Serval: Slack-Free ℓ_2-Sound Polynomial Commitments from Lattices

Lizhen ZHANG[1], Sherman S. M. Chow[2], Shang GAO[1], and Bin XIAO[1]

[1] The Hong Kong Polytechnic University, Hong Kong, China
lizh6.zhang@connect.polyu.hk, shanggao@polyu.edu.hk
b.xiao@polyu.edu.hk
[2] Department of Information Engineering
The Chinese University of Hong Kong
smchow@ie.cuhk.edu.hk

Abstract. Polynomial commitment schemes (PCSs) are a core primitive in modern proof systems, especially SNARKs. Lattice-based PCSs are a natural post-quantum alternative, but existing constructions typically enforce lattice smallness only up to a multiplicative slack in the extracted ℓ_2 norm. We present Serval, a lattice-based PCS with *slack-free* ℓ_2 soundness (norm-preserving extraction) under standard SIS/LWE-type assumptions. Following the norm-proof paradigm of Lyubashevsky *et al.* (Crypto '22), we reduce exact ℓ_2 enforcement to a self-inner-product argument plus a binary validation that avoids modular wraparound. Our main technical insight is a reduction that rewrites the binary check as inner-product-shaped constraints that match the algebraic form of the PCS evaluation and commitment-consistency checks. This alignment yields a unified divide-and-conquer protocol that proves all constraints simultaneously in $O(\log L)$ rounds, giving succinct evaluation proofs and efficient verification. At $L = 2^{20}$ and matched security parameters ($\lambda = 128$), our instantiated proof size is $8\times$ smaller than Fenzi *et al.* (JoC '24) and $85\times$ smaller than SLAP (Eurocrypt '24), while remaining compatible with the recent construction of Cini *et al.* (Crypto '24).

Keywords: Lattice-based cryptography, Polynomial commitments

1 Introduction

A polynomial commitment scheme (PCS) lets a prover commit to a polynomial f using a commitment $\mathsf{Com}(f)$, and later prove that a claimed value y equals $f(x)$ (or $f(\vec{x})$ in the multilinear case) using a proof π. We write L for the dimension of the coefficient representation: $L = D + 1$ for a degree-D univariate polynomial, and $L = 2^n$ for an n-variate multilinear polynomial.

PCSs are a standard primitive in verifiable secret sharing [5], secure multiparty computation [10], authenticated data structures [26], and zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) [8,14,17,16]. In particular, PCS verification is often a major cost driver in recursive and proof-carrying settings, motivating schemes with sublinear verification in L.

Many deployed PCSs are built from discrete-logarithm assumptions [29,13,27] and are vulnerable to quantum adversaries. Code-based alternatives [7,25] are post-quantum, but they typically do not provide algebraic structures that enable efficient composition (*e.g.*, linear combinations and batching of claims). Lattice-based PCSs, in contrast, offer post-quantum security together with algebraic structure, making them a strong candidate for composable proof systems.

A central obstacle is to achieve *norm-preserving extraction* (no norm slack) without sacrificing efficiency. [3] At a high level, lattice soundness reduces to extracting a *small-norm* witness for a relation of the form:

$$\mathcal{R} \coloneqq \left\{ (A, F, \mathcal{B}; \vec{f}) \colon A\vec{f} = F \ \wedge \ \|\vec{f}\|_2 \leq \mathcal{B} \right\}.$$

The linear constraint $A\vec{f} = F$ is easy to verify, but enforcing the concrete ℓ_2 bound is substantially harder. Many lattice proof systems guarantee only that extraction succeeds with a slack factor, namely $\|\vec{f}\|_2 \leq \alpha\mathcal{B}$ for some $\alpha > 1$.

This gap has been studied since early work on identification and proof protocols based on lattices [30,37]. A common approach is to shrink the challenge space to reduce slack, but this increases the soundness error and forces many repetitions. As a result, proof sizes become large, and while ring or module structures improve constants, they do not remove the basic tension between slack, challenge space, and efficiency.

A different line of work targets *tight* ℓ_∞ bounds via algebraic identities. Representative works [21,20,4] transform element-wise constraints such as $\vec{f} \in \{0, \pm 1\}^n$ into the polynomial identity $\vec{f} \circ (\vec{f} - \vec{1}) \circ (\vec{f} + \vec{1}) = \vec{0}$, and verify it using techniques like Chinese remainder theorem (CRT) or number theoretic transform (NTT). These decompositions achieve slack-free ℓ_∞ soundness for small alphabets (*e.g.*, $\{0, \pm 1\}$), but their prover typically performs quasi-linear work in the vector length due to the required structured transforms. More importantly, a tight ℓ_∞ bound is often not the right target for lattice-based proof systems. Many lattice primitives and reductions naturally control $\|\cdot\|_2$ (*e.g.*, via Gaussian sampling or noise growth), and enforcing a stringent $\|\cdot\|_\infty$ bound can force conservative parameters (*e.g.*, larger moduli) while typically not strengthening the intended ℓ_2-based hardness statement.

More recent work targets ℓ_2 norm proofs directly. Lyubashevsky [33] decomposes an ℓ_2 constraint into a quadratic inner-product relation together with an auxiliary linear bound, and proves both with linear prover time, proof size, and verification time. Beullens *et al.* [9] further reduces the resulting ℓ_2-soundness slack to a constant factor of 2 using a modular Johnson–Lindenstrauss argument.

Despite these advances, integrating norm proofs into *structured* lattice proof systems remains delicate, especially for lattice-based PCSs. A PCS must support fast evaluation proofs and preserve commitment reuse, batching, and aggregation. Naive norm-proof integration can break one or more of these properties. As a result, existing lattice-based PCSs commonly handle lattice smallness using

[3] We target this guarantee under standard SIS/LWE-type assumptions, rather than k-SIS-style variants or trapdoor-dependent auxiliary assumptions.

small challenge spaces and a controlled slack factor, while concentrating optimization on the evaluation-proof component (see Appendix A).

This leaves a concrete gap: incorporate (exact) norm proofs into lattice-based PCSs while retaining sublinear evaluation verification and the structural advantages needed for composition. Our work addresses this gap.

1.1 Related Work

Recent lattice-based PCSs have achieved sublinear (often polylogarithmic) verification for evaluation proofs using either inner-product arguments or recursive folding. Inner-product-based schemes encode evaluation as an inner product and prove an IPA together with commitment consistency [2,15,38,18]. They typically obtain logarithmic proof size, but still require linear online work for at least one party. Some works further reduce online verification via preprocessing or delegation, at the cost of auxiliary keys/tables or additional setup complexity [11,23,29]. Recursive (split-and-fold) approaches verify evaluation through a recursion tree, yielding a polylogarithmic object checked by the verifier [7,22,3]. Hybrid designs combine folding with proof-of-proof compaction, exemplified by Greyhound [36] and by transparent SIS-based schemes using FRI-style folding [19,39].

Orthogonally, norm proofs introduce an additional soundness axis. Tight coefficient-set checks can be enforced via algebraic identities and structured transforms [21,20,4]. For ℓ_2 bounds, more recent techniques reduce the soundness slack to a small constant (notably 2) via modular Johnson–Lindenstrauss arguments, and have been incorporated into compact proof systems such as LaBRADOR [9]. In lattice-based PCSs, this norm axis is commonly handled by tolerating slack so that evaluation proofs retain sublinear structure. Achieving slack-free ℓ_2 soundness under standard SIS/LWE-type assumptions while preserving batching, aggregation, and sublinear verification remains less explored.

Serval addresses this gap by integrating slack-free ℓ_2 enforcement into a structured PCS framework without sacrificing succinct evaluation proofs and efficient verification. We provide a detailed comparison in Appendix A. Table 1 summarizes representative post-quantum PCSs along the axes most relevant to this work (setup, norm slack, and concrete proof size at matched security).

1.2 Our Results

We present a lattice-based PCS for univariate and multilinear polynomials that supports succinct evaluation proofs and enforces a slack-free ℓ_2 norm bound for the extracted witness.

At a high level, we cast evaluation, commitment consistency, and range constraints into arithmetic relations and reduce them to inner-product relations. For evaluation and commitment consistency, the inner product is taken with a public vector, while for range constraints it is taken with a private witness. Our construction follows a divide-and-conquer paradigm that treats all constraints uniformly. For a fixed security level, the proof size and verification time are

Table 1. Comparison among post-quantum polynomial-commitment schemes: All sizes report prover-to-verifier communication only (in bytes) at 128-bit security, following the accounting in the cited sources.

Work	**Basis**	**Slack**	**T.S.**	$L = 2^{15}$	$L = 2^{20}$	$L = 2^{25}$	$L = 2^{30}$
[25]	Hash	N/A	✓	5.32MB	10.25MB†	50.34MB	186.31MB†
[39]	Hash	N/A	✓	–	6.11MB	–	–
[22]	BASIS	poly(L)	×	–	3.4MB	–	8.3MB
[3]	M-SIS	poly(L)	×	–	36.5MB	–	767MB
[36]	M-SIS	poly(L)	✓	–	–	46KB	53KB
[19]	M-SIS	≈2	✓	120KB	501KB	1.51MB	5.17MB
Ours	M-SIS	1	✓	259.16KB	436.03KB	2.56MB	3.59MB

[1] **Basis**: hash/code-based, or lattice-based; M-SIS is standard (Module-)SIS, while BASIS is a non-standard SIS-with-hints assumption [38].
[2] **Slack**: the multiplicative factor α in the stated extraction/binding guarantee for the lattice object that must be short for the security argument, *i.e.*, $\|\tilde{\vec{f}}\|_2 \le \alpha\mathcal{B}$ (or the paper's analogous statement). It is N/A for non-lattice schemes, poly(L) when α grows with L, ≈2 for constant-factor slack, and 1 for slack-free extraction.
[3] **T.S.**: whether the scheme supports a transparent setup.
[4] **Sources**: Brakedown from [25, Fig. 8] (entries marked by † are interpolated from adjacent reported points); Basefold from [39, Tab. 2]; Greyhound from [36, Tab. 1]; Fenzi *et al.*, SLAP, and Cini *et al.* from the comparison in [19, Fig. 2].
[5] **Accounting**: "–" indicates missing data.

polylogarithmic in the size parameter L (measured in ring operations), and the prover runs in time linear in the witness length.

Efficient evaluation and commitment-consistency proofs. We follow an inner-product-based approach for evaluation and use a leveled lattice-based vector commitment [12] for the coefficient vector. Both the evaluation vector and the leveled commitment admit compatible tensor structures. This enables a $\log L$-round split-and-fold protocol without preprocessing or CRS generation.

Slack-free ℓ_2-norm enforcement via binary decomposition. Our norm enforcement follows the paradigm of Lyubashevsky *et al.* [33]. We prove the ℓ_2-norm constraint by proving (i) a quadratic self-inner-product relation with (ii) a coefficient bound strong enough to lift the modular statement to an integer statement. Concretely, for a claimed small norm $\vec{f} \in \mathbb{Z}_q^{L\cdot\ell}$, we prove a statement of the form (i) $\langle \vec{f}, \vec{f} \rangle \equiv b \pmod q$ with a range constraint $0 \le b \le B^2$ for a public bound B, and (ii) a bound $\|\vec{f}\|_\infty \le U$ such that $L \cdot \ell \cdot U^2 < q/2$. These imply $\|\vec{f}\|_2^2 \le L \cdot \ell \cdot \|\vec{f}\|_\infty^2 \le L \cdot \ell \cdot U^2 < q/2$, so the integer value $\langle \vec{f}, \vec{f} \rangle$ equals its residue modulo q. Thus, the bound $0 \le b \le B^2$ yields the desired slack-free guarantee $\|\vec{f}\|_2 \le B$. Accordingly, the extractor's output satisfies the *same*

bound B stated in the relation (no multiplicative loss), provided the coefficient bound ensures $\langle \vec{f}, \vec{f} \rangle \in [0, q/2)$ so that the centered representative coincides with the integer squared norm.

In our construction, the coefficient vector is constrained to be binary. This allows us to guarantee the ℓ_∞-norm bound by directly proving the binary property, rather than relying on an approximate ℓ_∞ bound as Lyubashevsky *et al.* [33].

Accordingly, for a large-norm vector over $\mathbb{Z}_q^L$, the decomposition length is set to $\ell = \lceil \log q \rceil$ via an unsigned base-2 decomposition. This is a standard design choice in various lattice-based primitives, most notably the GSW cryptosystem [24]. We note, however, that compared to schemes such as Greyhound [36] which employ a larger base, this choice increases the witness size by a constant factor, thereby marginally raising the prover's computational overhead in practice. Nonetheless, the asymptotic complexity remains unaffected, and this binary decomposition provides significant structural advantages for both the verification process and the efficiency of proof construction.

To prove the inner-product relation, we use a standard divide-and-conquer protocol that reduces a length-$L \cdot \ell$ inner product to length-$L \cdot \ell/2$ recursively, resulting in $\log L$ rounds. To prove binarity, we start from the arithmetic condition $\vec{f} \circ (\vec{f} - \vec{1}) = \vec{0}$ and reduce it to a single inner-product consistency check: for verifier-sampled $\vec{\alpha}$, $\langle \vec{f} \circ \vec{\alpha}, \vec{f} \rangle = \langle \vec{f}, \vec{\alpha} \rangle$. Crucially, this relation has the same algebraic shape as the self-IPA and the evaluation/commitment constraints, so it can be handled by the same split-and-fold machinery in $O(\log L)$ rounds.

Working over the modular ring. Our protocol is defined over a modular ring $\mathbb{R}_q$. Embedding coefficient constraints over $\mathbb{Z}_q$ into $\mathbb{R}_q$ is straightforward via standard coefficient mappings and ring automorphisms. The ring setting also admits large challenge spaces (*e.g.*, of size about 2^λ with appropriate parameters), which reduces the soundness error.

Concrete efficiency. After applying our optimizations, the resulting PCS achieves $O(\log L)$ proof size and $O(\log L)$ verification complexity for a fixed security level (measured in ring elements/operations). Table 1 summarizes the concrete comparison at 128-bit security along the axes most relevant to this work (setup, norm slack, and proof size). At $L = 2^{20}$, our instantiation is smaller than the reported proof sizes of Fenzi *et al.* [22] and SLAP [3], and remains compatible with the transparent SIS-based construction of Cini *et al.* [19]. Compared with Greyhound [36], our proof size is larger due to the structure limit and additional cost of exact range enforcement, while our verifier is faster at larger L as shown in Section 5.2.

1.3 Technique Overview

Let $\mathbb{R}_q \coloneqq \mathbb{Z}_q[X]/(X^d + 1)$ be the power-of-two cyclotomic ring of dimension d. We write $\sigma_{-1} \colon \mathbb{R}_q \to \mathbb{R}_q$ for the ring automorphism induced by $\sigma_{-1}(X) = X^{-1}$

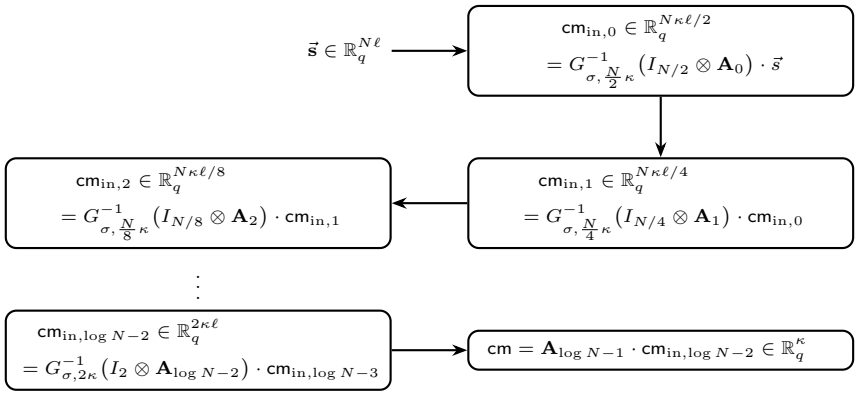


Fig. 1. A $\log N$-level Ajtai commitment scheme instantiated following [12] (G^{-1} denotes the decomposition operation)

in the quotient, $\mathsf{ct}(\cdot)$ for the constant-term map and $\mathsf{coef}(\cdot)$ for the coefficient vector over $\mathbb{Z}_q$ of a ring element. We write N for the PCS size parameter (with $N \cdot d = L$ in Section 1) and ℓ for the decomposition expansion factor.

A unified relation. We reduce both univariate and multilinear PCS in the lattice setting to the task of proving that a witness $\vec{\mathbf{s}} \in \mathbb{R}_q^{N\ell}$ satisfies the following four constraints: (i) The commitment constraint fixes the committed value: $\mathsf{Com}(\vec{\mathbf{s}}) = \mathsf{cm}$; (ii) The evaluation consistency constraint, which is a tensor-structured linear relation between $\mathsf{coef}(\vec{\mathbf{s}})$ and public vectors $(\vec{a}_i)$, yielding a public value v (Section 4); (iii) A quadratic self-inner-product condition: $\mathsf{ct}\big(\langle \vec{\mathbf{s}}, \sigma_{-1}(\vec{\mathbf{s}})\rangle\big) = b \bmod q$ together with a range bound $b \le \beta^2$; and (iv) The coefficient constraint enforces binarity: $\mathsf{coef}(\vec{\mathbf{s}}) \in \{0,1\}^{N\ell d}$. Lemma 2 shows that the quadratic constraint together with binarity implies a slack-free ℓ_2 bound by ruling out modular wraparound.

We instantiate Com using the $\log N$-level Ajtai commitment [12] (Figure 1), and the prover maintains intermediate commitments as private state.

One split-and-fold engine for all constraints. Each of the above constraints admits the same divide-and-conquer structure. At a high level, we repeatedly split the witness (along with private states) into halves, send constant-size auxiliary claims for the two halves, and fold them using a verifier challenge. After $\log N$ rounds, verification reduces to checking a base instance of constant size. We use this engine for commitment consistency, the tensor-structured linear constraint, and the quadratic self-inner-product constraint.

Binarity as an inner-product-shaped constraint. Let $\vec{s} := \mathsf{coef}(\vec{\mathbf{s}})$. Binarity is equivalent to the component-wise equation $\vec{s} \circ (\vec{s} - \vec{1}) = \vec{0}$. We reduce it to a single randomized inner-product consistency check: for a verifier-sampled $\vec{\alpha}$,

$\langle \vec{s} \circ \vec{\alpha}, \vec{s} \rangle = \langle \vec{s}, \vec{\alpha} \rangle$. Crucially, after embedding into $\mathbb{R}_q$ via the standard coefficient mapping, this check has the same algebraic shape as the linear and quadratic constraints, and is handled by the same split-and-fold engine.

Proof-size optimization via Fiat–Shamir and LaBRADOR. The protocol above has $\log N$ rounds and produces per-round constant-size subclaims. We apply the Fiat–Shamir transform in the random-oracle model by deriving each verifier challenge from the transcript-so-far. We then apply LaBRADOR to compress the resulting log-round transcript into a succinct proof. The final proof comprises this succinct proof along with a short reconstruction seed (see Section 5.1).

2 Preliminaries

We review the essential preliminaries. Further background and auxiliary technical material are deferred to Appendix B.

2.1 Notation

Let λ be the security parameter, implicitly given to all algorithms. All parameters are bounded by $\mathsf{poly}(\lambda)$ unless explicitly defined. A function $\mathsf{negl}(\lambda)$ is negligible. For $n \in \mathbb{N}$, define $[n] \coloneqq \{0, 1, \ldots, n-1\}$. For integers $a < b$, define $[a \colon b] \coloneqq \{a, a+1, \ldots, b-1\}$. Let q be a prime modulus. Let $\mathbb{Z}$ be the set of integers. Define $\mathbb{Z}_q \coloneqq \mathbb{Z}/q\mathbb{Z}$. When taking absolute values and norms, we identify each $x \in \mathbb{Z}_q$ with its centered representative in $\{-(q-1)/2, \ldots, (q-1)/2\}$. Vectors in $\mathbb{Z}_q^n$ are written as $\vec{f} = (f_0, \ldots, f_{n-1})$. Matrices over $\mathbb{Z}_q$ are denoted by uppercase letters. Let I_m be the identity matrix of size $m \times m$. Let $\otimes$ be the Kronecker product.

Let $\mathbb{Z}_q[X]$ be the ring of polynomials over $\mathbb{Z}_q$, and let d be a power of two. Define $\mathbb{R} \coloneqq \mathbb{Z}[X]/(X^d + 1)$ and $\mathbb{R}_q \coloneqq \mathbb{Z}_q[X]/(X^d + 1)$. We reserve the notation $\mathcal{R}$ for relations. Ring elements are in bold (*e.g.*, $\mathbf{f} \in \mathbb{R}_q$), except for challenges $c \in \mathcal{C}$. For $\mathbf{f} \coloneqq \sum_{i=0}^{d-1} f_i X^i \in \mathbb{R}_q$, define $\mathsf{ct}(\mathbf{f}) \coloneqq f_0$ and $\mathsf{coef}(\mathbf{f}) \coloneqq \vec{f} = (f_0, f_1, \ldots, f_{d-1})$. Bold uppercase letters denote matrices over $\mathbb{R}_q$. For size-n vectors $\vec{\mathbf{f}}$ and $\vec{\mathbf{g}}$, their concatenation is $(\vec{\mathbf{f}}, \vec{\mathbf{g}}) \in \mathbb{R}_q^{2n}$. For matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}_q^{m \times n}$, we write $[\mathbf{A}\ \mathbf{B}] \in \mathbb{R}_q^{m \times 2n}$. For a vector $\vec{\mathbf{a}} \in \mathbb{R}_q^n$, the i-th entry is denoted as $\mathbf{a}_i$ or $\mathbf{a}[i]$.

Norms. For $\mathbf{f} \coloneqq \sum_{i=0}^{d-1} f_i X^i \in \mathbb{R}_q$, we define $\|\mathbf{f}\|_1 \coloneqq \sum_{i=0}^{d-1} |f_i|$, $\|\mathbf{f}\|_2 \coloneqq \sqrt{\sum_{i=0}^{d-1} f_i^2}$, and $\|\mathbf{f}\|_\infty \coloneqq \max_{i \in [d]} |f_i|$. For $\vec{\mathbf{f}} \coloneqq (\mathbf{f}_0, \ldots, \mathbf{f}_{m-1}) \in \mathbb{R}_q^m$, define $\|\vec{\mathbf{f}}\|_1 \coloneqq \sum_{i=0}^{m-1} \|\mathbf{f}_i\|_1$, $\|\vec{\mathbf{f}}\|_2 \coloneqq \sqrt{\sum_{i=0}^{m-1} \|\mathbf{f}_i\|_2^2}$, and $\|\vec{\mathbf{f}}\|_\infty \coloneqq \max_{i \in [m]} \|\mathbf{f}_i\|_\infty$. The ℓ_∞-norm and ℓ_2-norm of $\vec{\mathbf{f}}$ satisfy $\|\vec{\mathbf{f}}\|_\infty \le \|\vec{\mathbf{f}}\|_2 \le \sqrt{md}\, \|\vec{\mathbf{f}}\|_\infty$. Unless stated otherwise, $\|\cdot\|$ refers to the ℓ_2-norm.

Challenge space. We call $\mathcal{C} \subset \mathbb{R}_q$ a challenge space if, for all distinct $c, c' \in \mathcal{C}$, the difference $c - c'$ is invertible in $\mathbb{R}_q$. We also require bounds $\|c\| \leq \tau$ and $\|c\|_{\mathsf{op},\infty} \leq T$ for all $c \in \mathcal{C}$, where $\tau, T \in \mathbb{Z}$. Here, $\|c\|_{\mathsf{op},\infty} \coloneqq \sup_{\mathbf{v} \in \mathbb{R}_q \setminus \{0\}} \frac{\|c\mathbf{v}\|_\infty}{\|\mathbf{v}\|_\infty}$. In our instantiation, $\mathcal{C}$ is chosen so that $c - c'$ falls in the invertible regime of Lemma 1 for all distinct $c, c' \in \mathcal{C}$.

Lemma 1 (Invertibility in Polynomial Rings [34]). *Let $d > 1$ be a power of two, and let q be a prime with $q \equiv 5 \bmod 8$. Then any $\mathbf{f} \in \mathbb{R}_q$ with $0 < \|\mathbf{f}\|_2 < \sqrt{q/2}$ is invertible in $\mathbb{R}_q$.*

Gadget-matrix decomposition. Let $a \in \mathbb{N}$. Define the gadget vector $\vec{g}_a \coloneqq (1, a, a^2, \ldots, a^{\iota-1})$, where $\iota \coloneqq \lceil \log_a q \rceil$. Define the gadget matrix $G_{a,m} \coloneqq I_m \otimes \vec{g}_a^{\mathsf{T}} \in \mathbb{Z}_q^{m \times m \cdot \iota}$. We use $G_{a,m}^{-1}(\cdot)$ as notation for the (base-a) gadget decomposition operator, not a matrix inverse. For $\mathbf{A} \in \mathbb{R}_q^{m \times n}$, define $\tilde{\mathbf{A}} \coloneqq G_{a,m}^{-1}(\mathbf{A}) \in \mathbb{R}_q^{\iota m \times n}$ (applied coefficient-wise over $\mathbb{R}_q$) such that $\mathbf{A} = G_{a,m}\tilde{\mathbf{A}}$. We write G_m^{-1} when the basis a is clear.

Conjugation automorphism. Let $\sigma_{-1} \colon \mathbb{R}_q \to \mathbb{R}_q$ be the conjugation automorphism induced by $\sigma_{-1}(X) = X^{-1}$ in the quotient $\mathbb{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$. Equivalently, for $\mathbf{f} = \sum_{i=0}^{d-1} f_i X^i \in \mathbb{R}_q$, $\sigma_{-1}(\mathbf{f}) = \sum_{i=0}^{d-1} f_i X^{-i}$, where exponents are interpreted modulo $X^d + 1$. Let $\vec{a}, \vec{b} \in \mathbb{Z}_q^{nd}$ be coefficient representations of $\vec{\mathbf{a}}, \vec{\mathbf{b}} \in \mathbb{R}_q^n$. The coefficient inner product satisfies $\langle \vec{a}, \vec{b} \rangle = \mathsf{ct}\Big(\Big\langle \sigma_{-1}(\vec{\mathbf{a}}), \vec{\mathbf{b}} \Big\rangle\Big)$.

Exact ℓ_2-norm constraint decomposition. An exact ℓ_2-norm constraint can be decomposed into the 2 sub-constraints as defined below.

Lemma 2. *When $q > 2N\ell d$, enforcing a slack-free ℓ_2-norm bound on $\vec{\mathbf{f}} \in \mathbb{R}_q^{N\ell}$ reduces to the following two constraints, ignoring zero-knowledge aspects:*

- *A self-inner-product constraint in the ring setting: $\mathsf{ct}\Big(\langle \vec{\mathbf{f}}, \sigma_{-1}(\vec{\mathbf{f}}) \rangle\Big) = b \bmod q$ and $0 \leq b \leq B^2$, where B is the prescribed valid norm bound.*
- *A bit constraint on all coefficients: $\mathsf{coef}(\vec{\mathbf{f}}) \in \{0,1\}^{N\ell d}$.*

Proof. Under the bit constraint, $\|\vec{\mathbf{f}}\|_\infty = 1$, so $|\vec{\mathbf{f}}|_2^2 \leq N\ell d$. For $q > 2N\ell d$, the centered representative of $\mathsf{ct}\Big(\langle \vec{\mathbf{f}}, \sigma_{-1}(\vec{\mathbf{f}}) \rangle\Big) \in \mathbb{Z}_q$ equals its integer value $|\vec{\mathbf{f}}|_2^2$. Hence, proving $b \leq B^2$ suffices to ensure $|\vec{\mathbf{f}}|_2^2 \leq B^2$. □

2.2 Lattice Problems and Leveled (Ajtai) Commitments

We recall the standard module-SIS (M-SIS) assumption [28].

Definition 1 ($\mathsf{M\text{-}SIS}_{\kappa,n,q,\beta}$). *Given a random matrix $\mathbf{A} \in \mathbb{R}_q^{\kappa \times n}$ for $n > \kappa$, M-SIS is to find a non-zero $\vec{\mathbf{z}} \in \mathbb{R}_q^n$ such that $\mathbf{A}\vec{\mathbf{z}} = \vec{\mathbf{0}}$ over $\mathbb{R}_q$ and $\|\vec{\mathbf{z}}\| \leq \beta$.*

Definition 2 (Ajtai Commitment [1]). *Let κ, q, β be positive integers. For messages that are an n-dimensional short vector $\vec{\mathbf{f}} \in \mathbb{R}_q^n$ with $\|\vec{\mathbf{f}}\| \leq \beta$:*

- $\mathsf{Setup}(1^\lambda)$*: Sample a matrix* $\mathbf{A} \leftarrow \mathbb{R}_q^{\kappa \times n}$*. Output* $\mathsf{pp} \coloneqq \mathbf{A}$*.*
- $\mathsf{Commit}(\mathsf{pp}, \vec{\mathbf{f}})$*: Output* $\mathsf{cm} \coloneqq \mathbf{A}\vec{\mathbf{f}}$*.*

A commitment scheme is binding if it is computationally infeasible to find two distinct openings for the same commitment, and hiding if it reveals no information about the committed message. Ajtai commitment is binding for openings with $\|\vec{\mathbf{f}}\| \leq \beta$ under the $\mathsf{M\text{-}SIS}_{\kappa,n,q,2\beta}$ assumption. Hiding can be achieved by appending a small random vector to the message, and throughout our protocols we treat this randomness as part of the witness.

2.3 LaBRADOR Relation

We recall the LaBRADOR relation [9]:

$$\mathcal{R}_{\mathsf{Lab}} = \left\{ ((\mathcal{F}, \mathcal{F}', \beta), (\vec{\mathbf{z}}_0, \ldots, \vec{\mathbf{z}}_{r-1})) \;\middle|\; \begin{array}{l} f(\vec{\mathbf{z}}_0, \ldots, \vec{\mathbf{z}}_{r-1}) = 0 \quad \forall f \in \mathcal{F}, \\ \mathsf{ct}(f'(\vec{\mathbf{z}}_0, \ldots, \vec{\mathbf{z}}_{r-1})) = 0 \quad \forall f' \in \mathcal{F}', \\ \sum_{i=0}^{r-1} \|\vec{\mathbf{z}}_i\|_2^2 \leq \beta^2 \end{array} \right\}.$$

Here, $\mathcal{F}$ is a family of quadratic dot product functions $f \colon \mathbb{R}_q^n \times \cdots \times \mathbb{R}_q^n \to \mathbb{R}_q$ of the form

$$f(\vec{\mathbf{z}}_0, \ldots, \vec{\mathbf{z}}_{r-1}) = \sum_{i,j=0}^{r-1} \alpha_{i,j} \langle \vec{\mathbf{z}}_i, \vec{\mathbf{z}}_j \rangle + \sum_{i=0}^{r-1} \langle \vec{\varphi}_i, \vec{\mathbf{z}}_i \rangle - \mathbf{b}, \tag{1}$$

where $\alpha_{i,j} \in \mathbb{R}_q$, $\mathbf{b} \in \mathbb{R}_q$, and $\vec{\varphi}_i \in \mathbb{R}_q^n$ for $n, r \geq 0$. The family $\mathcal{F}'$ contains functions f' of the same form as in Eq. 1, and the relation constrains only their constant terms via $\mathsf{ct}(\cdot)$.

The proof size of LaBRADOR is $O(\log \log N)$ for $N = n \cdot r$, where $r = N^{1/3}$ and $n = N^{2/3}$. It supports quasi-linear verification and proving time [9].

3 Serval: Efficient Proofs for Constructing PCS

3.1 Generalized Relation

We consider the relation $\mathcal{R}_{\mathsf{main}}$ below using the notation in Section 2.1. We show in Section 4 that both univariate and multilinear polynomial evaluation proofs in the lattice setting can be reduced to $\mathcal{R}_{\mathsf{main}}$.

$$\mathcal{R}_{\mathsf{main}} = \left\{ \begin{array}{l} (\mathsf{cm} \in \mathbb{R}_q^\kappa, \beta \in \mathbb{Z}_q), \\ (v, b \in \mathbb{Z}_q, (\vec{a}_i \in \mathbb{Z}_q^{b_i})_{i \in [\log N]}), \\ (\mathsf{cm}_{\mathrm{in},i} \in \mathbb{R}_q^{\frac{N}{2^{i+1}} \kappa \iota})_{i \in [\log N - 1]}, \\ (\vec{\mathbf{s}} \in \mathbb{R}_q^{N\ell}) \end{array} \;\middle|\; \begin{array}{l} \mathsf{Com}(\vec{\mathbf{s}}) = \mathsf{cm}, \\ \langle \mathsf{coef}(\vec{\mathbf{s}}), \bigotimes_{i=0}^{\log N - 1} \vec{a}_i \rangle = v, \\ \mathsf{ct}\Big(\langle \vec{\mathbf{s}}, \sigma_{-1}(\vec{\mathbf{s}}) \rangle\Big) \bmod q = b \leq \beta^2, \\ \mathsf{coef}(\vec{\mathbf{s}}) \in \{0,1\}^{N\ell d} \end{array} \right\}. \tag{2}$$

Here, we set $b_0 \coloneqq 2\ell d$ and $b_i \coloneqq 2$ for all $i \in [1 \colon \log N]$.

3.2 Core Protocol

We present the commitment, linear, and quadratic checks separately to describe their local claims.

Commitments. We use the leveled Ajtai commitment scheme [12] for the witness in $\mathcal{R}_{\mathsf{main}}$. Here, "leveled" refers to a tree-structured composition of linear maps together with gadget decomposition.

Concretely, let $\vec{\mathbf{s}} \in \mathbb{R}_q^{N\ell}$ be a short-norm witness and let $N = \prod_{i=0}^{k-1} m_i$. Let $\mathbf{A}_0 \in \mathbb{R}_q^{\kappa \times m_0 \ell}, \mathbf{A}_i \in \mathbb{R}_q^{\kappa \times m_i \kappa \iota}$ for $i \in [1 \colon k-1]$ be public matrices, where $\iota = \lceil \log_\sigma q \rceil$ for a decomposition base σ. We write $G_{\sigma,m}$ for the gadget matrix from Section 2.1, and we use $G^{-1}_{\sigma,m}(\cdot)$ for the standard gadget decomposition applied entry-wise. Define $n_i \coloneqq \prod_{j=i}^{k-1} m_j$ for $i \in [k]$. The commitment to $\vec{\mathbf{s}}$ is

$$\mathsf{cm} = \mathbf{A}_{k-1} \cdot G^{-1}_{\sigma,n_{k-1}\kappa}\Big(\cdots G^{-1}_{\sigma,n_2\kappa}\big((I_{n_2} \otimes \mathbf{A}_1) \cdot G^{-1}_{\sigma,n_1\kappa}\big((I_{n_1} \otimes \mathbf{A}_0)\,\vec{\mathbf{s}}\big)\big)\Big).$$

This structured composition yields a compact commitment and is widely used in lattice proof systems. For example, LaBRADOR [9] and Greyhound [36] employ a two-level construction (*i.e.*, $k = 2$), while Cini *et al.* [19] adopt a $\log N$-level variant with the additional constraint that all matrices $\mathbf{A}_i$ are identical.

In our protocol, we fix $N = 2^k$ and therefore have $k = \log N$. For every level $i \in [1 \colon \log N]$, we use matrices $\mathbf{A}_i \in \mathbb{R}_q^{\kappa \times 2\kappa\iota}$, together with a base-level matrix $\mathbf{A}_0 \in \mathbb{R}_q^{\kappa \times 2\ell}$. The prover maintains a private state consisting of all intermediate commitments $\mathsf{cm}_{\mathrm{in},i}$ for $i \in [\log N - 1]$, defined by

$$\mathsf{cm}_{\mathrm{in},i} \coloneqq G^{-1}_{\sigma,n_{i+1}\kappa}\Big((I_{n_{i+1}} \otimes \mathbf{A}_i)\, G^{-1}_{\sigma,n_i\kappa}\big(\cdots G^{-1}_{\sigma,n_1\kappa}\big((I_{n_1} \otimes \mathbf{A}_0)\,\vec{\mathbf{s}}\big)\big)\Big),$$

where $n_{i+1} \coloneqq N/2^{\,i+1}$. These intermediate values satisfy the recurrence relation

$$\mathsf{cm}_{\mathrm{in},i+1} = G^{-1}_{\sigma,n_{i+2}\kappa}\Big((I_{n_{i+2}} \otimes \mathbf{A}_{i+1})\,\mathsf{cm}_{\mathrm{in},i}\Big), \qquad i \in [0 \colon \log N - 3],$$

and the final commitment satisfies $\mathbf{A}_{\log N-1} \cdot \mathsf{cm}_{\mathrm{in},\log N-2} = \mathsf{cm}$.

Commitment Constraint Proof. The commitment consistency constraint in $\mathcal{R}_{\mathsf{main}}$ can be verified recursively by exploiting the leveled structure and the algebraic relations between intermediate commitments.

The top-level relation is

$$\mathbf{A}_{\log N-1} \cdot \mathsf{cm}_{\mathrm{in},\log N-2} = \mathsf{cm}. \tag{3}$$

The prover sends the pair $(\mathsf{cm}_{1,L}, \mathsf{cm}_{1,R}) \coloneqq \mathsf{cm}_{\mathrm{in},\log N-2} \in \mathbb{R}_q^{2\kappa\iota}$. The verifier checks Equation (3).

The verifier samples a random challenge $\vec{c}_1 = (c_{1,0}, c_{1,1}) \in \mathcal{C}^2$. The prover folds the witness $\vec{\mathbf{s}} = (\vec{\mathbf{s}}_L \,\|\, \vec{\mathbf{s}}_R)$ as $\vec{\mathbf{s}}_2 \coloneqq c_{1,0}\,\vec{\mathbf{s}}_L + c_{1,1}\,\vec{\mathbf{s}}_R$, and folds each stored intermediate commitment entry-wise in the same manner. The prover then sends

the next pair $(\mathsf{cm}_{2,L}, \mathsf{cm}_{2,R})$, corresponding to the folded intermediate commitment one level below.

The verifier checks the folding step by using the defining recurrence of the leveled commitment. In particular, letting $G_{\sigma,\kappa}$ be the gadget matrix from Section 2.1, it checks $\mathbf{A}_{\log N-2} \cdot \pi_2^{(\mathsf{cm})} \stackrel{?}{=} c_{1,0} \left(G_{\sigma,\kappa} \cdot \mathsf{cm}_{1,L}\right) + c_{1,1} \left(G_{\sigma,\kappa} \cdot \mathsf{cm}_{1,R}\right)$, where $\pi_2^{(\mathsf{cm})} \coloneqq (\mathsf{cm}_{2,L}, \mathsf{cm}_{2,R}) \in \mathbb{R}_q^{2\kappa\iota}$ denotes the updated message sent by the prover. The verification proceeds recursively for $\log N$ rounds. Each round reduces the commitment level by one and halves the witness length along the N axis.

After $\log N$ rounds, the prover obtains a folded witness $\vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$. The prover sends $\vec{\mathbf{s}}_{\log N}$ to the verifier. The verifier checks that $\vec{\mathbf{s}}_{\log N}$ correctly opens the final folded commitment obtained through the recursion.

Overall, the proof transcript is $\pi^{(\mathsf{cm})} = \left(\pi_1^{(\mathsf{cm})}, \ldots, \pi_{\log N}^{(\mathsf{cm})}\right)$, where $\pi_i^{(\mathsf{cm})} = (\mathsf{cm}_{i,L}, \mathsf{cm}_{i,R}) \in \mathbb{R}_q^{2\kappa\iota}$ for $i \in [1\colon \log N - 1]$ and $\pi_{\log N}^{(\mathsf{cm})} = \vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$.

Ajtai commitment is binding under M-SIS, which requires an explicit upper bound on the norm of the committed vector. To ensure that the norm of the folded witnesses remains within the binding regime throughout the recursion, the challenge space $\mathcal{C}$ must be chosen carefully. Section 3.6 discusses the corresponding parameter selection.

Linear Constraint Proof. We next prove the inner product between $\mathsf{coef}(\vec{\mathbf{s}})$ and $\bigotimes_{i=0}^{\log N-1} \vec{a}_i$. This constraint can be proved in $\log N$ rounds by exploiting the recursive decomposability of the tensor product, which reduces the original inner product to lower-dimensional inner products via successive folding.

Define the ring vector

$$\vec{\mathbf{a}}_0 \coloneqq \Big(\sum_{i=0}^{d-1} a_{0,i} X^{-i},\ \sum_{i=0}^{d-1} a_{0,i+d} X^{-i},\ \ldots,\ \sum_{i=0}^{d-1} a_{0,i+(2\ell-1)d} X^{-i}\Big) \in \mathbb{R}_q^{2\ell}.$$

Then

$$\Big\langle \mathsf{coef}(\vec{\mathbf{s}}),\ \bigotimes_{i=0}^{\log N-1} \vec{a}_i \Big\rangle = \mathsf{ct}\left(\Big\langle \vec{\mathbf{s}},\ \Big(\bigotimes_{i=1}^{\log N-1} \vec{a}_i\Big) \otimes \vec{\mathbf{a}}_0 \Big\rangle\right).$$

Let $\vec{\mathbf{a}}^{(\log N-2)} \coloneqq \left(\bigotimes_{i=1}^{\log N-2} \vec{a}_i\right) \otimes \vec{\mathbf{a}}_0$. Write $\vec{\mathbf{s}} = (\vec{\mathbf{s}}_L \,\|\, \vec{\mathbf{s}}_R)$. By the tensor-product decomposition of the last factor, we have

$$\left\langle \vec{\mathbf{s}}, \vec{a}_{\log N-1} \otimes \vec{\mathbf{a}}^{(\log N-2)} \right\rangle = \left\langle \vec{a}_{\log N-1}, \left(\langle \vec{\mathbf{s}}_L, \vec{\mathbf{a}}^{(\log N-2)} \rangle,\ \langle \vec{\mathbf{s}}_R, \vec{\mathbf{a}}^{(\log N-2)} \rangle\right) \right\rangle.$$

Then, the prover sends $\mathbf{v}_{1,L} \coloneqq \langle \vec{\mathbf{s}}_L, \vec{\mathbf{a}}^{(\log N-2)} \rangle$, $\mathbf{v}_{1,R} \coloneqq \langle \vec{\mathbf{s}}_R, \vec{\mathbf{a}}^{(\log N-2)} \rangle$. The verifier checks that $\mathsf{ct}\big(\langle \vec{a}_{\log N-1}, (\mathbf{v}_{1,L}, \mathbf{v}_{1,R}) \rangle\big) = v$. It then verifies the correctness of $(\mathbf{v}_{1,L}, \mathbf{v}_{1,R})$ via recursion.

The verifier samples a challenge $\vec{c}_1 = (c_{1,0}, c_{1,1}) \in \mathcal{C}^2$. The prover folds the witness as $\vec{\mathbf{s}}_2 \coloneqq c_{1,0}\, \vec{\mathbf{s}}_L + c_{1,1}\, \vec{\mathbf{s}}_R$. The verifier reduces the claim to

$$\left\langle \vec{\mathbf{s}}_2, \vec{\mathbf{a}}^{(\log N-2)} \right\rangle \stackrel{?}{=} c_{1,0}\, \mathbf{v}_{1,L} + c_{1,1}\, \mathbf{v}_{1,R}.$$

This procedure proceeds recursively for $\log N - 1$ rounds, halving the witness length in each round along the N axis.

In the final round, the prover sends the folded witness $\vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$. The verifier computes $\langle \vec{\mathbf{s}}_{\log N}, \vec{\mathbf{a}}_0 \rangle$ locally and checks that it matches the reduced claim. The proof transmitted by the prover consists of $\pi^{(\mathsf{lin})} = \big(\pi_1^{(\mathsf{lin})}, \ldots, \pi_{\log N}^{(\mathsf{lin})}\big)$, where $\pi_i^{(\mathsf{lin})} = (\mathbf{v}_{i,L}, \mathbf{v}_{i,R}) \in \mathbb{R}_q^2$ for $i \in [1\colon \log N - 1]$ and $\pi_{\log N}^{(\mathsf{lin})} = \vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$.

Quadratic Constraint Proof. We adopt a divide-and-conquer strategy to prove the quadratic (self-inner-product) constraint.

Write $\vec{\mathbf{s}} = (\vec{\mathbf{s}}_L \,\|\, \vec{\mathbf{s}}_R)$. Then $\langle \vec{\mathbf{s}}, \sigma_{-1}(\vec{\mathbf{s}}) \rangle = \langle \vec{\mathbf{s}}_L, \sigma_{-1}(\vec{\mathbf{s}}_L) \rangle + \langle \vec{\mathbf{s}}_R, \sigma_{-1}(\vec{\mathbf{s}}_R) \rangle$. The prover sends $\mathbf{L}_1 \coloneqq \langle \vec{\mathbf{s}}_L, \sigma_{-1}(\vec{\mathbf{s}}_L) \rangle$, $\mathbf{R}_1 \coloneqq \langle \vec{\mathbf{s}}_R, \sigma_{-1}(\vec{\mathbf{s}}_R) \rangle$ to the verifier. The verifier checks $\mathsf{ct}(\mathbf{L}_1 + \mathbf{R}_1) \equiv b \pmod q$.

However, $(\mathbf{L}_1, \mathbf{R}_1)$ alone do not support recursion. The verifier samples $\vec{c}_1 = (c_{1,0}, c_{1,1}) \in \mathcal{C}^2$ and the prover folds $\vec{\mathbf{s}}_2 \coloneqq c_{1,0}\, \vec{\mathbf{s}}_L + c_{1,1}\, \vec{\mathbf{s}}_R$. Expanding the next-round quadratic value gives

$$\begin{aligned}\big\langle \vec{\mathbf{s}}_2, \sigma_{-1}(\vec{\mathbf{s}}_2) \big\rangle = {} & c_{1,0}\, \sigma_{-1}(c_{1,0})\, \mathbf{L}_1 + c_{1,1}\, \sigma_{-1}(c_{1,1})\, \mathbf{R}_1 \\ & + c_{1,0}\, \sigma_{-1}(c_{1,1}) \big\langle \vec{\mathbf{s}}_L, \sigma_{-1}(\vec{\mathbf{s}}_R) \big\rangle + c_{1,1}\, \sigma_{-1}(c_{1,0}) \big\langle \vec{\mathbf{s}}_R, \sigma_{-1}(\vec{\mathbf{s}}_L) \big\rangle.\end{aligned}$$

This uses multiplicativity of σ_{-1} on $\mathbb{R}_q$, *i.e.*, $\sigma_{-1}(ab) = \sigma_{-1}(a)\, \sigma_{-1}(b)$ for all $a, b \in \mathbb{R}_q$. Therefore, the prover additionally sends the cross terms $\mathbf{M}_1^{(1)} \coloneqq \langle \vec{\mathbf{s}}_L, \sigma_{-1}(\vec{\mathbf{s}}_R) \rangle$, $\mathbf{M}_1^{(2)} \coloneqq \langle \vec{\mathbf{s}}_R, \sigma_{-1}(\vec{\mathbf{s}}_L) \rangle$. Given $\pi_1^{(\mathsf{qua})} = (\mathbf{L}_1, \mathbf{M}_1^{(1)}, \mathbf{M}_1^{(2)}, \mathbf{R}_1)$ and $\vec{c}_1$, the prover and verifier can then perform the recursive check on $\vec{\mathbf{s}}_2$.

Given $(\mathbf{L}_1, \mathbf{M}_1^{(1)}, \mathbf{M}_1^{(2)}, \mathbf{R}_1)$ and $\vec{c}_1$, the original self–inner-product constraint reduces to checking $\langle \vec{\mathbf{s}}_2, \sigma_{-1}(\vec{\mathbf{s}}_2) \rangle = \langle \vec{c}_1^{\,(\mathsf{qua})}, \pi_1^{(\mathsf{qua})} \rangle$, where

$$\vec{c}_1^{\,(\mathsf{qua})} = \big(c_{1,0}\, \sigma_{-1}(c_{1,0}),\ c_{1,1}\, \sigma_{-1}(c_{1,1}),\ c_{1,0}\, \sigma_{-1}(c_{1,1}),\ c_{1,1}\, \sigma_{-1}(c_{1,0})\big).$$

The recursion then proceeds by applying the same split-and-fold step to $\vec{\mathbf{s}}_2$.

To align with the previous two sub-proofs, we run $\log N$ (rather than $\log(N\ell)$) rounds by folding only along the N axis. The quadratic-constraint transcript is $\pi^{(\mathsf{qua})} = \big(\pi_1^{(\mathsf{qua})}, \ldots, \pi_{\log N}^{(\mathsf{qua})}\big)$, where $\pi_i^{(\mathsf{qua})} = (\mathbf{L}_i, \mathbf{M}_i^{(1)}, \mathbf{M}_i^{(2)}, \mathbf{R}_i) \in \mathbb{R}_q^4$ for $i \in [1\colon \log N - 1]$ and $\pi_{\log N}^{(\mathsf{qua})} = \vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$. The verifier locally computes $\langle \vec{\mathbf{s}}_{\log N}, \sigma_{-1}(\vec{\mathbf{s}}_{\log N}) \rangle$ and checks that it matches the final reduced claim.

Since all these proofs share the same structure, we use the same per-round challenge $\vec{c}_i$ across all sub-proofs, and the prover sends the final folded witness $\vec{\mathbf{s}}_{\log N}$ only once.

3.3 Binary Range Proof

We enforce $\mathsf{coef}(\vec{\mathbf{s}}) \in \{0,1\}^{N\ell d}$ by reducing the component-wise constraint $\vec{s} \circ (\vec{s} - \vec{1}) = \vec{0}$ to a collection of inner-product constraints, which are handled using the same $\log N$-round recursion as in Section 3.2. Here, $\vec{s} \coloneqq \mathsf{coef}(\vec{\mathbf{s}}) \in \mathbb{Z}_q^{N\ell d}$.

Randomized binarity check. The verifier samples $\alpha \leftarrow \mathbb{Z}_q$ and defines $\vec{\alpha} := (1, \alpha, \alpha^2, \ldots, \alpha^{N\ell d-1}) \in \mathbb{Z}_q^{N\ell d}$. If $\vec{s} \in \{0,1\}^{N\ell d}$, then

$$\langle \vec{\alpha} \circ \vec{s}, \vec{s} - \vec{1} \rangle = 0, \qquad \text{equivalently} \qquad \langle \vec{\alpha} \circ \vec{s}, \vec{s} \rangle = \langle \vec{\alpha}, \vec{s} \rangle.$$

Conversely, if $\vec{s} \notin \{0,1\}^{N\ell d}$, then $P(\alpha) := \sum_{i=0}^{N\ell d-1} (s_i^2 - s_i)\, \alpha^i$ is a non-zero polynomial of degree at most $N\ell d - 1$, and therefore $\Pr_{\alpha \leftarrow \mathbb{Z}_q}\big[P(\alpha) = 0\big] \le \frac{N\ell d}{q}$. Let $v_\alpha := \langle \vec{\alpha}, \vec{s} \rangle \in \mathbb{Z}_q$ and $\vec{s}_\alpha := \vec{s} \circ \vec{\alpha}$. We embed $\vec{s}_\alpha$ into the ring by introducing $\vec{\mathbf{s}}_\alpha \in \mathbb{R}_q^{N\ell}$ with $\mathsf{coef}(\vec{\mathbf{s}}_\alpha) = \vec{s}_\alpha$, so that $\mathsf{ct}\big(\langle \vec{\mathbf{s}}_\alpha, \sigma_{-1}(\vec{\mathbf{s}}) \rangle\big) = \langle \vec{s}_\alpha, \vec{s} \rangle = v_\alpha$.

To expose v_α as a tensor-structured linear constraint compatible with our recursion, we decompose $\vec{\alpha}$ as follows. Let

$$\vec{\alpha}_0 := (1, \alpha, \ldots, \alpha^{2\ell d-1}) \in \mathbb{Z}_q^{2\ell d}, \qquad \vec{\alpha}_i := \big(1, \alpha^{2^{i-1} \cdot 2\ell d}\big) \in \mathbb{Z}_q^2 \ \text{ for } i \in [1\colon \log N],$$

so that $\vec{\alpha} = \bigotimes_{i=0}^{\log N - 1} \vec{\alpha}_i$ and hence $\Big\langle \mathsf{coef}(\vec{\mathbf{s}}), \bigotimes_{i=0}^{\log N - 1} \vec{\alpha}_i \Big\rangle = v_\alpha$.

Certifying well-formedness of $\vec{\mathbf{s}}_\alpha$. To ensure $\mathsf{coef}(\vec{\mathbf{s}}_\alpha) = \vec{s} \circ \vec{\alpha}$, the verifier samples an additional challenge $r \leftarrow \mathbb{Z}_q$ and defines

$$\vec{r} := (1, r, \ldots, r^{N\ell d-1}), \qquad \vec{r}_\alpha := \vec{r} \circ \vec{\alpha} = (1, r\alpha, \ldots, (r\alpha)^{N\ell d-1}).$$

Then $\mathsf{coef}(\vec{\mathbf{s}}_\alpha) = \vec{s} \circ \vec{\alpha}$ implies $\langle \mathsf{coef}(\vec{\mathbf{s}}_\alpha), \vec{r} \rangle = \langle \mathsf{coef}(\vec{\mathbf{s}}), \vec{r}_\alpha \rangle$. Both $\vec{r}$ and $\vec{r}_\alpha$ admit tensor decompositions analogous to $\vec{\alpha}$; we therefore reduce this check to two tensor-structured linear constraints whose outputs are compared.

Commitment order. The challenge r must be sampled only after the prover has fixed $\vec{\mathbf{s}}_\alpha$. Otherwise, knowing (α, r) allows the prover to choose a fabricated $\tilde{\vec{\mathbf{s}}}_\alpha$ that matches the required inner products at the single evaluation point r while differing from $\vec{s} \circ \vec{\alpha}$ elsewhere. We therefore commit to (a decomposition of) $\vec{\mathbf{s}}_\alpha$ before receiving r.

Concretely, we decompose $\vec{\mathbf{s}}_\alpha$ in base σ_α into $\iota_\alpha := \lceil \log_{\sigma_\alpha} q \rceil$ pieces and concatenate them as $\hat{\mathbf{s}}_\alpha \in \mathbb{R}_q^{N\ell\iota_\alpha}$ such that $G_{\sigma_\alpha, N\ell}\, \hat{\mathbf{s}}_\alpha = \vec{\mathbf{s}}_\alpha$. We then commit to $\hat{\mathbf{s}}_\alpha$ using the same $\log N$-level structure as in Section 3.2, with public matrices $\mathbf{B}_i \in \mathbb{R}_q^{\kappa \times 2\kappa\iota}$ for $i \in [1\colon \log N]$, $\mathbf{B}_0 \in \mathbb{R}_q^{\kappa \times 2\ell\iota_\alpha}$.

$\hat{\mathbf{s}}_\alpha$ is used only to help establish binarity of the original witness, so a loose norm bound on $\hat{\mathbf{s}}_\alpha$ suffices.

Protocol repetition. In summary, the binary-check protocol proceeds as follows. The verifier samples α; the prover forms $\vec{\mathbf{s}}_\alpha$, decomposes it, and sends $\mathsf{cm}_\alpha := \mathsf{Com}(\hat{\mathbf{s}}_\alpha)$. The verifier then samples r and the parties prove the claims below via the recursion from Section 3.2 (with the quadratic protocol used in its bilinear form):

1. $\mathsf{Com}(\hat{\mathbf{s}}_\alpha) = \mathsf{cm}_\alpha$;
2. $\Big\langle \mathsf{coef}(\vec{\mathbf{s}}), \bigotimes_{i=0}^{\log N - 1} \vec{\alpha}_i \Big\rangle = v_\alpha$;

3. $\langle \mathsf{coef}(\vec{\mathbf{s}}), \vec{r}_\alpha \rangle = v_{r_\alpha}$;
4. $\langle \mathsf{coef}(G_{\sigma_\alpha, N\ell}\, \hat{\mathbf{s}}_\alpha), \vec{r} \rangle = v_r$ and $v_r = v_{r_\alpha}$;
5. $\mathsf{ct}(\langle G_{\sigma_\alpha, N\ell}\, \hat{\mathbf{s}}_\alpha, \sigma_{-1}(\vec{\mathbf{s}}) \rangle) = v_\alpha$.

That is, at each round $i \in [1, \log N]$, the binary proof consists of the following components:

$$\pi_i^{(\mathsf{bin})} = \big(\pi_i^{(\mathsf{cm},\alpha)}, \pi_i^{(\mathsf{lin}_1,\alpha)}, \pi_i^{(\mathsf{lin}_2,\alpha)}, \pi_i^{(\mathsf{lin}_3,\alpha)}, \pi_i^{(\mathsf{qua},\alpha)}\big) \in \mathbb{R}_q^{2\kappa\iota+10}.$$

In the final round, the prover directly sends $\hat{\mathbf{s}}_{\alpha,\log N} \in \mathbb{R}_q^{2\ell\iota}$.

Each randomized reduction in the binary-check protocol incurs a soundness error of at most $2N\ell d/q$. To drive this error down to the negligible level $2^{-\lambda}$, we repeat the protocol t times using fresh α values, where $t \coloneqq \left\lceil \frac{\lambda}{\log\left(q/(N\ell d)\right)} \right\rceil$.

Discussion (binary check alignment). A generic lattice range proof could certify $\mathsf{coef}(\vec{\mathbf{s}}) \in \{0,1\}^{N\ell d}$, but it would be structurally misaligned with the PCS recursion and would introduce an additional proof subsystem. We instead use the randomized identity $\langle \vec{\alpha} \circ \vec{s}, \vec{s} \rangle = \langle \vec{\alpha}, \vec{s} \rangle$ to express binarity as an inner-product-shaped constraint over $\mathbb{R}_q$. This makes binarity provable by the same $\log N$-round split-and-fold engine used for evaluation, commitment consistency, and the self-inner-product norm check, which keeps verifier work polylogarithmic in N and preserves composability.

3.4 Putting Everything Together

We combine the sub-proofs from Sections 3.2 and 3.3 into a single protocol.

All sub-proofs share the same $\log N$-round split-and-fold recursion. In round i, the prover folds every active witness (and its commitment state) using the same verifier challenge. This reuse enforces that every sub-proof refers to the same folded main witness $\vec{\mathbf{s}}_{i+1}$ and, for each binary repetition, the same folded auxiliary witness $\hat{\mathbf{s}}_{\alpha,i+1}$. Accordingly, the verifier can maintain one reduced claim per constraint family, and no additional cross-consistency checks are needed.

In each round $i \in [1\colon \log N]$, the prover sends the tuple

$$\pi_i \coloneqq \big(\pi_i^{(\mathsf{cm})}, \pi_i^{(\mathsf{lin})}, \pi_i^{(\mathsf{qua})}, (\pi_i^{(\mathsf{bin})})^t\big),$$

where $(\pi_i^{(\mathsf{bin})})^t$ denotes the t independent repetitions of the binary sub-proof (Section 3.3). In the final round, the prover sends $\pi_{\log N} = (\vec{\mathbf{s}}_{\log N}, (\hat{\mathbf{s}}^{(j)}_{\log N})_{j\in[t]})$.

Statement-level checks. The verifier first checks $0 \le b \le \beta^2$. This check depends only on the statement and is independent of the transcript.

Verifier checks in round $i = 1$. The verifier checks:

1. **Commitment validity:** $\mathbf{A}_{\log N-1} \cdot \pi_1^{(\mathsf{cm})} \stackrel{?}{=} \mathsf{cm}$.
2. **Linear claim (expose v):** $\mathsf{ct}\Big(\Big\langle \vec{a}_{\log N-1}, \pi_1^{(\mathsf{lin})} \Big\rangle\Big) \stackrel{?}{=} v$.
3. **Quadratic claim (expose b):** for $\pi_1^{(\mathsf{qua})} = (\mathbf{L}_1, \mathbf{M}_1^{(1)}, \mathbf{M}_1^{(2)}, \mathbf{R}_1)$, check $\mathsf{ct}(\mathbf{L}_1 + \mathbf{R}_1) \stackrel{?}{\equiv} b \pmod q$.
4. **Binary sub-proof:** For each repetition, check:
   $\mathbf{B}_{\log N-1} \cdot \pi_1^{(\mathsf{cm},\alpha)} = \mathsf{cm}_\alpha$, $\mathsf{ct}\Big(\Big\langle \vec{\alpha}_{\log N-1}, \vec{\pi}_1^{(\mathsf{lin}_1,\alpha)} \Big\rangle\Big) = v_\alpha$,
   $\mathsf{ct}\Big(\Big\langle \vec{r}_{\alpha,\log N-1}, \vec{\pi}_1^{(\mathsf{lin}_2,\alpha)} \Big\rangle\Big) = v_{r_\alpha}$, $\mathsf{ct}\Big(\Big\langle \vec{r}_{\log N-1}, \vec{\pi}_1^{(\mathsf{lin}_3,\alpha)} \Big\rangle\Big) = v_r$,
   and $\mathsf{ct}(\mathbf{L}_1^{(\alpha)} + \mathbf{R}_1^{(\alpha)}) = v_\alpha$

Verifier checks in rounds $i \in [2\colon \log N - 1]$. Write $\pi_i^{(\mathsf{cm})} = (\mathsf{cm}_{i,L}, \mathsf{cm}_{i,R})$ and $\vec{c}_{i-1} = (c_{i-1,0}, c_{i-1,1})$. The verifier checks:

1. **Commitment consistency:**
   $\mathbf{A}_{\log N-i} \cdot \pi_i^{(\mathsf{cm})} \stackrel{?}{=} \big[\, c_{i-1,0}\, G_{\sigma,\kappa} \quad c_{i-1,1}\, G_{\sigma,\kappa} \,\big] \cdot \pi_{i-1}^{(\mathsf{cm})}$.
2. **Linear recursion:** $\Big\langle \vec{a}_{\log N-i}, \pi_i^{(\mathsf{lin})} \Big\rangle \stackrel{?}{=} \Big\langle \vec{c}_{i-1}, \pi_{i-1}^{(\mathsf{lin})} \Big\rangle$.
3. **Quadratic recursion:** define $\vec{c}_{i-1}^{\,(\mathsf{qua})}$ to be
   $$\big(c_{i-1,0}\sigma_{-1}(c_{i-1,0}), c_{i-1,0}\sigma_{-1}(c_{i-1,1}), c_{i-1,1}\sigma_{-1}(c_{i-1,0}), c_{i-1,1}\sigma_{-1}(c_{i-1,1})\big),$$
   and check, for $\pi_i^{(\mathsf{qua})} = (\mathbf{L}_i, \mathbf{M}_i^{(1)}, \mathbf{M}_i^{(2)}, \mathbf{R}_i)$, $\mathbf{L}_i + \mathbf{R}_i \stackrel{?}{=} \Big\langle \vec{c}_{i-1}^{\,(\mathsf{qua})}, \pi_{i-1}^{(\mathsf{qua})} \Big\rangle$.
4. **Binary recursion:** for each repetition, run the analogous round-i checks from Section 3.3 using the same folding challenge $\vec{c}_{i-1}$.

Final round $i = \log N$. The prover sends the final folded openings $\vec{\mathbf{s}}_{\log N}$ and, for each repetition, $\hat{\mathbf{s}}_{\alpha,\log N}$. The verifier checks (omitting the explicit repetition bookkeeping):

1. **Norm admissibility:** $\|\vec{\mathbf{s}}_{\log N}\| \le \gamma$ and $\|\hat{\mathbf{s}}_{\alpha,\log N}\| \le \gamma_\alpha$.
2. **Base commitment opening:**
   $\mathbf{A}_0 \cdot \vec{\mathbf{s}}_{\log N} \stackrel{?}{=} \big[\, c_{\log N-1,0}\, G_{\sigma,\kappa} \quad c_{\log N-1,1}\, G_{\sigma,\kappa} \,\big] \cdot \pi_{\log N-1}^{(\mathsf{cm})}$.
3. **Auxiliary base opening (binary proof):**
   $\mathbf{B}_0 \cdot \hat{\mathbf{s}}_{\alpha,\log N} \stackrel{?}{=} \big[\, c_{\log N-1,0}\, G_{\sigma,\kappa} \quad c_{\log N-1,1}\, G_{\sigma,\kappa} \,\big] \cdot \pi_{\log N-1}^{(\mathsf{cm},\alpha)}$.
4. **Final linear check:** $\langle \vec{\mathbf{a}}_0, \vec{\mathbf{s}}_{\log N} \rangle \stackrel{?}{=} \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin})} \rangle$.
5. **Final binary linear check:** $\langle \vec{\alpha}_0, \vec{\mathbf{s}}_{\log N} \rangle \stackrel{?}{=} \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin}_1,\alpha)} \rangle$,
   $\langle \vec{\mathbf{r}}_{\alpha,0}, G_{\sigma_\alpha,2\ell} \cdot \hat{\mathbf{s}}_{\alpha,\log N} \rangle \stackrel{?}{=} \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin}_2,\alpha)} \rangle$,
   and $\langle \vec{\mathbf{r}}_0, \vec{\mathbf{s}}_{\log N} \rangle \stackrel{?}{=} \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin}_3,\alpha)} \rangle$
6. **Final quadratic check:** $\langle \vec{\mathbf{s}}_{\log N}, \sigma_{-1}(\vec{\mathbf{s}}_{\log N}) \rangle \stackrel{?}{=} \langle \vec{c}_{\log N-1}^{\,(\mathsf{qua})}, \pi_{\log N-1}^{(\mathsf{qua})} \rangle$.
7. **Final binary quadratic check:**
   $\langle G_{\sigma_\alpha,2\ell}\, \hat{\mathbf{s}}_{\alpha,\log N}, \sigma_{-1}(\vec{\mathbf{s}}_{\log N}) \rangle \stackrel{?}{=} \langle \vec{c}_{\log N-1}^{\,(\mathsf{qua})}, \pi_{\log N-1}^{(\mathsf{qua},\alpha)} \rangle$.

Figure 2 presents the complete protocol. The decomposition and admissibility parameters are discussed in Section 3.6.

$\mathsf{cm} = \mathsf{Com}(\vec{\mathbf{s}}), v, b, \beta \in \mathbb{Z}_q. (\vec{a}_i \in \mathbb{Z}_q^{b_i})_{i\in[\log N]}, (b_i = 2)_{i\in[1,\log N]}, b_0 = 2\ell d.$

Private Inputs: $\vec{\mathbf{s}} \in \mathbb{R}_q^{N\ell}, (\mathsf{cm}_{\mathrm{in},i} \in \mathbb{R}_q^{n_{i+1}\cdot\kappa\iota})_{i\in[\log N-1]}, n_{i+1} = N/2^{i+1}.$

$\mathcal{P}$		$\mathcal{V}$
		Check $0 \le b \le \beta^2$.
	$\xleftarrow{(\alpha^{(j)})_{j\in[t]}}$	Sample $(\alpha^{(j)})_{j\in[t]} \leftarrow \mathbb{Z}_q$.
		Construct $\vec{\alpha}_i^{(j)}$: $\bigotimes_{i=0}^{\log N-1} \vec{\alpha}_i = \vec{\alpha}$
Construct $\vec{\mathbf{s}}_\alpha^{(j)}$ with $\vec{s}_\alpha^{(j)} = \vec{\alpha}^{(j)} \circ \vec{s}$. $v_\alpha^{(j)} = \langle \vec{\alpha}^{(j)}, \vec{s} \rangle$, $\hat{\mathbf{s}}_\alpha^{(j)} = G^{-1}_{\sigma_\alpha, N\ell}(\vec{\mathbf{s}}_\alpha^{(j)})$, $\mathsf{cm}_\alpha^{(j)} = \mathsf{Com}(\hat{\mathbf{s}}_\alpha^{(j)})$.	$\xrightarrow{(\mathsf{cm}_\alpha^{(j)}, v_\alpha^{(j)})_{j\in[t]}}$	
		Sample $r^{(j)} \leftarrow \mathbb{Z}_q$.
	$\xleftarrow{(r^{(j)})_{j\in[t]}}$	Construct $\vec{r}_i^{(j)}$, $\vec{r}_{\alpha,i}^{(j)}$: $\bigotimes_{i=0}^{\log N-1} \vec{r}_i^{(j)} = \vec{r}^{(j)}$, $\bigotimes_{i=0}^{\log N-1} \vec{r}_{\alpha,i}^{(j)} = \vec{r}^{(j)} \circ \vec{\alpha}^{(j)}$.
$v_r^{(j)} = \langle \vec{s}_\alpha^{(j)}, \bigotimes_{i=0}^{\log N-1} \vec{r}_i^{(j)} \rangle$. $v_{r_\alpha}^{(j)} = \langle \vec{s}, \bigotimes_{i=0}^{\log N-1} \vec{r}_{\alpha,i}^{(j)} \rangle$.	$\xrightarrow{(v_r^{(j)}, v_{r_\alpha}^{(j)})_{j\in[t]}}$	Check $v_r^{(j)} \stackrel{?}{=} v_{r_\alpha}^{(j)}$.
Compute: $\vec{\pi}_1^{(\mathsf{cm})}, \vec{\pi}_1^{(\mathsf{lin})}, \vec{\pi}_1^{(\mathsf{qua})}, (\vec{\pi}_1^{(\mathsf{bin},j)})_{j\in[t]}$. $\vec{\pi}_1 = (\vec{\pi}_1^{(\mathsf{cm})}, \vec{\pi}_1^{(\mathsf{lin})}, \vec{\pi}_1^{(\mathsf{qua})}, (\vec{\pi}_1^{(\mathsf{bin},j)})_{j\in[t]})$.	$\xrightarrow{\vec{\pi}_1}$	
	$\xleftarrow{\vec{c}_1}$	Run $\mathsf{Verify}(\pi_1)$. Sample $\vec{c}_1 \leftarrow \mathcal{C}^2$.
Update: $\vec{\mathbf{s}}_2 = c_{1,0}\vec{\mathbf{s}}_L + c_{1,1}\vec{\mathbf{s}}_R$, $\hat{\mathbf{s}}_{\alpha,2}^{(j)} = c_{1,0}\hat{\mathbf{s}}_{\alpha,L} + c_{1,1}\hat{\mathbf{s}}_{\alpha,R}$, Compute: $\vec{\pi}_2$.	$\xrightarrow{\vec{\pi}_2}$	
	$\xleftarrow{\vec{c}_2}$	Run $\mathsf{Verify}(\pi_1, \pi_2)$. Sample $\vec{c}_2 \leftarrow \mathcal{C}^2$.
Update: $\vec{\mathbf{s}}_3$, $\hat{\mathbf{s}}_{\alpha,3}^{(j)}$ …	$\xleftrightarrow[\log N - 1 \text{ rounds}]{\text{Iterate}}$	…
Update: $\vec{\mathbf{s}}_{\log N}$, $\hat{\mathbf{s}}_{\alpha,\log N}^{(j)}$	$\xrightarrow{\vec{\mathbf{s}}_{\log N}, \hat{\mathbf{s}}_{\alpha,\log N}}$	Run $\mathsf{Verify}(\pi_k)$, $k \in [1, \log N + 1]$.

Fig. 2. Core Protocol for Proving $\mathcal{R}_{\mathsf{main}}$

$\mathsf{Verify}\big((\pi_k)_{k\in[1,i+1]}\big)$

1 : If $i = 1$, check:

$\mathbf{A}_{\log N-1}\cdot\Big(\pi_1^{(\mathsf{cm})}\Big) \stackrel{?}{=} \mathsf{cm}, \quad \mathbf{B}_{\log N-1}\cdot\Big(\pi_1^{(\mathsf{cm},\alpha,j)}\Big) \stackrel{?}{=} \mathsf{cm}_\alpha^{(j)},$

$\mathsf{ct}\Big(\langle \vec{a}_{\log N-1}, \pi_1^{(\mathsf{lin})}\rangle\Big) \stackrel{?}{=} v, \quad \mathsf{ct}\Big(\langle \vec{r}^{(j)}_{\log N-1}, \pi_1^{(\mathsf{lin}_1,\alpha,j)}\rangle\Big) \stackrel{?}{=} v_r^{(j)},$

$\mathsf{ct}\Big(\langle \vec{r}^{(j)}_{\alpha,\log N-1}, \pi_1^{(\mathsf{lin}_2,\alpha,j)}\rangle\Big) \stackrel{?}{=} v_{r_\alpha}^{(j)}, \quad \mathsf{ct}\Big(\langle \vec{\alpha}^{(j)}_{\log N-1}, \pi_1^{(\mathsf{lin}_3,\alpha,j)}\rangle\Big) \stackrel{?}{=} v_\alpha^{(j)},$

$\mathsf{ct}\Big(\langle (1,0,0,1), \pi_1^{(\mathsf{qua})}\rangle\Big) \stackrel{?}{=} b, \quad \mathsf{ct}\Big(\langle (1,0,0,1), \pi_1^{(\mathsf{qua},\alpha,j)}\rangle\Big) \stackrel{?}{=} v_\alpha^{(j)}.$

2 : If $i \in \{2,3,\ldots,\log N-1\}$, check:

$\mathbf{A}_{\log N-i}\cdot\big(\pi_i^{(\mathsf{cm})}\big) \stackrel{?}{=} \big[\, c_{i-1,0}G_{\sigma,\kappa} \;\; c_{i-1,1}G_{\sigma,\kappa}\,\big]\cdot\pi_{i-1}^{(\mathsf{cm})}$

$\langle \vec{a}_{\log N-i}, \pi_i^{(\mathsf{lin})}\rangle \stackrel{?}{=} \langle \vec{c}_{i-1}, \pi_{i-1}^{(\mathsf{lin})}\rangle.$

$(1,0,0,1)\cdot\pi_i^{(\mathsf{qua})} \stackrel{?}{=} \langle \vec{c}_{i-1}^{(\mathsf{qua})}, \pi_{i-1}^{(\mathsf{qua})}\rangle.$

$\mathbf{B}_{\log N-i}\cdot\big(\pi_i^{(\mathsf{cm},\alpha,j)}\big) \stackrel{?}{=} \big[\, c_{i-1,0}G_{\sigma,\kappa} \;\; c_{i-1,1}G_{\sigma,\kappa}\,\big]\cdot\pi_{i-1}^{(\mathsf{cm},\alpha,j)}.$

$\langle \vec{r}^{(j)}_{\log N-i}, \pi_i^{(\mathsf{lin}_1,\alpha,j)}\rangle \stackrel{?}{=} \langle \vec{c}_{i-1}, \pi_{i-1}^{(\mathsf{lin}_1,\alpha,j)}\rangle.$

$\langle \vec{r}^{(j)}_{\alpha,\log N-i}, \pi_i^{(\mathsf{lin}_2,\alpha,j)}\rangle \stackrel{?}{=} \langle \vec{c}_{i-1}, \pi_{i-1}^{(\mathsf{lin}_2,\alpha,j)}\rangle.$

$\langle \vec{\alpha}^{(j)}_{\log N-i}, \pi_i^{(\mathsf{lin}_3,\alpha,j)}\rangle \stackrel{?}{=} \langle \vec{c}_{i-1}, \pi_{i-1}^{(\mathsf{lin}_3,\alpha,j)}\rangle.$

$(1,0,0,1)\cdot\pi_i^{(\mathsf{qua},\alpha,j)} \stackrel{?}{=} \langle \vec{c}_{i-1}^{(\mathsf{qua})}, \pi_{i-1}^{(\mathsf{qua},\alpha,j)}\rangle.$

3 : If $i = \log N$, check:

$\|\vec{\mathbf{s}}_{\log N}\| \le \gamma, \|\hat{\mathbf{s}}^{(j)}_{\log N,\alpha}\| \le \gamma_\alpha,$

$\mathbf{A}_0\cdot\vec{\mathbf{s}}_{\log N} = [c_{\log N-1,0}G_{\sigma,\kappa},\, c_{\log N-1,1}G_{\sigma,\kappa}]\cdot\pi_{\log N-1}^{(\mathsf{cm})},$

$\mathbf{B}_0\cdot\hat{\mathbf{s}}^{(j)}_{\alpha,\log N} = [c_{\log N-1,0}G_{\sigma,\kappa},\, c_{\log N-1,1}G_{\sigma,\kappa}]\cdot\pi_{\log N-1}^{(\mathsf{cm},\alpha,j)},$

$\langle \vec{\mathbf{a}}_0, \vec{\mathbf{s}}_{\log N}\rangle = \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin})}\rangle,$

$\langle \vec{\mathbf{r}}_0^{(j)}, \hat{\mathbf{s}}^{(j)}_{\alpha,\log N}\rangle = \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin}_1,\alpha,j)}\rangle,$

$\langle \vec{\mathbf{r}}^{(j)}_{\alpha,0}, \vec{\mathbf{s}}_{\log N}\rangle = \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin}_2,\alpha,j)}\rangle,$

$\langle \vec{\alpha}_0^{(j)}, \vec{\mathbf{s}}_{\log N}\rangle = \langle \vec{c}_{\log N-1}, \pi_{\log N-1}^{(\mathsf{lin}_3,\alpha,j)}\rangle,$

$\langle \vec{\mathbf{s}}_{\log N}, \sigma_{-1}(\vec{\mathbf{s}}_{\log N})\rangle = \langle \vec{c}_{\log N-1}^{(\mathsf{qua})}, \pi_{\log N-1}^{(\mathsf{qua})}\rangle,$

$\langle G_{\sigma_\alpha,2\ell}\hat{\mathbf{s}}^{(j)}_{\alpha,\log N}, \sigma_{-1}(\vec{\mathbf{s}}_{\log N})\rangle = \langle \vec{c}_{\log N-1}^{(\mathsf{qua})}, \pi_{\log N-1}^{(\mathsf{qua},\alpha,j)}\rangle.$

Fig. 3. Verification Algorithm for The Core Protocol for Proving $\mathcal{R}_{\mathsf{main}}$ (at each round, it checks t repeated binary proofs (for each $j \in [t]$))

3.5 Security Analysis

We analyze the interactive protocol in Fig. 2 for proving $\mathcal{R}_{\mathsf{main}}$. Completeness is perfect. Soundness follows from round-by-round, coordinate-wise special soundness of the shared split-and-fold recursion, plus the independent error from the randomized binary check repeated t times.

Theorem 1 (Completeness). *Let the challenge space $\mathcal{C} \subset \mathbb{R}_q$ consist of ring elements whose ℓ_2-norms are bounded by τ and whose ℓ_∞-operator norm is bounded by T. Define the per-round ℓ_∞ bounds*

$$\gamma_{\log N} \coloneqq (2T)^{\log N - 1} \quad \textit{and} \quad \gamma_{\alpha,\log N} \coloneqq \frac{\sigma_\alpha}{2} \cdot (2T)^{\log N - 1}.$$

Define the final ℓ_2 admissibility bounds

$$\gamma \coloneqq \sqrt{2\ell d} \cdot \gamma_{\log N} \quad \textit{and} \quad \gamma_\alpha \coloneqq \sqrt{2\ell\, \iota_\alpha d} \cdot \gamma_{\alpha,\log N}.$$

Fig. 2's protocol for proving the relation $\mathcal{R}_{\mathsf{main}}$ achieves perfect completeness.

Proof. An honest prover follows the protocol and the verifier accepts with probability 1. It suffices to show that the final openings sent in round $\log N$ satisfy the stated norm bounds.

For $i \in [1\colon \log N - 1]$, let $\vec{\mathbf{s}}_{i+1} = c_{i,0}\vec{\mathbf{s}}_{i,L} + c_{i,1}\vec{\mathbf{s}}_{i,R}$ be the folded witness. Define $\hat{\mathbf{s}}_{\alpha,i+1}$ analogously for the auxiliary witness in the binary sub-proof. Let γ_i and $\gamma_{\alpha,i}$ be such that $\|\vec{\mathbf{s}}_i\|_\infty \le \gamma_i$ and $\|\hat{\mathbf{s}}_{\alpha,i}\|_\infty \le \gamma_{\alpha,i}$.

Using the operator-norm bound $\|c\mathbf{v}\|_\infty \le \|c\|_{\mathsf{op},\infty}\|\mathbf{v}\|_\infty \le T\|\mathbf{v}\|_\infty$ for all $c \in \mathcal{C}$, we obtain

$$\begin{aligned}\|\vec{\mathbf{s}}_{i+1}\|_\infty &\le \|c_{i,0}\vec{\mathbf{s}}_{i,L}\|_\infty + \|c_{i,1}\vec{\mathbf{s}}_{i,R}\|_\infty \\ &\le \|c_{i,0}\|_{\mathsf{op},\infty}\,\|\vec{\mathbf{s}}_{i,L}\|_\infty + \|c_{i,1}\|_{\mathsf{op},\infty}\,\|\vec{\mathbf{s}}_{i,R}\|_\infty \\ &\le 2T \cdot \gamma_i.\end{aligned}$$

Thus, $\gamma_{i+1} \coloneqq 2T\cdot\gamma_i$ is a valid bound. The same argument gives $\gamma_{\alpha,i+1} \coloneqq 2T\cdot\gamma_{\alpha,i}$.

For the initial witnesses, we use $\gamma_1 = 1$ (binary coefficients) and $\gamma_{\alpha,1} = \sigma_\alpha/2$ (base-σ_α decomposition). By induction over the $\log N - 1$ folding steps, we obtain

$$\|\vec{\mathbf{s}}_{\log N}\|_\infty \le \gamma_{\log N} \quad \text{and} \quad \|\hat{\mathbf{s}}_{\alpha,\log N}\|_\infty \le \gamma_{\alpha,\log N}.$$

Finally, $\vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$ and $\hat{\mathbf{s}}_{\alpha,\log N} \in \mathbb{R}_q^{2\ell\iota_\alpha}$, so

$$\|\vec{\mathbf{s}}_{\log N}\|_2 \le \sqrt{2\ell d} \cdot \|\vec{\mathbf{s}}_{\log N}\|_\infty \le \sqrt{2\ell d} \cdot \gamma_{\log N} = \gamma,$$

and

$$\|\hat{\mathbf{s}}_{\alpha,\log N}\|_2 \le \sqrt{2\ell\iota_\alpha d} \cdot \|\hat{\mathbf{s}}_{\alpha,\log N}\|_\infty \le \sqrt{2\ell\iota_\alpha d} \cdot \gamma_{\alpha,\log N} = \gamma_\alpha. \qquad \square$$

Theorem 2 (Coordinate-wise Special Soundness). *Assume that the* M-SIS *problem is hard for rank κ and norm bound*

$$\max\Big(N \cdot \sqrt{N\ell}\,\gamma_{\log N},\ N \cdot \sqrt{N\ell\,\iota_\alpha}\,\gamma_{\alpha,\log N}\Big),$$

where $\gamma_{\log N}$ and $\gamma_{\alpha,\log N}$ are as in Theorem 1.

Then Fig. 2 achieves 2-coordinate-wise 2-special soundness with additional soundness error $\frac{4\log N}{|\mathcal{C}|} + \left(\frac{2N\ell d}{q}\right)^t$, where the second term is the error from the randomized binary check repeated t times, and $t \coloneqq \left\lceil \frac{\lambda}{\log\left(q/(2N\ell d)\right)} \right\rceil$.

The formal proof of Theorem 2 is given in Appendix C.

Theorem 3 (Knowledge Soundness). *Fig. 2 for proving $\mathcal{R}_{\mathsf{main}}$ is knowledge-sound with total knowledge soundness error bounded by $\frac{6\log N}{|\mathcal{C}|} + \left(\frac{2N\ell d}{q}\right)^t$.*

Proof. By Lemma 3 and Theorem 2, Fig. 2 satisfies 2-coordinate-wise 2-special soundness with additional error $\frac{4\log N}{|\mathcal{C}|} + \left(\frac{2N\ell d}{q}\right)^t$. Applying the union bound as in Lemma 3 yields a total failure probability bounded by $\frac{6\log N}{|\mathcal{C}|} + \left(\frac{2N\ell d}{q}\right)^t$. □

3.6 Parameter Selection and Efficiency Analysis

Parameter Selection. We group parameters by their roles in the core protocol.

- **Decomposition parameters.** These parameters influence concrete efficiency and admissibility. They are the gadget-decomposition bases (σ, ι) for inner commitments, and $(\sigma_\alpha, \iota_\alpha)$ for the potentially larger-norm vector $\vec{\mathbf{s}}_\alpha$.
- **Hardness parameters.** These parameters determine the hardness of M-SIS. They are the rank κ, the prime modulus q, the ring dimension d, and the M-SIS norm bound $\beta_{\mathsf{M\text{-}SIS}}$ required by the reduction (*cf.* (4)).
- **Challenges.** These parameters control soundness and completeness of folding. They are the challenge space $\mathcal{C}$, its size $|\mathcal{C}|$, and bounds (τ, T) on the ℓ_2-norm and ℓ_∞-operator norm of its elements.

We next state the constraints induced by correctness and security, and then give asymptotic choices that satisfy them.

Decomposition. For a decomposition base $\sigma \geq 2$, recall $\iota = \lceil \log_\sigma q \rceil$ and $\iota_\alpha = \lceil \log_{\sigma_\alpha} q \rceil$. Smaller ι and ι_α reduce the number of decomposed components, while larger bases increase the magnitude of decomposition digits (and thus affect admissibility bounds such as γ_α). In our asymptotic summary (Table 2), we treat ι and ι_α as $O(1)$ by choosing σ and σ_α as fixed powers of q, but the concrete instantiation follows the same tradeoff.

M-SIS *hardness constraint.* Let $\beta_{\mathsf{M\text{-}SIS}}$ denote the Euclidean norm bound for the M-SIS instance produced by the reduction. By Theorem 2, it must satisfy

$$\beta_{\mathsf{M\text{-}SIS}} \;\geq\; \max\Big(N\sqrt{N\ell}\,\gamma_{\log N},\; N\sqrt{N\ell\,\iota_\alpha}\,\gamma_{\alpha,\log N}\Big)\,, \tag{4}$$

where $\gamma_{\log N}$ and $\gamma_{\alpha,\log N}$ are the folded-witness bounds from Theorem 1. Security requires that $\mathsf{M\text{-}SIS}$ is hard for the chosen (d, κ, q) at norm $\beta_{\mathsf{M\text{-}SIS}}$.

According to the heuristic methodology of [35], an $\mathsf{M\text{-}SIS}$ instance over a ring with element degree d and Euclidean bound $\beta_{\mathsf{M\text{-}SIS}}$ is hard when

$$\beta_{\mathsf{M\text{-}SIS}} \leq \min\Big(2^{\,2\sqrt{d\log q\log\delta}},\ q\Big),$$

where δ is the root-Hermite factor. Following [6], it can be expressed as

$$\delta = \left(\frac{b_k\,(\pi b_k)^{1/b_k}}{2\pi e}\right)^{1/(2(b_k-1))},$$

for a BKZ block size $b_k = O(\lambda)$.

Rearranging the above inequality yields $d\log q > \frac{(\log \beta_{\mathsf{M\text{-}SIS}})^2}{4\log\delta}$.

Furthermore, observing $\log\beta_{\mathsf{M\text{-}SIS}} = O(\log N + \log\ell)$, and using

$$\log\delta = \frac{1}{2(b_k-1)}\log\left(\frac{b_k(\pi b_k)^{1/b_k}}{2\pi e}\right) = \Theta\left(\frac{\log\lambda}{\lambda}\right) \quad \text{for } b_k = O(\lambda),$$

we will set $\log\ell = \log\log q < \log N$ in the following instantiation. We obtain

$$d\log q = O\left(\frac{\lambda\cdot\log^2 N}{\log\lambda}\right),$$

which corresponds to the asymptotic bit-length of a ring element in our setting.

Set $d = \Theta(\lambda)$, we get $\log q = O(\frac{\log^2 N}{\log\lambda})$. Since $N = \mathsf{poly}(\lambda)$ under the default setting, choosing q accordingly also ensures $q = \mathsf{poly}(\lambda)$, consistent with the default parameters.

Binary-check error and repetitions. Each randomized binary check contributes soundness error at most $2N\ell d/q$ (Section 3.3). Repeating it t times yields error $(2N\ell d/q)^t$, so we set $t := \left\lceil \frac{\lambda}{\log\left(q/(2N\ell d)\right)} \right\rceil$, which ensures $(2N\ell d/q)^t \leq 2^{-\lambda}$ as long as $q > 2N\ell d$. Noting that $\log q = O\left(\frac{\log^2 N}{\log\lambda}\right)$, we obtain $t = \left\lceil \frac{\lambda\cdot\log\lambda}{\log^2 N}\right\rceil = \Theta\left(\frac{\lambda}{\log\lambda}\right)$, and hence $t\cdot\log q = O(\lambda)$.

Challenge space. Recall that $\mathcal{C} \subset \mathbb{R}_q$ must satisfy: (i) for all distinct $c, c' \in \mathcal{C}$, the difference $c - c'$ is invertible in $\mathbb{R}_q$; (ii) $\|c\| \leq \tau$ and $\|c\|_{\mathsf{op},\infty} \leq T$ for all $c \in \mathcal{C}$. A concrete way to satisfy (i) is to choose $\mathcal{C}$ from a pool of short ring elements whose pairwise differences remain short, and then invoke Lemma 1.

Concretely, fix integers w_0, w_1 and $w_2 \in [d]$ consider the pool $\mathcal{P}_{w_0,w_1,w_2}$:

$$\Big\{\, c \in \mathbb{R}_q \;\Big|\; \mathsf{coef}(c) \in \{0, \pm1, \pm2\} \ \wedge\ \forall i \in \{0,1,2\} :\ \#\{j : |\mathsf{coef}(c)_j| = i\} = w_i \Big\}.$$

For any distinct $c, c' \in \mathcal{P}_w$, the difference $c - c'$ has the norm at most $2(w_1 + w_2)$ nonzero coefficients, each in $\{0, \pm1, \pm2, \pm3, \pm4\}$, hence $0 < \|c - c'\|_2 \leq$

Table 2. Notations with asymptotic parameters (with size parameter N instantiated with $N \cdot d = L$ in our PCS setting)

Notation	Explanation	Instantiation
q	Prime modulus, $q \equiv 5 \mod 8$	$\log q = O\Big(\frac{\log^2 N}{\log \lambda}\Big)$
d	Ring dimension	$\Theta(\lambda)$
κ	Height of matrices $\mathbf{A}_i$	$O(1)$
τ or T	ℓ_2- or operator-norm bound	$O(1)$
$\mathcal{C}$	Challenge space	$\lvert\mathcal{C}\rvert \approx 2^\lambda$
$(\sigma,\, \sigma_\alpha)$	Decomposition basis	$O(q^{1/O(1)})$
$(\iota,\, \iota_\alpha)$	Decomposition pieces	$O(1)$
$(\gamma_{\log N}, \gamma_{\alpha,\log N})$	Infinity norm bound	$O\big(q^{1/O(1)} \cdot N\big)$
t	Repetition times for binary check	$t \cdot \log q = O(\lambda)$

$4\sqrt{2w_1 + 2w_2}$. Thus, if q is a prime with $q \equiv 5 \bmod 8$ and $4\sqrt{2w_1 + 2w_2} < \sqrt{q/2}$ (equivalently, $q > 64(w_1 + w_2)$), then every nonzero difference $c - c'$ is invertible in $\mathbb{R}_q$ by Lemma 1, and condition (i) holds for any subset $\mathcal{C} \subseteq \mathcal{P}_w$. Moreover, for all $c \in \mathcal{P}_w$ we have $\|c\|_2 \le \sqrt{w_1 + 2w_2}$, so we may take $\tau \coloneqq \sqrt{w_1 + 2w_2}$. Also, using $\|c\mathbf{v}\|_\infty \le \|c\|_1 \cdot \|\mathbf{v}\|_\infty$ for polynomial multiplication in $\mathbb{R}_q$, we have $\|c\|_{\mathsf{op},\infty} \le \|c\|_1 = w_1 + 2w_2$, so we may take $T \coloneqq w_1 + 2w_2$. Finally, since $|\mathcal{P}_w| = \binom{d}{w_1} \cdot \binom{d-w_1}{w_2} \cdot 2^{w_1+w_2}$, we can choose w so that $|\mathcal{P}_w| \ge 2^\lambda$ and then select any subset $\mathcal{C} \subseteq \mathcal{P}_w$ of size $|\mathcal{C}| \approx 2^\lambda$. Table 2 summarizes the notation and the resulting asymptotic parameter choices.

Efficiency. We now estimate proof size and computation cost under the above choices. We measure proof size in bits by representing each ring element in $\mathbb{R}_q$ via its d coefficients in $\mathbb{Z}_q$, *i.e.*, $d \log q$ bits.

Proof Size. For rounds $i \in [1\colon \log N - 1]$, message π_i contains: $2\kappa\iota$ ring elements for $\pi_i^{(\mathsf{cm})}$, 2 ring elements for $\pi_i^{(\mathsf{lin})}$, 4 ring elements for $\pi_i^{(\mathsf{qua})}$, and $(2\kappa\iota + 10)$ ring elements per binary repetition (*cf.* Section 3.3). Thus, $|\pi_i| = \Big((2\kappa\iota + 6) + t \cdot (2\kappa\iota + 10)\Big) \cdot d \log q$. In the final round, the prover also sends $\vec{\mathbf{s}}_{\log N} \in \mathbb{R}_q^{2\ell}$ and (for each repetition) $\hat{\mathbf{s}}_{\alpha,\log N} \in \mathbb{R}_q^{2\ell\iota_\alpha}$, so $|\pi_{\log N}| = (2\ell + 2t\ell\iota_\alpha) \cdot d \log q$. Therefore, the total proof size is bounded by $\sum_{i=1}^{\log N - 1} |\pi_i| + |\pi_{\log N}| = O\Big(\big(\log N + \ell\big) \cdot (2\kappa\iota + 10) \cdot t\, d \log q\Big)$. Using the asymptotic parameters from Table 2 and the fact that $t \cdot \log q = O(\lambda)$, the total size is bounded by $O\big((\log N + \ell) \cdot \lambda^2\big)$ bits.

Prover Cost. In round $i \in [1\colon \log N]$, the prover updates folded witnesses of lengths $N\ell/2^{i-1}$ and (per binary repetition) $N\ell\iota_\alpha/2^{i-1}$. Each of the commitment, linear, and quadratic sub-proofs performs a constant number of ring operations per witness coordinate and per stored intermediate commitment. Summing over rounds yields a geometric series, giving total prover cost of $O\big(\ell N + t\ell N\big) = O(t\ell N)$ ring operations, up to constant factors depending on κ, ι and ι_α.

Table 3. Asymptotic Efficiency in terms of N, t, λ, and ℓ

Proof Size (in bits)	**Prover Cost** (in ring operations)	**Verification Cost** (in ring operations)
$O((\ell + \log N) \cdot \lambda^2)$	$O(t\ell N)$	$O((t+1)\log N + t\ell)$

Verifier Cost. Per round, the verifier performs a constant number of checks for (cm), (lin), and (qua), plus t copies of the binary checks, hence $O(t+1)$ ring operations per round. Over $\log N - 1$ rounds, this contributes $O((t+1)\log N)$ ring operations. In the final round, the verifier additionally processes $t+1$ vectors of length $\Theta(\ell)$, contributing $O(t\ell)$ ring operations.

The verifier also constructs tensor-decomposed exponent vectors. Using the decomposition from Section 3.3, for each repetition, it constructs base vectors of length $2\ell d$ (namely $\vec{\alpha}_0$, $\vec{r}_0$, and $\vec{r}_{\alpha,0}$), and then $\log N$ many length-2 vectors (namely $\vec{\alpha}_i$, $\vec{r}_i$, and $\vec{r}_{\alpha,i}$ for $i \geq 1$) via repeated squaring. Thus, across t repetitions, the total cost for challenge-vector construction is $O\big(t(\ell d + \log N)\big)$ field operations over $\mathbb{Z}_q$.

Putting everything together, the total verification cost is $O\big((t+1)\log N + t\ell\big)$ ring operations, plus $O\big(t(\ell d + \log N)\big)$ field operations for challenge-vector construction. Noting that one ring operation corresponds to $O(d \log d)$ field operations, the total verification cost can be expressed, in unified ring-operation units, as $O\big((t+1)\log N + t\ell\big)$.

Table 3 summarizes the asymptotic efficiency of the scheme.

4 Polynomial Commitment Scheme from Serval

We instantiate the protocol in Fig. 2 to construct PCSs for univariate and multilinear polynomials over $\mathbb{Z}_q$. We need to address the following tasks.

- Adapt polynomials over $\mathbb{Z}_q$ to the ring setting $\mathbb{R}_q$.
- Reduce univariate and multilinear evaluation constraints to linear constraints in $\mathcal{R}_{\mathsf{main}}$.
- Decompose large-norm coefficient vectors into small-norm witnesses compatible with the lattice setting.

Integer-to-Ring Mapping. Let $\vec{f} \in \mathbb{Z}_q^{Nd}$ be a coefficient vector. We map $\vec{f}$ to a ring vector $\vec{\mathbf{f}} \in \mathbb{R}_q^N$ by packing every d consecutive coefficients into one ring element. Concretely,

$$\vec{\mathbf{f}} = \Big(\sum_{i=0}^{d-1} f_i X^i,\ \sum_{i=0}^{d-1} f_{i+d} X^i,\ \ldots,\ \sum_{i=0}^{d-1} f_{(N-1)d+i} X^i\Big) \in \mathbb{R}_q^N.$$

Evaluation Constraint Reduction. We show that evaluation constraints reduce to linear constraints of the form in $\mathcal{R}_{\mathsf{main}}$.

Univariate. Let f be a univariate polynomial with coefficient vector $\vec{f} \in \mathbb{Z}_q^{Nd}$. For an evaluation point $u_{\mathsf{uni}} \in \mathbb{Z}_q$, the evaluation constraint is $\sum_{i=0}^{Nd-1} f_i \cdot u_{\mathsf{uni}}^i = v$. Equivalently, $\langle \vec{f}, \vec{u}_{\mathsf{uni}} \rangle = v$, $\vec{u}_{\mathsf{uni}} \coloneqq (1, u_{\mathsf{uni}}, \ldots, u_{\mathsf{uni}}^{Nd-1}) \in \mathbb{Z}_q^{Nd}$. Since N and d are powers of two, $\vec{u}_{\mathsf{uni}}$ admits the tensor decomposition $\vec{u}_{\mathsf{uni}} = \bigotimes_{i=0}^{\log N - 1} \vec{u}_{\mathsf{uni},i}$, where $\vec{u}_{\mathsf{uni},0} \coloneqq (1, u_{\mathsf{uni}}, \ldots, u_{\mathsf{uni}}^{2d-1}) \in \mathbb{Z}_q^{2d}$, $\vec{u}_{\mathsf{uni},i} \coloneqq (1, u_{\mathsf{uni}}^{2^i d}) \in \mathbb{Z}_q^2$, for $i = \{1, 2, \ldots, \log N - 1\}$. So, the evaluation constraint becomes

$$\langle \vec{f}, \vec{u}_{\mathsf{uni}} \rangle = \left\langle \mathsf{coef}(\vec{\mathbf{f}}), \bigotimes_{i=0}^{\log N - 1} \vec{u}_{\mathsf{uni},i} \right\rangle = v,$$

which matches the linear constraint form in $\mathcal{R}_{\mathsf{main}}$.

Multilinear. Let f be a multilinear polynomial over $\log(Nd)$ variables with coefficient vector $\vec{f} \in \mathbb{Z}_q^{Nd}$. For an evaluation point $(u_0, \ldots, u_{\log(Nd)-1}) \in \mathbb{Z}_q^{\log(Nd)}$, define the standard MLE vector $\vec{u}_{\mathsf{mul}} \coloneqq \bigotimes_{j=0}^{\log(Nd)-1} (1, u_j) \in \mathbb{Z}_q^{Nd}$. Then the evaluation constraint is $\langle \vec{f}, \vec{u}_{\mathsf{mul}} \rangle = v$. We regroup $\vec{u}_{\mathsf{mul}}$ into $\log N$ tensor factors as follows. Let $\vec{u}_{\mathsf{mul},0} \coloneqq \bigotimes_{j=0}^{\log(2d)-1} (1, u_j) \in \mathbb{Z}_q^{2d}$, $\vec{u}_{\mathsf{mul},i} \coloneqq (1, u_{\log(2d)+i-1}) \in \mathbb{Z}_q^2$ for $i \in [1\colon \log N]$. Then $\vec{u}_{\mathsf{mul}} = \bigotimes_{i=0}^{\log N - 1} \vec{u}_{\mathsf{mul},i}$, and hence $\langle \vec{f}, \vec{u}_{\mathsf{mul}} \rangle = \left\langle \mathsf{coef}(\vec{\mathbf{f}}), \bigotimes_{i=0}^{\log N - 1} \vec{u}_{\mathsf{mul},i} \right\rangle = v$, which matches the linear constraint form in $\mathcal{R}_{\mathsf{main}}$.

Large-Norm Decomposition. To fit the lattice setting, we decompose $\vec{\mathbf{f}}$ into a small-norm binary witness. Let $\ell \coloneqq \lceil \log q \rceil$. We apply base-2 decomposition coefficient-wise (modulo q) and obtain a witness vector $\vec{\mathbf{s}} \in \mathbb{R}_q^{N\ell}$ such that $\mathsf{coef}(\vec{\mathbf{s}}) \in \{0,1\}^{N\ell d}$ and $\|\vec{\mathbf{s}}\|_\infty \leq 1$. Under our norm definition, this implies $\|\vec{\mathbf{s}}\|_2^2 \leq N\ell d$, and hence $\|\vec{\mathbf{s}}\|_2 \leq \beta \coloneqq \sqrt{N\ell d}$. Accordingly, the evaluation constraint for $\vec{\mathbf{f}}$ can be replaced by an equivalent constraint over $\vec{\mathbf{s}}$ by absorbing the base-2 weights into the first tensor factor. We thus replace $\vec{u}_{\mathsf{uni},0}$ (resp. $\vec{u}_{\mathsf{mul},0}$) by

$$\vec{u}_{\mathsf{uni},0} \otimes (1, 2, 2^2, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^{2d\ell} \qquad (\text{resp. } \vec{u}_{\mathsf{mul},0} \otimes (1, 2, 2^2, \ldots, 2^{\ell-1}) \in \mathbb{Z}_q^{2d\ell}).$$

This yields a linear constraint on $\mathsf{coef}(\vec{\mathbf{s}})$ of the form required by $\mathcal{R}_{\mathsf{main}}$ with $b_0 = 2\ell d$ and $b_i = 2$ for all $i \in [1\colon \log N]$.

We can now invoke the protocol in Fig. 2 to prove knowledge of the decomposed witness $\vec{\mathbf{s}}$ and the claimed evaluation value.

The complete polynomial commitment scheme is in Fig. 4. We provide the batching support in Appendix E.

Theorem 4. *The PCS in Fig. 4 satisfies evaluation completeness, weak binding, and knowledge soundness.*

Proof. Theorem 1 establishes evaluation completeness, Theorem 2 guarantees (weak) binding, and Theorem 3 ensures knowledge soundness. □

The protocol can be made zero-knowledge by hiding the ℓ_2-norm of the witness and proving it using standard techniques such as range proofs.

Public Inputs:

$\mathsf{pp} = ((\mathbf{A}_i, \mathbf{B}_i \in \mathbb{R}_q^{\kappa \times 2\kappa\iota})_{i\in[1,\log N]},$
$\mathbf{A}_0 \in \mathbb{R}_q^{\kappa \times 2\lceil \log q \rceil}, \mathbf{B}_0 \in \mathbb{R}_q^{\kappa \times 2\lceil \log q \rceil \iota}).$

Private Inputs:

$f \in \mathbb{Z}_q^{<Nd}[X]$ or $f \in \mathbb{Z}_q[X_0, \ldots, X_{\log(Nd)-1}]$.

$\mathsf{Commit}\ (\mathsf{pp}, f) \to (\mathsf{cm}, \mathsf{st})$

Represent f as $\vec{f} = (f_0, \ldots, f_{Nd-1})$.

Compute $\mathbf{f}_i = \sum_{j=0}^{d-1} f_{id+j} X^j$, $i \in [N]$.

Let $\vec{\mathbf{f}} = (\mathbf{f}_0, \ldots, \mathbf{f}_{N-1}) \in \mathbb{R}_q^N$.

Employ unsigned base-2 on $\mathbf{f}$ to get $\vec{\mathbf{s}} \in \mathbb{R}_q^{N \cdot \lceil \log q \rceil}$.

Compute $(\mathsf{cm}, \mathsf{st}) = \mathsf{Com}(\vec{\mathbf{s}})$.

Return (cm, st).

$\mathsf{Open}\ (\mathsf{pp}, \mathsf{cm}, f, \mathsf{st}) \to \{0, 1\}$

Represent f as $\vec{f}$.

Construct $\vec{\mathbf{f}} = (\mathbf{f}_0, \ldots, \mathbf{f}_{N-1})$.

Employ unsigned base-2 on $\mathbf{f}$ to get $\vec{\mathbf{s}}$.

Return 1 iff $\mathsf{Com}(\vec{\mathbf{s}}) = (\mathsf{cm}, \mathsf{st})$.

$\mathsf{Eval}(\mathsf{pp}, \mathsf{cm}, u\ (\text{or } \vec{u} \in \mathbb{Z}_q^{\log(Nd)}), v; (f, \mathsf{st}))$

Compute $\vec{\mathbf{s}}$ based on f.

Construct $(\vec{u}_{\mathsf{uni},i})_{i\in[\log N]}$ (resp., $(\vec{u}_{\mathsf{mul},i})_{i\in[\log N]}$).

Let $\vec{a}_i = \vec{u}_{\mathsf{uni},i}$ (resp. $\vec{u}_{\mathsf{mul},i}$) for $i \in [\log N]$).

Run the protocol in Fig. 2 to prove $((\mathsf{cm}, \beta, v, b, (\vec{a}_i)_{i\in[\log N]}); \vec{\mathbf{s}})$.

Fig. 4. PCS constructed from Fig. 2

Theorem 5. *Using the parameters in Table 2 with $\ell = O(\log q)$, the PCS depicted in Fig. 4 achieves the following efficiency:*

- *Proof Size: $O(\lambda^2 \cdot \log^2 N)$ bits;*
- *Prover Cost: $O(N \cdot \lambda)$ ring operations;*
- *Verification Cost: $O(\log N \cdot \mathsf{poly}(\lambda))$ ring operations.*

Proof. The bounds follow directly from Table 3. Substituting $\ell = O(\log q)$ and $\log q = \Theta\Big(\frac{\log^2 N}{\log \lambda}\Big)$ yields the claimed bound on the proof size. The prover and verification costs follow by the relation $t \cdot \log q = O(\lambda)$ and $t = \Theta(\frac{\lambda}{\log \lambda})$. □

5 Optimization and Implementation

5.1 Proof Compaction

We present an optional optimization of Fig. 2 that reduces concrete proof size by aggregating the verifier's checks. The starting point is structural: Fig. 3 consists almost entirely of dot-product–shaped constraints over $\mathbb{R}_q$ (and a small number of linear-map consistency checks), and our protocol already supplies the tensor and decomposition machinery needed to express them uniformly. We therefore aggregate all per-round verification constraints into a single LaBRADOR instance [9], yielding a proof-of-proof.

This optimization enforces two requirements. First, the $\log N$-round split-and-fold phase must be non-interactive so that the full set of checks is fixed before aggregation. Second, each verifier check must be written in the LaBRADOR relation format (quadratic dot products plus linear terms).

Fiat–Shamir transform. We apply Fiat–Shamir to the recursive phase after the prover sends (v_r, v_{r_α}). The prover derives each folding challenge $\vec{c}_i \in \mathcal{C}^2$ as a deterministic function of the transcript prefix up to round $i-1$. We include the round index and a domain separator in the hash input to avoid cross-round collisions. To ensure that $\vec{c}_i \in \mathcal{C}^2$ (satisfying invertibility and coefficient/norm constraints), we design the challenge-generation algorithm shown in Algorithm 1, which enforces these requirements.

In our instantiation, $\mathcal{C} \subset \mathbb{R}_q$ consists of ring elements with coefficients in $\{0, \pm 1, \pm 2\}$ and an ℓ_∞-operator norm bounded by T. Let $\mathsf{Com}(\mathsf{tr}_{i-1})$ denote the Ajtai commitment to the transcript prefix up to round $i-1$, *i.e.*, $\mathsf{Com}(\mathsf{tr}_{i-1}) = \mathsf{Com}\big(\mathsf{pp}, \mathsf{cm}, (\mathsf{cm}_\alpha^{(j)})_{j\in[t]}, \pi_1, \ldots, \pi_{i-1}\big)$.

We claim that all challenges generated by Algorithm 1 belong to $\mathcal{C}^2$, satisfying invertibility and holding an acceptable ℓ_∞ operator norm.

Invertibility is ensured by the choice of a suitable q in our main parameter setting. For the operator-norm condition, it can be proved that $\|c\|_{\mathsf{op},\infty} = \|c\|_1$. Specifically, by Young's inequality for negacyclic convolution, multiplication by c satisfies $\|c\mathbf{v}\|_\infty \le \|c\|_1 \cdot \|\mathbf{v}\|_\infty$, and therefore $\|c\|_{\mathsf{op},\infty} = \|c\|_1$.

In our concrete instantiation (Section 5.2), we set the operator-norm bound $T = 48$, and all challenge elements output by Algorithm 1 satisfy $\|c\|_1 = 32 + 8 \cdot 2 = 48$, thereby fulfilling the operator-norm requirement.

Consequently, the verifier can reconstruct the same $\vec{c}_i$ from the committed prefix without repeating rejection sampling, guaranteeing that the check is performed with the identical challenge.

Algorithm 1 Folding Challenge Generation in FS

$\mu_i \leftarrow \mathcal{H}(\mathsf{Com}(\mathsf{tr}_{i-1}))$; $\rho_i \leftarrow \mathsf{PRG}(\mu_i)$;
for $j \in \{0,1\}$ **do**
  $A \leftarrow [\underbrace{0,\ldots,0}_{24},\underbrace{1,\ldots,1}_{32},\underbrace{2,\ldots,2}_{8}]$; // the absolute-value multiset
  for $k = d-1 \to 1$ **do** // Fisher–Yates shuffle using bits from ρ_i
    $x \leftarrow$ next bits from ρ_i interpreted as an integer in $[0,k]$;
    swap $A[k] \leftrightarrow A[x]$;
  endfor
  for $k = 0 \to d-1$ **do** // assign random signs
    if $A[k] = 0$ then $c_{i,j,k} \leftarrow 0$; else $s \leftarrow$ next bit from ρ_i;
    $c_{i,j,k} \leftarrow A[k] \cdot (-1)^s$;
  endfor
  $c_{i,j} \leftarrow \sum_{k=0}^{d-1} c_{i,j,k} X^k \in \mathbb{R}_q$;
endfor
return $\vec{c}_i = (c_{i,0}, c_{i,1})$;

LaBRADOR Compaction. We now satisfy the second requirement by encoding the verifier's checks as a single LaBRADOR relation. Fix the (now non-interactive) proof transcript

$$\pi = \big(\pi_{\mathsf{pre}}, \pi_{\mathsf{inter}}, (\mathsf{Com}(\mathsf{tr}_i))_{i\in[1,\log N]}\big), \qquad \pi_{\mathsf{inter}} = (\pi_1,\ldots,\pi_{\log N}).$$

Each round-i verification condition in Fig. 3 is one of the following types.
(1) A dot product over $\mathbb{R}_q$ (*e.g.*, linear recursion and the quadratic recursion written as a linear combination of $(\mathbf{L}, \mathbf{M}^{(1)}, \mathbf{M}^{(2)}, \mathbf{R})$).
(2) A linear-map consistency check $\mathbf{A}\cdot\mathbf{x} = \mathbf{y}$, which can be written as a set of dot-product constraints by expanding coordinates against standard basis vectors.
(3) A final-round norm admissibility check, which we reduce to dot-product constraints by (i) proving a self inner product bound and (ii) enforcing a binary decomposition of the opened coefficients via Section 3.3.

We also need to enforce transcript-seed consistency: the committed seeds $(\mathsf{Com}(\mathsf{tr}_i))$ must match the transcript blocks used to derive Fiat–Shamir challenges. Since Ajtai commitments are linear, these consistency checks are also linear constraints and therefore can be reduced to dot products.

Let W denote the concatenation of all witness blocks required to express the conjunction of the above constraints, after applying the same base decompositions to any large-norm components that appear in $\pi_{\log N}$. The prover produces a LaBRADOR proof π_{L} attesting that W satisfies the resulting system of quadratic

dot-product constraints. The aggregated relation has the LaBRADOR form

$$\sum_{i,j} \alpha_{i,j} \langle \vec{\mathbf{z}}_i, \vec{\mathbf{z}}_j \rangle + \sum_i \langle \vec{\varphi}_i, \vec{\mathbf{z}}_i \rangle - \mathbf{b} = 0,$$

where the blocks $(\vec{\mathbf{z}}_i)$ are slices of W and the coefficients $(\alpha_{i,j}, \vec{\varphi}_i, \mathbf{b})$ are verifier-computable from public inputs and the transcript.

Let M be the total number of ring elements in W. Under this encoding, LaBRADOR yields a proof whose size is $O(\log \log M)$ ring elements [9]. In our setting, M is dominated by the per-round transcript blocks in π_{inter} and the decomposed final openings, and it grows as $M = \Theta\big((t+1) \cdot \log N\big) + \Theta\big(t\ell \cdot \log \gamma_\alpha + \ell \cdot \log \gamma\big)$ which is $O(\lambda \cdot \log N)$ under our PCS instantiation using the parameters from Table 2. The resulting compact proof sent to the verifier is $\pi = \big(\pi_{\mathsf{pre}}, \pi_{\mathrm{L}}, (\mathsf{Com}(\mathsf{tr}_i))_{i \in [1, \log N]}\big)$.

This transformation moves security to the random oracle model (via Fiat–Shamir) while preserving the underlying module-SIS binding and extraction guarantees. We analyze the soundness of the optimized protocol in Appendix D.

5.2 Implementation

We provide a concrete instantiation and baseline benchmarks. We first state the parameters used in our prototype and then report timing and size measurements.

Concrete Parameters. For security parameter $\lambda = 128$, we select parameters satisfying Table 2. We report communication in bytes under the resulting $\log q$-bit encoding of ring elements, so that all concrete comparisons fix λ rather than hiding security dependence in asymptotics. We instantiate $\mathcal{C}$ as ring elements with 24 zero coefficients, 32 coefficients in $\{\pm 1\}$, and 8 coefficients in $\{\pm 2\}$. The number of such challenges is $\binom{64}{32} \cdot \binom{32}{8} \cdot 2^{40} \approx 2^{124.0}$. We set $d = 64$ and $T = 48$; by construction, the challenge sampler ensures $\|c\|_{\mathsf{op},\infty} \leq \|c\|_1 = 48$ (via Young's inequality), so the required operator-norm bound holds.

Table 4 reports proof sizes as (i) the baseline protocol and (ii) an *estimated* post-compaction size under LaBRADOR-style aggregation. The post-compaction sizes are computed analytically from the proof-size formulas/tables of LaBRADOR [9] together with our $\log q$-bit ring-element encoding, and are not included in the measured microbenchmarks.

If one restricts $\mathcal{C}$ further (*e.g.*, by tightening the operator-norm target), the entropy of a single challenge decreases and can be compensated by repetition at the protocol level. Table 4 illustrates this tradeoff for larger degree bounds.

Prototype and benchmarking setup. Our prototype is implemented on an Apple Silicon platform (M2, 16 GB unified memory). The implementation code and a subset of the raw experimental data are available at `https://gitlab.com/LatticePCSReview/servalPCS`.

Table 4. Concrete parameter choices and proof sizes for polynomial length $L = 2^e$

	KB ($e \leq 20$)						**MB** ($e \geq 22$)				
e	**15**	**16**	**17**	**18**	**19**	**20**	**22**	**24**	**26**	**28**	**30**
$\log q$	88	100	108	120	128	128	84	94	104	114	124
d	64	64	64	64	64	64	64	64	64	64	64
κ	13	13	13	13	13	13	13	13	13	13	13
ι	5	5	5	5	5	8	5	5	5	5	5
Raw[a]	2.01	2.40	2.73	3.17	3.54	5.45	11.27	14.14	17.22	20.72	24.43
Opt.	259.16	302.88	337.03	385.22	423.03	436.03	1.97	2.24	2.73	3.15	3.59

[a] Raw sizes are in MB.
Optimized (Opt.) sizes are in KB for $e \leq 20$ and in MB for $e \geq 22$.

The current code covers the baseline interactive protocol in Fig. 2. We do not integrate LaBRADOR compaction in this prototype because the available LaBRADOR implementation[4] is not yet stable enough on our platform.

We benchmark prover time, verifier time, and proof size for the commitment-constraint proof, the linear-constraint proof, and the quadratic-constraint proof. We treat the binary-property proof as a composition of these three components and therefore omit it from the microbenchmarks. To accommodate varying N in a single build, we use a fixed q of 100 bits, so the measured sizes are not optimized per-N by modulus tuning. We use small-coefficient challenges in $\{0, \pm 1\}$ with an operator-norm target $T = 9$. To target at least 90-bit security in this configuration, we run the full protocol twice via standard parallel repetition.

Our implementation uses NTT-based multiplication for ring operations and batched matrix-vector multiplication. We do not use SIMD-specific optimizations or multi-threaded accumulation in this baseline, so the results should be read as conservative reference measurements.

Figure 5 reports prover time, verifier time, and proof size for the three proof components. Prover time grows as expected with N under fixed security, reflecting $O(N)$ ring work for folding and auxiliary multiplications. Verification is dominated by the commitment proof due to matrix-vector multiplications, while the linear and quadratic checks are comparatively lightweight. Proof size is largest for the commitment proof because it carries the folded intermediate commitment state.

Comparison with Greyhound. We compare against Greyhound [36], which provides a public implementation and is representative of lattice-based PCS evaluation proofs. Our hardware does not support the same SIMD instruction set used by the Greyhound implementation, so we do not compare wall-clock timings across codebases. Instead, we compare ring-operation counts derived from each

[4] `https://github.com/lattice-dogs/labrador`

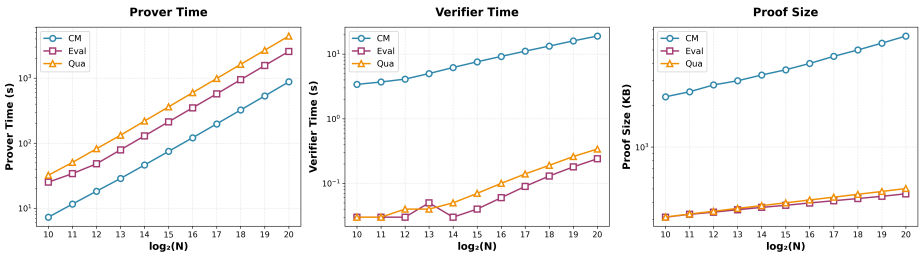


Fig. 5. Benchmark comparison of commitment, linear, and quadratic proofs

Table 5. Ring-operation counts for Greyhound [36] and Serval

	Prover ops		**Verifier ops**	
Phase	[36]	**Ours**	[36]	**Ours**
Init.	$N + \kappa\sqrt{N}$	$N\left(8\ell + (\iota+1)\frac{\kappa+\ell}{2}\right)$	–	$4\kappa^2\iota + 12$
Mid.	–	$N\left(4\ell + (\iota+1)\frac{\kappa+\ell}{4}\right)$	–	$(8\kappa^2\iota + 28)(\log N - 2)$
Final	N	$2\ell(1+\iota)$	$\sqrt{N}\left(\iota(\kappa^2+2\kappa+4)\right.$ $\left.+\kappa+1\right)$	$\ell(12+2\kappa+4\iota$ $+2\kappa\iota)+16$

protocol's specification and cross-check them against the reference implementation, focusing on evaluation-proof routines. We report these counts in Table 5 under a common choice of (q, d) and comparable challenge distributions.

Figure 6 plots the resulting prover and verifier ring-operation counts as a function of N. For small N, our verifier can be modestly more costly due to explicit norm checks and the smaller gap between $\sqrt{N}$ and $\log N$. As N grows, the verifier advantage becomes pronounced, matching the asymptotic polylogarithmic verification cost of Serval. On the prover side, the main overhead comes from the base-2 decomposition factor ℓ, and the ratio is correspondingly stable.

6 Conclusion

Serval (*Slack-free Euclidean Radius Verification for Algebraic Lattices*), is a post-quantum PCS that gives succinct evaluation proofs with a *slack-free* ℓ_2 binding guarantee under standard M-SIS and a transparent setup. Like its namesake, Serval targets the bound sharply: it enforces the Euclidean radius *without slack*. Our design reduces commitment consistency, evaluation correctness, and lattice smallness to one inner-product-shaped relation over $\mathbb{R}_q$, so a single $\log N$ split-and-fold engine drives the entire protocol and yields polylogarithmic verification with essentially linear proving.

The crux is an exact norm mechanism that fits this recursion. We decompose slack-free ℓ_2 soundness into a ring self-inner-product claim that prevents modular wraparound and a binary coefficient constraint that pins ℓ_∞ exactly, and we

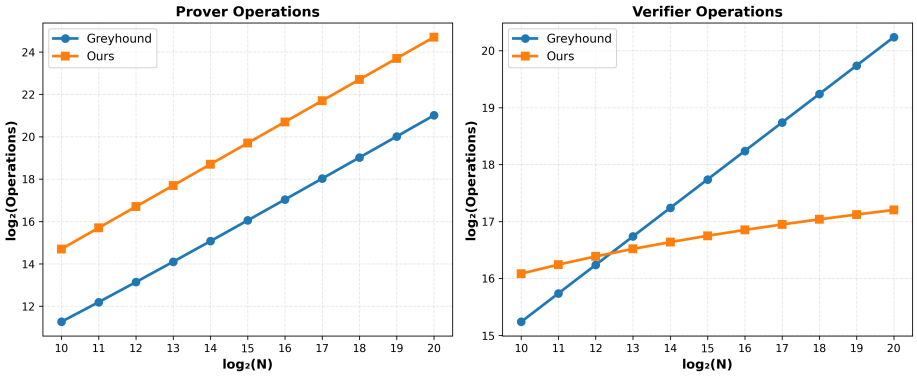


Fig. 6. Ring-operation counts for Greyhound and Serval (y-axis in log scale)

express binarity itself via randomized inner-product checks with tensor structure. To preclude adaptive fabrication, the prover commits to the derived auxiliary witness before the verifier samples the follow-on challenges. Because every check shares the same recursion and state, the composed protocol avoids a separate range-proof subsystem and stays naturally batching- and recursion-friendly.

Under the stated parameter regime (prime q with the required invertibility condition for challenges, and $q > 2N\ell d$ for the exact-norm reduction), soundness follows from coordinate-wise special soundness of the shared recursion plus the negligible error from t binary-check repetitions. In this scope, our concrete comparison shows competitive proof sizes and fast verification while providing the stronger no-slack norm guarantee that prior post-quantum PCSs typically relax. An immediate direction is tighter constant-alphabet range enforcement that preserves the same $\log N$ engine, and a full end-to-end instantiation with Fiat–Shamir and LaBRADOR-style compaction to reduce constants in practice.

Acknowledgment

We used generative AI tools (*e.g.*, ChatGPT) for phrasing and grammar correction. All technical content, data, code, and conclusions are by the authors, who have reviewed the manuscript and take full responsibility for it.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC. pp. 99–108 (1996). https://doi.org/10.1145/237814.237838
2. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.K.: Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In: CRYPTO Part II. pp. 102–132 (2022). https://doi.org/10.1007/978-3-031-15979-4_4
3. Albrecht, M.R., Fenzi, G., Lapiha, O., Nguyen, N.K.: SLAP: Succinct lattice-based polynomial commitments from standard assumptions. In: EUROCRYPT Part VI. pp. 90–119 (2024). https://doi.org/10.1007/978-3-031-58754-2_4
4. Attema, T., Lyubashevsky, V., Seiler, G.: Practical product proofs for lattice commitments. In: CRYPTO Part II. pp. 470–499 (2020)
5. Backes, M., Datta, A., Kate, A.: Asynchronous computational VSS with reduced communication complexity. In: CTRSA. pp. 259–276 (2013). https://doi.org/10.1007/978-3-642-36095-4_17
6. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: SODA. pp. 10–24 (2016), https://doi.org/10.1137/1.9781611974331.ch2
7. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast Reed-Solomon interactive oracle proofs of proximity. In: ICALP. pp. 14:1–14:17 (2018). https://doi.org/10.4230/LIPICS.ICALP.2018.14
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptol. ePrint Arch. 2018/046 (2018)
9. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from module-SIS. In: CRYPTO Part V. pp. 518–548 (2023). https://doi.org/10.1007/978-3-031-38554-4_17
10. Bhadauria, R., Hazay, C., Venkitasubramaniam, M., Wu, W., Zhang, Y.: Private polynomial commitments and applications to MPC. In: PKC Part II. pp. 127–158 (2023). https://doi.org/10.1007/978-3-031-31371-4_5
11. Bootle, J., Chiesa, A., Sotiraki, K.: Lattice-based succinct arguments for NP with polylogarithmic-time verification. In: CRYPTO Part II. pp. 227–251 (2023). https://doi.org/10.1007/978-3-031-38545-2_8
12. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: A non-PCP approach to succinct quantum-safe zero-knowledge. In: CRYPTO Part II. pp. 441–469 (2020). https://doi.org/10.1007/978-3-030-56880-1_16
13. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: S&P. pp. 315–334 (2018). https://doi.org/10.1109/SP.2018.00020
14. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: EUROCRYPT Part I. pp. 677–706 (2020). https://doi.org/10.1007/978-3-030-45721-1_24
15. de Castro, L., Peikert, C.: Functional commitments for all functions, with transparent setup and from SIS. In: EUROCRYPT Part III. pp. 287–320 (2023). https://doi.org/10.1007/978-3-031-30620-4_10
16. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: EUROCRYPT Part II. pp. 499–530 (2023). https://doi.org/10.1007/978-3-031-30617-4_17

17. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT Part I. pp. 738–768 (2020). https://doi.org/10.1007/978-3-030-45721-1_26
18. Cini, V., Lai, R.W.F., Malavolta, G.: Lattice-based succinct arguments from vanishing polynomials - (extended abstract). In: CRYPTO Part II. pp. 72–105 (2023). https://doi.org/10.1007/978-3-031-38545-2_3
19. Cini, V., Malavolta, G., Nguyen, N.K., Wee, H.: Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. In: CRYPTO Part X. pp. 207–242 (2024). https://doi.org/10.1007/978-3-031-68403-6_7
20. Esgin, M.F., Nguyen, N.K., Seiler, G.: Practical exact proofs from lattices: New techniques to exploit fully-splitting rings. In: Moriai, S., Wang, H. (eds.) ASIACRYPT Part II. pp. 259–288 (2020)
21. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In: Crypto Part I. pp. 115–146 (2019)
22. Fenzi, G., Moghaddas, H., Nguyen, N.K.: Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. J. Cryptol. **37**(3), 31 (2024). https://doi.org/10.1007/S00145-024-09511-8
23. Fisch, B., Liu, Z., Vesely, P.: Orbweaver: Succinct linear functional commitments from lattices. In: CRYPTO Part II. pp. 106–131 (2023). https://doi.org/10.1007/978-3-031-38545-2_4
24. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: CRYPTO Part I. pp. 75–92 (2013), https://doi.org/10.1007/978-3-642-40041-4_5
25. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: CRYPTO Part II. pp. 193–226 (2023). https://doi.org/10.1007/978-3-031-38545-2_7
26. Hall-Andersen, M., Simkin, M., Wagner, B.: Foundations of data availability sampling. IACR Commun. Cryptol. **1**(4), 34 (2024), https://doi.org/10.62056/a09qudhdj
27. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT. pp. 177–194 (2010). https://doi.org/10.1007/978-3-642-17373-8_11
28. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr. **75**(3), 565–599 (2015). https://doi.org/10.1007/S10623-014-9938-4
29. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: TCC Part II. pp. 1–34 (2021). https://doi.org/10.1007/978-3-030-90453-1_1
30. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: PKC. pp. 107–124 (2013). https://doi.org/10.1007/978-3-642-36362-7_8
31. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM **39**(4), 859–868 (1992). https://doi.org/10.1145/146585.146605
32. Lyubashevsky, V.: Lattice signatures without trapdoors. In: EUROCRYPT. pp. 738–755 (2012). https://doi.org/10.1007/978-3-642-29011-4_43
33. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: CRYPTO Part II. pp. 71–101 (2022). https://doi.org/10.1007/978-3-031-15979-4_3

34. Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: EUROCRYPT Part I. pp. 204–224 (2018). https://doi.org/10.1007/978-3-319-78381-9_8
35. Micciancio, D., , Regev, O.: Lattice-based cryptography. In: Post-Quantum Cryptography, pp. 147–191. Springer (2009). https://doi.org/10.1007/978-3-540-88702-7_5
36. Nguyen, N.K., Seiler, G.: Greyhound: Fast polynomial commitments from lattices. In: CRYPTO Part X. pp. 243–275 (2024). https://doi.org/10.1007/978-3-031-68403-6_8
37. Stern, J.: A new identification scheme based on syndrome decoding. In: CRYPTO. pp. 13–21 (1993). https://doi.org/10.1007/3-540-48329-2_2
38. Wee, H., Wu, D.J.: Succinct vector, polynomial, and functional commitments from lattices. In: EUROCRYPT Part III. pp. 385–416 (2023). https://doi.org/10.1007/978-3-031-30620-4_13
39. Zeilberger, H., Chen, B., Fisch, B.: BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: CRYPTO Part X. pp. 138–169 (2024). https://doi.org/10.1007/978-3-031-68403-6_5

A Additional Related Works

We provide additional context for Table 1 by summarizing representative post-quantum PCS frameworks and where norm-soundness slack enters their analyses. This section is intended as a high-level comparison along the axes used in Table 1 (basis/setup/slack/proof size); for precise statements, we refer to the cited works.

Inner-product-based approaches. A common lattice-based route commits to the coefficient vector and proves evaluation by reducing it to an inner product $\langle \vec{f}, \vec{x} \rangle$ with $\vec{x} = (1, x, \ldots, x^{L-1})$ in the univariate case [2,15,38,18]. These schemes typically yield logarithmic-size evaluation proofs (measured in $\mathbb{Z}_q$ or ring elements), while at least one party performs work linear in L due to commitment consistency or binding checks. Some works reduce *online* verification via preprocessing or delegation (*e.g.*, Dory-style techniques or structured evaluation support), at the cost of auxiliary keys/tables or stronger setup requirements [29,11,23]. Along the norm axis, most inner-product-based lattice PCSs guarantee extraction only up to a multiplicative slack, or tighten slack by strengthening the underlying assumption (*e.g.*, k-SIS) rather than by proving an exact ℓ_2 bound under standard SIS/LWE-type assumptions [23,2,18].

Recursive approaches. Recursive (split-and-fold) PCSs verify an evaluation proof by a smaller proof system, which is itself verified recursively, yielding a polylogarithmic object checked by the verifier [7]. Optimized variants achieve very small verifier time but often rely on large auxiliary data and/or structured assumptions, and their norm guarantees follow from the underlying extractor for the commitment primitive and thus inherit multiplicative slack [22,3]. Consequently, these constructions typically do not provide norm-preserving extraction under standard SIS/LWE-type assumptions.

Hybrid approaches. Hybrid designs combine folding with proof-of-proof compaction. Greyhound [36] builds on Brakedown [25] and achieves sublinear verification and sub-logarithmic proof via recursive proof-of-proof inner-product arguments, using LaBRADOR-style compaction to compress intermediate proofs. Cini *et al.* [19] propose a transparent SIS-based PCS (univariate and multilinear) that adopts FRI-style folding as in Basefold [39], achieving (poly-)logarithmic verification time and logarithmic proof size without third-party compaction. Both lines integrate modern norm-proof techniques and obtain a small constant slack (*e.g.*, ≈ 2) in the stated norm guarantee for the extracted object, which suffices for many one-shot binding applications but falls short of slack-free ℓ_2 soundness.

Norm proofs. Tight coefficient-set checks (and thus tight ℓ_∞ bounds for very small sets) can be enforced via algebraic identities such as $\vec{f} \circ (\vec{f} - \vec{1}) \circ (\vec{f} + \vec{1}) = \vec{0}$ together with structured transforms [21,20,4]. For ℓ_2 bounds, the norm-proof paradigm of Lyubashevsky [32] decomposes the constraint into a quadratic relation together with an auxiliary bound, and more recent techniques reduce the resulting slack to a small constant using modular Johnson–Lindenstrauss arguments [9]. Our work differs in targeting slack-free ℓ_2 soundness within a PCS evaluation-proof framework under standard SIS/LWE-type assumptions, as reflected in Table 1.

B Additional Preliminaries

B.1 Interactive Proofs

Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ be a ternary relation. For $(\mathsf{pp}, x, w) \in \mathcal{R}$, we call pp the public parameters, x the statement, and w a witness. For fixed pp and x, we write $\mathcal{R}(\mathsf{pp}, x) \coloneqq \{\, w \colon (\mathsf{pp}, x, w) \in \mathcal{R} \,\}$.

Definition 3 (Interactive proof system). *An interactive proof system $\Pi = (\mathcal{S}, \mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ has a probabilistic polynomial-time (PPT) setup algorithm $\mathcal{S}$ and an interactive protocol between a PPT prover $\mathcal{P}$ and a verifier $\mathcal{V}$. $\mathcal{P}$ convinces $\mathcal{V}$ that there exists w with $(\mathsf{pp}, x, w) \in \mathcal{R}$. Both $\mathcal{P}$ and $\mathcal{V}$ receive x and public parameters $\mathsf{pp} \leftarrow \mathcal{S}(1^\lambda)$. $\mathcal{P}$ also holds a private witness $w \in \mathcal{R}(\mathsf{pp}, x)$. At the end, $\mathcal{V}$ outputs accept or reject. The transcript is the sequence of messages exchanged.*

Definition 4 (Completeness). *A proof system $\Pi = (\mathcal{S}, \mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}$ satisfies completeness with completeness error $\epsilon(\cdot)$ if for all adversaries $\mathcal{A}$,*

$$\Pr\left[b = 0 \wedge (\mathsf{pp}, x, w) \in \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (tr, b) \leftarrow \langle \mathcal{P}(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x) \rangle \end{array} \right] = \epsilon(\lambda) + \mathsf{negl}(\lambda).$$

Furthermore, Π is said to satisfy perfect completeness whenever $\epsilon = 0$.

Definition 5 (Knowledge Soundness). *A proof system $\Pi = (\mathcal{S}, \mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}$ is knowledge sound with knowledge error $\varepsilon(\cdot)$ if there exists an expected PPT extractor $\mathcal{E}$ such that for any stateful PPT adversary $\mathcal{P}^*$:*

$$\Pr\left[b = 1 \wedge (\mathsf{pp}, x, w) \notin \mathcal{R} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda) \\ (x, w, \mathsf{st}) \leftarrow \mathcal{P}^*(\mathsf{pp}) \\ (tr, b) \leftarrow \langle \mathcal{P}^*(\mathsf{pp}, x, w), \mathcal{V}(\mathsf{pp}, x)\rangle \\ w \leftarrow \mathcal{E}_{\mathcal{P}^*}(\mathsf{pp}, tr, x) \end{array}\right] = \varepsilon(\lambda) + \mathsf{negl}(\lambda).$$

The extractor $\mathcal{E}$ has black-box oracle access to the (malicious) prover $\mathcal{P}^*$ and can rewind it to any point in the interaction.

B.2 Coordinate-wise Special Soundness

Coordinate-wise special soundness generalizes special soundness, and implies knowledge soundness for a $(2\mu+1)$-round interactive proof system [22].

Let $x = (x_1, x_2, \ldots, x_\ell)$, $y = (y_1, y_2, \ldots, y_\ell) \in S^\ell$. For $i \in [\ell]$, define $\equiv_i$ so that $\vec{x} \equiv_i \vec{y}$ means $\vec{x}$ and $\vec{y}$ agree in every coordinate except the i-th, *i.e.*,

$$\vec{x} \equiv_i \vec{y} \Leftrightarrow (x_i \neq y_i \wedge \forall j \in [\ell] \setminus \{i\}, x_j = y_j).$$

Next, for $K = \ell(k-1)+1$, define the set:

$$SS(S, \ell, k) \coloneqq \left\{ \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_K\} \subseteq (S^\ell)^K \;\middle|\; \begin{array}{l} \exists e \in [K],\, \forall i \in [\ell], \\ \exists J_i \subseteq [K] \setminus \{e\}, |J_i| = k-1, \\ \forall j \in J_i,\ \vec{x}_e \equiv_i \vec{x}_j \end{array} \right\}.$$

Equivalently, there is a "central" vector $\vec{x}_e$ such that for each coordinate i, there are $k-1$ other vectors that differ from it only at coordinate i; thus, for each i, the k vectors $\{\vec{x}_e\} \cup \{\vec{x}_j : j \in J_i\}$ vary only in the i-th coordinate.

Definition 6 (ℓ-Coordinate-wise k-Special Soundness (Multi-round)). *Let $\Pi = (\mathcal{S}, \mathcal{P}, \mathcal{V})$ be a public-coin $(2\mu+1)$-round interactive proof for a relation $\mathcal{R}$. In each round the verifier samples a challenge $c \in \mathcal{C} = S^\ell$.*

Consider a full $(\ell(k-1)+1)$-ary tree of depth μ whose outgoing edges at every internal node are labeled by $\ell(k-1)+1$ distinct challenges that, as a set, lie in $\mathsf{SS}(S, \ell, k)$. Each root-to-leaf path is one transcript, so the tree collects $K = (\ell(k-1)+1)^\mu$ transcripts for the same statement x.

Π has ℓ-coordinate-wise k-special soundness if there exists a polynomial-time extractor that, given x and any such transcript tree for x, outputs a witness w with $(x, w) \in \mathcal{R}$. This reduces to the standard k-special soundness when $\ell = 1$.

Lemma 3 ([22, Lemma 2.31]). *Let $\Pi = (\mathcal{S}, \mathcal{P}, \mathcal{V})$ be a public-coin $(2\mu+1)$-round interactive proof system for relation $\mathcal{R}$ and S^ℓ be the challenge space of $\mathcal{V}$ in each round. If Π is ℓ-coordinate-wise k-special sound and $(\ell(k-1))^\mu = \mathsf{poly}(\lambda)$, it is knowledge sound with knowledge error $\mu\ell(k-1)/|S|$.*

B.3 Polynomial Commitment Schemes

Definition 7 (Polynomial Commitment Scheme (PCS)). *A PCS* PC *over message space* $\mathcal{M} = \mathbb{Z}_q^{<N}[X]$ *follows the syntax below:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: Take security parameter* λ *and output public parameter* pp*.*
- $\mathsf{Commit}(\mathsf{pp}, f) \to (\mathsf{cm}, \mathsf{st})$*: Take public parameters* pp*, a message* f*, output a commitment* cm *and de-commitment state* st*.*
- $\mathsf{Open}(\mathsf{pp}, \mathsf{cm}, \mathsf{st}, f) \to \{0, 1\}$*: Take public parameters* pp*, a commitment* cm*, a message* f*, a de-commitment state* st *and output a bit indicating whether* cm *is a valid commitment to* f *under* pp*.*
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{cm}, x, y; f, \mathsf{st})$ *is a protocol to let prover* $\mathcal{P}$ *convince verifier* $\mathcal{V}$ *that* f *is an opening to commitment* cm *and* $f(x) = y$ *(or* $\vec{x}$ *in the multilinear case.* $\mathcal{P}$ *also knows a polynomial* f *and the de-commitment state* st*).*

A PCS must satisfy three main properties below.

Definition 8 (Evaluation Completeness). *A PCS* PC *satisfies completeness if for any polynomial* f *and evaluation point* $u \in \mathbb{Z}$*:*

$$\Pr\left[\begin{array}{l|l}\mathsf{Eval}.\mathcal{V}(\mathsf{pp}, \mathsf{cm}, & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ u, f(u), \pi) = 0 & (\mathsf{cm}, \mathsf{st}) \leftarrow \mathsf{Commit}(\mathsf{pp}, f) \\ & \pi \leftarrow \mathsf{Eval}.\mathcal{P}(\mathsf{pp}, \mathsf{cm}, u, f(u); f, \mathsf{st})\end{array}\right] = \mathsf{negl}(\lambda).$$

Definition 9 (Binding). *A polynomial commitment scheme* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *satisfies binding if for every PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{c|l}f \neq f' \wedge & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{Open}(\mathsf{pp}, \mathsf{cm}, f, \mathsf{st}) & (\mathsf{cm}, f, \mathsf{st}, f', \mathsf{st}') \leftarrow \mathcal{A}(\mathsf{pp}) \\ = \mathsf{Open}(\mathsf{pp}, \mathsf{cm}, f', \mathsf{st}') = 1 & \end{array}\right] = \mathsf{negl}(\lambda).$$

Definition 10 (Knowledge Soundness). *A polynomial commitment scheme* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *is knowledge sound with knowledge error* $\varepsilon(\cdot)$ *if for all stateful PPT adversary* $\mathcal{P}^*$*, there exists an expected PPT extractor* $\mathcal{E}$ *where:*

$$\Pr\left[\begin{array}{l|l}o = \mathsf{Open}(\mathsf{pp}, \mathsf{cm}, f, \mathsf{st}) & \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda) \\ b = 1 \wedge (o \neq 1 \vee f(u) \neq z) & (\mathsf{cm}, u, z, \pi) \leftarrow \mathcal{P}^*(\mathsf{pp}) \\ & b \leftarrow \mathsf{Eval}.\mathcal{V}(\mathsf{pp}, \mathsf{cm}, u, z, \pi) \\ & (f, \mathsf{st}) \leftarrow \mathcal{E}_{\mathcal{P}^*}(\mathsf{pp}, \mathsf{cm}, u, z, \pi)\end{array}\right] = \mathsf{negl}(\lambda).$$

C Proof For Theorem 2

Theorem 6 (Coordinate-wise Special Soundness). *Assume that the* M-SIS *problem is hard for rank* κ *and norm bound*

$$\max\left(N \cdot \sqrt{N\ell}\,\gamma_{\log N},\ N \cdot \sqrt{N\ell\,\iota_\alpha}\,\gamma_{\alpha,\log N}\right),$$

where $\gamma_{\log N}$ *and* $\gamma_{\alpha,\log N}$ *are defined as in the proof of Theorem 1.*

Then Fig. 2 achieves 2*-coordinate-wise* 2*-special soundness with an additional soundness error of* $\frac{4\log N}{|\mathcal{C}|} + (\frac{2N\ell d}{q})^t$*.*

Proof. We prove with the following steps.

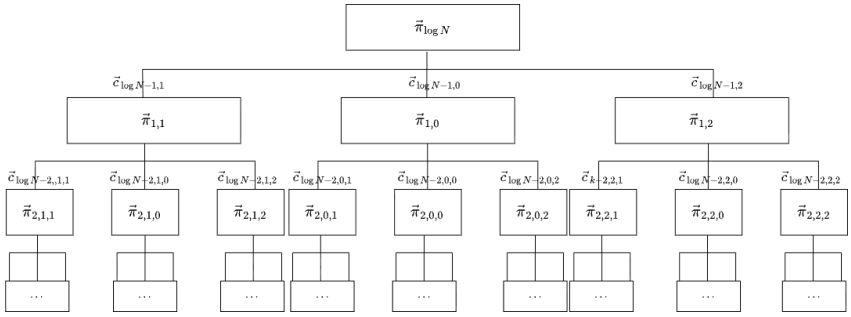


Fig. 7. A tree of accepted transcripts

I. Extract the weak commitment opening. Given a tree of accepted transcripts of size $K = 3^{\log N - 1}$ as specified in Definition 6 and organized as in Figure 7, we can extract a relaxed opening $(\bar{\mathbf{s}}, \bar{\mathbf{s}}_\alpha)$ where

$$\mathsf{Com}(\bar{c} \cdot \bar{\mathbf{s}}) = \bar{c} \cdot \mathsf{cm} \quad \text{and} \quad \|\bar{c}\bar{\mathbf{s}}\|_\infty \leq 2^{\log N - 1}\gamma_{\log N},$$
$$\mathsf{Com}(\bar{c} \cdot \bar{\mathbf{s}}_\alpha) = \bar{c} \cdot \mathsf{cm}_\alpha \quad \text{and} \quad \|\bar{c}\bar{\mathbf{s}}_\alpha\|_\infty \leq 2^{\log N - 1}\gamma_{\alpha,\log N},$$

with $\bar{c} = \prod_{i=1}^{\log N - 1} \bar{c}_i$, each $\bar{c}_i \in \mathcal{C} - \mathcal{C}$, where $\mathcal{C} - \mathcal{C}$ is an abuse of notation denoting the set formed by individual differences, *i.e.*, $\{\, c_1 - c_2 \mid c_1, c_2 \in \mathcal{C} \,\}$.

The extraction procedure for the witness $\vec{\mathbf{s}}$ proceeds as follows. An identical process applies to $\hat{\mathbf{s}}_\alpha$; we omit the details to avoid redundancy.

1. Fix the challenges $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{\log N - 2}$, and let $(\vec{c}^{(i)}_{\log N - 1})_{i \in [3]} \in \mathsf{SS}(\mathcal{C}, 2, 2)$ be three valid challenge vectors at round $\log N - 1$. Consider the following accepting transcripts:

$$\begin{aligned}
c^{(0)}_{\log N-1,0}\vec{\mathbf{s}}_{\log N-1,L} + c^{(0)}_{\log N-1,1}\vec{\mathbf{s}}_{\log N-1,R} &= \vec{\mathbf{s}}^{(0)}_{\log N}, \\
c^{(1)}_{\log N-1,0}\vec{\mathbf{s}}_{\log N-1,L} + c^{(1)}_{\log N-1,1}\vec{\mathbf{s}}_{\log N-1,R} &= \vec{\mathbf{s}}^{(1)}_{\log N}, \\
c^{(2)}_{\log N-1,0}\vec{\mathbf{s}}_{\log N-1,L} + c^{(2)}_{\log N-1,1}\vec{\mathbf{s}}_{\log N-1,R} &= \vec{\mathbf{s}}^{(2)}_{\log N}.
\end{aligned} \tag{5}$$

Assume that $c^{(1)}_{\log N-1,1} = c^{(0)}_{\log N-1,1}$ and $c^{(2)}_{\log N-1,0} = c^{(0)}_{\log N-1,0}$. Then the extractor computes a relaxed witness

$$\bar{\mathbf{s}}_{\log N-1} \coloneqq \left(\frac{\vec{\mathbf{s}}^{(1)}_{\log N} - \vec{\mathbf{s}}^{(0)}_{\log N}}{\bar{c}^{(1)}_{\log N-1}}, \; \frac{\vec{\mathbf{s}}^{(2)}_{\log N} - \vec{\mathbf{s}}^{(0)}_{\log N}}{\bar{c}^{(2)}_{\log N-1}} \right),$$

where $\bar{c}^{(1)}_{\log N-1} = c^{(1)}_{\log N-1,0} - c^{(0)}_{\log N-1,0}$ and $\bar{c}^{(2)}_{\log N-1} = c^{(2)}_{\log N-1,1} - c^{(0)}_{\log N-1,1}$. By linearity of the commitment scheme, according to Eq. 5 we obtain

$$\mathsf{Com}(\bar{c}^{(1)}_{\log N-1} \cdot \bar{\mathbf{s}}_{\log N-1,L}) = \mathbf{A}_0(\bar{c}^{(1)}_{\log N-1} \cdot \bar{\mathbf{s}}_{\log N-1,L}) = \bar{c}^{(1)}_{\log N-1} \cdot \mathbf{A}_0\vec{\mathbf{s}}_{\log N-1,L}, \tag{6}$$

$$\mathsf{Com}(\bar{c}^{(2)}_{\log N-1} \cdot \bar{\mathbf{s}}_{\log N-1,R}) = \mathbf{A}_0(\bar{c}^{(1)}_{\log N-1} \cdot \bar{\mathbf{s}}_{\log N-1,R}) = \bar{c}^{(2)}_{\log N-1} \cdot \mathbf{A}_0\vec{\mathbf{s}}_{\log N-1,R}. \tag{7}$$

In the $\bar{c}^{(1)}_{\log N-1} = \bar{c}^{(2)}_{\log N-1} = \bar{c}_{\log N-1}$ case, which can be easily achieved for challenges sampled from the space $\mathsf{SS}(\mathcal{C}, 2, 2)$, the extractor obtains the witness satisfying

$$\left\|\bar{c}_{\log N-1} \cdot \bar{\mathbf{s}}_{\log N-1}\right\|_\infty \leq 2\gamma_{\log N}.$$

2. Next, we repeat the above rewinding procedure by fixing $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_{\log N-3}$, and sampling $(\vec{c}_{\log N-2,i})_{i\in[3]} \leftarrow \mathsf{SS}(\mathcal{C}, 2, 2)$. From the resulting accepting transcripts, we obtain

$$\begin{aligned}
c^{(0)}_{\log N-2,0}(\vec{\mathbf{s}}_{\log N-2,L})_L + c^{(0)}_{\log N-2,1}(\vec{\mathbf{s}}_{\log N-2,R})_L &= \bar{\mathbf{s}}^{(0)}_{\log N-1,L},\\
c^{(1)}_{\log N-2,0}(\vec{\mathbf{s}}_{\log N-2,L})_L + c^{(1)}_{\log N-2,1}(\vec{\mathbf{s}}_{\log N-2,R})_L &= \bar{\mathbf{s}}^{(1)}_{\log N-1,L},\\
c^{(2)}_{\log N-2,0}(\vec{\mathbf{s}}_{\log N-2,L})_L + c^{(2)}_{\log N-2,1}(\vec{\mathbf{s}}_{\log N-2,R})_L &= \bar{\mathbf{s}}^{(2)}_{\log N-1,L}.
\end{aligned}$$

An analogous set of equations holds for the right half.
By taking pairwise differences of the above equations, the extractor derives the relaxed witness

$$\bar{\mathbf{s}}_{\log N-2,L} = \left(\frac{\bar{\mathbf{s}}^{(1)}_{\log N-1,L} - \bar{\mathbf{s}}^{(0)}_{\log N-1,L}}{\bar{c}^{(1)}_{\log N-2}}, \frac{\bar{\mathbf{s}}^{(1)}_{\log N-1,R} - \bar{\mathbf{s}}^{(0)}_{\log N-1,R}}{\bar{c}^{(1)}_{\log N-2}}\right),$$

and analogously

$$\bar{\mathbf{s}}_{\log N-2,R} = \left(\frac{\bar{\mathbf{s}}^{(2)}_{\log N-1,L} - \bar{\mathbf{s}}^{(0)}_{\log N-1,L}}{\bar{c}^{(2)}_{\log N-2}}, \frac{\bar{\mathbf{s}}^{(2)}_{\log N-1,R} - \bar{\mathbf{s}}^{(0)}_{\log N-1,R}}{\bar{c}^{(2)}_{\log N-2}}\right),$$

where $\bar{c}^{(1)}_{\log N-2} = c^{(1)}_{\log N-2,0} - c^{(0)}_{\log N-2,0}$ and $\bar{c}^{(2)}_{\log N-2} = c^{(2)}_{\log N-2,1} - c^{(0)}_{\log N-2,1}$.
In the event that $\bar{c}^{(1)}_{\log N-2} = \bar{c}^{(2)}_{\log N-2} = \bar{c}_{\log N-2}$, the extracted witness satisfies the following consistency relations:

$$\begin{aligned}
\mathbf{A}_0 \cdot (\bar{c}_{\log N-2} \cdot (\bar{\mathbf{s}}_{\log N-2,L})_L) &= \bar{c}_{\log N-2} \cdot \mathbf{A}_0(\vec{\mathbf{s}}_{\log N-1,L})_L,\\
\mathbf{A}_0 \cdot (\bar{c}_{\log N-2} \cdot (\bar{\mathbf{s}}_{\log N-2,L})_R) &= \bar{c}_{\log N-2} \cdot \mathbf{A}_0(\vec{\mathbf{s}}_{\log N-1,L})_R\\
\mathbf{A}_0 \cdot (\bar{c}_{\log N-2} \cdot (\bar{\mathbf{s}}_{\log N-2,R})_L) &= \bar{c}_{\log N-2} \cdot \mathbf{A}_0(\vec{\mathbf{s}}_{\log N-1,R})_L\\
\mathbf{A}_0 \cdot (\bar{c}_{\log N-2} \cdot (\bar{\mathbf{s}}_{\log N-2,R})_R) &= \bar{c}_{\log N-2} \cdot \mathbf{A}_0(\vec{\mathbf{s}}_{\log N-1,R})_R.
\end{aligned}$$

Thus, we obtain

$$\begin{aligned}
\mathsf{Com}(\bar{c}_{\log N-2} \cdot \bar{\mathbf{s}}_{\log N-2,L}) &= \mathbf{A}_1 \cdot G^{-1}_{2\kappa} \begin{bmatrix} \mathbf{A}_0 \cdot \left(\bar{c}_{\log N-2} \cdot (\vec{\mathbf{s}}_{\log N-2,L})_L\right) \\ \mathbf{A}_0 \cdot \left(\bar{c}_{\log N-2} \cdot (\vec{\mathbf{s}}_{\log N-2,L})_R\right) \end{bmatrix}\\
&= \bar{c}_{\log N-2} \cdot \mathsf{Com}(\vec{\mathbf{s}}_{\log N-2,L}),
\end{aligned}$$

and similarly,

$$\begin{aligned}
\mathsf{Com}(\bar{c}_{\log N-2} \cdot \bar{\mathbf{s}}_{\log N-2,R}) &= \mathbf{A}_1 \cdot G^{-1}_{2\kappa} \begin{bmatrix} \mathbf{A}_0 \cdot \left(\bar{c}_{\log N-2} \cdot (\vec{\mathbf{s}}_{\log N-2,R})_L\right) \\ \mathbf{A}_0 \cdot \left(\bar{c}_{\log N-2} \cdot (\vec{\mathbf{s}}_{\log N-2,R})_R\right) \end{bmatrix}\\
&= \bar{c}_{\log N-2} \cdot \mathsf{Com}(\vec{\mathbf{s}}_{\log N-2,R}).
\end{aligned}$$

Moreover, the extracted (scaled) witness satisfies

$$\left\|\bar{c}_{\log N-1}\cdot\left(\bar{c}_{\log N-2}\cdot\bar{\mathbf{s}}_{\log N-2}\right)\right\|_\infty \;\le\; 2\left\|\bar{c}_{\log N-1}\cdot\bar{\mathbf{s}}_{\log N-1}\right\|_\infty \;\le\; 2^2\gamma_{\log N}.$$

3. Repeating the above extraction procedure up to depth $\log N$, the extractor eventually recovers a relaxed witness $\bar{\mathbf{s}}$ defined as

$$\bar{\mathbf{s}} = \left(\frac{\bar{\mathbf{s}}_2^{(1)}-\bar{\mathbf{s}}_2^{(0)}}{\bar{c}_1^{(1)}},\ \frac{\bar{\mathbf{s}}_2^{(2)}-\bar{\mathbf{s}}_2^{(0)}}{\bar{c}_2^{(2)}}\right).$$

Moreover, the extracted witness satisfies the following consistency relations:

$$\mathsf{Com}(\bar{c}_2\cdot\bar{\mathbf{s}}_L)=\bar{c}_2\cdot\mathsf{Com}(\vec{\mathbf{s}}_L)\,,\qquad \mathsf{Com}(\bar{c}_2\cdot\bar{\mathbf{s}}_R)=\bar{c}_2\cdot\mathsf{Com}(\vec{\mathbf{s}}_R)\,.$$

Finally, the scaled witness obeys the ℓ_∞-norm bound

$$\left\|\Big(\prod_{i=1}^{\log N-1}\bar{c}_i\Big)\cdot\bar{\mathbf{s}}\right\|_\infty \;\le\; 2^{\log N-1}\,\gamma_{\log N}.$$

II. Prove the extracted witnesses satisfy linear and quadratic constraints. Next, we show that the extracted witness $(\bar{\mathbf{s}},\bar{\mathbf{s}}_\alpha)$ satisfies required linear constraint and quadratic constraints, respectively.

The above extraction procedure proceeds through $\log N$ levels and outputs a collection of relaxed witnesses[5]

$$\big((\bar{\mathbf{s}}_i,\bar{\mathbf{s}}_{\alpha,i})\big)_{i\in[\log N]}.$$

For the extracted witness

$$\bar{\mathbf{s}}_{\log N-1}\coloneqq\big(\bar{\mathbf{s}}_{\log N-1,L},\bar{\mathbf{s}}_{\log N-1,R}\big)=\left(\frac{\vec{\mathbf{s}}_{\log N}^{(1)}-\vec{\mathbf{s}}_{\log N}^{(0)}}{\bar{c}_{\log N-1}^{(1)}},\ \frac{\vec{\mathbf{s}}_{\log N}^{(2)}-\vec{\mathbf{s}}_{\log N}^{(0)}}{\bar{c}_{\log N-1}^{(2)}}\right).$$

From the linear verification equations corresponding to $\vec{\mathbf{s}}_{\log N}^{(0)}$, $\vec{\mathbf{s}}_{\log N}^{(1)}$, and $\vec{\mathbf{s}}_{\log N}^{(2)}$, we obtain

$$\langle\vec{\mathbf{a}}_0,\,\vec{\mathbf{s}}_{\log N}^{(1)}-\vec{\mathbf{s}}_{\log N}^{(0)}\rangle=\langle\bar{c}_{\log N-1}^{(1)}-\bar{c}_{\log N-1}^{(0)},\pi_{\log N-1}^{(\mathsf{lin})}\rangle=\bar{c}_{\log N-1}^{(1)}\cdot\mathbf{v}_{\log N-1,L},$$
$$\langle\vec{\mathbf{a}}_0,\,\vec{\mathbf{s}}_{\log N}^{(2)}-\vec{\mathbf{s}}_{\log N}^{(0)}\rangle=\langle\bar{c}_{\log N-1}^{(2)}-\bar{c}_{\log N-1}^{(0)},\pi_{\log N-1}^{(\mathsf{lin})}\rangle=\bar{c}_{\log N-1}^{(2)}\cdot\mathbf{v}_{\log N-1,R}.$$

Dividing by the corresponding challenge differences yields

$$\langle\vec{\mathbf{a}}_0,\bar{\mathbf{s}}_{\log N-1,L}\rangle=\mathbf{v}_{\log N-1,L},\qquad\langle\vec{\mathbf{a}}_0,\bar{\mathbf{s}}_{\log N-1,R}\rangle=\mathbf{v}_{\log N-1,R}.$$

Next, consider the quadratic verification. By construction, for each accepting transcript $i\in\{0,1,2\}$,

$$\vec{\mathbf{s}}_{\log N}^{(i)}=c_{\log N-1,0}^{(i)}\bar{\mathbf{s}}_{\log N-1,L}+c_{\log N-1,1}^{(i)}\bar{\mathbf{s}}_{\log N-1,R}.$$

[5] We identify $(\bar{\mathbf{s}}_1,\bar{\mathbf{s}}_{\alpha,1})$ with the final extracted witness pair and denote it by $(\bar{\mathbf{s}},\bar{\mathbf{s}}_\alpha)$.

Substituting this expression into the quadratic check shows that the verifier is evaluating a degree-2 polynomial in the challenge $\vec{c}_{\log N-1}$ whose coefficients are precisely

$$\langle \bar{\mathbf{s}}_{\log N-1,L}, \sigma_{-1}(\bar{\mathbf{s}}_{\log N-1,L})\rangle,\ \langle \bar{\mathbf{s}}_{\log N-1,L}, \sigma_{-1}(\bar{\mathbf{s}}_{\log N-1,R})\rangle,$$
$$\langle \bar{\mathbf{s}}_{\log N-1,R}, \sigma_{-1}(\bar{\mathbf{s}}_{\log N-1,L})\rangle,\ \langle \bar{\mathbf{s}}_{\log N-1,R}, \sigma_{-1}(\bar{\mathbf{s}}_{\log N-1,R})\rangle.$$

Therefore, if the quadratic verification holds for randomly chosen $\vec{c}_{\log N-1} \in \mathcal{C}$ but the claimed values $(\mathbf{L}_{\log N-1}, \mathbf{M}^{(1)}_{\log N-1}, \mathbf{M}^{(2)}_{\log N-1}, \mathbf{M}_{\log N-1})$ do not correspond to the true inner products of $\bar{\mathbf{s}}_{\log N-1}$, then this yields a nonzero degree-2 polynomial that vanishes at a random point. By the Schwartz–Zippel lemma, this occurs with probability at most $2/|\mathcal{C}|$. A similar analysis applies to all other rounds. Therefore, the overall soundness error of the quadratic relation proof is bounded by $2 \log N/|\mathcal{C}|$.

The same argument applies to all other linear and quadratic constraints, which we omit for brevity.

III. Prove the extracted $\bar{\mathbf{s}}$ *has binary coefficients.* Recall that we reduce the binary constraint on $\vec{s}$ to a collection of constraints on the constructed vector $\vec{\mathbf{s}}_\alpha$. In particular, the well-formedness of $\vec{\mathbf{s}}_\alpha$, together with its relation with $\vec{s}$ have been enforced via commitment, linear, and quadratic constraints, implies that the original vector $\vec{s}$ satisfies the desired binary constraint.

Next, we analyze the soundness error introduced by this constraint reduction. Specifically, we reduce the binary constraint

$$\vec{s} \circ (\vec{s} - \vec{1}) = \vec{0}$$

to the randomized linear constraint

$$\langle \vec{\alpha} \circ \vec{s},\ \vec{s} - \vec{1}\rangle = 0,$$

which introduces a soundness error bounded by $\frac{N\ell d}{q}$.

Moreover, to certify the well-formedness of $\vec{\mathbf{s}}_\alpha = \vec{s} \circ \vec{\alpha}$, we further reduce it to checking the linear consistency constraint

$$\langle \vec{\mathbf{s}}_\alpha, \vec{r}\rangle = \langle \vec{s},\ \vec{r} \circ \vec{\alpha}\rangle,$$

which incurs an additional soundness error of at most $\frac{N\ell d}{q}$.

For the case where $q = \mathsf{poly}(\lambda)$, to achieve negligible soundness error, it suffices to repeat the binary proof $t = \Theta\left(\frac{\lambda}{\log q}\right)$ times using independent randomness. □

D Soundness Proof for Optimized Version in Section 5.1

Theorem 7 (Knowledge Soundness for Composed Scheme). *The composed scheme described in Section 5.1 is knowledge-sound in the random-oracle*

model under the standard adaptive security definition, with overall soundness error

$$\epsilon_{\mathsf{se}} + \epsilon_{\mathsf{fs}} + \epsilon_{\mathsf{La}}.$$

Here, ϵ_{se} denotes the knowledge-soundness error contributed by Serval*, as established in Theorem 3; ϵ_{fs} captures the additional soundness error introduced by the Fiat–Shamir transformation; and ϵ_{La} is the knowledge-soundness error inherited from LaBRADOR.*

Proof. Owing to the knowledge-soundness guarantee of LaBRADOR, an extractor is guaranteed to obtain a set of witnesses that satisfy the claimed dot-product constraint. This, in turn, ensures that the verification procedure of the non-interactive instantiation of Serval is passed.

In our Fiat–Shamir transformation, the challenge in each round is derived by hashing the commitment to the entire transcript accumulated up to that point. Since the hash function is modeled as a random oracle, the resulting challenges are indistinguishable from uniformly random challenges and therefore satisfy the requirements for knowledge soundness in the ROM. This allows the standard rewinding-and-reprogramming extractor to obtain two accepting transcripts with divergent challenges and extract the corresponding witness.

Consequently, if all non-interactive proofs produced within the protocol verify successfully, then, by the knowledge-soundness theorem of Serval (Theorem 3), one can extract a witness that satisfies all constraints imposed by the Serval protocol. □

E Batching

We propose batching techniques for multiple polynomials at a single point, a single polynomial at multiple points, and multiple polynomials at multiple points.

Multiple Polynomials Evaluated at a Single Point. For claims $f_i(u) = v_i$ (or $f_i(\vec{u}) = v_i$) for $(f_i)_{i\in[n]}$ at a common point, the verifier samples $\vec{\alpha} \leftarrow \mathbb{Z}_q^n$. The prover forms a linear combination $f \coloneqq \sum_{i=0}^{n-1} \alpha_i f_i$ and both set $y \coloneqq \sum_{i=0}^{n-1} \alpha_i v_i$. They run one evaluation proof for f at u (or $\vec{u}$) with claimed value y. Completeness is immediate. Knowledge soundness follows from rewinding plus the PCS extractor: after extracting f, repeated rewinding recovers each f_i.

Single Polynomial Evaluated at Multiple Points. Suppose the claims are $f(u_i) = v_i$ for $i \in [n]$. Multilinear and univariate cases are slightly different.

Multilinear Case: The prover first constructs $\tilde{f}(\vec{x}) = \sum_{\vec{b}\in\{0,1\}^{\log N}} f(\vec{b}) \cdot \widetilde{\mathsf{eq}}(\vec{b}, \vec{x})$, where $\widetilde{\mathsf{eq}}(\vec{b}, \vec{x}) \coloneqq \prod_{i=0}^{\log N-1}(b[i]x[i] + (1 - b[i])(1 - x[i]))$. Since the two polynomials coincide on all Boolean vectors $\vec{b} \in \{0,1\}^{\log N}$ and f is also multilinear, it can be proved that $\tilde{f}(\vec{x}) = f(\vec{x}), \forall \vec{x} \in \mathbb{Z}_q^{\log N}$. Hence, $\tilde{f}(\vec{u}_i) = f(\vec{u}_i) = v_i$.

The verifier samples $\vec{\alpha} \leftarrow \mathbb{Z}_q^n$ and the prover sets $g(\vec{x}) = \sum_{i=0}^{n-1} \alpha_i \cdot f(\vec{x}) \cdot \widetilde{\mathsf{eq}}(\vec{x}, \vec{u}_i)$. They run a sum-check protocol [31] for $\sum_{i=0}^{n-1} \alpha_i v_i = \sum_{\vec{b} \in \{0,1\}^{\log N}} g(\vec{b})$. This reduces to checking whether $f(\vec{r}) = v$ and $\widetilde{\mathsf{eq}}(\vec{r}, \vec{u}_i) = z_i$ for some random $\vec{r}$ determined by the verifier. The latter can be computed by the verifier, while the former can be verified via the PCS. Completeness and knowledge soundness follow from those of the sum-check protocol and the PCS in Section 4.

Univariate Case: If we define $X_i \coloneqq X^{2^i}$, the univariate polynomial $f(X) = \sum_{i=0}^{N-1} f_i X^i$ can be rewritten in a multilinear form as

$$f(X_0, X_1, \ldots, X_{\ell-1}) = f_0 + f_1 X_0 + f_2 X_1 + \cdots + f_{N-1} X_0 X_1 \cdots X_{\ell-1},$$

with the same coefficient vector $\vec{f} = (f_0, f_1, \ldots, f_{N-1})$. By constructing $\vec{u}_i = (u_i, u_i^2, \ldots, u_i^{2^{\ell-1}})$, we obtain that $f(\vec{u}_i) = f(u_i)$.

Thus, the protocol reduces to the multilinear setting, with $\vec{u}_i$ in a special structured form, and the remaining steps proceed as in the multilinear case, ultimately reducing to a single-point evaluation check at a random point $\vec{r}$.

Multiple Polynomials Evaluated at Multiple Points. Now the claims are $f_i(\vec{u}_i) = v_i$ for $i \in [n]$. The construction extends naturally from the second case. The verifier first samples a random challenge vector $\vec{\alpha} \in \mathbb{Z}_q^n$, and the prover constructs the combined polynomial $g(\vec{x}) = \sum_{i=0}^{n-1} \alpha_i \cdot f_i(\vec{x}) \cdot \widetilde{\mathsf{eq}}(\vec{x}, \vec{u}_i)$. Both then engage in a sum-check protocol for $\sum_{i=0}^{n-1} \alpha_i v_i \stackrel{?}{=} \sum_{\vec{b} \in \{0,1\}^{\log N}} g(\vec{b})$.

As before, verification reduces to checking evaluations of individual f_i at a random point $\vec{r}$, which can be handled using the batching technique described in the first case. Similarly, completeness and knowledge soundness are ensured by the corresponding properties of both the sum-check protocol and the PCS.