mathcal{R}_q^{k_v},$$

and in both cases, the sign of $\langle \mathbf{z}_2, c(\mathbf{r}_s, \mathbf{r}_v)\rangle$ is known.
This is the two-secret case of Definition 7, with $\mathbf{r}_s$ and $\mathbf{r}_v$ as the two secrets, the single challenge $c$ of the transcript shared by both, and the sign of one joint inner product as the leak. Lemma 7 below bounds the loss of this step.

– The third simulator $\mathcal{S}_2$ still knows about $\tilde{\mathbf{s}}$ and runs identically to $\mathcal{S}_1$, except that it sets $(\mathbf{t}_s, u, \mathbf{t}_v) := \mathbf{u} \leftarrow \mathcal{R}_q^{k_s+k_v+1}$. Then the distribution of the output of $\mathcal{S}_2$ is exactly the same as of $\mathcal{S}_1$. Note that here the only place where the knowledge of $\tilde{\mathbf{s}}$ or $\tilde{\mathbf{v}}$ is needed is in computing the low-order bits $v_i^{(0)}$. Here the commitments being uniform, no longer depend on $\tilde{\mathbf{s}}$ or $\tilde{\mathbf{v}}$.

– The fourth simulator $\mathcal{S}_3$ does not know any information about $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$ and runs identically as $\mathcal{S}_2$, except that now replaces $v_i^{(0)}$ as $v_i^{(0)} \leftarrow [-2^{k_i-1}, 2^{k_i-1} - 1]^{256}$, for $i \in [4]$, which by Lemma 4, are all within statistical distance at most $4(1 - p_1) \leq \frac{1}{2^{128}}$ (see Section D) from all the $v_i^{(0)}$ from the distribution of the output of $\mathcal{S}_2$ (and from any valid accepting transcript), by triangle inequality, as these are sampled independently. Here we instantiate Lemma 4 for $X = [-2^{k_i-1}, 2^{k_i-1} - 1]^{256}$ and $A$ the subset of $X$ of values $v_i^{(0)}$ can take such that the masked projection has less than $N$ carries. Thus, the distribution of the output of $\mathcal{S}_3$ is statistically close to the one of $\mathcal{S}_2$. We get the conclusion by using hybrid argument.

**Lemma 7.** *Let $\mathcal{A}$ be a PPT adversary that distinguishes the outputs of $\mathcal{S}_0$ and $\mathcal{S}_1$ with probability $\epsilon$. Then there is a PPT adversary $\mathcal{B}$ against the 2-secret Extended M-LWE problem of Definition 7 with $(m_1, m_2) = (k_s + 1, k_v)$, $(n_1, n_2) = (n_s, n_v)$, $\chi_1 = \chi_2 = \chi$ the uniform distribution over binary polynomials, $\xi = D_{\sigma_2}^{n_s+n_v}$, and $\zeta$ the distribution of $(c, c)$ for $c \leftarrow \mathcal{C}$, with advantage at least $\epsilon/2$.*

*Proof.* $\mathcal{B}$ receives $(\mathbf{B}_1, \mathbf{B}_2, (c, c), \mathbf{z}, h, \mathbf{u}_1, \mathbf{u}_2)$ from the challenger of Definition 7. By the choice of $\zeta$ both challenge slots carry the same $c \leftarrow \mathcal{C}$, and the hint is $h = \mathtt{sign}(\langle \mathbf{z}, c(\mathbf{r}_s, \mathbf{r}_v)\rangle)$ for the secrets $\mathbf{r}_s \leftarrow \chi^{n_s}$ and $\mathbf{r}_v \leftarrow \chi^{n_v}$. $\mathcal{B}$ identifies

$$\mathbf{B}_1 = \begin{pmatrix} \mathbf{A}_1' \\ \varphi_{rand}^T \end{pmatrix} \in \mathcal{R}_q^{(k_s+1)\times n_s}, \qquad \mathbf{B}_2 = \mathbf{A}_2' \in \mathcal{R}_q^{k_v \times n_v}$$

with the public matrices, which are uniformly random, so $\mathcal{B}$ may take them from its instance.

If $h = 0$, $\mathcal{B}$ outputs a uniform bit. If $h = 1$, it sets $\mathbf{z}_2 := \mathbf{z}$, computes $(\mathbf{t}_s, u, \mathbf{t}_v)$ from $\mathbf{u}_1, \mathbf{u}_2$ exactly as $\mathcal{S}_1$ does in Eqs. (25) and (26), and produces every remaining component of the transcript, such as $\mathcal{S}_0$. It hands the transcript to $\mathcal{A}$ and returns $\mathcal{A}$'s output.

The hint $h$ is a function of $(\mathbf{z}, c, \mathbf{r}_s, \mathbf{r}_v)$ alone, and these are distributed identically in the two worlds of Definition 7. Hence $p := \Pr[h = 1]$ is the same in both worlds, and $p \geq 1/2$ by the symmetry of $D_{\sigma_2}$ together with $\texttt{sign}(0) = 1$. Conditioned on $h = 1$, the joint distribution of $(c, \mathbf{z}, \mathbf{r}_s, \mathbf{r}_v)$ is the nominal one restricted to the event $\langle \mathbf{z}, c(\mathbf{r}_s, \mathbf{r}_v)\rangle \geq 0$, which is the restriction the simulators implement.[6] Since that event involves only $(\mathbf{z}, c, \mathbf{r}_s, \mathbf{r}_v)$, the restriction commutes with sampling the remaining, independently generated components. Therefore, conditioned on $h = 1$, the assembled transcript is distributed exactly as the output of $\mathcal{S}_0$ if $(\mathbf{u}_1, \mathbf{u}_2) = (\mathbf{B}_1\mathbf{r}_s, \mathbf{B}_2\mathbf{r}_v)$, and exactly as the output of $\mathcal{S}_1$ if $(\mathbf{u}_1, \mathbf{u}_2)$ is uniform. On $h = 0$ the output of $\mathcal{B}$ is a coin independent of the world, so these contributions cancel, and

$$\texttt{Adv}(\mathcal{B}) = p \cdot \left| \Pr[\mathcal{A}(\mathcal{S}_0) = 1] - \Pr[\mathcal{A}(\mathcal{S}_1) = 1] \right| \;\geq\; \epsilon/2.$$

**Soundness.** Here we present the proof of Theorem 1.

*Proof.* As the outer commitment preserves knowledge soundness (Lemma 5), it is enough to focus on proving knowledge soundness of our zero-knowledge layer without outer commitments. Recall the output vectors of LaBRADOS, $(\mathbf{s}_1, \ldots, \mathbf{s}_4)$, which satisfy the checks of LaBRADOS and the norm bounds $\beta_i$, $i \in [4]$. Let us collect the full-constraints of the LaBRADOS checks in $\mathcal{F}$ and the constant-term ones in $\mathcal{F}'$. Note that bin-check for proving $\mathbf{s}_4$ being binary considers $\mathbf{s}_5 = \sigma_{-1}(\mathbf{s}_4)$ and the corresponding constant-term constraints, already collected in $\mathcal{F}'$. Keep in mind that an approximate proof for $\mathbf{s}_5$ is not needed. All of these output vectors, together with the additional vector $\mathbf{s}_6$ made by the masks, form the witness $\tilde{\mathbf{s}}$ of our zero-knowledge layer and are all committed by hiding the Ajtai commitment, with the randomness vector $\mathbf{r}_s$. For simplicity, we extend the domains of the constraints from $\mathcal{F}$ and $\mathcal{F}'$ to be satisfied by $\tilde{\mathbf{s}}$, although $\mathbf{s}_6$ is not involved. Formally, our zero-knowledge protocol LNP-Lite gives a proof-of-knowledge for the following relation

$$R_{\texttt{LNP}} = \left\{ \begin{array}{c|c} ((\mathbf{t}_s, \mathcal{F}, \mathcal{F}', (\beta_i)_i), & \mathbf{t}_s = \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}'_1\mathbf{r}_s \\ (\tilde{\mathbf{s}}, \mathbf{r}_s) & f(\tilde{\mathbf{s}}) = 0,\ \forall f \in \mathcal{F} \\ & \texttt{ct}(f'(\tilde{\mathbf{s}})) = 0,\ \forall f' \in \mathcal{F}' \\ & \|\mathbf{s}_1\| \leq \bar{\gamma}\beta_1, \ldots, \|\mathbf{s}_4\| \leq \bar{\gamma}\beta_4 \end{array} \right\}.$$

for $\bar{\gamma} := \frac{4 \cdot 9.75}{0.49}$. Recall that in the first projection step, each of the vectors $\mathbf{s}_i$, for $i \in [4]$ receives a masked projection, using random projection matrices $\mathbf{\Pi}_i$, all coming from the same $\mathbf{\Pi}$ of the verifier, and the uniform masks $\mathbf{u}_i$, part of the vector $\mathbf{s}_6$. Decomposing these projections, we get the high-order values and low-order bits:

$$\mathbf{\Pi_i}\mathbf{s}_i + \mathbf{u}_i = 2^{k_i}(\mathbf{w}_i - \mathbf{t}_i) + \mathbf{v}_i^{(0)},\ \forall i \in [4], \tag{27}$$

where $k_i = \lceil \log(9.75\beta_i) \rceil$. All low-order bits, $\mathbf{v}_i^{(0)}$, could be sent in the clear, as they are uniform. All high-order values, represented by the binary vectors $\mathbf{w}_i$ and $\mathbf{t}_i$, are collected in the vector $\tilde{\mathbf{v}}_1 \in \mathcal{R}^8$. These masked projections are accepted if they contain less than $N$ nonzero high-order values, i.e. $\|(\mathbf{w}_i, \mathbf{t}_i)\| \leq \sqrt{N}$, each with probability $p_1$. Then, (with probability $1 - 4(1 - p_1)$), we can construct a new witness vector $\tilde{\mathbf{v}}_1$, for which we prove binary and $\|\tilde{\mathbf{v}}_1\| \leq \sqrt{4N}$. For this, we also consider $\tilde{\mathbf{v}}_2 = \sigma_{-1}(\tilde{\mathbf{v}}_1)$ and, thus a hiding Ajtai commitment $\mathbf{t}_v$ to $\tilde{\mathbf{v}} = (\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2)$ is provided, under randomness $\mathbf{r}_v$. The projections from Equation (27) get translated into new constant-term constraints to be proven to be satisfied by $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{v}}$, and we collect them all into $\mathcal{F}''$. Note that starting from the second step of projection, the zero-knowledge protocol LNP-Lite follows exactly [34, Sec. 5], so we can treat the following steps together as a subprotocol. Therefore, formally

[6] The sign condition on $\mathbf{z}_2$ restricts the joint distribution of all sampled values, not that of $\mathbf{z}_2$ alone: a rejected attempt restarts the whole simulation. This matches the distribution of accepted transcripts produced by the rejection step, cf. the algorithm $\mathcal{F}$ of [36], Fig. 3.

written, the first projection step could be viewed as reducing $R_{\texttt{LNP}}$ to the following relation

$$R'_{\texttt{LNP}} = \left\{ \begin{array}{c|c} & \mathbf{t}_s = \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}'_1\mathbf{r}_s \\ & \mathbf{t}_v = \mathbf{A}_2\tilde{\mathbf{v}} + \mathbf{A}'_2\mathbf{r}_v \\ ((\mathbf{t}_s, \mathbf{t}_v, (\mathbf{v}_i^{(0)})_i, & f(\tilde{\mathbf{s}}) = 0,\ \forall f \in \mathcal{F} \\ \mathcal{F}, \mathcal{F}', (\beta_i)_i, & \texttt{ct}(f'(\tilde{\mathbf{s}})) = 0,\ \forall f' \in \mathcal{F}' \\ N, \mathbf{\Pi}), & \texttt{ct}(f''(\tilde{\mathbf{s}}, \tilde{\mathbf{v}})) = 0,\ \forall f'' \in \mathcal{F}'' \\ (\tilde{\mathbf{s}}, \tilde{\mathbf{v}}, \mathbf{r}_s, \mathbf{r}_v)) & \texttt{bin-check}(\tilde{\mathbf{v}}) \\ & \|\tilde{\mathbf{v}}_1\| \leq \gamma'\sqrt{4N} \end{array} \right\}.$$

for $\gamma' \coloneqq 2\sqrt{2 \cdot 256/26}\gamma\sqrt{337}$, for a repetition rate $\gamma$. (see [34, Prop.5.1.])

Our extractor $\mathcal{E}$ for the zero-knowledge protocol `LNP-Lite` uses as a subroutine the extractor $\mathcal{E}'$ of the subprotocol. Therefore, we can rely on the knowledge soundness of the latter to extract a witness $\bar{\mathbf{s}}, \bar{\mathbf{v}}, \bar{\mathbf{r}}_s, \bar{\mathbf{r}}_v$ that satisfies

$$\begin{aligned} &\mathbf{t}_s = \mathbf{A}_1\bar{\mathbf{s}} + \mathbf{A}'_1\bar{\mathbf{r}}_s \\ &\mathbf{t}_v = \mathbf{A}_2\bar{\mathbf{v}} + \mathbf{A}'_2\bar{\mathbf{r}}_v \\ &f(\bar{\mathbf{s}}) = 0,\ \forall f \in \mathcal{F} \\ &\texttt{ct}(f'(\bar{\mathbf{s}})) = 0,\ \forall f' \in \mathcal{F}' \\ &\texttt{ct}(f''(\bar{\mathbf{s}}, \bar{\mathbf{v}})) = 0,\ \forall f'' \in \mathcal{F}'' \qquad (28) \\ &\texttt{bin-check}(\tilde{\mathbf{v}}_1) \qquad (29) \end{aligned}$$

except with probability $\kappa' \coloneqq 1/2^{128} + (1/q)^{d/2} + (1/q)^{\lambda} + 2/|\mathcal{C}|$ (see [34, Prop.5.1]). Moreover, if the value of any of $\bar{\mathbf{s}}, \bar{\mathbf{v}}, \bar{\mathbf{r}}_s, \bar{\mathbf{r}}_v$ was dependent on the challenge $\mathbf{\Pi}$, then we would be able to use the knowledge extractor $\mathcal{E}'$ to break the binding of the commitments $\mathbf{t}_s$ and $\mathbf{t}_v$, i.e. the M-SIS problems for $\ell_2$ norm bound $8T\sqrt{B_s^2 + B_r^2}$ and rank $k_s$ and for $\ell_2$ norm bound $8T\sqrt{B_v^2 + B_r^2}$ and rank $k_v$. Therefore, $\bar{\mathbf{s}}, \bar{\mathbf{v}}, \bar{\mathbf{r}}_s, \bar{\mathbf{r}}_v$ are unique and, in particular, do not depend on $\mathbf{\Pi}$. Thus, it suffices to compute the probability, over the randomness of $\mathbf{\Pi}$ that the extracted vectors do not satisfy the relation $R_{\texttt{LNP}}$. As the extracted vectors already satisfy the `LaBRADOS` checks from $\mathcal{F}$ and $\mathcal{F}'$, it suffices to compute the probability that one of the vectors $\bar{\mathbf{s}}_i$, for some $i \in [4]$, does not satisfy the norm bound, i.e. $\|\bar{\mathbf{s}}_i\| > \bar{\gamma}\beta_i$.

First, we argue that the extractor $\mathcal{E}'$ of the subprotocol extracts the vector $\bar{\mathbf{v}}_1 \in \mathcal{R}^8$ that is indeed binary. Note that this vector satisfies $\|\bar{\mathbf{v}}_1\| \leq 2\sqrt{2 \cdot 256/26}\gamma\sqrt{337}\sqrt{4N} = 2\sigma\sqrt{512/26}$. ( [34, Prop 5.1.]). For this approximate norm bound, it was used $q \geq 41 \cdot 8d \cdot 2\sigma\sqrt{512/26}$, where the ring degree $d$ is 256. As this vector $\bar{\mathbf{v}}_1$ satisfies `bin-check`, i.e. $\bar{\mathbf{v}}_2 = \sigma_{-1}(\bar{\mathbf{v}}_1)$ and $\langle \bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2 \rangle - \langle \bar{\mathbf{v}}_1, \sigma_{-1}(\mathbf{1}) \rangle = 0 \mod q$, we can upper-bound (the absolute value of) the left hand side of the latter equation, $2\|\bar{\mathbf{v}}_1\|^2 \leq 2 \cdot (2\sigma\sqrt{512/26})^2 \leq q$, so we are guaranteed that this equation actually holds over $\mathbb{Z}$. This proves that $\bar{\mathbf{v}}_1$, and, in particular, the vectors $\bar{\mathbf{w}}_i, \bar{\mathbf{t}}_i$ $\bar{\mathbf{v}}_1$ is made of, are indeed binary.

Second, we argue that the extracted vectors $\bar{\mathbf{s}}_i$ satisfy $\|\mathbf{\Pi}_i\bar{\mathbf{s}}_i\|_\infty \leq 2^{k_i+1}$. As the vectors extracted from $\mathcal{E}'$ satisfy Equation (28), they also satisfy Equation (27) written as follows

$$\mathbf{\Pi}_i\bar{\mathbf{s}}_i = 2^{k_i}(\bar{\mathbf{w}}_i - \bar{\mathbf{t}}_i) + \mathbf{v}_i^{(0)} - \bar{\mathbf{u}}_i \bmod q.$$

Recall that the vectors $\mathbf{v}_i^{(0)} \in [-2^{k_i-1}, 2^{k_i-1}-1]^{256}$ are public and the extractor $\mathcal{E}$ finds $\bar{\mathbf{u}}_i \in [-2^{k_i-1}, 2^{k_i-1}-1]^{256}$ from the length of its bit-decomposition. Moreover, since $\bar{\mathbf{w}}_i, \bar{\mathbf{t}}_i$ are binary, $\|\bar{\mathbf{w}}_i - \bar{\mathbf{t}}_i\|_\infty \leq 1$. Therefore, by the triangle inequality, the infinity norm of the right hand side could be at most $2^{k_i} + 2^{k_i-1} + 2^{k_i-1} = 2^{k_i+1}$. Moreover, $q$ is set so big that no wraparound mod $q$ is happening on this side.

Therefore, our extracted vectors $\bar{\mathbf{s}}_i$ satisfy $\|\mathbf{\Pi}_i\bar{\mathbf{s}}_i \bmod q\|_\infty \leq 2^{k_i+1}$, for any $i \in [4]$. By Theorem 3, if it happens that some of the extracted vectors $\bar{\mathbf{s}}_i$ are too long, i.e. they do not satisfy the norm bound $\|\bar{\mathbf{s}}_i\| \leq \frac{2^{k_i+1}}{0.49} \leq \frac{4 \cdot 9.75}{0.49}\beta_i$, then $\|\mathbf{\Pi}_i\bar{\mathbf{s}}_i \bmod q\|_\infty \leq 2^{k_i+1}$ holds only with probability $1/2^{128}$, because of Theorem 3 (here we use $q > 61 \cdot 2^{k_i+1}/0.49$ for any $i \in [4]$ and $k_i = \lceil \log(9.75\beta_i) \rceil$). By union bound, the probability that at least one of the four norm checks fails is at most $4/2^{128}$.

Therefore, we obtain the knowledge error of the zero-knowledge protocol LNP-Lite as $\kappa \coloneqq \kappa' + 4/2^{128}$. Thus, the extracted vectors $\bar{\mathbf{s}}_i$ satisfy $\|\bar{\mathbf{s}}_i\| \leq \frac{4 \cdot 9.75}{0.49}\beta_i$, and the LaBRADOS checks, collected in $\mathcal{F}, \mathcal{F}'$, unless with small probability. What is left to check is $\bar{\mathbf{s}}_4$ being binary: note that it already satisfies the bin-check function, so then, as long as $2\|\bar{\mathbf{s}}_4\|^2 \leq 2 \cdot d \cdot n_4 \cdot \left(\frac{4 \cdot 9.75}{0.49}\right)^2 \leq q$, the binary equation holds over $\mathbb{Z}$, which leads to the conclusion.

### B.4 The Fiat-Shamir Transformation

*Public-coin interactive* proofs (those where all the verifier's messages are randomly sampled values) can be turned *non-interactive* using the Fiat-Shamir transformation [26]. The security of the composition of LNP (with outer commitments), as the starting protocol, with LaBRADOR was already studied in [41]. In our case, we have iterations of LaBRADOS (which is LaBRADOR with outer commitments), before going to LNP-Lite, a version of LNP, which is further fed into iterations of LaBRADOS and LaBRADOR. Thus, the security of our composition should follow almost the same lines as the security analyzed in [41], with the differences we detail below.

Completeness follows directly from the construction of the Fiat-Shamir transformation, whereas non-interactive zero-knowledge is derived by the fact that the composition itself is honest-verifier zero-knowledge [33]. Thus, we focus only on the aspect of knowledge soundness, looking at each kind of protocol, LaBRADOS, LNP-Lite and LaBRADOR.

[1] proved the knowledge soundness of LaBRADOR in the random oracle model (ROM) [9], using the *predicate special soundness* framework. This framework could be viewed as a natural extension of the special soundness that comes with a collection of *predicates*, i.e. properties satisfied (with an overwhelming probability) by the extracted openings and the challenges at each prover-verifier interaction. As the property of *predicate special soundness* is preserved when applying outer commitments on a proof system $\Pi$ (as noted also by [41, Sec.B.6.1], due to the binding of the outer commitments, they never open to distinct values, a similar property to Lemma 5), then, due to [1, Thm 5.2.], we can heuristically rely on the soundness of LaBRADOR and LaBRADOS compiled via Fiat-Shamir. [1] also showed that predicate special soundness protocols are composed nicely and thus their composition can be soundly compiled via Fiat-Shamir (see [1, Thm 5.1]). Therefore, it suffices to analyze the soundness of our zero-knowledge layer in the discussed framework. The knowledge soundness of LNP in a non-interactive version was analyzed in [34, 41], where the latter explicitly used the *predicate* special-soundness machinery. Using our zero-knowledge protocol LNP-Lite (without outer commitments), we could derive its soundness proof as [41] cast in the *predicate* special-soundness framework. We recall that this protocol has, compared to LNP, an additional prover-verifier interaction, corresponding to the first projection step. In this step, note that the upper bound on the fraction of projection matrices (i.e. *failure density*), used in this step is at most $1/2^{128}$ (Theorem 3), since the projections of the invalid (i.e. of too big norm) extracted witness vectors to have a certain number of bits is at most $1/2^{128}$. Thus, in the analysis of [41, Lem.B.10], we inquire about an extra *failure density* value of $1/2^{128}$. As in LaBRADOS, the predicate special soundness of LNP-Lite easily extends when considering the outer commitments. Therefore, the whole composition of LNP-Lite with LaBRADOS and LaBRADOR has predicate special soundness, and thus shall be considered (heuristically) secure after applying the Fiat–Shamir transformation.

Following the observations from [41, Thm 5.1], the knowledge error $\kappa$ of our construction in the ROM shall be adjusted for a multiplicative loss arising from the number of queries the adversary is allowed to make to the random oracle.

## C Multi-Secret Extended LWE

The zero-knowledge simulation of LNP-Lite (Section B.3) rests on the hiding of the two inner commitments $\mathbf{t}_s$ and $\mathbf{t}_v$, which are randomized by two independent binary vectors $\mathbf{r}_s \in \mathcal{R}^{n_s}$ and $\mathbf{r}_v \in \mathcal{R}^{n_v}$. The rejection step for the combined masked opening $\mathbf{z}_2 = \mathbf{y}_2 + c \cdot (\mathbf{r}_s, \mathbf{r}_v)$ (Lemma 3 (2)) leaks one additional bit of information about $(\mathbf{r}_s, \mathbf{r}_v)$: an accepted transcript satisfies $\langle \mathbf{z}_2, c(\mathbf{r}_s, \mathbf{r}_v)\rangle \geq 0$, i.e. it reveals

$$\mathtt{sign}\left(\langle \mathbf{z}_2^s, c\,\mathbf{r}_s\rangle + \langle \mathbf{z}_2^v, c\,\mathbf{r}_v\rangle\right).$$

The sign bit depends on both randomness vectors jointly, so the hiding step is not captured by two instances of the (single-secret) Extended M-LWE problem of [36]: two such instances would leak the signs of the two summands separately and they would carry two independent challenges, while the transcript contains a single $c$.

**Definition 7 (Multi-Secret Extended M-LWE).** *For integers $\eta > 0$ (number of secrets), $q > 0$ (modulus) and $n_j, m_j > 0$ (M-LWE dimensions), $n := \sum_j n_j$ (for $j \in [\eta]$), let $\chi_1, \ldots, \chi_\eta$ be distributions over $\mathcal{R}$ (secrets' distributions), let $\xi$ be an efficiently sampleable distribution over $\mathcal{R}^n$ (masked opening distribution), and let $\zeta$ be an efficiently sampleable distribution over $\mathcal{R}^\eta$ (challenge distribution). We define the $\eta$-secret Extended M-LWE$_{q,(m_j)_j,(n_j)_j,(\chi_j)_j,\zeta}$ problem as follows: for $\mathbf{r}_1 \leftarrow \chi_1^{n_1}, \ldots, \mathbf{r}_\eta \leftarrow \chi_\eta^{n_\eta}$, distinguish between the following two worlds*

- $\big((\mathbf{B}_j)_{j\in[\eta]},\ (c_j)_{j\in[\eta]},\ \mathbf{z},\ \mathtt{sign}\big(\sum_{j\in[\eta]}\langle \mathbf{z}_j, c_j\,\mathbf{r}_j\rangle\big),\ (\mathbf{u}_j)_{j\in[\eta]}\big)$, *for* $\mathbf{B}_j \leftarrow \mathcal{R}_q^{m_j\times n_j}$, $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_\eta) \leftarrow \xi$, $(c_1, \ldots, c_\eta) \leftarrow \zeta$, $\mathbf{u}_j := \mathbf{B}_j\mathbf{r}_j$ $(j \in [\eta])$
- $\big((\mathbf{B}_j)_{j\in[\eta]},\ (c_j)_{j\in[\eta]},\ \mathbf{z},\ \mathtt{sign}\big(\sum_{j\in[\eta]}\langle \mathbf{z}_j, c_j\,\mathbf{r}_j\rangle\big),\ (\mathbf{u}_j)_{j\in[\eta]}\big)$, *for* $\mathbf{B}_j \leftarrow \mathcal{R}_q^{m_j\times n_j}$, $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_\eta) \leftarrow \xi$, $(c_1, \ldots, c_\eta) \leftarrow \zeta$, $\mathbf{u}_j \leftarrow \mathcal{R}_q^{m_j}$ $(j \in [\eta])$

*where each product $c_j\mathbf{r}_j$ is computed in $\mathcal{R}$ (without reduction mod $q$).*

In both cases the sign is computed from the real secrets $\mathbf{r}_1, \ldots, \mathbf{r}_\eta$, in the second case the secrets appear *only* inside the sign.

*Properties of $\xi$ and $\zeta$.* The hardness analysis uses additional properties of the space of masked openings and the challenge space (essentially, their elements need to be "small", the properties are met by the masked openings and challenge spaces we consider):

(C1) *Bounded inner products:* there are (polynomially bounded) positive integers $T_1, \ldots, T_\eta$ such that

$$\Pr[|\langle \mathbf{z}_j, c_j\mathbf{r}_j\rangle| \geq T_j]$$

is negligible, over $(c_1, \ldots, c_\eta) \leftarrow \zeta$, $\mathbf{r}_j \leftarrow \chi_j^{n_j}$ and $\mathbf{z} \leftarrow \xi$.

(C2) *Smallness relative to $q$:* $2T_j$ is smaller than the smallest prime factor of $q$ (so the difference of two distinct inner products must be invertible modulo $q$).

*Special cases.* For $\eta = 1$, Definition 7 coincides with the Extended M-LWE problem of [36, Def.3.3], with $k = 1$ challenges (Several challenges per secret are captured by letting $c_j$ itself be a tuple acting on a corresponding split of $\mathbf{z}_j$, respectively by the stacked matrices in the unstructured form, which recovers the multiple-challenge variant of Extended M-LWE).

For $\eta = 2$ and $\xi = D_\sigma^{n_1+n_2}$ it is exactly Definition 5. For $\zeta$ the distribution of $(c, \ldots, c)$ with $c \leftarrow \mathcal{C}$ (sample the diagonal of $\mathcal{C}^\eta$) all secrets share one challenge (which is the case the zero-knowledge proof uses).

### C.1 Hardness of Multi-Secret Extended LWE

In this section, we analyze the hardness of the multi-secret Extended LWE problem analogous to Definition 7 but without any polynomial structure.

The proof combines the guessing technique of [5, 30] with a hybrid argument. For $\eta = 1$, the problem specializes to that of [36, Lem. D.1].

For $j \in [\eta]$, let $\chi_j$ be an efficiently sampleable distribution over $\mathbb{Z}^{n_j}$, let $\xi$ be one over $\mathbb{Z}^{\ell_1+\ldots+\ell_\eta}$ (masked openings $\mathbf{z}_1, \ldots, \mathbf{z}_\eta$) and $\zeta$ one over $\mathbb{Z}^{\ell_1\times n_1} \times \ldots \times \mathbb{Z}^{\ell_\eta\times n_\eta}$ (challenge matrices $\mathbf{C}_1, \ldots, \mathbf{C}_\eta$). The challenger generates $\mathbf{B}_j \leftarrow \mathbb{Z}_q^{m_j\times n_j}$, $\mathbf{r}_j \leftarrow \chi_j$, $\mathbf{z} \leftarrow \xi$, $(\mathbf{C}_1, \ldots, \mathbf{C}_\eta) \leftarrow \zeta$ and $\mathbf{u}_j \leftarrow \mathbb{Z}_q^{m_j}$. It flips a bit $b \leftarrow \{0, 1\}$, and if $b = 1$ it sets $\mathbf{u}_j := \mathbf{B}_j\mathbf{r}_j$ for all $j$. Eventually, it outputs

$$\big((\mathbf{B}_j)_{j\in[\eta]},\ (\mathbf{C}_j)_{j\in[\eta]},\ \mathbf{z},\ \mathtt{sign}(X_1 + \cdots + X_\eta),\ (\mathbf{u}_j)_{j\in[\eta]}\big),$$

where $X_j := \langle \mathbf{z}_j, \mathbf{C}_j \mathbf{r}_j \rangle \in \mathbb{Z}$.

Under the coefficient embedding, Definition 7 takes this form with all dimensions multiplied by $d$: $\mathbf{C}_j$ is the multiplication matrix of the challenge $c_j$ acting on the coefficient vector of $\mathbf{r}_j$, the distributions $\xi$ and $\zeta$ are over integer vectors and matrices, and the uniform ring matrices are uniform integer matrices.

**Lemma 8.** *Let $T_1, \ldots, T_\eta$ be (polynomially bounded) positive integers such that, for every $j \in [\eta]$, the probability that $|X_j| < T_j$ (over $\mathbf{r}_j \leftarrow \chi_j$, $\mathbf{z} \leftarrow \xi$, $(\mathbf{C}_1, \ldots, \mathbf{C}_\eta) \leftarrow \zeta$) is overwhelming, and $2T_j$ is smaller than the smallest prime factor of $q$ (for e.g. $2T_j < q$ for a prime $q$). Suppose there is a PPT adversary $\mathcal{A}$ with advantage $\epsilon := \mathit{Adv}(\mathcal{A})$ against the $\eta$-secret Extended LWE problem above. Then there are PPT adversaries $\mathcal{B}_1, \ldots, \mathcal{B}_\eta$, where $\mathcal{B}_j$ solves Decisional-LWE (in knapsack form) with block-$j$ parameters (i.e. distinguishes $(\mathbf{B}_j, \mathbf{B}_j \mathbf{r}_j)$ from $(\mathbf{B}_j, \mathbf{u}_j)$ for $\mathbf{B}_j \leftarrow \mathbb{Z}_q^{m_j \times n_j}$, $\mathbf{r}_j \leftarrow \chi_j$, $\mathbf{u}_j \leftarrow \mathbb{Z}_q^{m_j}$) such that*

$$\epsilon \;\leq\; \sum_{j \in [\eta]} (2T_j - 1) \cdot \mathit{Adv}(\mathcal{B}_j) \;+\; \mathsf{negl}.$$

*In particular, if $\epsilon$ is non-negligible, then for at least one $j$ the advantage $\mathit{Adv}(\mathcal{B}_j) \geq (\epsilon - \mathsf{negl})/(\eta\,(2T_j - 1))$ is non-negligible.*

*Proof.* For $i \in \{0, 1, \ldots, \eta\}$, let $H_i$ denote the distribution of the challenger's output tuple with $\mathbf{u}_j \leftarrow \mathbb{Z}_q^{m_j}$ uniform for $j \leq i$ and $\mathbf{u}_j = \mathbf{B}_j \mathbf{r}_j$ for $j > i$, the hint $\mathtt{sign}(X_1 + \cdots + X_\eta)$ computed from the real secrets as before. For $j \leq i$ the secret $\mathbf{r}_j$ appears only inside the sign. $H_0$ and $H_\eta$ are the $b = 1$ and $b = 0$ cases, and with $p_i := \Pr[\mathcal{A}(H_i) = 1]$,

$$\epsilon = |p_0 - p_\eta| \;\leq\; \sum_{i \in [\eta]} \epsilon_i, \qquad \epsilon_i := |p_{i-1} - p_i|.$$

We construct (for each $i$) a reduction $\mathcal{B}_i$ with $\epsilon_i \leq (2T_i - 1)\mathtt{Adv}(\mathcal{B}_i) + \mathsf{negl}$. The lemma then follows from summing over $i$.

$\mathcal{B}_i$ receives an instance $(\mathbf{B}_i, \mathbf{u}_i^*)$, where either $\mathbf{u}_i^* = \mathbf{B}_i \mathbf{r}_i$ for $\mathbf{r}_i \leftarrow \chi_i$ (the real instance), or $\mathbf{u}_i^*$ is uniform (the uniform instance). It samples $\mathbf{z} \leftarrow \xi$, $(\mathbf{C}_j)_j \leftarrow \zeta$, secrets $\mathbf{r}_j \leftarrow \chi_j$ for all $j \neq i$, a guess $X' \leftarrow W$ from a *window* $W := \{-T_i + 1, \ldots, T_i - 1\}$, *uniformly*, for the value of $\langle \mathbf{z}_i, \mathbf{C}_i \mathbf{r}_i \rangle$, and $\mathbf{v} \leftarrow \mathbb{Z}_q^{m_i}$; for $j < i$ it samples $\mathbf{B}_j, \mathbf{u}_j$ uniform, and for $j > i$ it samples $\mathbf{B}_j$ uniform and sets $\mathbf{u}_j := \mathbf{B}_j \mathbf{r}_j$. (For $j < i$ the secret $\mathbf{r}_j$ is needed only inside the sign, and $\mathcal{B}_i$ can sample it itself. This keeps $K$ below computable at every step.) It computes (over $\mathbb{Z}$)

$$K := \sum_{j \neq i} \langle \mathbf{z}_j, \mathbf{C}_j \mathbf{r}_j \rangle,$$

sets

$$\mathbf{B}_i' := \mathbf{B}_i - \mathbf{v}\,(\mathbf{z}_i^T \mathbf{C}_i) \in \mathbb{Z}_q^{m_i \times n_i}, \qquad \mathbf{u}_i' := \mathbf{u}_i^* - X' \cdot \mathbf{v} \in \mathbb{Z}_q^{m_i},$$

runs $\mathcal{A}$ on the tuple with $(\mathbf{B}_i', \mathbf{u}_i')$ in slot $i$, all other slots as sampled above, and hint $\mathtt{sign}(X' + K)$, and returns $\mathcal{A}$'s output.

For the analysis, let $X_i := \langle \mathbf{z}_i, \mathbf{C}_i \mathbf{r}_i \rangle$ (in the uniform case, via an auxiliary $\mathbf{r}_i \leftarrow \chi_i$ introduced for the analysis only), and condition both experiments on the overwhelming event $|X_i| < T_i$. This costs an additive negligible term, and afterwards $X_i$ always lies in the guess window, which has $2T_i - 1$ elements. Since, for fixed $(\mathbf{v}, \mathbf{z}_i, \mathbf{C}_i)$, the map $\mathbf{B}_i \mapsto \mathbf{B}_i - \mathbf{v}(\mathbf{z}_i^T \mathbf{C}_i)$ is a bijection of $\mathbb{Z}_q^{m_i \times n_i}$, the matrix $\mathbf{B}_i'$ is uniform and independent of $\mathbf{v}$ and of all other sampled values, and $X'$ takes each of the $2T_i - 1$ window values with probability $1/(2T_i - 1)$, independently of all other randomness.

Write $\mathcal{T}$ for the tuple that $\mathcal{B}_i$ hands to $\mathcal{A}$. A run of a hybrid $H$ produces an output tuple and, in its slot $i$, a secret $\mathbf{r}_i$: write $\mathcal{T}_H$ for that output tuple and

$$Y_H \;:=\; \langle \mathbf{z}_i, \mathbf{C}_i \mathbf{r}_i \rangle.$$

We fix a window value $x \in W$, condition on $X' = x$, and distinguish three cases.

– $\mathbf{u}_i^*$ *uniform and* $X_i = x$. Then $\mathbf{u}'_i$ is uniform and independent of everything else, and on this event the manufactured hint equals $\mathtt{sign}(X_i + K)$. So the slot $i$ of the tuple $\mathcal{T}$ carries a uniform vector while $\mathbf{r}_i$ enters only through the sign, which is the shape of $H_i$, and $X_i$ is the inner product of that secret. Hence

$$\big((\mathcal{T}, X_i) \,\big|\, X_i = x\big) \;\equiv\; \big((\mathcal{T}_{H_i}, Y_{H_i}) \,\big|\, Y_{H_i} = x\big).$$

– $\mathbf{u}_i^* = \mathbf{B}_i\mathbf{r}_i$ *and* $X_i = x$. The manufactured hint is correct, $\mathtt{sign}(X' + K) = \mathtt{sign}(X_i + K)$, and

$$\mathbf{u}'_i = \mathbf{B}_i\mathbf{r}_i - X'\mathbf{v} = \mathbf{B}'_i\mathbf{r}_i + (X_i - X')\,\mathbf{v} = \mathbf{B}'_i\mathbf{r}_i.$$

So the slot $i$ of the tuple $\mathcal{T}$ carries the matrix $\mathbf{B}'_i$ together with the genuine secret $\mathbf{r}_i$, the slots below $i$ are uniform and the slots above $i$ are real, which is the shape of $H_{i-1}$, and $X_i$ is the inner product of that slot's secret. Hence

$$\big((\mathcal{T}, X_i) \,\big|\, X_i = x\big) \;\equiv\; \big((\mathcal{T}_{H_{i-1}}, Y_{H_{i-1}}) \,\big|\, Y_{H_{i-1}} = x\big).$$

– $X_i \neq x$ *(either instance).* Then $X_i - X'$ is a nonzero integer of absolute value less than $2T_i$, which is smaller than every prime factor of $q$, so $X_i - X'$ is invertible modulo $q$ and thus $(X_i - X')\,\mathbf{v}$ is uniform. In the uniform instance $\mathbf{u}'_i$ is uniform outright. So in both experiments $\mathcal{T}$ consists of uniform independent $(\mathbf{B}'_i, \mathbf{u}'_i)$, the hint $\mathtt{sign}(x + K)$ and the remaining slots, and $X_i$ is correlated with $\mathcal{T}$ only through $(\mathbf{z}_i, \mathbf{C}_i)$. The law of $\big((\mathcal{T}, X_i) \,\big|\, X_i \neq x\big)$ is therefore the same for the real and the uniform instance, and these contributions cancel in the advantage, for every $x$ separately.

None of these identifications involves $X'$ beyond the conditioning on $X' = x$. The events $\{X_i = x\}$, for $x$ in the window $W$, are disjoint and, after the conditioning, exhaustive, so summing the first case over the window with the slice weights $\Pr[X' = x] = 1/(2T_i - 1)$ gives

$$\frac{1}{2T_i - 1} \sum_{x \in W} \Pr\left[\mathcal{A}(\mathcal{T}_{H_i}) = 1 \;\wedge\; Y_{H_i} = x\right] \;=\; \frac{\Pr[\mathcal{A}(H_i) = 1]}{2T_i - 1}$$

for the uniform instance, and the second case gives the same with $H_{i-1}$ in place of $H_i$ for the real instance, both taken in the conditioned experiments. Hence, exactly,

$$\mathtt{Adv}^*(\mathcal{B}_i) \;=\; \frac{\left|p^*_{i-1} - p^*_i\right|}{2T_i - 1},$$

where the star denotes the conditioned experiments. Undoing the conditioning changes each probability by at most an additive negligible amount, so $\mathtt{Adv}(\mathcal{B}_i) \geq (\epsilon_i - \mathsf{negl})/(2T_i - 1) - \mathsf{negl}$; rearranging, and using that $T_i$ is polynomially bounded, gives $\epsilon_i \leq (2T_i - 1)\mathtt{Adv}(\mathcal{B}_i) + \mathsf{negl}$, as claimed.

# D Deferred Parameters

## D.1 Parameters for LaBRADOS

**The Modulus $q$.** From Theorem 4, we need $\frac{9.75}{0.49}\beta_i \leq \frac{q}{61}$. For proving `bin-check` over $\mathbb{Z}$, (Section A.3), as all witness vectors of LaBRADOS, including $\mathbf{x}_{\mathtt{bin}} \in \mathcal{R}^{n_{\mathtt{bin}}}$, benefit from approximate range proofs with slack $c_{\mathtt{LaBRADOS}} = \frac{9.75}{0.49}$, we set $2dn_{\mathtt{bin}} \cdot c^2_{\mathtt{LaBRADOS}} \leq q$.

The rest of the parameters involved in LaBRADOS and LaBRADOR are chosen following the same rationale of [13, 15].

### D.2 Parameters for LNP-Lite

We set the parameters for LNP-Lite described in Section 3.3. We use the challenge space $\mathcal{C}$ from Section 2.1 of ring elements of operator norm $T$.

**The Dimension and Bound of the Additional Witness $\mathbf{s}_6$.** Recall $s_6 \in \mathcal{R}^{n_6}$ is made of the binary decompositions of the masks $\mathbf{u}, \mathbf{y}, \mathbf{g}$. As $\mathbf{y} \leftarrow D_\sigma$, with high probability $\|\mathbf{y}\|_\infty \leq 14\sigma$. Each $u_i$ has coefficients uniform in $[-2^{k_i-1}, 2^{k_i-1}-1]$, so on $k_i$ bits. Each of the $\lceil \frac{128}{\log q} \rceil$ polynomials $g_i$ has uniform coefficients mod $q$, except the constant coefficient which is 0, so on $\lceil \log q \rceil$ bits each. Then $n_6 \coloneqq \sum_{i\in[4]} k_i + (\lceil \log 14\sigma \rceil + 1) + \lceil \frac{128}{\log q} \rceil \cdot \lceil \log q \rceil$ and $\|\mathbf{s}_6\| \leq \beta_6$ for

$$\beta_6 \coloneqq \sqrt{256(\sum_{i\in[4]} k_i) + 256(\lceil \log 14\sigma \rceil + 1) + 255 \cdot \lceil \frac{128}{\log q} \rceil \cdot \lceil \log q \rceil}.$$

**The Number of Carries $N$.** We set the number of carries such that the probability that $\|\mathbf{v}_i^{(1)}\|^2 \leq N$ is at least $1 - \frac{1}{2^{130}}$, where $\mathbf{v}_i^{(1)} \in \{-1, 0, 1\}^{256}$ is the vector of high-order values. If we denote by $p \coloneqq \Pr[|\mathbf{v}_i^{(1)}[j]| = 1]$ the probability of a carry, for any entry of index $j \in [256]$, then we can model $\|\mathbf{v}_i^{(1)}\|^2 = \sum_{j\in[256]} (\mathbf{v}_i^{(1)}[j])^2$ as $X \sim \mathrm{Bin}(256, p)$, so we can find $N$ such that its cumulative distribution function, $\Pr[X \leq N] = \sum_{k=0}^{N} \binom{256}{k} p^k (1-p)^{256-k}$ is at least $1 - \frac{1}{2^{130}}$. Thus, it suffices to compute the probability $p$. The masked projection of some witness vector $\mathbf{s}$ at any of the 256 indices looks like $\langle \pi, \mathbf{s} \rangle + u = 2^k v^{(1)} + v^{(0)}$, for some projection row $\pi$, uniform mask $u \leftarrow [-2^{k-1}, 2^{k-1}-1]$, $v^{(1)} \in \{-1, 0, 1\}$ and $v^{(0)} \in [-2^{k-1}, 2^{k-1}-1]$ and $k$ depends on the norm bound on $\mathbf{s}$, as $k \coloneqq \lceil \log(9.75\|\mathbf{s}\|) \rceil$. Then a carry happens if the following event $E$ happens: $\langle \pi, \mathbf{s} \rangle + u \geq 2^{k-1}$ or $\langle \pi, \mathbf{s} \rangle + u \leq -2^{k-1} - 1$. Then

$$\begin{aligned} p = \Pr_{\pi,u}[E] &= \sum_{a=-2^k}^{2^k} \Pr_u[E | \langle \pi, \mathbf{s} \rangle = a] \cdot \Pr_\pi[\langle \pi, \mathbf{s} \rangle = a] \\ &= \sum_{a=-2^k}^{2^k} p_a \cdot \Pr_\pi[\langle \pi, \mathbf{s} \rangle = a], \end{aligned}$$

where we denoted the first term of the product by $p_a$. We claim that, for $a = 0$, $p_a = 0$ and else, equals $\frac{|a|}{2^k}$. In this case, $p = \frac{E[|\langle \pi, \mathbf{s} \rangle|]}{2^k} = \frac{E[|\langle \pi, \mathbf{s} \rangle|]/\|\mathbf{s}\|}{2^k/\|\mathbf{s}\|} = \frac{E[|\langle \pi, \mathbf{s} \rangle|]/\|\mathbf{s}\|}{9.75}$.

About the claim on $p_a$, note that

$$p_a = \Pr_u[a + u \geq 2^{k-1} \text{ or } a + u \leq -2^{k-1} - 1 | \langle \pi, \mathbf{s} \rangle = a].$$

Keep in mind that $u$ takes uniform values over $[-2^{k-1}, 2^{k-1}-1]$. Then, if $a = 0$, for sure $p_a = 0$. If $0 < a \leq 2^k$, $p_a = \Pr_u[u \geq 2^{k-1} - a] = \frac{a}{2^k}$, whereas, if $-2^k \leq a < 0$, $p_a = \Pr_u[u \leq -2^{k-1} - a - 1] = \frac{|a|}{2^k}$.

For computing $p$, it is thus enough to see how big $E[|\langle \pi, \mathbf{s} \rangle|]/\|\mathbf{s}\|$ is. One naive approach would be to approximate $\langle \pi, \mathbf{s} \rangle \simeq N(0, \sigma)$, for $\sigma \coloneqq \|\mathbf{s}\|/\sqrt{2}$, by a central limit theorem argument, as $\pi \leftarrow \mathtt{Bin}_1$ has entries of standard deviation $1/\sqrt{2}$. This approximation is in general good to use when the entries of $\mathbf{s}$ are roughly all the same size. Thus, as $E[|Y|] = \frac{\sigma\sqrt{2}}{\sqrt{\pi}}$, for a normal variable $Y$ of standard deviation $\sigma$, one gets $p = \frac{1}{9.75\sqrt{\pi}}$, and for $N = 83$ the probability of having more than $N$ carries becomes $1 - \sum_{k=0}^{N} \binom{256}{k} p^k (1-p)^{256-k} < \frac{1}{2^{130}}$. But as for a more general vector $\mathbf{s}$, this approximation from the central limit theorem might incur a small error, we could upper-bound instead $E[|\langle \pi, \mathbf{s} \rangle|]/\|\mathbf{s}\|$ by $1/\sqrt{3}$, (Lemma 9) leading to $p = \frac{1}{9.75\sqrt{3}}$. Using this value of $p$ and $N = 83$, the same probability of more than $N$ carries is $1 - \sum_{k=0}^{N} \binom{256}{k} p^k (1-p)^{256-k} \simeq \frac{1}{2^{128}}$ instead.

For independent interest, we are giving below a proof for the claimed upper-bound of $E[|\langle \pi, \mathbf{s} \rangle|]/\|\mathbf{s}\|$. The idea is to express the inner product $X \coloneqq \langle \pi, \mathbf{s} \rangle$ as the difference of two independent copies of a simpler

inner product $Y \coloneqq \langle \pi', \mathbf{s} \rangle$, where the vector $\pi'$ has now independent uniform bits. Then $E|X|$ measures how spread out $Y$ is. We show this spread is controlled by how well $Y$ lines up with a uniform random number on $[0,1]$ and then we can upper-bound the expectation using Cauchy–Schwarz.

**Lemma 9.** *For any nonzero vector* $\mathbf{s} \in \mathbb{R}^n$, $\frac{E[|\langle \pi, \mathbf{s} \rangle|]}{\|\mathbf{s}\|} < \frac{1}{\sqrt{3}}$.

*Proof.* Let $\mathbf{s}$ be a fixed vector and $X \coloneqq \langle \pi, \mathbf{s} \rangle$. As $\pi \leftarrow \mathtt{Bin}_1$, we can rewrite $\pi = \pi' - \pi''$, for $\pi'$ and $\pi''$ vectors of independent uniform bits. Let $Y \coloneqq \langle \pi', \mathbf{s} \rangle$ and $Y' \coloneqq \langle \pi'', \mathbf{s} \rangle$ the other. Keep in mind that these two variables are independent identically distributed (i.i.d) and take finitely many values. Note that $\mathrm{Var}(Y) = \sum_j \mathbf{s}_j^2 \cdot \frac{1}{4} = \frac{\|\mathbf{s}\|^2}{4}$ and similarly for $Y'$. Then we get $X = Y - Y'$, where $Y'$ can be seen as an independent copy of $Y$.

Now we are aiming at removing the absolute value when computing the expectation. Since $|z| = z \ \mathrm{sign}(z)$,

$$E[|X|] = E[|Y - Y'|] = E\big[Y \ \mathrm{sign}(Y - Y')\big] - E\big[Y' \ \mathrm{sign}(Y - Y')\big].$$

Swapping the names $Y \leftrightarrow Y'$ does not change any expectation, because as they are i.i.d. We can rewrite then the second term as $E[Y \ \mathrm{sign}(Y' - Y)] = -E[Y \ \mathrm{sign}(Y - Y')]$. Hence

$$E[|X|] = 2\, E\big[Y \ \mathrm{sign}(Y - Y')\big]. \tag{30}$$

Now list all the possible (finitely many) values of $Y$ as $y_1 < y_2 < \cdots < y_k$, with probabilities $p_1, \ldots, p_k$. Cut $[0,1]$ into consecutive intervals $I_i \coloneqq (c_i,\, c_i + p_i]$, $c_i \coloneqq p_1 + \cdots + p_{i-1}$, so $I_i$ has length $p_i$. Notice that $Y$ could be defined in terms of a random variable $U$ uniform over $[0,1]$ as $Y = y_i$ when $U \in I_i$, and similarly $Y'$ (independent on $Y$).

Fix $Y = y_i$. Then

$$E\big[\,\mathrm{sign}(y_i - Y')\big] = \Pr[Y' < y_i] - \Pr[Y' > y_i] = c_i - (1 - c_i - p_i) = 2\Big(c_i + \frac{p_i}{2}\Big) - 1.$$

Given $Y = y_i$, the variable $U$ is uniform on $I_i$, whose expectation is $c_i + p_i/2$. So

$$E\big[\,\mathrm{sign}(Y - Y') \,\big|\, Y\big] = 2\, E[U \mid Y] - 1.$$

Using the tower rule twice, $E\big[YZ\big] = E\big[E\big[YZ \mid Y\big]\big] = E\big[Y E\big[Z \mid Y\big]\big]$ for any random variable $Z$,

$$\begin{aligned} E\big[Y \ \mathrm{sign}(Y - Y')\big] &= E\big[E\big[Y \ \mathrm{sign}(Y - Y') \mid Y\big]\big] = E\big[Y E\big[\,\mathrm{sign}(Y - Y') \mid Y\big]\big] \\ &= E\big[Y \cdot (2E\big[U \mid Y\big] - 1)\big] = 2E\big[E\big[YU \mid Y\big]\big] - E\big[Y\big] \\ &= 2\, E[YU] - E[Y]. \end{aligned}$$

As $U$ is a uniform variable over $[0,1]$, $E[U] = \frac{1}{2}$, and therefore $2\, E[YU] - E[Y] = 2\big(E[YU] - E[Y]E[U]\big) = 2\,\mathrm{Cov}(Y,U)$. Combining with Equation (30),

$$E\big[|X|\big] = 4\ \mathrm{Cov}(Y,U). \tag{31}$$

We can see that since $Y = \langle \pi', \mathbf{s} \rangle = \sum_{i \in [n]} b_i \mathbf{s}_i$, for $\pi' \coloneqq (b_1, \ldots, b_n)$ made of uniform bits, $E\big[Y\big] = \sum_{i \in [n]} \frac{\mathbf{s}_i}{2}$ and

$$\begin{aligned} E\big[Y^2\big] = E\big[(\sum_{i \in [n]} b_i \mathbf{s}_i)^2\big] &= E\big[\sum_{i \in [n]} b_i^2 \mathbf{s}_i^2 + \sum_{i \neq j} b_i b_j \mathbf{s}_i \mathbf{s}_j\big] = \sum_{i \in [n]} \mathbf{s}_i^2 E\big[b_i^2\big] + \sum_{i \neq j} \mathbf{s}_i \mathbf{s}_j E\big[b_i b_j\big] \\ &= \frac{\|\mathbf{s}\|^2}{2} + \frac{\sum_{i \neq j} \mathbf{s}_i \mathbf{s}_j}{4} = \frac{\|\mathbf{s}\|^2}{2} + \frac{(\sum_{i \in [n]} \mathbf{s}_i)^2 - \sum_{i \in [n]} \mathbf{s}_i^2}{4} = \frac{\|\mathbf{s}\|^2}{4} + \frac{(\sum_{i \in [n]} \mathbf{s}_i)^2}{4}, \end{aligned}$$

which leads to $\mathrm{Var}(Y) = \frac{\|\mathbf{s}\|^2}{4}$. Moreover, we know that for a uniform variable $U$ over $[0,1]$, $E\big[U\big] = \frac{1}{2}$ and $E\big[U^2\big] = \frac{1}{3} < \infty$, which leads to $\mathrm{Var}(U) = \frac{1}{12}$. As $\mathrm{Var}(Y) = E\big[(Y - E\big[Y\big])^2\big]$, $\mathrm{Var}(U) = E\big[(U - E\big[U\big])^2\big] <$

$\infty$, we can apply the Cauchy–Schwarz inequality, $|E[XY]| \leq \sqrt{E[X^2]} \cdot \sqrt{E[Y^2]}$, for the variables $X = Y - E[Y]$ and $Y = U - E[U]$ as: $|\operatorname{Cov}(Y,U)| \leq \sqrt{\operatorname{Var}(Y)} \cdot \sqrt{\operatorname{Var}(U)}$. Using Equation (31), we obtain

$$E[|X|] \;\leq\; 4 \cdot \frac{\|\mathbf{s}\|}{2} \cdot \frac{1}{2\sqrt{3}} \;=\; \frac{\|\mathbf{s}\|}{\sqrt{3}}.$$

But the equality in Cauchy–Schwarz requires $Y - E[Y] = t \cdot (U - E[U])$, almost surely for a constant $t$ which is equivalent with $Y = aU + b$ almost surely for some constants $a, b$. That is impossible here:

- if $a \neq 0$, then $aU + b$ is continuous, but $Y$ takes finitely many values;
- if $a = 0$, then $Y$ is constant, so $\operatorname{Var}(Y) = \|\mathbf{s}\|^2/4 = 0$, contradicting $\mathbf{s} \neq 0$.

This analysis makes the inequality to be strict $\frac{E|\langle \pi, \mathbf{s} \rangle|}{\|\mathbf{s}\|} < \frac{1}{\sqrt{3}}$.

**The Standard Deviations $\sigma, \sigma_1, \sigma_2$.** In general, the standard deviation of the Gaussian distribution, used in rejection sampling (Lemma 3) is set as $\gamma \cdot \alpha$, where $\alpha$ is an upper bound of $\|\mathbf{x}\|$, for a vector $\mathbf{x}$ over $\mathcal{R}$ we want to mask, and a repetition constant $\gamma$. Recall that by definition of the challenge space $\mathcal{C}$ from Section 2.1, $\|c \cdot \mathbf{x}\| \leq T\alpha$, for any challenge $c \in \mathcal{C}$.

- In the first rejection sampling application, we want to mask $\bar{\mathbf{\Pi}}\mathbf{v}_1$, where $\mathbf{v}_1$ is the vector of high-order values. As $\|\mathbf{v}_1\| \leq \sqrt{4 \cdot N}$, we have with high probability $\|\bar{\mathbf{\Pi}}\mathbf{v}_1\| \leq \sqrt{337 \cdot 4 \cdot N}$. (Theorem 2) Therefore, we can set $\sigma \coloneqq \gamma \cdot \sqrt{337 \cdot 4 \cdot N}$.
- In the last rejection sampling application, we want to mask $c(\tilde{\mathbf{s}}, \tilde{\mathbf{v}})$, and $c(\mathbf{r}_s, \mathbf{r}_v)$, for a given challenge $c \in \mathcal{C}$. Note that $\|(\tilde{\mathbf{s}}, \tilde{\mathbf{v}})\|^2 = \sum_{i \in [6]} \|\mathbf{s}_i\|^2 + \|\tilde{\mathbf{v}}_1\|^2 + \|\tilde{\mathbf{v}}_2\|^2 \leq \sum_{i \in [6]} \beta_i^2 + 2 \cdot 4 \cdot N$. Then, we can set $\sigma_1 \coloneqq \gamma_1 \cdot T \cdot \sqrt{\sum_{i \in [6]} \beta_i^2 + 8N}$. Moreover, $\|(\mathbf{r}_s, \mathbf{r}_v)\|^2 \leq d \cdot (n_s + n_v)$, as these random vectors are made of binary ring elements. Then, we can set $\sigma_2 \coloneqq \gamma_2 \cdot T \cdot \sqrt{256 \cdot (n_s + n_v)}$.

The parameters $\gamma_1, \gamma_2$ are repetition rate-related constants coming from the rejection sampling algorithms.

**The Modulus $q$ for the Next LaBRADOS Iteration.** We want the LaBRADOS modulus to coincide with the one of LNP-Lite. To prove correctness of the binary constraint on the vector $\mathbf{x}_{\mathtt{bin}}$, we prove an approximate norm bound of it, $\|\mathbf{x}_{\mathtt{bin}}\| \leq \sqrt{d \cdot n_{\mathtt{bin}}}$, with the slack factor $c_{\mathtt{LaBRADOS}} = \frac{9.75}{0.49}$. To make sure this constraint does not wraparound mod $q$, it suffices to set $2\|\mathbf{x}_{\mathtt{bin}}\|^2 \leq q$. This leads to setting

$$2 \cdot d \cdot n_{\mathtt{bin}} \cdot c^2_{\mathtt{LaBRADOS}} \leq q.$$

**The LNP-Lite Modulus $q$.** One can set the modulus $q$ as indicated by Theorem 1 and as above.

# E Security of the Primitives

## E.1 Collision Resistant Hash Function

The hash function in Section 4.2 is collision resistant, assuming the intractability of the M-SIS problem with parameters $M = 1$, $N = 16$, $q = 256$, $\beta_\infty = 1$ over $\mathcal{R}$ of degree $d = 512$.

*Proof.* Let $\mathcal{A}_{\text{CR-hash}}$ be a probabilistic polynomial-time adversary that finds collisions with some advantage $\epsilon$. One constructs a probabilistic polynomial-time adversary $\mathcal{A}_{\text{M-SIS}}$ that solves M-SIS instances with the same advantage as follows:

- $\mathcal{A}_{\text{M-SIS}}$ calls $\mathcal{A}_{\text{CR-hash}}$ to obtain a pair $\mathbf{x} \neq \mathbf{x}' \in \mathcal{R}_2^*$, and aborts if it is not a collision.
- If the inputs $\mathbf{x}, \mathbf{x}'$ have different dimensions, the last block of the padded input is different, $\mathbf{x}_{k-1} \neq \mathbf{x}'_{k'-1}$. Then, there is a collision in the last compression, i.e. $(\mathbf{y}_{k-2}, \mathbf{x}_{k-1}) \neq (\mathbf{y}'_{k'-2}, \mathbf{x}'_{k'-1}) \in \mathcal{R}_2^{16}$ with $\mathbf{A}(\mathbf{y}_{k-2}, \mathbf{x}_{k-1}) = \mathbf{A}(\mathbf{y}'_{k'-2}, \mathbf{x}'_{k'-1})$. $\mathcal{A}_{\text{M-SIS}}$ returns the difference $(\mathbf{y}_{k-2}, \mathbf{x}_{k-1}) - (\mathbf{y}'_{k'-2}, \mathbf{x}'_{k'-1})$, which is a non-zero vector $\mathbf{v}$ with coefficients in $\{-1, 0, 1\}$ satisfying $\mathbf{A}\mathbf{v} = 0 \bmod 256$.

– Else, since $\mathbf{x} \neq \mathbf{x}'$ and the number of blocks of the padded inputs is equal, there exists $\mathbf{x}_i \neq \mathbf{x}'_i$. It follows that either $(\mathbf{y}_{i-1}, \mathbf{x}_i) \neq (\mathbf{y}'_{i-1}, \mathbf{x}'_i)$ and $\mathbf{A}(\mathbf{y}_{i-1}, \mathbf{x}_i) = \mathbf{A}(\mathbf{y}'_{i-1}, \mathbf{x}'_i)$ or $\mathbf{y}_i \neq \mathbf{y}'_i$, hence $(\mathbf{y}_i, \mathbf{x}_{i+1}) \neq (\mathbf{y}'_i, \mathbf{x}'_{i+1})$. In the first case, $\mathcal{A}_{\text{M-SIS}}$ returns the difference $(\mathbf{y}_{i-1}, \mathbf{x}_i) - (\mathbf{y}'_{i-1}, \mathbf{x}'_i)$. In the second case, it proceeds until it finds an iteration in the compression function and returns the difference.

We estimate the concrete hardness of the SIS problem with $d$ rows, $16d$ columns, $\ell_2$ norm bound $\beta = \sqrt{16d}$ as at least 256-bits secure, by the Lattice Estimator [4], commit 6019056.

### E.2 Pseudorandom Generator

The function in Section 4.1 is a pseudorandom generator, assuming the hardness of some Module-LWR problem.

*Proof.* Let $\mathcal{A}_{\text{PRG}}$ be a probabilistic polynomial-time algorithm that distinguishes the PRG output from a uniformly random one with advantage $\epsilon_{\text{PRG}}$, i.e.

$$\big|\Pr\big[\mathcal{A}_{\text{PRG}}(\mathbf{A}, \mathbf{s}) = 1 |\ \mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 2},\ s_0^{(6)}, s_0^{(7)} \leftarrow \mathcal{R}_2,\ \mathbf{s} = \text{PRG}(s_0^{(6)}, s_0^{(7)})\big] - \Pr\big[\mathcal{A}_{\text{PRG}}(\mathbf{A}, \mathbf{s}) = 1 |\ \mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 2},\ \mathbf{s} \leftarrow \mathcal{R}_2^{4k+2}\big]\big| = \epsilon_{\text{PRG}}.$$

One constructs a probabilistic polynomial-time algorithm $\mathcal{A}_{\text{M-LWR}}$ that distinguishes M-LWR samples from uniformly random ones with advantage $\epsilon_{\text{MLWR}} = \epsilon_{\text{PRG}}/k$ as follows.

– Receive as input a pair $(\mathbf{A}, t)$ where $\mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 2}$ and $t \in \mathcal{R}_{64}$.
– Sample $i \leftarrow [k]$ and let $\mathbf{s}_i$ be the decomposition of $t$, $t = \mathbf{G}_2\mathbf{s}_i$.
– Sample $s_j^{(2)}, \ldots, s_j^{(5)} \leftarrow \mathcal{R}_2$ for $j \in [i-1]$.
– Compute $\lceil \mathbf{A}(s_j^{(6)}, s_j^{(7)})\rfloor_{256\to 64} = \mathbf{G}_2\mathbf{s}_{j+1}$ for $j \in [i, k-1]$.
– Return $\mathcal{A}_{\text{PRG}}(\mathbf{A}, \mathbf{s})$.

For $i = 0, \ldots, k$, let the *hybrid* $H_i$ describe the distribution

$$H_i = \{(\mathbf{A}, \mathbf{s}) :\ \mathbf{A} \leftarrow \mathcal{R}_{256}^{1\times 2},\ s_j^{(2)}, \ldots, s_j^{(5)} \leftarrow \mathcal{R}_2 \text{ for } j \in [i], \\ s_i^{(6)}, s_i^{(7)} \leftarrow \mathcal{R}_2,\ \lceil \mathbf{A}(s_j^{(6)}, s_j^{(7)})\rfloor_{256\to 64} = \mathbf{G}_2\mathbf{s}_{j+1} \text{ for } j \in [i, k-1]\}.$$

Note that $H_0$ describes the evaluation of the pseudorandom generator on a random input $s_0^{(6)}, s_0^{(7)} \leftarrow \mathcal{R}_2$, while $H_k$ presents a random binary output. Then, $\epsilon_{\text{PRG}}$ can be written as

$$\big|\Pr\big[\mathcal{A}_{\text{PRG}}(H_0) = 1\big] - \Pr\big[\mathcal{A}_{\text{PRG}}(H_k) = 1\big]\big|.$$

By the construction of $\mathcal{A}_{\text{M-LWR}}$ and the union bound, we have $\Pr\big[\mathcal{A}_{\text{M-LWR}}(A, t) = 1 \mid t = \lceil A\mathbf{s}\rfloor_{256\to 64}\big] = \frac{1}{k}\sum_{i=1}^{k}\Pr\big[\mathcal{A}(H_{i-1}) = 1\big]$ and $\Pr\big[\mathcal{A}_{\text{M-LWR}}(A, t) = 1 \mid t \leftarrow \mathcal{R}_{64}\big] = \frac{1}{k}\sum_{i=1}^{k}\Pr\big[\mathcal{A}(H_i) = 1\big]$, then

$$\epsilon_{\text{M-LWR}} = \frac{1}{k}\Big|\sum_{i=1}^{k}\Pr\big[\mathcal{A}_{\text{PRG}}(H_{i-1}) = 1\big] - \Pr\big[\mathcal{A}_{\text{PRG}}(H_i) = 1\big]\Big| \\ = \frac{1}{k}\big|\Pr\big[\mathcal{A}_{\text{PRG}}(H_0) = 1\big] - \Pr\big[\mathcal{A}_{\text{PRG}}(H_k) = 1\big]\big| = \frac{1}{k}\epsilon_{\text{PRG}}.$$

To estimate the concrete hardness of the Module-LWR problem with parameters $q = 256$, $p = 64$, $m' = 1$, $n' = 2$, $\chi = \mathcal{U}(\mathcal{R}_2)$, we model the LWR problem as an LWE problem, with $d$ rows, $2 \cdot d$ columns, modulus $q$, secret distribution as $\chi$, and error distribution as Gaussian of standard deviation $\sqrt{((q/p)^2 - 1)/12}$. ( [2]). According to the Lattice Estimator, [4], commit 6019056, this instance costs at least $2^{300}$ ring operations.

### E.3 Output Transformation

The modulus transformation in Section 4.1 is, intuitively, an expander that converts a binary input into ring elements. For simplicity, we chose the output entropy of the PRG to be the same as the size of the ring and we perform the mapping via a matrix-vector multiplication where the matrix consists of random ring elements from the target ring. If the binary input to this multiplication is uniformly-random, then one can view the output distribution as a Ring-LWE instance. Because the entropy of the input is as large as the output, this in fact the hardest LWE instance for the ring. The security of this transformation is therefore much larger than the security of the PRG (roughly 7000 bits), and we include the formal analysis below merely for completeness.

If we treat the input to this expander as random, then the output is indistinguishable from a random element, assuming the hardness of some Module-LWE problem.

*Proof.* The proof is analogous to the previous one. From the adversary $\mathcal{A}_{\text{mod}}$ that distinguishes the output of the function from a random one with advantage $\epsilon_{\text{mod}}$, one constructs a Module-LWE adversary $\mathcal{A}_{\text{M-LWE}}$ with advantage $\epsilon_{\text{mod}}/l$ as follows

- sample $i \leftarrow [l]$ and set $t_i = t$,
- sample $t_j \leftarrow \mathcal{R}_q$ for $j \in [0, i-1]$,
- compute $t_j = \mathbf{B}\tilde{\mathbf{s}}_j$ with $\tilde{\mathbf{s}}_j \leftarrow \mathcal{R}_2^\ell$ for $j \in [i+1, l-1]$.
- return $\mathcal{A}_{\text{mod}}(\mathbf{B}, \mathbf{t})$ with $\mathbf{t} = (t_0, t_1, \dots)^T$.

The hybrids are defined, for $i \in [0, l]$ as

$$H_i = \{(\mathbf{B}, \mathbf{t}) : t_j \leftarrow \mathcal{R}_q \text{ for } j \in [0, i-1],\ \tilde{\mathbf{s}}_j \leftarrow \mathcal{R}_2^\ell \\ t_j = \mathbf{B}\tilde{\mathbf{s}}_j \mod q \quad \text{for } j \in [i, l-1]\}.$$

The concrete hardness of the LWE problem with $d$ rows, $d \cdot \lceil \log(q) \rceil$ columns, secret and error distributions uniform binary, for the Falcon modulus $q = 12289$, is estimated to be at least 7000-bits secure by the Lattice Estimator. [4].

## F Figures

We include Figures 5 and 6 for `LaBRADOR`, Figures 7 and 8 for `LaBRADOS`, Figures 9 and 10 for `LNP`, and Figures 11 and 12 for `LNP-Lite`. We recall the polynomial ring we work with is $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$, where $d = 256$, $\mathcal{R}_q \coloneqq \mathcal{R}/(q\mathcal{R})$, and $\mathcal{R}_2 \coloneqq \mathcal{R}/(2\mathcal{R})$. By $\texttt{Decomp}_b$ we mean decomposing with respect to some integer $b$, by `BinDecomp` we mean decomposing in binary, whereas by `OutCom` applying the (Ajtai) outer commitment layer.

Instance: constraints $f_i, f_i'$ as in (1), global bound $\beta$
Witness: $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_r), \mathbf{s}_i \in \mathcal{R}^n$

**1. Commit**

$\mathbf{t}_i \coloneqq \mathbf{A}\mathbf{s}_i$
$\mathbf{g}_{ij} \coloneqq \langle \mathbf{s}_i, \mathbf{s}_j \rangle$
$\hat{\mathbf{t}} \coloneqq \mathtt{Decomp}_b((\mathbf{t}_i)_i)$
$\hat{\mathbf{g}} \coloneqq \mathtt{Decomp}_b((g_{ij})_{ij})$
Send $\mathbf{u} \coloneqq \mathbf{B}(\hat{\mathbf{t}}, \hat{\mathbf{g}})$

**2. Approximate Norm Proof**

Receive $\mathbf{\Pi} \in \{-1, 0, 1\}^{256 \times drn}$
Send $\mathbf{p} \coloneqq \mathbf{\Pi}(\mathbf{s}_1, \ldots, \mathbf{s}_r) \in \mathbb{Z}_q^{256}$

**3. Aggregate the Constraints**

Receive $\mu_j^{(i)} \in \mathbb{Z}_q$
$h_i \coloneqq \sum_j \mu_j^{(i)} f_j', i \in [\lambda]$
Send $(h_i)_{i \leq \lambda}$
Receive $\nu_j \in \mathcal{R}_q$
$f \coloneqq \sum_j \nu_j f_j$
Compute $a_{ij}, b \in \mathcal{R}_q, \varphi_i \in \mathcal{R}_q^n$ s.t.
$f = \sum_{i,j \in [r]} a_{ij} \langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i \in [r]} \langle \varphi_i, \mathbf{s}_i \rangle - b$
$\mathbf{h}_{ij} \coloneqq \frac{1}{2}(\langle \varphi_i, \mathbf{s}_j \rangle + \langle \varphi_j, \mathbf{s}_i \rangle)$
$\hat{\mathbf{h}} \coloneqq \mathtt{Decomp}_b((\mathbf{h}_{ij})_{ij})$
Send $\mathbf{v} \coloneqq \mathbf{D}(\hat{\mathbf{h}})$

**4. Prove the Final Constraint**

Receive $c_i \in \mathcal{C}$
$\mathbf{z} \coloneqq \sum_i c_i \mathbf{s}_i$
Send $\mathbf{z}, \hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$

**Fig. 5.** Prover of the LaBRADOR protocol [13].

Instance: constraints $f_i, f_i'$ as in (1), global bound $\beta$

**1. Commit**

Receive $\mathbf{u}$

**2. Approximate Norm Proof**

Send $\mathbf{\Pi} \leftarrow \mathtt{Bin}_1^{256 \times drn}$
Receive $\mathbf{p}$
**Check** $\|\mathbf{p}\| \leq \sqrt{128}\beta$

**3. Aggregate the Constraints**

Send $\mu_j^{(i)} \leftarrow \mathbb{Z}_q$
Receive $(h_i)_{i \leq \lambda}$
**Check** $\mathtt{ct}(h_i) = 0$
Send $\nu_j \leftarrow \mathcal{R}_q$
Compute $a_{ij}, b \in \mathcal{R}_q$, $\varphi_i \in \mathcal{R}_q^n$ s.t.
$f = \sum_{i,j \in [r]} a_{ij} \langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i \in [r]} \langle \varphi_i, \mathbf{s}_i \rangle - b$
Receive $\mathbf{v}$

**4. Prove the Final Quadratic**

Send $c_i \leftarrow \mathcal{C}$
Receive $\mathbf{z}, \hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$
Recompose $\hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$ into $\mathbf{t_i}, \mathbf{g_{ij}}, \mathbf{h_{ij}}$
**Check** correctness of $\mathbf{u}, \mathbf{v}$ w.r.t. $\hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$
**Check** $\mathbf{z}$, $\hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$ are small
**Check** $\mathbf{A}\mathbf{z} = \sum_{i \in [r]} c_i \mathbf{t}_i$
**Check** $\langle \mathbf{z}, \mathbf{z} \rangle = \sum_{i,j \in [r]} c_i c_j \mathbf{g}_{ij}$
**Check** $\sum_{i \in [r]} \langle \varphi_i, \mathbf{z} \rangle c_i = \sum_{i,j \in [r]} c_i c_j \mathbf{h}_{ij}$
**Check** $\sum_{i,j \in [r]} a_{ij} \mathbf{g}_{ij} + \sum_{i \in [r]} \mathbf{h}_{ii} - b = 0$

**Fig. 6.** Verifier of the LaBRADOR protocol [13].

Instance: constraints $f_i, f_i'$ as in (1) and bounds $\beta_i$
Witness: $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_r), \mathbf{s}_i \in \mathcal{R}^n$

**1. Commit**

$\mathbf{t}_i \coloneqq \mathbf{A}\mathbf{s}_i$
$\mathbf{g}_{ij} \coloneqq \langle \mathbf{s}_i, \mathbf{s}_j \rangle$
$\hat{\mathbf{t}} \coloneqq \mathtt{Decomp}_b((\mathbf{t}_i)_i)$
$\hat{\mathbf{g}} \coloneqq \mathtt{Decomp}_b((g_{ij})_{ij})$
$\mathbf{u}_1' \coloneqq \mathtt{BinDecomp}(\mathbf{B}(\hat{\mathbf{t}}, \hat{\mathbf{g}}))$
Send $\mathbf{u}_1 \coloneqq \mathtt{OutCom}(\mathbf{u}_1')$

**2. Approximate Norm Proof**

Receive $\mathbf{\Pi} \in \{-1, 0, 1\}^{256 \times dn}$
$\mathbf{p}_i \coloneqq \mathbf{\Pi}\mathbf{s}_i \in \mathbb{Z}_q^{256}$
$\mathbf{u}_2' = \mathtt{BinDecomp}((\mathbf{p}_i)_i)$
Send $\mathbf{u}_2 \coloneqq \mathtt{OutCom}(\mathbf{u}_2')$

**3. Aggregate the Constraints**

Receive $\mu_j^{(i)} \in \mathbb{Z}_q$
$h_i \coloneqq \sum_j \mu_j^{(i)} f_j', i \in [\lambda]$
$\mathbf{u}_3' \coloneqq \mathtt{BinDecomp}((h_i)_i)$
Send $\mathbf{u}_3 \coloneqq \mathtt{OutCom}(\mathbf{u}_3')$
Receive $\nu_j \in \mathcal{R}_q$
$f \coloneqq \sum_j \nu_j f_j$
Compute $a_{ij}, b \in \mathcal{R}_q, \varphi_i \in \mathcal{R}_q^n$ s.t.
$f = \sum_{i,j \in [r]} a_{ij} \langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i \in [r]} \langle \varphi_i, \mathbf{s}_i \rangle - b$
$\mathbf{h}_{ij} \coloneqq \frac{1}{2}(\langle \varphi_i, \mathbf{s}_j \rangle + \langle \varphi_j, \mathbf{s}_i \rangle)$
$\hat{\mathbf{h}} \coloneqq \mathtt{Decomp}_b((\mathbf{h}_{ij})_{ij})$
$\mathbf{u}_4' \coloneqq \mathtt{BinDecomp}(\mathbf{D}(\hat{\mathbf{h}}))$
Send $\mathbf{u}_4 \coloneqq \mathtt{OutCom}(\mathbf{u}_4')$

**4. Prove the Final Constraint**

Receive $c_i \in \mathcal{C}$
$\mathbf{z} \coloneqq \sum_i c_i \mathbf{s}_i$
Send $\mathbf{z}, \mathbf{x}_{\mathtt{mid}} \coloneqq (\hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}})$, $\mathbf{x}_{\mathtt{bin}} \coloneqq (\mathbf{u}_1', \mathbf{u}_2', \mathbf{u}_3', \mathbf{u}_4')$

**Fig. 7.** Prover of the LaBRADOS protocol [15].

Instance: constraints $f_i, f_i'$ as in (1) and bounds $\beta_i$

**1. Commit**

Receive $\mathbf{u}_1$

**2. Approximate Norm Proof**

Send $\mathbf{\Pi} \leftarrow \mathtt{Bin}_1^{256 \times dn}$
Receive $\mathbf{u}_2$

**3. Aggregate the Constraints**

Send $\mu_j^{(i)} \leftarrow \mathbb{Z}_q$
Receive $\mathbf{u}_3$
Send $\nu_j \leftarrow \mathcal{R}_q$
Compute $a_{ij}, b \in \mathcal{R}_q$, $\varphi_i \in \mathcal{R}_q^n$ s.t.
$f = \sum_{i,j \in [r]} a_{ij} \langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i \in [r]} \langle \varphi_i, \mathbf{s}_i \rangle - b$
Receive $\mathbf{u}_4$

**4. Prove the Final Constraint**

Send $c_i \leftarrow \mathcal{C}$
Receive $\mathbf{z}, \mathbf{x}_{\mathtt{mid}} = (\hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}})$, $\mathbf{x}_{\mathtt{bin}} = (\mathbf{u}_1', \mathbf{u}_2', \mathbf{u}_3', \mathbf{u}_4')$
Recompose $\mathbf{u}_3', \hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$ into $h_i, \mathbf{t_i}, \mathbf{g_{ij}}, \mathbf{h_{ij}}$
**Check** $\mathbf{u_i} = \mathtt{OutCom}(\mathbf{u}_i'), i \in [4]$
**Check** correctness of $\mathbf{u_1'}, \mathbf{u_4'}$ w.r.t. $\hat{\mathbf{t}}, \hat{\mathbf{g}}, \hat{\mathbf{h}}$
**Check** $\mathbf{x}_{\mathtt{bin}}$ is binary
**Check** $\mathtt{ct}(h_i) = 0$
**Check** $\mathbf{z}, \mathbf{x}_{\mathtt{mid}}$ are small
**Check** $\mathbf{A}\mathbf{z} = \sum_{i \in [r]} c_i \mathbf{t}_i$
**Check** $\langle \mathbf{z}, \mathbf{z} \rangle = \sum_{i,j \in [r]} c_i c_j \mathbf{g}_{ij}$
**Check** $\sum_{i \in [r]} \langle \varphi_i, \mathbf{z} \rangle c_i = \sum_{i,j \in [r]} c_i c_j \mathbf{h}_{ij}$
**Check** $\sum_{i,j \in [r]} a_{ij} \mathbf{g}_{ij} + \sum_{i \in [r]} \mathbf{h}_{ii} - b = 0$

**Fig. 8.** Verifier of the LaBRADOS protocol [15].

Instance: constraints $f_i$, $f'_i$ as in (1) and global bound $\beta$
Witness: $\mathbf{s}_i \in \mathcal{R}^{n_i}, i \in [r]$

**1. Commit**

$\mathbf{y} \leftarrow D_\sigma^{256/d}$
$g_i \leftarrow \{x \in \mathcal{R}_q : \mathtt{ct}(x) = 0\},\ i \in [\lambda],\ \lambda \coloneqq \lceil 128/\log q \rceil$
$b \leftarrow \{-1, 1\}$
$\mathbf{s}_{r+1} \coloneqq \mathtt{BinDecomp}(y, (g_i)) \in \mathcal{R}^{n_{r+1}}$
$\tilde{\mathbf{s}} \coloneqq (\mathbf{s}_1, \ldots, \mathbf{s}_{r+1})$
$\mathbf{t}_s \coloneqq \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}'_1\mathbf{r}_s \in \mathcal{R}_q^{k_s},\ \mathbf{r}_s \leftarrow \mathcal{R}_2^{n_s}$
Send $\mathbf{t}_s$

**2. Approximate Norm Proof**

Receive $\mathbf{\Pi} \in \{-1, 0, 1\}^{256 \times d \sum_{i \in [r]} n_i}$
$\mathbf{z} \coloneqq b\mathbf{\Pi}(\mathbf{s}_1, \ldots, \mathbf{s}_r) + \mathbf{y} \in \mathbb{Z}_q^{256}$
if $(\mathtt{BimodRej}(\mathbf{z}, \mathbf{\Pi}(\mathbf{s}_1, \ldots, \mathbf{s}_r), b, \sigma) = \perp)$ abort
Send $\mathbf{z}$

**3. Aggregate the Constraints**

Receive $\mu_j^{(i)} \in \mathbb{Z}_q$
$h_i \coloneqq g_i + \sum_j \mu_j^{(i)} f'_j,\ i \in [\lambda]$
Send $(h_i)_{i \le \lambda}$
Receive $\nu_j \in \mathcal{R}_q$
$f \coloneqq \sum_j \nu_j f_j$
Compute $a_{ij}, b \in \mathcal{R}_q$ and $\varphi \in \mathcal{R}_q^{n_i}$ such that
$f = \sum_{i,j \in [r]} a_{ij}\langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i \in [r]} \langle \varphi_i, \mathbf{s}_i \rangle - b$

**4. Prove the Final Constraint**

$\mathbf{y}_1 \leftarrow D_{\sigma_1}^{\sum_{i \in [5]} n_i},\ \mathbf{y}_2 \leftarrow D_{\sigma_2}^{n_s}$
$\mathbf{w} \coloneqq \mathbf{A}_1\mathbf{y}_1 + \mathbf{A}'_1\mathbf{y}_2 \in \mathcal{R}_q^{k_s}$
Compute garbage terms $u', v' \in \mathcal{R}_q$
to make Eq. (32) hold
$u \coloneqq u' + \langle \varphi_{rand}, \mathbf{r}_s \rangle,\ v \coloneqq v' + \langle \varphi_{rand}, \mathbf{y}_2 \rangle$.
Send $(\mathbf{w}, u, v)$
Receive $c \in \mathcal{C}$
$\mathbf{z}_1 \coloneqq \mathbf{y}_1 + c \cdot \tilde{\mathbf{s}},\ \mathbf{z}_2 \coloneqq \mathbf{y}_2 + c \cdot \mathbf{r}_s$
if $(\mathtt{Rej}_1(\mathbf{z}_1, c\tilde{\mathbf{s}}, \sigma_1) = \perp) \vee (\mathtt{Rej}_2(\mathbf{z}_2, c\mathbf{r}_s, \sigma_2) = \perp)$ abort
Send $(\mathbf{z}_1, \mathbf{z}_2)$

**Fig. 9.** Prover of the LNP protocol [34].

Instance: constraints $f_i$, $f'_i$ as in (1) and global bound $\beta$

**1. Commit**

Receive $\mathbf{t}_s$

**2. Approximate Norm Proof**

Send $\mathbf{\Pi} \leftarrow \mathtt{Bin}_1^{256 \times d \sum_i n_i}$
Receive $\mathbf{z}$
**Check** $\|\mathbf{z}\| \le \sigma\sqrt{512}$

**3. Aggregate the Constraints**

Send $\mu_j^{(i)} \leftarrow \mathbb{Z}_q$
Receive $(h_j)$
**Check** $\mathtt{ct}(h_j) = 0,\ j \in [\lambda]$
Send $\nu_j \leftarrow \mathcal{R}_q$
Compute $a_{ij}, b \in \mathcal{R}_q$, $\varphi_i \in \mathcal{R}_q^{n_i}$ s.t.
$f = \sum_{i,j \in [r]} a_{ij}\langle \mathbf{s}_i, \mathbf{s}_j \rangle + \sum_{i \in [r]} \langle \varphi_i, \mathbf{s}_i \rangle - b$

**4. Prove the Final Constraint**

Receive $\mathbf{w}, u, v$
Send $c \leftarrow \mathcal{C}$
Receive $\mathbf{z}_1, \mathbf{z}_2$ and split $\mathbf{z}_1 = (z_{1,i})$
**Check** $\mathbf{z}_1, \mathbf{z}_2$ are small
**Check** $\mathbf{A}_1\mathbf{z}_1 + \mathbf{A}'_1\mathbf{z}_2 = c\mathbf{t}_s + \mathbf{w}$
**Check**

$$\sum_{i,j \in [r]} a_{ij}\langle \mathbf{z}_{1,i}, \mathbf{z}_{1,j} \rangle + c \cdot \langle \varphi, \mathbf{z}_1 \rangle + c^2 \cdot b = cu + v - \langle \varphi_{rand}, \mathbf{z}_2 \rangle \tag{32}$$

**Fig. 10.** Verifier of the LNP protocol [34].

Instance: LaBRADOS checks $f_i, f_i'$, bounds $\beta_i$ (Fig. 8)
Witness: LaBRADOS output vectors $\mathbf{s}_i \in \mathcal{R}^{n_i}, i \in [4]$

**1. Commit**

$\mathbf{s}_5 \coloneqq \sigma_{-1}(\mathbf{s}_4)$
$\mathbf{u}_i \leftarrow [-2^{k_i-1}, 2^{k_i-1}-1]^{256}, i \in [4]$
$\mathbf{y} \leftarrow D_\sigma^{256/d}$
$g_i \leftarrow \{x \in \mathcal{R}_q : \mathtt{ct}(x) = 0\}$, $i \in [\lambda], \lambda \coloneqq \lceil 128/\log q \rceil$
$\mathbf{s}_6 \coloneqq \mathtt{BinDecomp}((\mathbf{u}_i), y, (g_i)) \in \mathcal{R}^{n_6}$
$\tilde{\mathbf{s}} \coloneqq (\mathbf{s}_1, \ldots, \mathbf{s}_6)$
$\mathbf{t}_s \coloneqq \mathbf{A}_1\tilde{\mathbf{s}} + \mathbf{A}_1'\mathbf{r}_s \in \mathcal{R}_q^{k_s}$, $\mathbf{r}_s \leftarrow \mathcal{R}_2^{n_s}$
$\hat{\mathbf{t}}_s \coloneqq \mathtt{BinDecomp}(\mathbf{t}_s)$
Send $\mathbf{b}_1 \coloneqq \mathtt{OutCom}(\hat{\mathbf{t}}_s)$

**2. Approximate Norm Proof**

Receive $\mathbf{\Pi}_i \in \{-1, 0, 1\}^{256 \times dn_i}$
$\mathbf{\Pi}_i\mathbf{s}_i + \mathbf{u}_i = 2^{k_i}\mathbf{v}_i^{(1)} + \mathbf{v}_i^{(0)} \in \mathbb{Z}_q^{256}$
if $\|\mathbf{v}_i^{(1)}\|^2 > N$ abort
$\mathbf{v}_i^{(1)} \coloneqq \mathbf{w}_i - \mathbf{t}_i, i \in [4]$
$\tilde{\mathbf{v}}_1 \coloneqq (w_1, t_1, \ldots, w_4, t_4) \in \mathcal{R}^8$, $\tilde{\mathbf{v}}_2 \coloneqq \sigma_{-1}(\tilde{\mathbf{v}}_1)$
$\tilde{\mathbf{v}} \coloneqq (\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2)$
$\mathbf{t}_v \coloneqq \mathbf{A}_2\tilde{\mathbf{v}} + \mathbf{A}_2'\mathbf{r}_v \in \mathcal{R}_q^{k_v}, \mathbf{r}_v \leftarrow \mathcal{R}_2^{n_v}$
$\hat{\mathbf{v}} \coloneqq \mathtt{BinDecomp}(\mathbf{t}_v, (v_i^{(0)}))$
Send $\mathbf{b}_2 \coloneqq \mathtt{OutCom}(\hat{\mathbf{v}})$
Receive $\bar{\mathbf{\Pi}} \in \{-1, 0, 1\}^{256 \times 8d}$
$\mathbf{z} \coloneqq \bar{\mathbf{\Pi}}\tilde{\mathbf{v}}_1 + \mathbf{y}$
if $(\mathtt{Rej}_1(\mathbf{z}, \bar{\mathbf{\Pi}}\tilde{\mathbf{v}}_1, \sigma) = \perp)$ abort
Send $\mathbf{z}$

**3. Aggregate the Constraints**

Receive $\mu_j^{(i)} \in \mathbb{Z}_q$
$h_i \coloneqq g_i + \sum_j \mu_j^{(i)} f_j'$, $i \in [\lambda]$
$\hat{h} \coloneqq \mathtt{BinDecomp}((h_i))$
Send $\mathbf{b}_3 \coloneqq \mathtt{OutCom}(\hat{h})$
Receive $\nu_j \in \mathcal{R}_q$
$f \coloneqq \sum_j \nu_j f_j$

**4. Prove the Final Constraint**

$\mathbf{y}_1 \coloneqq (\mathbf{y}_1^s, \mathbf{y}_1^v) \leftarrow D_{\sigma_1}^{\sum_{i\in[6]} n_i+16}$,
$\mathbf{y}_2 \coloneqq (\mathbf{y}_2^s, \mathbf{y}_2^v) \leftarrow D_{\sigma_2}^{n_s+n_v}$
$\mathbf{w}_s \coloneqq \mathbf{A}_1\mathbf{y}_1^s + \mathbf{A}_1'\mathbf{y}_2^s \in \mathcal{R}_q^{k_s}$,
$\mathbf{w}_v \coloneqq \mathbf{A}_2\mathbf{y}_1^v + \mathbf{A}_2'\mathbf{y}_2^v \in \mathcal{R}_q^{k_v}$
Compute garbage terms $u', v' \in \mathcal{R}_q$
to make Eq. (7) hold
$u \coloneqq u' + \langle \varphi_{rand}, \mathbf{r}_s \rangle$, $v \coloneqq v' + \langle (\varphi_{rand}, 0), \mathbf{y}_2 \rangle$.
$\hat{w} \coloneqq \mathtt{BinDecomp}(\mathbf{w}_s, \mathbf{w}_v, u, v)$
Send $\mathbf{b}_4 \coloneqq \mathtt{OutCom}(\hat{w})$
Receive $c \in \mathcal{C}$
$\mathbf{z}_1 \coloneqq \mathbf{y}_1 + c \cdot (\tilde{\mathbf{s}}, \tilde{\mathbf{v}})$, $\mathbf{z}_2 \coloneqq \mathbf{y}_2 + c \cdot (\mathbf{r}_s, \mathbf{r}_v)$
if $(\mathtt{Rej}_1(\mathbf{z}_1, c(\tilde{\mathbf{s}}, \tilde{\mathbf{v}}), \sigma_1) = \perp) \vee$
$(\mathtt{Rej}_2(\mathbf{z}_2, c(\mathbf{r}_s, \mathbf{r}_v), \sigma_2) = \perp)$ abort
Send $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{x}_{\mathtt{bin}} \coloneqq (\hat{\mathbf{t}}_s, \hat{\mathbf{v}}, \hat{\mathbf{h}}, \hat{\mathbf{w}}))$

**Fig. 11.** Prover of the LNP-Lite protocol.

Instance: LaBRADOS checks $f_i, f_i'$, bounds $\beta_i$ (Fig. 8)

**1. Commit**

Receive $\mathbf{b}_1$

**2. Approximate Norm Proof**

Send $\mathbf{\Pi}_i \leftarrow \mathtt{Bin}_1^{256 \times dn_i}$, $i \in [4]$
Receive $\mathbf{b}_2$
Send $\bar{\mathbf{\Pi}} \leftarrow \mathtt{Bin}_1^{256 \times 8d}$
Receive $\mathbf{z}$
**Check** $\|\mathbf{z}\| \le \sigma\sqrt{2 \cdot 256}$

**3. Aggregate the Constraints**

Send $\mu_j^{(i)} \leftarrow \mathbb{Z}_q$
Receive $\mathbf{b}_3$
Send $\nu_j \leftarrow \mathcal{R}_q$
Compute $a, a_{ij} \in \mathcal{R}_q$, $\varphi \in \mathcal{R}_q^{\sum_{i\in[6]} n_i+16}$, $f_0$
from Eq. (4)

**4. Prove the Final Constraint**

Receive $\mathbf{b}_4$
Send $c \leftarrow \mathcal{C}$
Receive $\mathbf{z}_1, \mathbf{z}_2$ and split as in Eqs. (5) and (6)
Receive $\mathbf{x}_{\mathtt{bin}}$ and recompose into
$\mathbf{t}_s, \mathbf{t}_v, (v_i^{(0)}), (h_i), \mathbf{w}_s, \mathbf{w}_v, u, v$
**Check** $\mathbf{A}_1\mathbf{z}_1^s + \mathbf{A}_1'\mathbf{z}_2^s = \mathbf{w}_s + c\mathbf{t}_s$
**Check** $\mathbf{A}_2\mathbf{z}_1^v + \mathbf{A}_2'\mathbf{z}_2^v = \mathbf{w}_v + c\mathbf{t}_v$
**Check**

$$\sum_{i,j\in[5]} a_{ij}\langle \mathbf{z}_{1,i}^s, \mathbf{z}_{1,j}^s \rangle + a\langle \mathbf{z}_{1,1}^v, \mathbf{z}_{1,2}^v \rangle + c\langle \varphi, \mathbf{z}_1 \rangle + c^2 f_0 = cu + v - \langle (\varphi_{rand}, 0), \mathbf{z}_2 \rangle$$

**Check** $\mathtt{ct}(h_i) = 0$, $i \in [\lambda]$
**Check** $\|\mathbf{z}_{1,i}^s\| \le \sigma_1\sqrt{2 \cdot dn_i}$, $i \in [6]$
**Check** $\|\mathbf{z}_{1,1}^v\|, \|\mathbf{z}_{1,2}^v\| \le \sigma_1\sqrt{2 \cdot 8d}$
**Check** $\|\mathbf{z}_2\| \le \sigma_2\sqrt{2 \cdot d(n_s + n_v)}$
**Check** $\mathbf{x}_{\mathtt{bin}}$ is binary.
**Check** correctness of $\mathbf{b}_1, \ldots, \mathbf{b}_4$ w.r.t $\hat{\mathbf{t}}_s, \hat{\mathbf{v}}, \hat{\mathbf{h}}, \hat{\mathbf{w}}$

**Fig. 12.** Verifier of the LNP-Lite protocol.