Post-Quantum Sublinear Lookup Arguments from Polynomial Preprocessing

Adarsh Basumata[1], Chaya Ganesh[2], Varun Narayanan[3], Sikhar Patranabis[4], Surbhi Shaw[5], and Nitin Singh[6]

[1,2]Indian Institute of Science
[3]Chennai Mathematical Institute
[4,6]IBM Research India
[5]Indian Institute of Technology Kanpur

Abstract

Lookup arguments with sublinear prover time typically rely on bilinear pairings. The only exception is the recent work of Campanelli, Carrillo, Cascudo, Fiore, Francati, and Gennaro (ePrint 2025/238); however, they rely on general-purpose SNARKs to certify expensive preprocessing, which requires non-black-box usage of cryptographic primitives. Thus practical realizations of sublinear lookups with post-quantum security remain challenging. We develop the first plausibly post-quantum sublinear lookup arguments that avoid general-purpose SNARKs via polynomial preprocessing and a novel adaptation of proximity testing for Reed-Muller codes into the Interactive Oracle Proof (IOP) framework.

Our starting point is a knowledge-sound polynomial commitment scheme (PCS) with prover-side preprocessing based on the Kedlaya-Umans framework. Rather than proving the expensive preprocessing computation inside a general-purpose SNARK, we directly certify its output using Reed-Muller proximity testing and an associated evaluation protocol. For polynomials of size N, our approach yields multivariate and univariate PCS with $N^{1+\epsilon}\mathsf{polylog}(N)$ preprocessing and only $\mathsf{polylog}(N)$ online prover time, while retaining $\mathsf{polylog}(N)$ proof size and verification.

We leverage this PCS to obtain a lookup argument for *static* tables of size N and $k \ll N$ lookups with $k \cdot \mathsf{polylog}(N)$ online prover time and $\mathsf{polylog}(N)$ verification and proof size. We further extend the approach to *updatable* tables, obtaining $O(k(N^{\epsilon} + \mathsf{polylog}(N)))$ amortized prover time for a k-sized batch of reads and writes, where $\epsilon \in (0, 1)$. We retain $\mathsf{polylog}(N)$ proof size and verification. The IOPs underlying our lookup arguments satisfy round-by-round (knowledge) soundness, which can be compiled into plausibly post-quantum arguments in the (quantum) random oracle model with transparent setup.

Contents

1 Introduction **3**
- 1.1 Our Contributions . . . 4

2 Technical Overview **6**
- 2.1 Multivariate Polynomial Commitment . . . 7
- 2.2 Static Sublinear Lookup Argument . . . 8
- 2.3 Updatable Sublinear Lookup Argument . . . 9
- 2.4 Amortization via Staggered Preprocessing . . . 9
- 2.5 Lookup from Sparse Tables . . . 10
- 2.6 Hierarchical Update Strategy . . . 12

3 Preliminaries **14**
- 3.1 Notation . . . 14
- 3.2 Interactive Oracle Proofs of Proximity . . . 14
- 3.3 Polynomial Commitment Scheme . . . 16
- 3.4 KU Preprocessing . . . 17
- 3.5 Stochastic Bounds . . . 18

4 Polynomial Commitments from KU Preprocessing **18**
- 4.1 IOPs for Multivariate Polynomials . . . 18
- 4.2 Multivariate PCS from KU Preprocessing . . . 24
- 4.3 Univariate PCS from KU Preprocessing . . . 24

5 IOP for Sublinear Static Lookup **25**

6 IOP for Sublinear Updatable Lookup **28**
- 6.1 Definitions . . . 28
- 6.2 The IOP . . . 29
- 6.3 Auxiliary IOPs . . . 31

7 Blueprint for Round-by-Round Soundness **34**

1 Introduction

A Succinct Non-interactive ARgument of Knowledge (SNARK) enables a prover to provide a compact certificate attesting to the correctness of a computation, while requiring the verifier to perform substantially less work than executing the computation itself. More formally, for an NP relation $\mathcal{R}$, given a statement $\mathbb{x}$, the prover produces a non-interactive proof π certifying knowledge of a witness $\mathbb{w}$ satisfying $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$. Beyond their theoretical development [Kil92, Mic94, Gro10, Lip12, BCCT12, BCI+13, GGPR13, PHGR13, BCG+13, Lip13, BCTV14], SNARKs have become an important practical cryptographic primitive, with applications spanning a variety of real-world settings [BCG+14, Lab17, rol21, BCJ+24, LKKO20, ZFZS20, SDP22].

Prover-Side Preprocessing for SNARKs. The defining efficiency requirement for SNARKs is succinctness: after preprocessing a fixed circuit C, both the proof length and the verifier's running time should be $\mathrm{poly}(\lambda, \log |C|)$, up to the time required to read the public input. Standard preprocessing already benefits the verifier by encoding circuit-dependent information into the public parameters. Our focus is whether preprocessing can similarly benefit the *prover*. Standard SNARK provers typically perform work at least linear in the size of C for every proof. When the same circuit is used repeatedly, it is therefore natural to allow an expensive one-time *prover-side preprocessing* step that produces auxiliary information reusable across many subsequent proofs. The goal is to reduce the online prover time to sublinear in $|C|$, while retaining succinct proofs and verification. In this work, we study SNARK building blocks in this prover-side preprocessing paradigm.

Polynomial Commitment Scheme. Polynomial commitment schemes (PCS) are a central building block of modern SNARKs [WTs+18, GWC19, XZZ+19, CHM+20, Set20, SL20, BCHO22, XZS22, CBBZ23, GLS+23]. A PCS allows a prover to commit succinctly to a polynomial f and later (succinctly) prove claims of the form $f(x) = v$. For many applications, we additionally require *knowledge soundness* which guarantees extractability of a polynomial consistent with the commitment and accepted evaluation claims.

A common SNARK design combines an information-theoretic Polynomial Interactive Oracle Proof (PIOP) with a cryptographic PCS. The PIOP prover supplies polynomial oracles encoding the computation and witness, and the verifier queries these oracles at a small number of points. The PCS realizes these oracle accesses cryptographically, and Fiat–Shamir [FS87] removes interaction in the resulting public-coin succinct argument. The cost of constructing and opening these polynomials contributes directly to the prover complexity of the resulting SNARK. This paradigm encompasses both univariate PIOPs compiled with univariate PCS [MBKM19, CHM+20, GWC19, DMS24] and multilinear PIOPs compiled with multilinear PCS [WTs+18, XZZ+19, Set20, SL20, BCHO22, XZS22, GLS+23, CBBZ23].

PCS with Prover-Side Preprocessing. In this work, we investigate whether a PCS can support prover-side preprocessing. Concretely, the prover may perform an expensive one-time computation on the committed polynomial and retain auxiliary information that is reusable across evaluation queries. After this preprocessing, we seek evaluation proofs whose prover cost is sublinear (ideally polylogarithmic) in the size of the polynomial.

An elegant starting point for realizing this goal is the polynomial preprocessing framework of Kedlaya and Umans [KU08, Uma08]. Roughly speaking, after preprocessing an N-coefficient polynomial in $N^{1+\epsilon}$ time, their data structure supports evaluations in $\mathrm{polylog}(N)$ time, for any constant $\epsilon > 0$. Campanelli et al. [CCC+25] recently showed how this algebraic machinery can be used to construct multivariate PCS with prover-side preprocessing.

A key obstacle in the approach of [CCC+25] is *certifying the preprocessing*. The preprocessing algorithm produces evaluation data for a multivariate representation of the committed polynomial, and a malicious prover could provide malformed data. To obtain knowledge soundness, [CCC+25] proves correctness of the entire preprocessing computation inside an outer general-purpose SNARK. This is precisely the largest and least arithmetization-friendly computation in the construction: it has $N^{1+\epsilon}$ complexity and may involve operations without an efficient circuit representation. As a result, the outer SNARK substantially degrades the desired prover efficiency and makes the construction less modular. This motivates our first question:

Can polynomial preprocessing yield a knowledge-sound PCS without relying on a general-purpose SNARK to certify the preprocessing computation?

Lookup Arguments. Lookup arguments [BCG+18, GW20, ZBK+22, EFG22, ZGK+25] allow a prover to show succinctly that a collection of values belongs to a prescribed table. They are particularly useful for computations that are expensive to arithmetize directly, such as range checks, bitwise operations, instruction tables, and accesses to large read-only datasets. They have consequently become important components of modern proof systems, including zkVMs [STW24, AST24, ST25], proofs for non-arithmetic operations in machine-learning computations [SLZ24], and "à la carte" proof systems for non-uniform computation [DXNT24, CGG+24, GPPS24].

Preprocessing for Static Lookup Arguments. Existing preprocessing-based lookup arguments with sublinear online prover, such as Caulk [ZBK+22], CQ [EFG22], and Baloo [ZGK+25], crucially exploit specialized features of pairing-based univariate PCS. Homomorphic KZG commitments [KZG10] allow the prover to combine commitments to table-dependent polynomials, while bilinear pairings verify multiplicative polynomial relations without requiring the prover to explicitly construct the corresponding high-degree polynomials. This approach is highly effective, but ties these protocols to bilinear groups and hence does not provide a plausibly post-quantum realization based purely on hashing. The preprocessing framework of [CCC+25], by contrast, can yield sublinear lookup proving without pairings and can be instantiated using cryptographic hash functions. It nevertheless inherits the outer-SNARK bottleneck described above. We therefore ask:

Can preprocessing yield sublinear static lookup arguments without relying on bilinear pairings or a general-purpose SNARK?

Preprocessing for Updatable Lookup Arguments? An *updatable* lookup argument supports a table whose contents evolve over time, providing a natural abstraction for mutable memory, batched RAM operations [BFR+13, OWWB20, CFH+22, DGP+24, PCP26], and applications such as blockchain rollups [rol21].

This dynamic setting raises a further challenge – preprocessing tied to the current table becomes stale as the table changes. Reprocessing an N-entry table after every update would cost $N^{1+\epsilon}$ and eliminate the benefit of sublinear online proving. Instead, one would like to *maintain* the preprocessing information efficiently as the table evolves, so that both lookups and updates can be certified without repeatedly processing the entire table. Prior work on verifiable batched memory updates [DGP+24, PCP26] achieves sublinear amortized prover complexity by crucially using pairing-based machinery. On the other hand, it is unclear how the preprocessing approach of [CCC+25] can be adapted to this dynamic setting. This leads to our third question:

Can preprocessing yield sublinear updatable lookup arguments without relying on bilinear pairings?

1.1 Our Contributions

In this paper, we answer all three questions affirmatively. We construct prover-side preprocessing-based knowledge-sound PCS, static sublinear lookup arguments, and updatable lookup arguments with sublinear *amortized* prover complexity, all with succinct verification, *without* relying on either bilinear pairings or an outer, general-purpose SNARK for preprocessing certification. All of our constructions admit plausibly post-quantum realizations with transparent setup based purely on cryptographically secure hash functions.

PCS from Polynomial Preprocessing without SNARKs. Our first contribution is a new approach to constructing knowledge-sound PCS from polynomial preprocessing that significantly departs from [CCC+25]. Rather than proving the expensive preprocessing computation inside a general-purpose outer SNARK, we directly certify the *algebraic* object produced by this computation. To this end, we follow the blueprint behind popular hash-based SNARKs [ZCF24, ACFY25, ACFY24] as follows: we use a *Reed-Muller proximity* test to establish that the preprocessed evaluation oracle is close to a low-degree polynomial, and a companion evaluation protocol to recover evaluations of the unique nearby polynomial. Together, these checks precisely establish correct preprocessing: the committed evaluation data consistently represents a well-defined polynomial of the prescribed degree. This enforces knowledge soundness directly at the polynomial interface, yielding a more direct and modular approach that avoids non black-box representation of cryptographic objects within an outer SNARK circuit.

Our approach yields both multivariate and univariate PCS with the following efficiency profile: after a one-time $N^{1+\epsilon}$ preprocessing step, subsequent evaluation proofs require only polylog(N) prover work, communication, and verification. Our result is captured by the following informal theorem.

Informal Theorem 1.1 (PCS from Polynomial Preprocessing)**.** *Assuming cryptographic hash functions, for every constant $\epsilon > 0$, there exist knowledge-sound multivariate and univariate PCS for polynomials of size N with $N^{1+\epsilon}\,\mathrm{polylog}(N)$ preprocessing, after which evaluation proofs require* $\mathrm{polylog}(N)$ *online prover time and* $\mathrm{polylog}(N)$ *verifier cost. The construction does not use bilinear pairings or a general-purpose SNARK to certify preprocessing.*

See Section 4 for the formal construction and Theorem 4.1 for its (round-by-round) knowledge-soundness and efficiency guarantees. Section 2.1 presents an overview of the corresponding techniques.

Post-Quantum Static Sublinear Lookup Argument. We use the above PCS construction to design the first static lookup argument with sublinear online prover time that requires neither bilinear pairings (unlike [ZBK+22, EFG22, ZGK+25]) nor a general-purpose SNARK for preprocessing certification (unlike [CCC+25]). Our construction admits a plausibly post-quantum realization based only on cryptographic hashing.

The natural polynomial formulation of a lookup composes the large table polynomial with a smaller polynomial encoding the queried addresses. Direct composition makes the degree, and hence the prover's work, scale with the table size. Prior sublinear static lookup arguments [ZBK+22, EFG22, ZGK+25] avoid this using specialized pairing-based machinery. Instead, we exploit the multivariate representation underlying our preprocessing-based PCS, reducing each coordinate induced by the address polynomial modulo the lookup domain before composition. This keeps the resulting polynomials small, allowing the prover to construct the lookup relation using only $k\,\mathrm{polylog}(N)$ queries to the preprocessed table. Thus, rather than manipulating a prohibitively large polynomial implicitly using pairings, we change its representation so that the computation itself is small.

Informal Theorem 1.2 (Static Sublinear Lookup)**.** *Assuming cryptographic hash functions, for every constant $\epsilon > 0$, there exists a static lookup argument for a table of size N and $k \ll N$ lookups with $N^{1+\epsilon}\,\mathrm{polylog}(N)$ preprocessing, $k\,\mathrm{polylog}(N)$ online prover time, and* $\mathrm{polylog}(N)$ *verification and proof size. The construction requires neither bilinear pairings nor a general-purpose SNARK to certify preprocessing.*

See Section 5 for the formal construction, Theorem 5.1 for its knowledge-soundness analysis, and Lemma 7.1 (Section 7) for an analysis of its round-by-round knowledge soundness. Section 2.2 presents a self-contained overview of the corresponding techniques.

Extension to Updatable Sublinear Lookup Argument. Finally, we extend the above preprocessing technique for static lookups to *dynamic* tables. This yields the first updatable lookup argument that simultaneously achieves sublinear amortized prover time and succinct verification *without* bilinear pairings (unlike [DGP+24, PCP26]). As in the static case, our construction admits a plausibly post-quantum realization based only on cryptographic hashing.

Note that a naïve extension of our static preprocessing techniques to dynamic tables encounters the following challenge: the preprocessed output is tied to the initial version of the table; naïvely rebuilding it after every batch of updates would cost essentially as much as preprocessing the entire table again. We overcome this through two complementary ideas. First, following the logarithmic method of Bentley and Saxe [BS80], we maintain the table as a hierarchy of increasingly large components, so that small components are rebuilt frequently while expensive large components are rebuilt only rarely. Second, we exploit the sparsity of recently changed components through a compact polynomial representation whose preprocessing cost depends on the number of changed entries rather than the size of the full table. Crucially, we demonstrate that this compact representation still supports efficient lookup proofs, allowing both preprocessing and subsequent proving to benefit from sparsity.

Together, these ideas allow preprocessing to be efficiently maintained as the table evolves, amortizing expensive rebuilding across many update batches. Our result is captured by the following informal theorem.

Informal Theorem 1.3 (Updatable Sublinear Lookup)**.** *Assuming cryptographic hash functions, for every constant $\epsilon > 0$, there exist updatable lookup arguments for a table of size N supporting batches of $k \ll N$ reads and writes with $O(k(N^{\epsilon} + \text{polylog}(N)))$ amortized prover time and* $\text{polylog}(N)$ *verification and proof size. The construction requires neither bilinear pairings nor a general-purpose SNARK to certify preprocessing.*

See Section 6 for the construction and its knowledge-soundness analysis, and Section 7 for the round-by-round knowledge soundness. Self-contained overviews of the underlying techniques appear in Sections 2.3 to 2.5.

On Our PIOP-based Presentation. For clarity and ease of exposition, we present our core constructions as PIOPs. This allows us to make their underlying algebraic structures explicit, while separating the polynomial identities and information-theoretic soundness arguments from their cryptographic realizations. The cryptographic arguments stated in the informal theorems above follow immediately from applying the BCS compiler [BCS16], which instantiates the polynomial oracles in these PIOPs using Merkle commitments and removes interaction via Fiat-Shamir [FS87] in the (classical) ROM.

Finally, we remark that BCS compilation also preserves knowledge soundness against quantum adversaries in the Quantum ROM (QROM) when the underlying IOP satisfies round-by-round knowledge soundness [CMS19]. In this paper, we establish round-by-round knowledge soundness for all of our constructions, thus providing a natural route to plausibly post-quantum realizations by instantiating the Merkle commitments with plausibly quantum-secure cryptographic hash functions.

2 Technical Overview

We now provide an overview of the techniques underlying the key contributions of this work, namely, the univariate polynomial commitment scheme (PCS) with *succinct evaluation*, a sublinear lookup argument for static tables, and finally an updatable sublinear lookup argument for dynamic tables, all of which are based purely on cryptographic hashing and are plausibly post-quantum secure.

As mentioned in the introduction, our approach to using the polynomial preprocessing framework in this work is a significant departure from that of [CCC+25]. Instead of using a SNARK to attest to the correctness of the committed preprocessed function as a multivariate polynomial of appropriate degree, we borrow the blueprint behind popular hash-based SNARKs [ZCF24, ACFY25, ACFY24]; an IOP of proximity (IOPP) followed by an evaluation IOP. The former establishes the function oracle to be *close* to an appropriate polynomial function, which the latter IOP can appropriately *de-noise* to ensure only the true evaluation claim is accepted. Both the IOPP and evaluation IOP are based on results from proximity testing literature, specifically the *line*-tests to check proximity of a function to a Reed-Muller code. We discuss them shortly.

Polynomial Preprocessing. We start with a discussion of the polynomial preprocessing framework of Kedlaya and Umans [KU08, Uma08] and distill the key primitive it hands us.

Theorem 2.1 (Preprocessing for Multivariate Polynomials [KU08, Uma08] (Informal))**.** *Let $q \in \mathbb{N}$ and let $d, m \in \mathbb{N}$. Let $\bar{f}$ be an m-variate polynomial with degree $< d$ in each variable. Then, there exists a data structure* DS *of size*

$$S = d^m \cdot \text{poly}(\log q,\, m,\, d) \cdot O\big(m(\log m + \log d + \log\log q)\big)^m,$$

computable in time $O(S)$, such that for any evaluation point $\mathbf{x} \in \mathbb{F}^m$ the value $\bar{f}(\mathbf{x})$ can be computed in time $\text{poly}(\log q, d, m)$ *using only* $\text{poly}(\log q, d, m)$ *field operations and random-access lookups into* DS*.*

Using a succinct vector commitment scheme such as Merkle-trees for a polynomial of size $N = d^m$, and setting d and m appropriately, the above can be synthesized into a polynomial commitment scheme, with *commitment* time of $N^{1+\epsilon}$, while evaluations are $O(\log^{O(1/\epsilon)} N)$ (which we will henceforth denote as $O(\text{polylog}(N))$) for *both the prover and the verifier*[1]. To prevent an adversarial prover from committing to functions *without low-degree polynomial* representation, we use results from proximity-testing literature

[1]Note, that generally evaluation is $\Omega(N)$ for prover; polynomial evaluation being the obvious bottleneck

invoked within the framework of *Interactive Oracle Proofs* (IOPs). An IOP of proximity (IOPP) shows the committed function to be *close* to a polynomial; the evaluation IOP recovers the correct evaluation from the noisy version of the polynomial. We sketch the construction below.

2.1 Multivariate Polynomial Commitment

To construct a multivariate polynomial commitment scheme from the above function commitment, we use the following characterization of proximity of a function $f : \mathbb{F}^m \to \mathbb{F}$ to an m-variate polynomial of total degree D.

Lemma 2.1 (Theorem 7, [FS95])**.** *For any $\epsilon > 0$, there exists a constant C_ϵ such that for $|\mathbb{F}| > C_\epsilon \cdot D$ and any degree family of D polynomials $\{\hat{P}_{\mathbf{x},\mathbf{h}}\}$ satisfying:*

$$\Pr_{\mathbf{x},\mathbf{h},t}\left[\hat{P}_{\mathbf{x},\mathbf{h}}(t) \neq f(\mathbf{x} + t \cdot \mathbf{h})\right] = \gamma < 1/8 - \epsilon$$

there exists a degree D polynomial $\bar{g}$ such that $\Delta(f, \bar{g}) \leq 2\gamma$.

IOPP. The above characterization can be turned into an IOPP $(\mathcal{P}, \mathcal{V})$, which checks proximity of a function to multivariate polynomial of total degree D as follows. To check if the function committed by an oracle $[\![f]\!]$ is 1/8-close to an m-variate polynomial of total degree D, the verifier sends uniform pairs $(\mathbf{x}_i, \mathbf{h}_i) \leftarrow \mathbb{F}^m \times \mathbb{F}^m$ for $i \in [s]$. The prover responds with polynomials $\hat{p}_i(z) = P_{\mathbf{x},\mathbf{h}}(z)$, $i \in [s]$ of degree at most D. The verifier then samples $t_1, \ldots, t_s \leftarrow \mathbb{F}$ and checks $\hat{p}_i(t_i) = f(\mathbf{x}_i + t_i \cdot \mathbf{h}_i)$. It accepts if all the checks succeed.

To see the soundness of the above IOPP, let us set $\gamma \approx 1/16$ and $\epsilon := 1/16$ in Lemma 2.1. Assume a deterministic adversarial prover $\widetilde{\mathcal{P}}$ which sends polynomial $P_{\mathbf{x},\mathbf{h}}$ for the challenge pair $(\mathbf{x}, \mathbf{h}) \in \mathbb{F}^m \times \mathbb{F}^m$ from some family of polynomials $\mathcal{P}$. Observe that if $\Delta(f, \bar{g}) > 1/8$ for every polynomial $\bar{g}$ of total degree D, the contrapositive of the characterization implies that for each uniform $(\mathbf{x}, \mathbf{h}, t)$, the test $\hat{P}_{\mathbf{x},\mathbf{h}}(t) = f(\mathbf{x} + t \cdot \mathbf{h})$ rejects with probability $\approx 1/16$. Since polynomials are tested at independent points, the success probability splits bounding the soundness error to $2^{-\lambda}$ for $s = O(\lambda)$. In Lemma 4.1, we extend our analysis to prove the stronger *round-by-round* (knowledge) soundness of the above IOPP.

Evaluation IOP. We now extend the proximity claim on a function oracle $[\![f]\!]$ to an evaluation claim on the nearby (unique) polynomial $\bar{f}$. Given $\mathbf{x} \in \mathbb{F}^m$, the interactive oracle proof (IOP) for checking $\bar{f}(\mathbf{x}) = v$ proceeds as:

- Invoke the above IOPP. Reject if it fails.
- $\mathcal{V}$ sends $\mathbf{h}_i \leftarrow \mathbb{F}^m$ for $i \in [s]$.
- $\mathcal{P}$ responds with polynomials $\hat{p}_i(z)$ ($\hat{p}_i(z) = \bar{f}(\mathbf{x} + z \cdot \mathbf{h}_i)$ in honest case) of degree at most D.
- $\mathcal{V}$ samples $t_i \leftarrow \mathbb{F}$, checks $\hat{p}_i(0) = v$ and $\hat{p}_i(t_i) = [\![f]\!](\mathbf{x} + t_i \cdot \mathbf{h}_i)$ for all $i \in [s]$.

Again, we sketch the basic soundness here, deferring the proof of round-by-round knowledge soundness of the evaluation IOP to Lemma 4.2. If the IOPP accepts, then f is 1/8-close to a polynomial $\bar{f}$, except with negligible probability. Observe that $\Delta(f, \bar{f})$ is almost the expectation $\mathbb{E}_{\mathbf{h}}[\,\Delta(f|_{\mathbf{x},\mathbf{h}}, \bar{f}|_{\mathbf{x},\mathbf{h}})\,]$, over the distance between f and $\bar{f}$ restricted to lines through $\mathbf{x}$. A simple Markov's bound suggests that with probability $\geq 1/2$, the two line restrictions are at most 1/4 far. We call it the *good* case. For i^{th} query which falls in the good case, we have: if the prover sends the honest restriction $\hat{p}_i := \bar{f}(\mathbf{x} + t_i \cdot \mathbf{h}_i)$, it fails $\hat{p}_i(0) = v$, as $\bar{f}(\mathbf{x}) \neq v$. If it sends a different polynomial, by triangle inequality, it is at least $1 - D/|\mathbb{F}| - 1/4$ far from $f|_{\mathbf{x},\mathbf{h}_i}$, and thus it succeeds with probability at most $1/4 + D/|\mathbb{F}|$. Over both good and bad cases, the prover succeeds in i^{th} query with probability at most $1/2 + 1/4 + D/|\mathbb{F}| = 3/4 + D/|F|$. Thus, for $s = O(\lambda)$, the probability of malicious prover succeeding is bound by $2^{-\lambda}$.

At a high level, the above IOP(P)s give us a multivariate polynomial commitment scheme in the IOP model. The honest prover $\mathcal{P}$ commits to a polynomial $\bar{f}$ as a function oracle $[\![f]\!]$ (using the polynomial preprocessing technique of [KU08, Uma08]). The evaluation claim $f(\mathbf{x}) = v$ is checked as above. Except with negligible probability, the IOPP and evaluation IOP accept only when f is 1/8-close to a polynomial, which is guaranteed to be unique under our field-size assumption. Denoting this unique polynomial by $\bar{f}$, the evaluation IOP accepts the claim $f(\mathbf{x}) = v$ if and only if $\bar{f}(\mathbf{x}) = v$. Thus, $[\![f]\!]$ commits to the unique polynomial $\bar{f}$. We describe the full PCS construction based on the above IOPs in Section 4.

Univariate Polynomial Commitment. Given an oracle $[\![f]\!]$ for an m-variate polynomial with degree $< d$ in each variable, we can implicitly define an oracle $[\![\hat{f}]\!]$ for a univariate polynomial as $[\![\hat{f}]\!](x) = [\![f]\!](x, x^d, \ldots, x^{d^{m-1}})$. When f is a multivariate polynomial of degree $< d$ in each variable, $\hat{f}$ is a univariate polynomial of degree $< N = d^m$. Similarly, for any univariate polynomial $\hat{f}$ of degree $< N$, there exists a m-variate polynomial $\bar{f}$ with degree $< d$ in each variable, so that the above identity holds for all $x \in \mathbb{F}$. This allows us to realize a univariate polynomial commitment via a multivariate polynomial commitment, where the evaluation claim $\hat{f}(x) = v$ is expressed as equivalent claim $[\![f]\!](x, x^d, \ldots, x^{d^{m-1}}) = v$ on the multivariate polynomial.

In summary, using the polynomial preprocessing framework of [Uma08, KU08], folding it into the IOP framework using proximity tests of [FS95], we synthesize a PCS with the efficiency properties summarized informally in Informal Theorem 1.1. Next, we sketch how to leverage the prover-succinct evaluation phase to realize sublinear lookup arguments, both in absence and presence of table updates.

2.2 Static Sublinear Lookup Argument

Given integers N, k with $k \ll N$, and committed vectors $\mathbf{t} \in \mathbb{F}^N$ and $\mathbf{v} \in \mathbb{F}^k$, the lookup argument $(\mathcal{P}, \mathcal{V})$ proves $\mathbf{v}$ is a *sub-vector* of $\mathbf{t}$, i.e, there exist a_i such that $v_i = t_{a_i}$ for all $i \in [k]$. We denote the same by $\mathbf{v} \preceq \mathbf{t}$. The verifier in the lookup protocol holds commitments to $\mathbf{t}$ and $\mathbf{v}$ respectively. A sublinear lookup argument requires $\mathcal{P}$ to run in time sublinear in N, the size of the table. Several constructions of sublinear lookup arguments such as Caulk [ZBK+22], Baloo [ZGK+25], CQ [EFG22] and their variants are realized as univariate polynomial IOPs. The respective polynomial IOPs construct polynomials $\hat{t}(X)$ and $\hat{v}(X)$ which interpolate $\mathbf{t}$ and $\mathbf{v}$ over the domains $\mathbb{L}$ and $\mathbb{V}$ consisting of N^{th} and k^{th} roots of unity respectively. Writing $\mathbb{L} = \langle\omega\rangle$ and $\mathbb{V} = \langle\nu\rangle$, we have $\hat{t}(\omega^i) = t_i$ and $\hat{v}(\nu^j) = v_j$. A natural PIOP for proving lookups takes the form of polynomial composition. Let $(a_0, \ldots, a_{k-1})$ be a vector such that $v_i = t_{a_i}$ for all $i \in [k]$ and define $\hat{a}(X)$ to be the polynomial interpolating $(\omega^{a_0}, \ldots, \omega^{a_{k-1}})$ over $\mathbb{V}$. Then, the correctness of lookup reduces to showing that $\hat{t}(\hat{a}(\nu^j)) = \hat{v}(\nu^j)$ for all $j \in [k]$. Expressed as a polynomial identity, one checks the composition:

$$\hat{t}(\hat{a}(X)) \equiv \hat{v}(X) \pmod{X^k - 1}$$

The above "natural" PIOP is expensive for the prover, which needs to materialize a high-degree quotient $(\hat{t}(\hat{a}(X)) - \hat{v}(X))/(X^k - 1)$. Therefore, the breakthrough constructions in [ZBK+22, ZGK+25, EFG22] eschew polynomial composition and instead design their PIOPs around sparse matrix-vector products [ZGK+25] or logarithmic derivatives [Hab22, EFG22]. These approaches yield sublinear lookup exploiting very niche properties of trapdoor based polynomial commitments (e.g. [KZG10]) over bilinear groups. In particular, they homomorphically compute commitments to high-degree quotients in the resulting PIOPs as a k-sparse linear combination of pre-computed commitments, which depend only on the table $\mathbf{t}$ – allowing them to be computed as *offline* preprocessing. Moreover, they check the polynomial identities using only commitments via the bilinear pairing, thereby avoiding explicit evaluation queries to the polynomials. Naturally, replicating the aforementioned machinery outside the specific setting of bilinear groups has been challenging.

More recently, the authors in [CCC+25] construct a sublinear lookup argument using the *sparse-dense* sumcheck protocol of [STW24], where the lookup relation is formulated as a sparse-dense sumcheck over interpolating polynomials. The prover cost scales as $s \cdot \mathsf{Evaltime}(\bar{t})$ where s captures the sparsity of polynomials (which is $O(k)$ for our setting) and $\mathsf{Evaltime}(\bar{t})$ is the cost of an evaluation query to the table represented by $\bar{t}$. By instantiating $\bar{t}$ using KU preprocessing, the authors achieve $\mathsf{Evaltime}(\bar{t}) \approx \mathrm{polylog}(|\bar{t}|)$ which results in a sublinear lookup argument. Again, the knowledge-soundness rests on outer SNARK proving the correctness of KU preprocessing.

We present a simpler sublinear lookup argument for static tables, without the non-black-box use of SNARKs. Like [ZBK+22, ZGK+25, EFG22], the prover encodes the table $\mathbf{t}$ using polynomial $\hat{t}(X)$ such that $\hat{t}(\omega^i) = t_i$ and $\mathbf{v}$ using the polynomial $\hat{v}(X)$ such that $\hat{v}(\nu^j) = v_j$. However, the prover realizes $\hat{t}(X)$ via the multivariate polynomial $\bar{t}$ such that $\hat{t}(X) = \bar{t}(X, X^d, \ldots, X^{d^{m-1}})$ for some d, m satisfying $N = d^m$. We can choose both $d, m = \mathrm{polylog}(N)$ (we defer precise choices to later sections). Now, we write the compositional claim in terms of multivariate representation as follows:

$$\begin{aligned} \hat{t}(\hat{a}(X)) &\equiv \hat{v}(X) \pmod{X^k - 1} \\ \Leftrightarrow \bar{t}(a_0(X), a_1(X), \ldots, a_{m-1}(X)) &\equiv \hat{v}(X) \pmod{X^k - 1} \end{aligned}$$

where, in the above $a_i(X) = (\hat{a}(X))^{d^i}$. To avoid degree expansion, we can reduce inner polynomials modulo $X^k - 1$, to keep their degrees $< k$. Thus, for $b_i(X) = a_i(X) \bmod X^k - 1$, the prover can show:

$$\bar{t}(b_0(X), \ldots, b_{m-1}(X)) \equiv \hat{v}(X) \pmod{X^k - 1} \tag{1}$$

We note that the quotient in Equation (1) has degree $O(k \cdot m \cdot d) = O(k \operatorname{polylog}(N))$, so the prover can compute the same using $O(k \operatorname{polylog}(N))$ evaluation queries to $\bar{t}$. Thus, the multivariate representation makes the composition feasible by lowering the degree expansion. Finally, the correctness of $b_0(X), \ldots, b_{m-1}(X)$ is easily established by verifier checking the following identities:

$$b_0(X) = a_0(X), \quad b_i(X) = (b_{i-1}(X))^d \pmod{X^k - 1} \tag{2}$$

The verifier must also check that $\hat{a}(\nu^j)$ is an N^{th} root of unity for every $j \in [k]$. Otherwise, $\hat{a}(\nu^j)$ could lie outside $\mathbb{L}$, where $\bar{t}$ is not bound to $\mathbf{t}$. Since $N = d^m$, this reduces to the following identity:

$$(b_{m-1}(X))^d \equiv 1 \pmod{X^k - 1} \tag{3}$$

Again, the above involves polynomials of degree $O(kd) = O(k \operatorname{polylog}(N))$. Finally, the verifier can check identities in Equations (1) to (3) by querying the constituent polynomials at a random point. This requires one query to $[\![\bar{t}]\!]$ in addition to queries to smaller polynomial oracles. This yields a sublinear lookup argument for static tables (in IOP form) with the efficiency properties summarized in Informal Theorem 1.2.

2.3 Updatable Sublinear Lookup Argument

We again consider the lookup from tables as in Section 2.2, but in a setting where the table $\mathbf{t}$ admits updates of size k. More precisely, the prover runs the following operation on $\mathbf{t}$

$$\mathsf{Update}(\mathbf{t}, \mathbf{a}, \mathbf{v}', \mathsf{aux}) \to (\mathbf{v}, \mathbf{t}', \mathsf{aux}')$$

where $\mathbf{a} \in [N]^k$ is a vector of indices of $\mathbf{t}$ involved in read/write, while aux represents some pre-processed information about $\mathbf{t}$ maintained by the prover. To show *correct* update, the prover needs to show that $\mathbf{v} = \mathbf{t}[\mathbf{a}]$, the updated table $\mathbf{t}'$ satisfies $\mathbf{t}'[\mathbf{a}] = \mathbf{v}'$ and $\mathbf{t} = \mathbf{t}'$ outside indices in $\mathbf{a}$. For succinct verification, the verifier checks the above holding commitments to $\mathbf{t}, \mathbf{t}', \mathbf{a}, \mathbf{v}$ and $\mathbf{v}'$. We would again want prover cost of both operations to be sublinear in N and quasi-linear in k. The setting has been considered in [DGP+24] in context of proving batch operations on a random access memory (RAM), where the authors show *amortized* prover cost to scale as $O(k \log k + \sqrt{kN})$. The aforementioned approach also works only in the bilinear group, leveraging many of the same tools underlying the construction of static lookup arguments such as [ZBK+22, EFG22]. Here, we obtain a plausibly post-quantum safe updatable lookup argument with amortized prover complexity of $O(k(N^\epsilon + \operatorname{polylog}(N)))$. The correctness of the update operation as defined follows from correctness of the pair of lookups $(\mathbf{v}, \mathbf{v}') = (\mathbf{t}[\mathbf{a}], \mathbf{t}'[\mathbf{a}])$ and proving $\mathbf{t}, \mathbf{t}'$ to be *identical* outside indices in $\mathbf{a}$. Following [DGP+24], the latter reduces to a PIOP with prover cost of $\widetilde{O}(k)$. Next, we focus on the cost of lookup, in the presence of updates.

2.4 Amortization via Staggered Preprocessing

In the case of static lookups, the expensive $(N^{1+\epsilon})$ preprocessing step makes subsequent lookups relatively cheap $(O(k \operatorname{polylog}(N)))$. Unfortunately, updates break this cost dynamic as subsequent lookups are over a table that differs from the one that was preprocessed. Our idea is to split the table $\mathbf{t}$ as

$$\mathbf{t} = \mathbf{t}_0 + \mathbf{t}_1 + \cdots + \mathbf{t}_\ell$$

and maintain auxiliary (pre-processed) information for each component separately as $\mathsf{aux} = (\mathsf{aux}_0, \ldots, \mathsf{aux}_\ell)$. The intuition being that a particular update does not change all the components and thus recomputation of components of aux gets staggered. We further impose *sparsity* constraints on the constituent vectors: the vector $\mathbf{t}_i$ has at most $k \cdot 2^i$ non-zero entries. We assume $N = k \cdot 2^\ell$, so the vector $\mathbf{t}_\ell$ can be arbitrarily dense. Then, the update $\mathsf{Update}(\mathbf{t}, \mathbf{a}, \mathbf{v}', \mathsf{aux}) \to (\mathbf{v}, \mathbf{t}', \mathsf{aux}')$ factors over the constituent vectors as:

$$\mathsf{Update}(\mathbf{t}, \mathbf{a}, \mathbf{v}', \mathsf{aux}) \to (\mathbf{v}, \mathbf{t}', \mathsf{aux}') \equiv \left(\mathsf{Update}(\mathbf{t}_i, \mathbf{a}, \mathbf{v}'_i, \mathsf{aux}_i) \to (\mathbf{v}_i, \mathbf{t}'_i, \mathsf{aux}'_i)\right)_{i=0}^{\ell}$$

The correctness of $\mathbf{v} = \mathbf{t}[\,\mathbf{a}\,]$ is implied by $\mathbf{v}_i = \mathbf{t}_i[\,\mathbf{a}\,]$ for all i together with $\mathbf{v} = \sum_{i=0}^{\ell} \mathbf{v}_i$. Thus, the correctness of overall update is implied by correctness of constituent updates, subject to $\mathbf{v} = \sum_{i=0}^{\ell} \mathbf{v}_i$ and similarly for $\mathbf{v}'$. We account the cost of i^{th} component update under two categories:

– a *mutating* update when $\mathbf{t}'_i \notin \{\mathbf{t}_i, \mathbf{0}\}$, i.e, the new component $\mathbf{t}'_i$ is changed and non-trivial. We apportion the cost $\mathcal{C}_{\mathsf{pp}}(i)$ to the operation as it requires re-computing aux'_i.

– a *non-mutating* update when $\mathbf{t}_i = \mathbf{t}'_i$, where we only incur the "static lookup" cost $\mathcal{C}_{\mathsf{lk}}(i)$.

To keep the amortized cost low, we must ensure that mutating updates are *rarer* for denser tables $\mathbf{t}_i$. To achieve this, in Section 2.6, we describe a hierarchical update rule for $\mathbf{t}$, where the component $\mathbf{t}_i$ changes at most once in every 2^i updates (each of size k). Additionally, in Section 2.5, we show that table $\mathbf{t}_i$ with at most $s_i = k \cdot 2^i$ non-zero entries can be pre-processed in time $O(s_i^{1+\epsilon})$, i.e, $\mathcal{C}_{\mathsf{pp}}(i) = O(s_i^{1+\epsilon})$ and supports online lookup with cost $\mathcal{C}_{\mathsf{lk}}(i) = O(k\,\mathrm{polylog}(N))$. Thus, the amortized cost incurred by i^{th} update is $\mathcal{C}_{\mathsf{pp}}(i) \cdot \frac{1}{2^i} + \mathcal{C}_{\mathsf{lk}}(i)$. Averaging over all i, the amortized cost is:

$$\begin{aligned} C_{\mathrm{av}} &= \sum_{i=0}^{\ell} \left(\mathcal{C}_{\mathsf{pp}}(i) \cdot \frac{1}{2^i} + \mathcal{C}_{\mathsf{lk}}(i)\right) = \sum_{i=0}^{\ell} O(k \cdot k^{\epsilon} \cdot 2^{i\cdot\epsilon} + k \cdot \mathrm{polylog}(N)) \\ &= O\left(k \cdot (N^{\epsilon} + \mathrm{polylog}(N))\right) \end{aligned}$$

Before describing the hierarchical update strategy in Section 2.6, we discuss the PIOP for component lookups $\mathbf{v}_i = \mathbf{t}_i[\,\mathbf{a}\,]$ and analyze its efficiency.

2.5 Lookup from Sparse Tables

Consider an s-sparse vector $\mathbf{t}_* \in \mathbb{F}^N$ (with $\mathsf{char}(\mathbb{F}) \nmid N$). Let $\hat{t}_*(X)$ be the univariate polynomial interpolating $\mathbf{t}_*$ on the set $\mathbb{L}$ of N^{th} roots of unity. Constructing an evaluation table for $\hat{t}_*$ using KU preprocessing naïvely would cost $N^{1+\epsilon}$, whereas we prefer something like $O(s^{1+\epsilon})$. We therefore exploit additional structure of $\hat{t}_*$. Let $\mathcal{K}$ denote the subset of $[N]$ such that $\hat{t}_*(\omega^j) \neq 0$ for $j \in \mathcal{K}$. Then, we have:

$$\hat{t}_*(X) = \prod_{j \notin \mathcal{K}} (X - \omega^j) \cdot \hat{p}_*(X) = (X^N - 1) \cdot \frac{\hat{p}_*(X)}{\hat{q}_*(X)} \tag{4}$$

where $\hat{p}_*(X)$ is a polynomial of degree $< s$ and $\hat{q}_*(X) = \prod_{j \in \mathcal{K}} (X - \omega^j)$ is a degree s polynomial. We thus represent the polynomial $\hat{t}_*$ as a pair of degree $\leq s$ polynomials $(\hat{p}_*, \hat{q}_*)$. Let $\bar{p}$ and $\bar{q}$ be multivariate polynomials, with m_* variables each of degree at most d_* such that $\hat{p}_*(X) = \bar{p}(X, X^{d_*}, \ldots, X^{d_*^{m_*-1}})$ and likewise of $\hat{q}_*$. Now, using KU preprocessing technique, efficient evaluation tables for $\bar{p}$ and $\bar{q}$ can be constructed in time $O(d_*^{m_*} \cdot {m_*}^{m_*})$ which for appropriate choices of d_* and m_* can be made $O(s^{1+\epsilon})$. Clearly, an evaluation query to $\hat{t}_*$ can be efficiently simulated using evaluation queries to $\hat{p}_*$ and $\hat{q}_*$. We now consider proving the component lookup $\mathbf{v}_* = \mathbf{t}_*[\,\mathbf{a}\,]$ where $\mathbf{a} \in \mathbb{F}^k$ is a position vector. As before, it reduces to the following compositional identity:

$$\begin{aligned} \hat{t}_*(\hat{a}(X)) &\equiv \hat{v}_*(X) \bmod X^k - 1 \\ \Rightarrow \frac{\left((\hat{a}(X))^N - 1\right)}{\hat{q}_*(\hat{a}(X))} \cdot \hat{p}_*(\hat{a}(X)) &\equiv \hat{v}_*(X) \bmod X^k - 1 \end{aligned} \tag{5}$$

where $\hat{a}(X)$ interpolates the vector $(\omega^{a_0}, \ldots, \omega^{a_{k-1}})$ over $\mathbb{V}$. Let $\mathcal{Z}(\hat{q}_*)$ denote the roots of $\hat{q}_*$. We partition the set $\mathbb{V}$ into two depending on whether or not $\hat{a}(\eta) \in \mathcal{Z}(\hat{q}_*)$ and define vanishing polynomials over the corresponding sets:

$$\hat{g}_*(X) = \prod_{\eta \in \mathbb{V} : \hat{a}(\eta) \notin \mathcal{Z}(\hat{q}_*)} (X - \eta), \quad \hat{h}_*(X) = \prod_{\eta \in \mathbb{V} : \hat{a}(\eta) \in \mathcal{Z}(\hat{q}_*)} (X - \eta)$$

Notice that $\hat{h}_*(X) = \gcd(\hat{q}_*(\hat{a}(X)), X^k - 1)$, and thus $\hat{g}_*(X) = (X^k - 1)/\hat{h}_*(X)$. When $\eta \in \mathcal{Z}(\hat{g}_*)$, i.e, $\hat{q}_*(\hat{a}(\eta)) \neq 0$, we have:

$$\hat{v}_*(\eta) = \left(\frac{X^N - 1}{\hat{q}_*(X)}\right)(\hat{a}(\eta)) \cdot \hat{p}_*(\hat{a}(\eta)) = 0$$

Protocol $\mathsf{MutatingUpdate}(*, \mathbf{t}_*)$: re-computing aux_* for a mutated component $\mathbf{t}_*$
$\mathcal{P}$: $O(s^{1+\epsilon})$, $\mathcal{V}$: $O(\text{polylog}(N))$ − incurred once in 2^i updates for the component $\mathbf{t}_i$

1. **Sparse decomposition**: for $\mathcal{K} = \{j \in [N] : \mathbf{t}_*[j] \neq 0\}$ with $|\mathcal{K}| \leq s$, $\mathcal{P}$ computes $\hat{q}_*(X) = \prod_{j \in \mathcal{K}}(X - \omega^j)$, its formal derivative $\hat{q}'_*(X)$, and $\hat{p}_*(X)$ of degree $< s$ satisfying $\hat{t}_*(X) = (X^N - 1) \cdot \hat{p}_*(X)/\hat{q}_*(X)$.
2. **KU preprocessing**: for d_*, m_* with $d_*^{m_*} = O(s)$, $\mathcal{P}$ computes evaluation tables for the m_*-variate polynomials $\bar{p}_*, \bar{q}_*, \bar{q}'_*$ of degree $< d_*$ in each variable, satisfying $\hat{p}_*(X) = \bar{p}_*(X, X^{d_*}, \ldots, X^{d_*^{m_*-1}})$ and likewise for $\hat{q}_*, \hat{q}'_*$. This takes time $O(d_*^{m_*} \cdot m_*^{m_*}) = O(s^{1+\epsilon})$, and accounts for $\mathcal{C}_{\mathsf{pp}}(*)$.
3. **Commitment**: $\mathcal{P}$ sends the function oracles $[\![\bar{p}_*]\!]$, $[\![\bar{q}_*]\!]$, $[\![\bar{q}'_*]\!]$.
4. **Proximity check**: $\mathcal{V}$ runs the IOPP of Section 2.1 on each of the three oracles, accepting only if each is 1/4-close to an m_*-variate polynomial of the prescribed degree; $\bar{p}_*, \bar{q}_*, \bar{q}'_*$ henceforth denote these unique nearby polynomials.
5. **Divisibility & Derivative proof**: $\mathcal{P}$ sends a proof π attesting that $\hat{q}_*(X)$ divides $(X^N - 1)$ and that $\hat{q}'_*(X)$ is the formal derivative of $\hat{q}_*(X)$; $\mathcal{V}$ checks π. The oracles $[\![\bar{p}_*]\!], [\![\bar{q}_*]\!], [\![\bar{q}'_*]\!]$ constitute aux_*.

Figure 1: Protocol $\mathsf{MutatingUpdate}$: re-computing the pre-processed oracles aux_* for a mutated sparse component $\mathbf{t}_*$.

since $\hat{a}(\eta)^N = 1$ for all such η. Thus:

$$\hat{v}_*(X) \equiv 0 \pmod{\hat{g}_*(X)} \tag{6}$$

For $\eta \in \mathcal{Z}(\hat{h}_*)$, i.e, $\hat{q}_*(\hat{a}(\eta)) = 0$, we have:

$$\left(\frac{X^N - 1}{\hat{q}_*(X)}\right)(\hat{a}(\eta)) = \left(\frac{NX^{N-1}}{\hat{q}'_*(X)}\right)(\hat{a}(\eta))$$

and therefore,

$$\hat{v}_*(X) \cdot \hat{q}'_*(\hat{a}(X)) - N(\hat{a}(X))^{N-1} \cdot \hat{p}_*(\hat{a}(X)) = 0 \pmod{\hat{h}_*(X)} \tag{7}$$

In the above $\hat{q}'_*$ denotes the formal derivative of the polynomial $\hat{q}_*$ and we used the following elementary fact about polynomials expressed as quotients:

Lemma 2.2. *For polynomials $\hat{u}(X), \hat{v}(X)$ and $\hat{w}(X)$ such that $\hat{w}(X) = \hat{u}(X)/\hat{v}(X)$ and $\hat{u}(X), \hat{v}(X)$ have no repeated roots, we have $\hat{w}(x) = \hat{u}'(x)/\hat{v}'(x)$ for x such that $\hat{v}(x) = 0$.*

Thus, the congruence in Equation (5) decomposes into the congruences:

$$\begin{aligned} \hat{v}_*(X) &\equiv 0 \pmod{\hat{g}_*(X)} \\ \hat{v}_*(X) \cdot \hat{q}'_*(\hat{a}(X)) &\equiv N(\hat{a}(X))^{N-1} \cdot \hat{p}_*(\hat{a}(X)) \pmod{\hat{h}_*(X)} \end{aligned} \tag{8}$$

We need to address three further issues: (i) checking correctness of polynomials $\hat{q}_*, \hat{q}'_*, \hat{g}_*, \hat{h}_*$ etc.; (ii) querying the derivative polynomial $\hat{q}'_*$ in time $O(\text{polylog}(N))$ per query; and (iii) checking congruences in Equation (8) using low-degree polynomials (i.e, degree bounded by $O(k\,\text{polylog}(N))$).

We start with the first. The verifier checks $\hat{g}_*(X) \cdot \hat{h}_*(X) = X^k - 1$ to check that $\hat{g}_*$ and $\hat{h}_*$ together (and disjointly) span all the linear factors of $X^k - 1$. To ensure that $\hat{h}_*$ spans the roots $\eta \in \mathbb{V}$ where $\hat{q}_*(a(X))$ vanishes, the verifier checks $\gcd(\hat{q}_*(\hat{a}(X)), X^k - 1) = \hat{h}_*(X)$. Summarizing, the PIOP verifier checks:

$$\begin{aligned} \hat{v}_*(X) &\equiv 0 \pmod{\hat{g}_*(X)} \\ \hat{v}_*(X) \cdot \hat{q}'_*(\hat{a}(X)) - N(\hat{a}(X))^{N-1} \cdot \hat{p}_*(\hat{a}(X)) &\equiv 0 \pmod{\hat{h}_*(X)} \qquad (*) \\ \hat{g}_*(X) \cdot \hat{h}_*(X) &= X^k - 1 \\ \gcd(\hat{q}_*(\hat{a}(X)), X^k - 1) &= \hat{h}_*(X) \qquad (*) \\ \hat{a}(X)^N &\equiv 1 \pmod{X^k - 1} \end{aligned} \tag{9}$$

where the last congruence checks that the polynomial $\hat{a}$ interpolates N^{th} roots of unity. The "starred" identities in the above involve polynomials of degree $O(s)$, which exceeds our budget of $O(k\,\mathrm{polylog}(N))$ for the frequent *non-mutating* operations involving online lookup. We use the earlier trick of using multivariate representations of univariate polynomials $\hat{p}_*, \hat{q}_*$ and $\hat{q}'_*$ which we denote by $\bar{p}_*$, $\bar{q}_*$ and $\bar{q}'_*$ respectively. The prover computes these representations (using KU preprocessing), only during the infrequent *mutating* updates, where it additionally produces a proof π attesting that $\hat{q}'_*(X)$ is indeed the derivative of $\hat{q}_*(X)$. Again computing polynomials $\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)$ obtained by successive modular reduction w.r.t $X^k - 1$, i.e,

$$\hat{b}_0(X) = \hat{a}_0(X), \quad \hat{b}_i(X) = (\hat{b}_{i-1}(X))^{d_*} \pmod{X^k - 1} \text{ for } i > 0$$

The prover now claims the following reduced polynomials of size at most k.

$$\begin{aligned}
\hat{P}_*(X) &= \bar{p}_*(\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)) \pmod{X^k - 1} \\
\hat{Q}_*(X) &= \bar{q}_*(\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)) \pmod{X^k - 1} \\
\hat{Q}'_*(X) &= \bar{q}'_*(\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)) \pmod{X^k - 1} \\
\hat{R}(X) &= \big(\hat{a}(X)\big)^{N-1} \pmod{X^k - 1}
\end{aligned} \tag{10}$$

In terms of the reduced polynomials, the starred congruences can be written as:

$$\begin{aligned}
\hat{v}_*(X) \cdot \hat{Q}'_*(X) - N \cdot \hat{R}(X) \cdot \hat{P}_*(X) &\equiv 0 \pmod{\hat{h}_*(X)} \\
\gcd(\hat{Q}_*(X), X^k - 1) &= \hat{h}_*(X)
\end{aligned} \tag{11}$$

We see that the reductions in Equation (10) involve quotients of degree $O(k\,\mathrm{polylog}(N))$, and subsequently, the identities in Equation (11) are also of polynomials of degree $O(k)$. The prover sends Bézout polynomials $\hat{c}_*(X)$ and $\hat{d}_*(X)$ of degree at most k to reduce the gcd claim to:

$$\hat{q}_1(X) \cdot \hat{h}_*(X) = \hat{Q}_*(X), \quad \hat{q}_1(X) \cdot \hat{c}_*(X) + \hat{g}_*(X) \cdot \hat{d}_*(X) = 1$$

Here the first identity establishes that $\hat{h}_*$ divides $\hat{Q}_*$, while the second identity shows no other factors of $X^k - 1$, which are all contained in $\hat{g}_*$, divide $\hat{Q}_*$, establishing $\hat{h}_*$ as the claimed gcd of $\hat{Q}_*$ and $X^k - 1$. We summarize the component lookup protocol as two separate sub-protocols: MutatingUpdate in Figure 1, the infrequent *mutating* phase which recomputes the pre-processed oracles for a mutated component $\mathbf{t}_*$, and NonMutatingUpdate in Figure 2, the frequent *non-mutating* phase which proves a lookup against the already pre-processed oracles.

2.6 Hierarchical Update Strategy

We now specify the update rule promised in Section 2.4, which follows the *logarithmic method* of [BS80]. Each component is either *empty* ($\mathbf{t}_i = \mathbf{0}$) or holds up to its budget of $k \cdot 2^i$ non-zero entries. Given a batch update at positions $\mathbf{a} \in [N]^k$ with new values $\mathbf{v}'$, the prover reads $\mathbf{v} = \mathbf{t}[\,\mathbf{a}\,]$ by invoking NonMutatingUpdate on each non-empty component, and forms the update vector $\mathbf{u} = \mathbf{v}' - \mathbf{v}$, which is k-sparse as it is supported on $\mathbf{a}$. Note that we do not assume that the entries of $\mathbf{a}$ are distinct, but require repeated addresses to specify the same updated value, i.e., $a_i = a_j$ implies $v'_i = v'_j$. It then locates the smallest empty component $i = \min\{j : \mathbf{t}_j = \mathbf{0}\}$, collapses everything below level i into it,

$$\mathbf{t}'_i \leftarrow \mathbf{u} + \sum_{j<i} \mathbf{t}_j, \quad \mathbf{t}'_j \leftarrow \mathbf{0} \text{ for } j < i, \quad \mathbf{t}'_j \leftarrow \mathbf{t}_j \text{ for } j > i$$

and runs MutatingUpdate on level i alone. Level i stays within budget, as the components below it are full and contribute $\sum_{j<i} k \cdot 2^j = k(2^i - 1)$ entries, to which $\mathbf{u}$ adds at most k more. If no component is empty we take $i = \ell$ and absorb $\mathbf{t}_\ell$ into the collapse as well, which is unconstrained since $N = k \cdot 2^\ell$. Figure 3 summarizes the protocol.

Observe that level i is the smallest empty level once in every 2^i updates, precisely the frequency assumed in Section 2.4: each update incurs one invocation of MutatingUpdate at cost $\mathcal{C}_{\mathsf{pp}}(i) = O((k2^i)^{1+\epsilon})$ and

Protocol NonMutatingUpdate$(*, \mathbf{a})$: proving the lookup $\mathbf{v}_* = \mathbf{t}_*[\mathbf{a}]$
$\mathcal{P}$: $O(k\,\mathrm{polylog}(N))$, $\mathcal{V}$: $O(\mathrm{polylog}(N))$ − incurred at every update

---

Common input: the oracles $[\![\bar{p}_*]\!], [\![\bar{q}_*]\!], [\![\bar{q}'_*]\!]$ produced by the mutating phase, and $[\![\hat{v}_*]\!]$.

1. **Index polynomial**: $\hat{a}(X)$ interpolating the lookup vector $(\omega^{a_0}, \ldots, \omega^{a_{k-1}})$ over $\mathbb{V}$.
2. **Successive modular reductions of a-powers**:
   - $\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)$ satisfying $\hat{b}_0(X) = \hat{a}(X)$, $\hat{b}_i(X) = \big(\hat{b}_{i-1}(X)\big)^{d_*} \pmod{X^k - 1}$ for $1 \le i < m_*$.
   - $\hat{R}_0(X) = (\hat{a}(X))^{d-1}$, $\hat{R}_i(X) = \big(\hat{R}_{i-1}(X)\big)^{d} \cdot \hat{R}_0(X) \pmod{X^k - 1}$, so that $\hat{R}(X) = \hat{R}_{m-1}(X) = (\hat{a}(X))^{N-1} \pmod{X^k - 1}$.
3. **Reduction using multivariate polynomials**: using $O(k\,\mathrm{polylog}(N))$ evaluation queries to the pre-processed oracles,
$$\hat{P}_*(X) = \bar{p}_*(\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)), \quad \hat{Q}_*(X) = \bar{q}_*(\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)),$$
$$\hat{Q}'_*(X) = \bar{q}'_*(\hat{b}_0(X), \ldots, \hat{b}_{m_*-1}(X)) \quad \text{all reduced} \pmod{X^k - 1}$$
4. **Modular decomposition polynomials**: $\hat{g}_*(X)$, $\hat{h}_*(X)$. **Bezout polynomials**: $\hat{c}_*(X)$, $\hat{d}_*(X)$. **Quotient polynomial**: $\hat{q}_1(X)$.
5. **Verifier checks**: all of the following involve polynomials of degree $O(k\,\mathrm{polylog}(N))$, and are checked at a random point.
$$\begin{aligned}
\hat{b}_i(X) &= \Big(\hat{b}_{i-1}(X)\Big)^{d_*} && \pmod{X^k - 1} \text{ for } 1 \le i < m_*,\\
\hat{R}_0(X) &= (\hat{a}(X))^{d-1} && \pmod{X^k - 1},\\
\hat{R}_i(X) &= \Big(\hat{R}_{i-1}(X)\Big)^{d} \cdot \hat{R}_0(X) && \pmod{X^k - 1} \text{ for } 1 \le i < m,\\
&\text{the three reductions defining } \hat{P}_*, \hat{Q}_*, \hat{Q}'_* \text{ in item 3} && \pmod{X^k - 1},\\
\hat{v}_*(X) &= 0 && \pmod{\hat{g}_*(X)},\\
\hat{g}_*(X) \cdot \hat{h}_*(X) &= X^k - 1, &&\\
\hat{v}_*(X) \cdot \hat{Q}'_*(X) - N \cdot \hat{R}(X) \cdot \hat{P}_*(X) &= 0 && \pmod{\hat{h}_*(X)},\\
\hat{q}_1(X) \cdot \hat{h}_*(X) &= \hat{Q}_*(X), &&\\
\hat{q}_1(X) \cdot \hat{c}_*(X) + \hat{g}_*(X) \cdot \hat{d}_*(X) &= 1. &&
\end{aligned}$$

Figure 2: Protocol NonMutatingUpdate: proving a lookup $\mathbf{v}_* = \mathbf{t}_*[\mathbf{a}]$ against the oracles pre-processed by Figure 1.

$O(\ell)$ invocations of NonMutatingUpdate at cost $\mathcal{C}_{\mathsf{lk}} = O(k\,\mathrm{polylog}(N))$ each, recovering $C_{\mathrm{av}} = O(k \cdot (N^{\epsilon} + \mathrm{polylog}(N)))$. Note that the choice of level i and the collapse are prover-side bookkeeping and need not be verified: a correct update is characterized by the lookups $\mathbf{v} = \mathbf{t}[\mathbf{a}]$, $\mathbf{v}' = \mathbf{t}'[\mathbf{a}]$ and $\mathbf{t}, \mathbf{t}'$ agreeing outside $\mathbf{a}$, which is oblivious to how the prover distributes the table across levels. We defer the PIOP for the last check to Section 6.

Combining the ideas from Sections 2.3 to 2.6 yields an IOP for sublinear lookup over dynamic tables with the efficiency guarantees summarized in Informal Theorem 1.3. In Sections 5 and 6, we formalize the intuition developed here and demonstrate that the static and updatable lookup protocols of Sections 2.2 and 2.3, when instantiated with the PCS from Section 2.1, yield knowledge-sound IOPs for the corresponding relations. Moreover, Lemma 7.1 (Section 7) establishes their round-by-round knowledge soundness via a more general blueprint for PIOPs composed with round-by-round sound polynomial commitment schemes, which also encompasses our constructions.

Protocol $\mathsf{Update}(\mathbf{a}, \mathbf{v}')$ − one batch update of size k on $\mathbf{t} = \mathbf{t}_0 + \cdots + \mathbf{t}_\ell$
amortized $\mathcal{P}$: $O(k(N^\epsilon + \mathrm{polylog}(N)))$, $\mathcal{V}$: $O(\mathrm{polylog}(N))$

State: components $\mathbf{t}_0, \ldots, \mathbf{t}_\ell$ with $\mathbf{t}_i$ either $\mathbf{0}$ or $k2^i$-sparse and $N = k2^\ell$; pre-processed $\mathsf{aux} = (\mathsf{aux}_0, \ldots, \mathsf{aux}_\ell)$.
Input: positions $\mathbf{a} \in [N]^k$ and new values $\mathbf{v}' \in \mathbb{F}^k$.

1. **Read**: for each non-empty level j, run $\mathsf{NonMutatingUpdate}(j, \mathbf{a})$ to obtain and prove $\mathbf{v}_j = \mathbf{t}_j[\mathbf{a}]$; empty levels contribute $\mathbf{v}_j = \mathbf{0}$. Set $\mathbf{v} = \sum_{j=0}^{\ell} \mathbf{v}_j$, and $\mathcal{V}$ checks $\hat{v}(X) = \sum_{j=0}^{\ell} \hat{v}_j(X)$.
2. **Update vector**: $\mathbf{u} \leftarrow \mathbf{v}' - \mathbf{v}$, supported on $\mathbf{a}$ and hence k-sparse.
3. **Carry**: $i \leftarrow \min\{j \in \{0, \ldots, \ell\} : \mathbf{t}_j = \mathbf{0}\}$; if no such j exists, $i \leftarrow \ell$.
4. **Collapse**: the new components are
$$\mathbf{t}'_i = \mathbf{u} + \sum_{j<i} \mathbf{t}_j, \quad \mathbf{t}'_j = \mathbf{0} \ \text{ for } j < i, \quad \mathbf{t}'_j = \mathbf{t}_j \ \text{ for } j > i$$
where, if no level was empty (so $i = \ell$), the first sum instead runs over all $j \le \ell$, i.e, $\mathbf{t}'_\ell = \mathbf{u} + \sum_{j \le \ell} \mathbf{t}_j$.
5. **Re-pre-process**: run $\mathsf{MutatingUpdate}(i, \mathbf{t}'_i)$ to obtain aux'_i. Levels $j < i$ discard their oracles; levels $j > i$ retain aux_j unchanged.
6. **Verifier**: accepts iff every invoked $\mathsf{NonMutatingUpdate}$ and $\mathsf{MutatingUpdate}$ accepts, $\hat{v} = \sum_j \hat{v}_j$, and $\mathbf{t}, \mathbf{t}'$ agree outside $\mathbf{a}$ (Section 6). Steps 3 and 4 are prover-side bookkeeping and are not checked.

Cost: one $\mathsf{MutatingUpdate}$ at level i costing $\mathcal{C}_{\mathsf{pp}}(i) = O((k2^i)^{1+\epsilon})$, incurred once in every 2^i updates, and $O(\ell)$ invocations of $\mathsf{NonMutatingUpdate}$ costing $\mathcal{C}_{\mathsf{lk}} = O(k\,\mathrm{polylog}(N))$ each.

Figure 3: Batch update protocol. Only the smallest empty level i is mutated, and $\mathsf{MutatingUpdate}$ is invoked on it once in every 2^i updates.

3 Preliminaries

In this section, we introduce the notations used throughout the paper, and summarize results from prior work needed to describe our constructions.

3.1 Notation

We write $\mathbb{F}$ for a finite field and $\mathbb{F}_q$ when explicitly denoting a field of prime order q. The integers and natural numbers are $\mathbb{Z}$ and $\mathbb{N}$, and $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. We use $[n]$ to denote the set $\{0, 1, \ldots, n-1\}$ and $\{0,1\}^m$ for the set of m-bit binary strings. A security parameter is denoted λ; functions that vanish faster than every inverse polynomial are $\mathrm{negl}(\lambda)$.

Polynomials. We use the following convention throughout: $\hat{f}$ denotes a *univariate* polynomial, $\widetilde{f}$ denotes a *multilinear* polynomial, and $\bar{f}$ denotes a general *multivariate* polynomial. The polynomial ring in variables $X_0, \ldots, X_{m-1}$ is written $\mathbb{F}[X_0, \ldots, X_{m-1}]$, and $\mathbb{F}[X]$ for the univariate ring. We use D and d to denote bounds on total degree and individual degrees for multivariate polynomials. For a polynomial f we use $[\![f]\!]$ to denote oracle access to f, i.e, query access to f. For a multiplicative subgroup $\mathbb{G} \subseteq \mathbb{F}^*$ with generator g, we write $\hat{Z}_{\mathbb{G}}(X) = X^{|\mathbb{G}|} - 1$ for its vanishing polynomial. We write $\{\hat{\mu}_y^{\mathbb{G}}\}_{y \in \mathbb{G}}$ for its Lagrange basis, i.e., the polynomials of degree $< |\mathbb{G}|$ with $\hat{\mu}_y^{\mathbb{G}}(y') = 1$ if $y = y'$ and 0 otherwise, and we also write $\hat{\mu}_i^{\mathbb{G}} = \hat{\mu}_{g^i}^{\mathbb{G}}$ for $i \in [|\mathbb{G}|]$.

3.2 Interactive Oracle Proofs of Proximity

An *Interactive Oracle Proof* (IOP) [BCS16, RRR16] is an information theoretic object that generalizes the notion of a Probabilistically Checkable Proof (PCP) [BFLS91, FGL+91, AS92, ALM+92]. An IOP is an interactive protocol $(\mathcal{P}, \mathcal{V})$ where $\mathcal{V}$ only has oracle access to prover's messages. A k-round public coin *Interactive Oracle Proof of Proximity* (IOPP) for a ternary relation $\mathcal{R}$ on tuples $(\mathbb{x}, \mathbb{y}, \mathbb{w})$ is an interactive

proof $(\mathcal{P}, \mathcal{V})$ where, in each round $i \in [1, k]$, the verifier $\mathcal{V}$ sends uniformly sampled randomness ρ_i, while the prover $\mathcal{P}$ replies with $\Pi_i \in \Sigma^*$. The verifier $\mathcal{V}$ reads $\mathbb{x}$ in its entirety, but only has oracle access to the remaining prover messages. At the end of k rounds, the verifier $\mathcal{V}$ queries oracles $[\![\mathbb{y}]\!], [\![\Pi_1]\!], \ldots, [\![\Pi_k]\!]$ at a few locations and outputs a decision bit b.

Definition 3.1 (IOPP Completeness)**.** *An* IOPP $(\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *is complete if for every* $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$*,*

$$\Pr_{\rho_1, \ldots, \rho_k}\left[\mathcal{V}\big(\mathbb{x}, [\![\mathbb{y}]\!], \big([\![\Pi_i]\!]\big)_{i=1}^k, (\rho_i)_{i=1}^k\big) = 1 \;\middle|\; \begin{array}{c} \Pi_1 \leftarrow \mathcal{P}(\mathbb{x}, \mathbb{y}, \mathbb{w}, \rho_1) \\ \cdots \\ \Pi_k \leftarrow \mathcal{P}(\mathbb{x}, \mathbb{y}, \mathbb{w}, \rho_1, \ldots, \rho_k) \end{array}\right] = 1.$$

Definition 3.2 (IOPP Knowledge-Soundness)**.** *An* IOPP $(\mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *is* knowledge-sound *with knowledge error* $\kappa(\cdot)$ *if there exists an efficient extractor* $\mathcal{E}$ *such that for any adversarial prover* $\widetilde{\mathcal{P}}$*,*

$$\Pr\left[\begin{array}{c} \mathcal{V}\big(\mathbb{x}, [\![y]\!], \big([\![\Pi_i]\!]\big)_{i=1}^k, (\rho_i)_{i=1}^k\big) = 1 \\ \wedge \\ (\mathbb{x}, \mathbb{y}, \mathbb{w}) \notin \mathcal{R} \end{array} \;\middle|\; \begin{array}{c} \big((\Pi_i)_{i=1}^k, (\rho_i)_{i=1}^k\big) \leftarrow \langle \widetilde{P}, \mathcal{V}\rangle \\ \mathbb{w} \leftarrow \mathcal{E}(\mathbb{x}, \mathbb{y}, (\Pi_i)_{i=1}^k) \end{array}\right] \leq \kappa(\mathbb{x}, \mathbb{y})$$

We now describe a stronger notion of soundness for IOPPs, namely, the *round-by-round (knowledge) soundness*, which directly relates to the (knowledge) soundness of the non-interactive argument in ROM, obtained by compiling them using the BCS [BCS16] compiler.

Definition 3.3 (State Function)**.** *Let* $(\mathcal{P}, \mathcal{V})$ *be a* k*-round* IOPP *for a relation* $\mathcal{R}$*. A state function for* $(\mathcal{P}, \mathcal{V})$ *is a (possibly inefficient) function* State *that receives as input an instance* $(\mathbb{x}, \mathbb{y})$ *and an interaction transcript* tr*, and outputs a bit (where* 0 *signifies that the prover is* doomed *and* 1 *signifies that the prover is* not-doomed*) such that the following properties hold.*

1. **Empty transcript.** *If* $\mathsf{tr} = \emptyset$ *is the empty transcript, then* $\mathrm{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$.[2]
2. **Prover moves.** *If* $\mathsf{tr} = (\Pi_1, \rho_1, \ldots, \Pi_i, \rho_i)$ *is a transcript of* i *rounds, where* $i < k$*, and* $\mathrm{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$*, then for every prover message* $\Pi_{i+1} \in \Sigma^{i+1}$*,* $\mathrm{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \Pi_{i+1}) = 0$.
3. **Full transcript.** *If* $\mathsf{tr} = (\Pi_1, \rho_1, \ldots, \Pi_k, \rho_k)$ *is a complete transcript and* $\mathrm{State}(\mathbb{x}, \mathbb{y}, \mathsf{tr}) = 0$*, then* $\mathcal{V}$ *rejects given this interaction transcript.*

Definition 3.4 (Round-by-Round Knowledge Soundness)**.** *Let* $(\mathcal{P}, \mathcal{V})$ *be a* k*-round* IOPP *for a relation* $\mathcal{R}$*. We say that* $(\mathcal{P}, \mathcal{V})$ *satisfies* round-by-round knowledge soundness *with knowledge error* $\kappa^{\mathrm{rbr}}(\cdot)$ *if there exists polynomial-time extractor* $\mathcal{E}^{\mathrm{rbr}}$ *and a state function* State *such that, for every instance* $\mathbb{x}, \mathbb{y}$*, every round* $i \in \{1, \ldots, k\}$*, and every malicious prover* $\widetilde{\mathcal{P}}$ *it holds that:*

$$\Pr_{\rho_i}\left[\begin{array}{c} \mathsf{State}(\mathbb{x}, \mathbb{y}, (\Pi_1, \rho_1, \ldots, \Pi_i)) = 0 \\ \wedge \\ \mathsf{State}(\mathbb{x}, \mathbb{y}, (\Pi_1, \rho_1, \ldots, \Pi_i, \rho_i)) = 1 \\ \wedge \\ (\mathbb{x}, \mathbb{y}, \mathcal{E}^{\mathrm{rbr}}(\mathbb{x}, \mathbb{y}, (\Pi_j)_{j=1}^i)) \notin \mathcal{R} \end{array} \;\middle|\; \big(\mathbb{x}, \mathbb{y}, (\Pi_j)_{j=1}^i, (\rho_j)_{j=1}^{i-1}\big) \leftarrow \widetilde{P} \right] \leq \kappa^{\mathrm{rbr}}(\mathbb{x}, \mathbb{y}).$$

Definition 3.5 (Round-by-Round Soundness [CCH+19])**.** *Let* $(\mathcal{P}, \mathcal{V})$ *be a* k*-round* IOPP *for a relation* $\mathcal{R}$*, and let* $\mathcal{L}_{\mathcal{R}} = \{(\mathbb{x}, \mathbb{y}) : \exists\, \mathbb{w},\ (\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}\}$*. A state function for round-by-round soundness is defined as above, except that the empty-transcript property only requires* $\mathsf{State}(\mathbb{x}, \mathbb{y}, \emptyset) = 0$ *for* $(\mathbb{x}, \mathbb{y}) \notin \mathcal{L}_{\mathcal{R}}$*. We say that* $(\mathcal{P}, \mathcal{V})$ *satisfies* round-by-round soundness *with error* $\kappa^{\mathrm{rbr}}(\cdot)$ *if there exists such a state function* State *with the following property. For every* $(\mathbb{x}, \mathbb{y}) \notin \mathcal{L}_{\mathcal{R}}$*, every round* $i \in \{1, \ldots, k\}$ *and every malicious prover* $\widetilde{\mathcal{P}}$*,*

$$\Pr_{\rho_i}\left[\begin{array}{c} \mathsf{State}(\mathbb{x}, \mathbb{y}, (\Pi_1, \rho_1, \ldots, \Pi_i)) = 0 \\ \wedge \\ \mathsf{State}(\mathbb{x}, \mathbb{y}, (\Pi_1, \rho_1, \ldots, \Pi_i, \rho_i)) = 1 \end{array} \;\middle|\; \big((\Pi_j)_{j=1}^i, (\rho_j)_{j=1}^{i-1}\big) \leftarrow \widetilde{P}(\mathbb{x}, \mathbb{y}) \right] \leq \kappa^{\mathrm{rbr}}(\mathbb{x}, \mathbb{y}).$$

Round-by-round knowledge soundness implies round-by-round soundness: for $(\mathbb{x}, \mathbb{y}) \notin \mathcal{L}_{\mathcal{R}}$*, the extractor in Definition 3.4 always fails.*

[2]Here, "doomed" refers to the round-by-round knowledge-soundness condition, not to the non-existence of a witness. The empty transcript is designated doomed by convention, with extraction required upon a $0 \to 1$ state transition.

In IOPPs, the soundness error bounds $\kappa(\cdot)$ (for standard soundness) and $\kappa^{\mathrm{rbr}}(\cdot)$ for round-by-round soundness may additionally depend on proximity $\delta = \delta(\mathbb{x}, \mathbb{y})$ defined as:

$$\delta(\mathbb{x}, \mathbb{y}) = \inf\{\Delta(\mathbb{y}, \mathbb{y}') : \exists \mathbb{w} \text{ s.t. } (\mathbb{x}, \mathbb{y}', \mathbb{w}) \in \mathcal{R}\}$$

In other words, we write $\kappa(\mathbb{x}, \mathbb{y}, \delta(\mathbb{x}, \mathbb{y}))$ and similarly for κ^{rbr}.

Interactive Oracle Proofs (IOP) . An *interactive oracle proof* or an IOP is simply an IOPP for relations $\mathcal{R}$ over tuples of the form $(\mathbb{x}, \perp, \mathbb{w})$, i.e, where $\mathbb{y} = \perp$. We typically write them as binary relations.

3.3 Polynomial Commitment Scheme

We first define the notion of a polynomial commitment scheme (PCS) in the interactive oracle proof (IOP) model. Our definition mimics the definition of a polynomial commitment scheme in the standard model, with function oracles $[\![f]\!]$ playing the role of commitments. Our definition is similar to that from [ZCF24].

Definition 3.6 (Polynomial Commitment Scheme)**.** *Let $m \in \mathbb{Z}$ and $\mathbf{d} = (d_1, \ldots, d_m) \in \mathbb{Z}^m_{\geq 0}$. Let $D = \sum_{i \in [m]} d_i$. A polynomial commitment scheme* PC *for m-variate polynomials over a field $\mathbb{F}$ with degree bounds $\mathbf{d}$ consists of a tuple of oracle algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{CheckProximity}, \mathsf{Eval})$ *as follows.*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *takes the security parameter λ and outputs public parameters* pp*.*
- $\mathsf{Commit}(\mathsf{pp}, \bar{f}) \to [\![\bar{f}]\!]$ *takes inputs* pp*, polynomial $\bar{f} \in \mathbb{F}^{\leq \mathbf{1} \cdot d}[X_1, \ldots, X_m]$ and outputs an m-variate function oracle $[\![\bar{f}]\!]$ (this is an analogue of a commitment in the standard model).*
- $\mathsf{Open}(\mathsf{pp}, f, \bar{f}, \delta)$ *is a relation that takes inputs* pp*, an m-variate function (word) f, an m-variate polynomial $\bar{f}$ of total degree at most D, a proximity parameter, and outputs 1 if f and $\bar{f}$ disagree on at most δ fraction of the domain $\mathbb{F}^m$.*
- $\mathsf{CheckProximity}\langle \mathcal{P}(\bar{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \delta) \to b$ *is an interactive oracle protocol between a PPT prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ with common inputs consisting of $[\![f]\!]$ and a proximity parameter δ. $\mathcal{P}$ additionally knows an m-variate polynomial $\bar{f}$ of total degree at most D. $\mathcal{P}$ wants to convince $\mathcal{V}$ that $\bar{f}$ is an opening to $[\![f]\!]$ (i.e.,* $\mathsf{Open}(\mathsf{pp}, f, \bar{f}, \delta) = 1$*). $\mathcal{V}$ outputs a bit b at the end of the protocol indicating accept or reject.*
- $\mathsf{Eval}\langle \mathcal{P}(\bar{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \mathbf{x}, v) \to b$ *is an interactive oracle protocol between a PPT prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$ with common inputs consisting of $[\![f]\!]$, an evaluation point $\mathbf{x} \in \mathbb{F}^m$ and a value $v \in \mathbb{F}$. $\mathcal{P}$ additionally knows an m-variate polynomial $\bar{f}$ of total degree at most D. $\mathcal{P}$ wants to convince $\mathcal{V}$ that $\bar{f}(\mathbf{x}) = v$. $\mathcal{V}$ outputs a bit b at the end of the protocol indicating accept or reject.*

A polynomial commitment scheme PC *must satisfy completeness, δ-binding and knowledge soundness as defined below.*

- PC *is* complete *if for any $\bar{f} \in \mathbb{F}^{\leq \mathbf{1} \cdot d}[X_1, \cdots, X_m]$ and $\mathbf{x} \in \mathbb{F}^m$*

$$\Pr\left[\mathsf{Eval}\langle \mathcal{P}(\bar{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \mathbf{x}, v) \to 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ [\![f]\!] \leftarrow \mathsf{Commit}(\mathsf{pp}, \bar{f}); \\ v \leftarrow \bar{f}(\mathbf{x}) \end{array} \right] = 1$$

  We do not separately require completeness of $\mathsf{CheckProximity}$ *in the above definition. The evaluation protocol* Eval *invokes* $\mathsf{CheckProximity}$ *as a sub-protocol, so completeness of* Eval *already accounts for its honest execution.*

- PC *is* δ-binding *if for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} b \wedge b_0 \wedge b_1 = 1 \\ \wedge \\ \bar{f}_0 \neq \bar{f}_1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ ([\![f]\!], \bar{f}_0, \bar{f}_1, \mathsf{aux}) \leftarrow \mathcal{A}_0(\mathsf{pp}); \\ b \leftarrow \mathsf{CheckProximity}\langle \mathcal{A}_1(\mathsf{aux}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \delta) \\ b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, f, \bar{f}_0, \delta); \\ b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, f, \bar{f}_1, \delta) \end{array} \right] \leq \mathrm{negl}(\lambda)$$

- PC *is* knowledge sound *if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, there exists an expected polynomial time algorithm* $\mathcal{E}$ *such that*

$$\Pr\left[\begin{array}{c} b = 1 \\ \wedge \\ \mathcal{R}_{Ev}(\mathsf{pp}, [\![f]\!], \mathbf{x}, v; \bar{f}) = 0 \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \\ ([\![f]\!], \mathbf{x}, v, \mathsf{aux}) \leftarrow \mathcal{A}_0(\mathsf{pp}), \\ b \xleftarrow{\mathsf{tr}} \mathsf{Eval}\langle \mathcal{A}_1(\mathsf{aux}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \mathbf{x}, v) \\ \bar{f} \leftarrow \mathcal{E}^{\mathcal{A}_1}(\mathsf{pp}, \mathsf{tr}) \end{array}\right]$$

is negligible in λ*, where the oracle relation* $\mathcal{R}_{Ev}$ *is defined as*

$$\mathcal{R}_{Ev} = \{((\mathsf{pp}, [\![f]\!], \mathbf{x}, v); \bar{f}) : \mathsf{Open}(\mathsf{pp}, f, \bar{f}, \delta) = 1 \wedge \bar{f}(\mathbf{x}) = v\}.$$

The notation $\mathcal{E}^{\mathcal{A}_1}$ *captures both* straightline *and* rewinding *extractors. In the former,* $\mathcal{E}$ *outputs a witness using only the interaction transcript* tr *of the adversary* $\mathcal{A}$*, while in the latter it can implement rewinding using the copy of* $\mathcal{A}_1$ *initialized with the private state of the successful* $\mathcal{A}_1$*.*

3.4 KU Preprocessing

Throughout this subsection, let $\mathbb{F} = \mathbb{F}_q$. Let $d, m \in \mathbb{N}$, set $M = d^m$, and let $\mathbb{F}^{<M}[X]$ denote the vector space of univariate polynomials of degree less than M. Let $\mathbb{F}^{<d\cdot\mathbf{1}}[X_0, \ldots, X_{m-1}]$ denote the vector space of m-variate polynomials with individual degrees less than d. The KU coefficient encoding $\Psi_{d,m} : \mathbb{F}^{<M}[X] \to \mathbb{F}^{<d\cdot\mathbf{1}}[X_0, \ldots, X_{m-1}]$ is the linear isomorphism defined on the monomial basis by

$$X^a \mapsto X_0^{a_0} \cdots X_{m-1}^{a_{m-1}}$$

for all $a \in [M]$, where $a_0, \ldots, a_{m-1}$ are the d-ary digits of a. For $\hat{f} \in \mathbb{F}^{<M}[X]$, let $\bar{f} = \Psi_{d,m}(\hat{f})$ denote the associated multivariate polynomial. It is immediate that $\hat{f}(x) = \bar{f}(\alpha(x))$ where $\alpha(x) = (x, x^d, x^{d^2}, \ldots, x^{d^{m-1}})$.

Theorem 3.1 (KU Preprocessing [KU08, Uma08])**.** *Let* $q \in \mathbb{N}$ *and let* $d, m \in \mathbb{N}$*. There exists a data structure* DS *of size*

$$S = d^m \cdot \mathrm{poly}(\log q,\, m,\, d) \cdot O\big(m(\log m + \log d + \log\log q)\big)^m,$$

computable from any $\bar{f} \in \mathbb{F}^{<d\cdot\mathbf{1}}[X_0, \ldots, X_{m-1}]$ *in time* $O(S)$*, such that for any evaluation point* $\mathbf{x} \in \mathbb{F}^m$ *the value* $\bar{f}(\mathbf{x})$ *can be computed in time* $\mathrm{poly}(\log q, d, m)$ *using only* $\mathrm{poly}(\log q, d, m)$ *field operations and random-access lookups into* DS*.*

Corollary 3.1 (KU Preprocessing for Univariate Polynomials)**.** *Let* $q, N, d, m \in \mathbb{N}$ *with* $d^m = N$*, and let* $\hat{f} \in \mathbb{F}^{<N}[X]$*. Set* $\bar{f} = \Psi_{d,m}(\hat{f})$ *and let* DS *be the data structure for* $\bar{f}$ *guaranteed by Theorem 3.1. Then for any* $x \in \mathbb{F}$*,*

$$\hat{f}(x) = \bar{f}\big(\alpha(x)\big), \qquad \alpha(x) = (x,\, x^d,\, x^{d^2},\, \ldots,\, x^{d^{m-1}}),$$

so $\hat{f}(x)$ *can be evaluated in time* $\mathrm{poly}(\log q, d, m)$ *after preprocessing* $\hat{f}$ *into* DS *of size* $S = d^m \cdot \mathrm{poly}(\log q, m, d) \cdot O(m(\log m + \log d + \log\log q))^m$*. In particular, taking* $d = \log^{3/\varepsilon}(N)$ *and* $m = \frac{\varepsilon}{3}\log(N)/\log\log(N)$ *for any constant* $\varepsilon > 0$ *gives* DS *of size* $O(N^{1+\varepsilon}) \cdot \mathrm{poly}(\log q, \log N)$ *and evaluation time* $\mathrm{poly}(\log q, \log N)$*.*

The data structure DS of Theorem 3.1 and Corollary 3.1 can be committed as Merkle-tree(s). The claimed evaluation $y = f(\mathbf{x})$ can be verified succinctly in time $\mathrm{poly}(\log q, d, m)$ by verifying membership of leaf nodes under the root hash.

For readability, we will not refer to DS explicitly in the sequel. Instead, we write $[\![\hat{f}]\!]$ (resp. $[\![\bar{f}]\!]$) to denote oracle access to a univariate (resp. multivariate) polynomial while describing interactive oracle proofs (IOPs) - here, the function oracle induced by the preprocessed data structure DS. During the compilation of IOP to an argument in the random oracle model (ROM), we will instantiate the oracle access using Merkle-trees in a standard manner.

Protocol: ProximityLinesTest

$\mathcal{P}$'s input: $(f, \bar{f})$ such $f : \mathbb{F}^m \to \mathbb{F}$, $\bar{f} \in \mathcal{C}$ and $f|_{\mathbb{F}^m} = \bar{f}|_{\mathbb{F}^m}$.
$\mathcal{V}$'s input: $[\![f]\!]$.

1. $\mathcal{V}$ sends $(\mathbf{x}_i, \mathbf{h}_i)$, $i \in [s]$ where $\mathbf{x}_i \leftarrow \mathbb{F}^m$ and $\mathbf{h}_i \leftarrow \mathbb{F}^m$ and $s = O(\lambda)$.
2. $\mathcal{P}$ sends polynomial $\ell_i(z) = \bar{f}(\mathbf{x}_i + z \cdot \mathbf{h}_i)$ of degree $< D$.
3. $\mathcal{V}$ samples $t_i \leftarrow \mathbb{F}$ for $i \in [s]$ and checks if $\ell_i(t_i) = [\![f]\!](\mathbf{x}_i + t_i \cdot \mathbf{h}_i)$
4. $\mathcal{V}$ accepts if all the checks pass, otherwise it rejects.

Figure 4: IOPP for relation $\mathcal{R}_{\mathcal{C}}$.

3.5 Stochastic Bounds

Lemma 3.1 (Chernoff Bound for Indicators)**.** *Let $X_1, \ldots, X_n$ be independent indicator random variables with $\Pr[X_i = 1] = p_i$, let $X = \sum_{i=1}^n X_i$, and let $\mu = \mathbb{E}[X] = \sum_{i=1}^n p_i$. Then:*

1. (Upper tail) *for every $\delta > 0$,*

$$\Pr[X \geq (1+\delta)\mu] \;\leq\; e^{-\mu\delta^2/3} \quad (0 < \delta \leq 1);$$

2. (Lower tail) *for every $0 < \delta < 1$,*

$$\Pr[X \leq (1-\delta)\mu] \;\leq\; e^{-\mu\delta^2/2}.$$

Moreover, the bounds hold with μ replaced by any upper bound $\bar{\mu} \geq \mathbb{E}[X]$ in the upper tail, and any lower bound $\underline{\mu} \leq \mathbb{E}[X]$ in the lower tail.

4 Polynomial Commitments from KU Preprocessing

In this section, we leverage the polynomial preprocessing framework of [KU08, Uma08] and the *line*-tests of multivariate polynomials from Lemma 4.1 to construct a plausibly post-quantum secure multivariate PCS with prover-side preprocessing that does not use general-purpose SNARK for preprocessing certification. We start with the formal descriptions of the constituent IOPs.

4.1 IOPs for Multivariate Polynomials

Fix integers d, m, N and D satisfying $N = d^m$ and $D = d \cdot m$. Let $\mathbb{F} = \mathbb{F}_q$ for some prime q and $\mathcal{C} = \mathbb{F}^{<D}[X_0, \ldots, X_{m-1}]$[3] denote the *Reed-Muller* code consisting of evaluations of polynomials in $\mathbb{F}^{<D}[X_0, \ldots, X_{m-1}]$ over $\mathbb{F}^m$. Let RS denote the *Reed-Solomon* code consisting of evaluations of univariate polynomials of degree $< D$ over $\mathbb{F}$. For a word $\mathbf{u}$ in a code C, we will use $\mathsf{Decode}(\mathbf{u})$ to denote the unique $\mathbf{v} \in \mathsf{C}$, such that $\Delta(\mathbf{u}, \mathbf{v}) < \delta/2$, where δ is the relative minimum distance of the code (we will assume C to be implicit from the context). When no such $\mathbf{v}$ exists, we write $\mathsf{Decode}(\mathbf{u}) = \perp$.

IOP of Proximity. Let the relation $\mathcal{R}_{\mathcal{C}}$ consist of pairs $(\mathcal{C}, f, \perp)$ where $f : \mathbb{F}^m \to \mathbb{F}$ is such that $f \in \mathcal{C}$. The IOPP for $\mathcal{R}_{\mathcal{C}}$ is described in Figure 4. We emphasize that R_C is an exact-codeword relation, and hence Figure 4 is stated with perfect completeness only for $f \in C$. In the evaluation protocol below, we use its round-by-round soundness analysis also when f is merely close to C; proximity to the decoded polynomial is handled separately in the composed protocol.

[3]We use $\mathbb{F}^{<D}[X_0, \ldots, X_{m-1}]$, so the maximum total degree in the code is $(D-1)$ whereas several subsequent sections refer to "degree D" and distance $(1 - D/q)$. Here, $D = dm$ is the strict total-degree bound and we intentionally use the weaker $(1 - D/q)$ distance bound for notational convenience.

Lemma 4.1. *The interactive oracle protocol* ProximityLinesTest *in Figure 4, for $s = O(\lambda)$ is a round-by-round knowledge sound IOPP for the relation $\mathcal{R}_{\mathcal{C}}$ with soundness error*

$$\kappa^{\text{iopp}} = \begin{cases} 2^{-\lambda} \text{ when } \Delta(\mathcal{C}, f) > 1/8 \\ 1 \text{ otherwise} \end{cases}$$

Proof. For $f \in \mathcal{C}$, it is easily seen that the verifier in ProximityLinesTest accepts. Thus completeness holds. For round-by-round soundness, we first describe the state function, denoted as State.

- **Empty Transcript**: We define $\mathsf{State}(\mathcal{C}, f, \emptyset) = 0$. Henceforth, for brevity we will suppress inputs $\mathcal{C}, f$ to the state function.
- **$\mathcal{V}$'s first message**: After verifier's first message $\mathsf{tr} = \left((\mathbf{x}_i, \mathbf{h}_i)_{i=1}^s\right) \in \mathbb{F}^{2ms}$. We define $\mathsf{State}(\mathcal{C}, f, \mathsf{tr}) = 1$ if f is 1/8-close to $\mathcal{C}$. When f is 1/8-far from $\mathcal{C}$, we define $\mathsf{State}(\mathcal{C}, f, \mathsf{tr}) = 0$, unless the following *exceptional* event E occurs:
$$E \equiv \begin{cases} \text{For a fixed parameter } s_1 < s, \text{ only few, i.e, at most } s_1 \\ \text{line restrictions } f(\mathbf{x}_j + z \cdot \mathbf{h}_j) \text{ are 1/32-far from } \mathsf{RS} \end{cases}$$
Thus, $\mathsf{State}(\mathsf{tr}) = (\Delta(\mathcal{C}, f) \leq 1/8) \vee E$.
- **$\mathcal{P}$'s first message**: After $\mathcal{P}$'s first message $(\hat{\ell}_i)_{i=1}^s$, we define
$$\mathsf{tr} = \left((\mathbf{x}_i, \mathbf{h}_i)_{i=1}^s, (\hat{\ell}_i)_{i=1}^s\right)$$
Note that we still have $\mathsf{State}(\mathsf{tr}) = (\Delta(\mathcal{C}, f) \leq 1/8) \vee E$, i.e., the prover's message does not change the state.
- **$\mathcal{V}$'s second message**: The full transcript after $\mathcal{V}$'s final message is given by:
$$\mathsf{tr} = \left((\mathbf{x}_i, \mathbf{h}_i)_{i=1}^s, (\hat{\ell}_i)_{i=1}^s, (t_i)_{i=1}^s\right)$$
where t_i, $i \in [s]$ are uniformly sampled from $\mathbb{F}$. For full transcript tr, we define $\mathsf{State}(\mathsf{tr}) = \mathcal{V}(\mathcal{C}, f, \mathsf{tr})$, i.e, the state function is $\mathcal{V}$'s output on the full transcript.

The extraction in this case is vacuous, so IOPP extractor $\mathcal{E}^{\text{iopp}}$ just outputs $\perp$. We show that the state switches from 0 to 1 with probability at most $\kappa^{\text{iopp}}(\mathcal{C}, f)$. The claim trivially holds for $\Delta(\mathcal{C}, f) \leq 1/8$, with κ^{iopp} defined to be 1 in this case. Thus, we only consider the case when $\Delta(\mathcal{C}, f) > 1/8$.
Switch at $\mathcal{V}$'s first message. We analyse the probability of the State function switch at verifier's first message. Here we have:
$$\mathsf{tr} = (\mathcal{C}, f), \rho_1 = \left((\mathbf{x}_i, \mathbf{h}_i)\right)_{i=1}^s$$
We consider the scenario when the switch occurs, and $\Delta(\mathcal{C}, f) > 1/8$. We show that the probability of switch is at most $2^{-\lambda}$ in this case. Since f is 1/8-far from $\mathcal{C}$, the state switches only when the exceptional event E occurs. For each $(\mathbf{x}, \mathbf{h})$, choose a polynomial $\hat{P}_{\mathbf{x},\mathbf{h}}$ of degree less than D minimizing its distance from $f|_{\mathbf{x},\mathbf{h}}$. Now, applying the contrapositive of Lemma 2.1 with degree bound $(D-1)$ and $\epsilon = 1/16$ implies that for the family of polynomials $\{\hat{P}_{\mathbf{x},\mathbf{h}}\}$, we have:

$$\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{h}}[\Delta(\mathsf{RS}, f|_{\mathbf{x},\mathbf{h}})] \geq \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{h}}[\Delta(\hat{P}_{\mathbf{x},\mathbf{h}}, f|_{\mathbf{x},\mathbf{h}})] > 1/16 = 1/8 - 1/16$$

where we use $\delta = 1/16$ and $\epsilon = 1/16$ in Lemma 2.1. Noticing that $\Delta(\cdot, \cdot) \leq 1$, a standard calculation shows that

$$\Pr_{\mathbf{x},\mathbf{h}}[\, f|_{\mathbf{x},\mathbf{h}} \text{ is 1/32-far from } \mathsf{RS}\,] \geq \frac{1}{31}$$

For $i \in [s]$, let X_i be the indicator of the event that $f|_{\mathbf{x}_i,\mathbf{h}_i}$ is 1/32-far from RS, and let $X = \sum_{i=1}^s X_i$. The pairs $(\mathbf{x}_i, \mathbf{h}_i)$ are sampled independently, so the X_i are independent indicators, and the bound above gives $\mathbb{E}[X] \geq s/31$. The event E is exactly $X \leq s_1$, and we take $s_1 \leq s/62$. Writing $\underline{\mu} := s/31$ for the lower bound

on $\mathbb{E}[X]$, we have $s_1 \leq (1-\delta)\underline{\mu}$ for $\delta = 1/2$. The lower tail of Lemma 3.1, applied with this δ and with $\underline{\mu}$ in place of $\mathbb{E}[X]$, then gives

$$\Pr[E] = \Pr[X \leq s_1] \leq e^{-\underline{\mu}\delta^2/2} = e^{-s/248},$$

which is at most $2^{-\lambda}$ for $s \geq 172\,\lambda$. Thus, we have:

$$\Pr_{\rho_1}\left[\begin{array}{c} \mathsf{State}(\mathcal{C}, f, \mathsf{tr}) = 0 \\ \wedge \\ \mathsf{State}(\mathcal{C}, f, \mathsf{tr}\|\rho_1) = 1 \\ \wedge \\ (\mathcal{C}, f, \mathcal{E}^{\mathrm{rbr}}(\mathcal{C}, f, \mathsf{tr}) = \perp) \notin \mathcal{R} \end{array} \middle| \; \left(\mathcal{C}, f, \mathsf{tr}\right) \leftarrow \widetilde{P} \right] \leq 2^{-\lambda}$$

Switch at $\mathcal{V}$'s second message. Next, we consider the $0 \to 1$ switch probability at verifier's second message. Here, we have:

$$\mathsf{tr} = \left((\mathbf{x}_i, \mathbf{h}_i)_{i=1}^s, (\hat{\ell}_i)_{i=1}^s\right), \rho_2 = (t_1, \ldots, t_s).$$

Since we must have $\mathsf{State}(\mathsf{tr}) = 0$, there exist $\geq s_1$ pairs $\{(\mathbf{x}_i, \mathbf{h}_i) : i \in S\}$ in tr for some $|S| \geq s_1$ such that $\Delta(f|_{\mathbf{x}_i, \mathbf{h}_i}, \mathsf{RS}) > 1/32$ for $i \in S$. Thus, for $i \in S$, $\Pr_{t_i}[\hat{\ell}_i(t_i) = f(\mathbf{x}_i + t_i \cdot \mathbf{h}_i)] \leq 31/32$. Since t_i are sampled independently and uniformly from $\mathbb{F}$, conditioned on $\mathsf{State}(\mathsf{tr}) = 0$, the verifier accepts with probability at most $(31/32)^{s_1}$, which is at most $2^{-\lambda}$ whenever $s_1 = c\lambda$ for a sufficiently large constant $c > 0$. In other words, by suitably setting $s_1 = c\lambda$ for a sufficiently large constant $c > 0$ and $s = O(\lambda)$, it can be seen that $\kappa^{\mathrm{rbr}} \leq 2^{-\lambda}$. This completes the proof. □

IOP for Evaluation. We now leverage the IOPP in Figure 4 to realize an IOP for the polynomial evaluation relation $\mathcal{R}_{Ev}$ defined by:

$$\mathcal{R}_{Ev} = \{((\mathsf{pp}, [\![f]\!], \mathbf{x}, v); \bar{f}) : \mathsf{Open}(\mathsf{pp}, f, \bar{f}, \delta) = 1 \wedge \bar{f}(\mathbf{x}) = v\},$$

where we set $\delta = 1/8$ and pp contains description of $\mathcal{C}, \mathsf{RS}$ etc. The evaluation IOP uses the IOPP of Figure 4, together with further line-tests from the evaluation point $\mathbf{x}$, to establish proximity to the decoded polynomial and correctness of its evaluation. The composition sequentially executes the *commitment rounds* of the two protocols, i.e, the rounds before the verifier's final message. The verifier's final messages in both the constituent protocols are deferred to the end, prior to computing the decision bit. To delineate the IOPP from evaluation checks in the composed IOP, we have the prover send dummy message "Start Eval" to preserve alternate messages from $\mathcal{P}$ and $\mathcal{V}$. The IOP appears in Figure 5.

Lemma 4.2. *The interactive oracle protocol in Figure 5 is a round-by-round knowledge sound IOP for the relation $\mathcal{R}_{Ev}$ with knowledge error $\kappa^{\mathrm{eval}} = \mathrm{negl}(\lambda)$ for $t = O(\lambda)$.*

Proof. We again skip completeness as it is straightforward to verify. Let $\mathsf{pp} = (\mathcal{C}, \mathsf{RS}, \ldots)$. We consider the following cases for the transcript of the composed IOP.

- **IOPP rounds**. We denote the partial transcript during IOPP rounds as $\mathsf{tr} = (\mathsf{tr}_{\mathrm{iopp}}, \emptyset, \emptyset)$.
- **Interface round**. At the interface of IOPP and evaluation tests, we can write the transcript as:

  $$\mathsf{tr} = \left(\mathsf{tr}_{\mathrm{iopp}}, \text{“Start Eval”}, \emptyset\right)$$

  where $\mathsf{tr}_{\mathrm{iopp}}$ denotes the full *commitment* transcript of the IOPP.
- **Evaluation rounds**. After the interface round, we denote the partial transcript as:

  $$\mathsf{tr} = \left(\mathsf{tr}_{\mathrm{iopp}}, \text{“Start Eval”}, \mathsf{tr}_{\mathrm{eval}}\right)$$

  where $\mathsf{tr}_{\mathrm{iopp}}$ denotes the full commitment transcript of IOPP and $\mathsf{tr}_{\mathrm{eval}}$ denotes the transcript of evaluation rounds.

Protocol: EvalLinesTest

$\mathcal{P}$'s input: $(\mathsf{pp}, f, \bar{f}, \mathbf{x}, v)$ such $f : \mathbb{F}^m \to \mathbb{F}$, $\bar{f} \in \mathcal{C}$, $f|_{\mathbb{F}^m} = \bar{f}|_{\mathbb{F}^m}$ and $\bar{f}(\mathbf{x}) = v$.
$\mathcal{V}$'s input: $(\mathsf{pp}, [\![f]\!], \mathbf{x}, v)$.

1. Let $\mathsf{tr} \leftarrow \langle \mathcal{P}(\mathsf{pp}, f, \bar{f}), \mathcal{V}(\mathsf{pp}, [\![f]\!]) \rangle$ be transcript of $\mathcal{P}$ and $\mathcal{V}$ executing IOPP ProximityLinesTest, except the final verifier's message.
2. $\mathcal{P}$ sends dummy message "Start Eval".
3. $\mathcal{V}$ sends $(\mathbf{r}_1, \ldots, \mathbf{r}_t)$, where $\mathbf{r}_i \leftarrow \mathbb{F}^m$ for $i \in [t]$.
4. $\mathcal{P}$ sends polynomial $\hat{p}_i(z) = \bar{f}(\mathbf{x} + z \cdot \mathbf{r}_i)$ of degree $< D$ for $i \in [t]$.
5. $\mathcal{V}$ uniformly samples and sends $(s_1, \ldots, s_t) \leftarrow \mathbb{F}^t$ and $(t_1, \ldots, t_s) \leftarrow \mathbb{F}^s$.
6. $\mathcal{V}$ checks if $\hat{p}_i(s_i) = [\![f]\!](\mathbf{x} + s_i \cdot \mathbf{r}_i)$ and $\hat{p}_i(0) = v$ for all $i \in [t]$.
7. $\mathcal{V}$ checks if $\mathcal{V}^{\text{iopp}}(\mathsf{tr} \| t_1, \ldots, t_s) = 1$.
8. $\mathcal{V}$ outputs 1 if both the above checks pass, else it outputs 0.

Figure 5: IOP for relation $\mathcal{R}_{Ev}$.

- **Full transcript**. The final verifier messages of both the IOPP and evaluation phase are sent as one final message ρ, giving the full transcript

$$\mathsf{tr} = \big(\mathsf{tr}_{\text{iopp}}, \text{"Start Eval"}, \mathsf{tr}_{\text{eval}}, \rho\big)$$

  where $\mathsf{tr}_{\text{iopp}}$ and $\mathsf{tr}_{\text{eval}}$ denote full commitment transcripts of the IOPP and evaluation phase respectively.

Next, we define the State function for the composed transcript. Let $\mathbb{P}(f)$ denote the predicate:

$$\mathbb{P}(f) = \begin{array}{c} (\Delta(\mathcal{C}, f) > 1/8) \\ \vee \\ (\Delta(\mathcal{C}, f) \leq 1/8) \wedge (\bar{f}(\mathbf{x}) = v) \end{array}$$

where $\bar{f} = \mathsf{Decode}(f)$. For *IOPP rounds* we define:

$$\mathsf{State}(\mathsf{tr}) = \mathsf{State}_{\text{iopp}}(\mathsf{tr}_{\text{iopp}}) \wedge \mathbb{P}(f).$$

At the *transition*, i.e., the verifier's first message in the evaluation round, we have:

$$\mathsf{tr} = \big(\mathsf{tr}_{\text{iopp}}, \text{"Start Eval"}, (\mathbf{r}_i)_{i=1}^t\big).$$

For this round, we specify the State function as follows. Let $t_1 < t$ be a parameter. For any f such $\Delta(\mathcal{C}, f) \leq 1/8$, let $\bar{f}$ be the unique member of $\mathcal{C}$ such that $\Delta(f, \bar{f}) \leq 1/8$. Define Bad to be the event that the set S of line restrictions of f that are $1/4$-close to that of $\bar{f}$ has cardinality less than t_1, i.e,

$$\mathsf{Bad} \equiv |\{i \in [t] : \Delta(f|_{\mathbf{x},\mathbf{r}_i}, \bar{f}|_{\mathbf{x},\mathbf{r}_i}) \leq 1/4\}| < t_1$$

For such transcripts tr, we define

$$\mathsf{State}(\mathsf{tr}) = \mathsf{State}_{\text{iopp}}(\mathsf{tr}_{\text{iopp}}) \wedge (\mathbb{P}(f) \vee (\Delta(\mathcal{C}, f) \leq 1/8 \wedge \mathsf{Bad}))$$

For the full composed transcript, we define:

$$\mathsf{State}(\mathsf{tr}) = \mathcal{V}_{\text{iopp}}(\mathsf{tr}_{\text{iopp}} \| \rho_{\text{iopp}}) \wedge \mathcal{V}_{\text{eval}}(\mathsf{tr}_{\text{eval}} \| \rho_{\text{eval}})$$

where we parse the final verifier message $\rho = (\rho_{\text{iopp}}, \rho_{\text{eval}})$.

Analyzing Switch Events. We first analyze the state switch within IOPP rounds. Since the predicate $\mathbb{P}$ is constant throughout the IOPP rounds, a state switch there implies that the function $\mathsf{State}_{\mathrm{iopp}}$ switches from 0 to 1. Invoking the round-by-round knowledge soundness of the IOPP in Figure 4, conditioned on $\mathsf{State}(\mathsf{tr}) = 0$ we have:

$$\Pr_{\rho}\left[(\mathsf{pp}, f, \mathcal{E}^{\mathrm{iopp}}(\mathsf{pp}, f, \mathsf{tr})) \notin \mathcal{R}_{\mathcal{C}} \wedge \mathsf{State}(\mathsf{tr}\|\rho)\right] \leq \kappa^{\mathrm{iopp}}.$$

If $\Delta(\mathcal{C}, f) \leq 1/8$, the extractor $\mathcal{E}^{\mathrm{eval}}$ described in the following lemma outputs $\bar{f} = \mathsf{Decode}(f)$ in time $O(\mathrm{poly}(N, \lambda))$, except with probability at most $2^{-\lambda}$. Moreover, since $\mathbb{P}(f) = 1$ at the switched state, the definition of $\mathbb{P}$ implies $\bar{f}(x) = v$; hence $\bar{f}$ is a valid witness for R_{Ev}.

Lemma 4.3. *Let $\bar{f}$ be an m-variate polynomial of total degree at most $D = d \cdot m$, where $d^m = N \leq |\mathbb{F}|/16$ and $d \geq 2$, such that $\Delta(f, \bar{f}) <= 1/8$. There exists a randomized algorithm $\mathcal{E}^{\mathrm{eval}}$ that, given oracle access to f, recovers $\bar{f}$ with failure probability at most $2^{-\lambda}$ using $O(\lambda \cdot N^9)$ field operations.*

Proof. In [GS92], Gemmell and Sudan describe a self correction procedure that makes $O(D)$ queries to f to correctly evaluate $\bar{f}$ at $x \in \mathbb{F}^m$ with probability 2/3 under the assumptions in the lemma; implementing its Berlekamp–Welch decoder by Gaussian elimination takes $O(D^3 + mD) = O(D^3)$ field operations. See [GRR18, Lemma 4.9]. Along with a simple concentration bound, this imples an efficient self correction procedure that makes $O(D \cdot \log \frac{1}{\delta})$ oracle calls to f and $O(D^3 \cdot \log \frac{1}{\delta})$ field operations to correctly evaluate $\bar{f}$ on any specified point with failure probability $\delta > 0$.

There exists $S \subset \mathbb{F}^m$ of size $\binom{m+D}{m}$ such that evaluations on S uniquely determine the interpolating m-variate polynomial of total degree at most D. The extractor $\mathcal{E}^{\mathrm{eval}}$ computes $\bar{f}$ by interpolating the evaluations of $\bar{f}$ on S. Size of S is upper-bounded by $d^{3m} = N^3$ since $d^m \leq N$ and $d \geq 2$. The extractor invokes the above self correction procedure on every element of S with failure probability $\delta = 2^{-\lambda}/N^3$ and interpolate these values to recover $\bar{f}$. The self correction procedure takes $N^3 \cdot O(\lambda \cdot D^3 \cdot \log N)$ steps; interpolation by Gaussian elimination takes $O(N^9)$ steps. By a union bound, the failure probability of the algorithm is at most $N^3\delta = 2^{-\lambda}$. □

On the other hand, if $\Delta(\mathcal{C}, f) > 1/8$, no valid witness for $\mathcal{R}_{\mathcal{C}}$ exists, and therefore round-by-round knowledge soundness of the IOPP bounds the probability of such a $0 \to 1$ switch by κ_{iopp}. Thus failure to extract a valid witness for R_{Ev} upon a switch within the IOPP rounds occurs with probability at most $\kappa_{\mathrm{iopp}} + 2^{-\lambda} = \mathrm{negl}(\lambda)$. Therefore, the round-by-round knowledge soundness condition is satisfied when the switch occurs within the IOPP rounds.

Next, assume that the state switch occurs at the transition. In this case, we have:

$$\mathsf{tr} = (\mathsf{tr}_{\mathrm{iopp}}, \text{“Start Eval”}), \quad \rho = (\mathbf{r}_i)_{i=1}^{t}$$

This switch can occur only if $\mathsf{State}_{\mathrm{iopp}}(\mathsf{tr}_{\mathrm{iopp}}) = 1$, $\mathbb{P}(f) = 0$. Moreover, the switch requires $\Delta(\mathcal{C}, f) \leq 1/8$ and the event Bad occurs. Thus, the switch probability is bounded by:

$$\Pr_{\rho}[\,\mathsf{Bad} = 1 \wedge \Delta(\mathcal{C}, f) \leq 1/8]$$

In Lemma 4.4, we prove that this probability is bounded by $2^{-\lambda}$ for $t \geq 4 \cdot t_1 \geq c'\lambda$ for a constant c'.

Next, assume that the state change occurs during the final (combined) verifier message. This event is contained in the union of the following two switch events:

$$\begin{aligned} E_1 :& \mathcal{V}_{\mathrm{iopp}}(\mathsf{tr}_{\mathrm{iopp}}\|\rho_{\mathrm{iopp}}) = 1 \wedge \mathsf{State}_{\mathrm{iopp}}(\mathsf{tr}_{\mathrm{iopp}}) = 0, \\ E_2 :& \mathcal{V}_{\mathrm{eval}}(\mathsf{tr}_{\mathrm{eval}}\|\rho_{\mathrm{eval}}) = 1 \wedge \mathsf{Bad} = 0 \wedge \Delta(\mathcal{C}, f) \leq 1/8 \end{aligned}$$

We have $\Pr_{\rho}[E_1] < \kappa^{\mathrm{iopp}}$ by the proof of Lemma 4.1, where the switch on verifier's final message is shown to occur with probability at most κ^{iopp}. For $\Pr_{\rho}[E_2]$, we bound it as:

$$\begin{aligned} \Pr_{\rho}[E_2] \leq & \Pr_{\rho}[\,\mathcal{V}_{\mathrm{eval}}(\mathsf{tr}_{\mathrm{eval}}\|\rho_{\mathrm{eval}}) = 1 \wedge \mathsf{Bad} = 0 \wedge \Delta(\mathcal{C}, f) \leq 1/8 \wedge \bar{f}(\mathbf{x}) \neq v\,] \\ & + \Pr_{\rho}[\,\mathcal{V}_{\mathrm{eval}}(\mathsf{tr}_{\mathrm{eval}}\|\rho_{\mathrm{eval}}) = 1 \wedge \mathsf{Bad} = 0 \wedge \Delta(\mathcal{C}, f) \leq 1/8 \wedge \bar{f}(\mathbf{x}) = v\,] \end{aligned}$$

In Lemma 4.5, we bound the first probability by $2^{-\lambda}$ as long as $t_1 = c\lambda$ for a sufficiently large constant $c > 0$, while in the second case, the decoded witness $\bar{f}$ is a valid witness for $\mathcal{R}_{Ev}$ which the extractor outputs. Thus failure to extract a witness conditioned on a switch event occurs with probability $\mathrm{negl}(\lambda)$.

Finally, we notice that whenever the State function is 0 on full transcript, so is the verifier's decision bit. This completes the proof. □

Additional Lemmas. We now prove the key additional lemmas needed in the proof of Lemma 4.2.

Lemma 4.4. *Let $f : \mathbb{F}^m \to \mathbb{F}$ be an arbitrary function with $\Delta(\mathcal{C}, f) \leq 1/8$ and let $\bar{f} \in \mathbb{F}^{<D}[X_1, \ldots, X_m]$ be the unique polynomial such that $\Delta(f, \bar{f}) \leq \frac{1}{8}$. For any $\mathbf{x} \in \mathbb{F}^m$, and $\mathbf{r}_i \leftarrow \mathbb{F}^m$ sampled uniformly for $i \in [t]$, define the event Bad as:*

$$\mathsf{Bad} \equiv \left|\left\{i \in [t] : \Delta(f|_{\mathbf{x},\mathbf{r}_i}, \bar{f}|_{\mathbf{x},\mathbf{r}_i}) \leq 1/4\right\}\right| < t_1$$

Then for $t = 4 \cdot t_1 \geq 16 \ln 2 \cdot \lambda$, we have $\Pr_\rho[\,\mathsf{Bad}\,] \leq \mathrm{negl}(\lambda)$, where $\rho = (\mathbf{r}_1, \ldots, \mathbf{r}_t)$.

Proof. To simplify calculations, we will consider m-variate functions over $\mathbb{F}^m \setminus \{\mathbf{x}\}$ and the line restrictions through $\mathbf{x}$ also without the point $\mathbf{x}$. This only impacts probabilities by $O(1/|\mathbb{F}|)$ additive factors, which we ignore. Then, lines through $\mathbf{x}$ form a partition of $\mathbb{F}^m$ (we ignore the point $\mathbf{x}$ from both). Thus we can write:

$$\Delta(f, \bar{f}) = \mathop{\mathbb{E}}_{\mathbf{r} \leftarrow \mathbb{F}^m}\left[\Delta(f|_{\mathbf{x},\mathbf{r}}, \bar{f}|_{\mathbf{x},\mathbf{r}})\right] \leq 1/8$$

Using Markov's inequality, we see that the above implies:

$$\Pr_{\mathbf{r} \leftarrow \mathbb{F}^m}[\,\Delta(f|_{\mathbf{x},\mathbf{r}}, \bar{f}|_{\mathbf{x},\mathbf{r}}) > 1/4\,] \leq 1/2$$

Let $X_i = \mathbf{1}[\Delta(f|_{\mathbf{x},\mathbf{r}_i}, \bar{f}|_{\mathbf{x},\mathbf{r}_i} \leq 1/4]$ denote independent indicator variables. Let $X = \sum_{i=1}^{t} X_i$. Then, $\mathsf{Bad} \equiv (X < t_1)$. Since $\mathbb{E}[X] \geq t/2$, taking $t_1 = t/4$, the Chernoff bound in Lemma 3.1 gives $\Pr[\mathsf{Bad}] \leq \Pr[X < t/4] \leq e^{-t/16}$, which is $< 2^{-\lambda}$ for $t \geq 16 \ln 2 \cdot \lambda$. This completes the proof.

□

Lemma 4.5. *Let $f : \mathbb{F}^m \to \mathbb{F}$ be an arbitrary function with $\Delta(\mathcal{C}, f) \leq 1/8$ and let $\bar{f} \in \mathbb{F}^{<D}[X_1, \ldots, X_m]$ be the unique polynomial such that $\Delta(f, \bar{f}) \leq \frac{1}{8}$. Let $\mathbf{x} \in \mathbb{F}^m$ and $v \in \mathbb{F}$ be such that $\bar{f}(\mathbf{x}) \neq v$. Let $\mathbf{r}_1, \ldots, \mathbf{r}_t$ be vectors in $\mathbb{F}^m$, such that*

$$\left|S = \left\{i \in [t] : \Delta(f|_{\mathbf{x},\mathbf{r}_i}, \bar{f}|_{\mathbf{x},\mathbf{r}_i}) \leq 1/4\right\}\right| \geq t_1$$

Then, for any degree $< D$ polynomials $\hat{p}_1, \ldots, \hat{p}_t$, the following holds:

$$\Pr_{(s_1,\ldots,s_t) \leftarrow \mathbb{F}^t}\left[\bigwedge_{i=1}^{t}\left(\hat{p}_i(0) = v \wedge \hat{p}_i(s_i) = f(\mathbf{x} + s_i \cdot \mathbf{r}_i)\right)\right] \leq 2^{-\lambda}$$

where $t_1 = c\lambda$ for a sufficiently large constant $c > 0$.

Proof. Since the randomness $s_1, \ldots, s_t$ applies to different clauses, the probability splits as a product.

$$\prod_{i=1}^{t}\left(\Pr_{s_i}[\,\hat{p}_i(0) = v \wedge \hat{p}_i(s_i) = f(\mathbf{x} + s_i \cdot \mathbf{r}_i)]\right)$$

Consider the factors corresponding to $i \in S$. Since $\hat{p}_i(0) = v \neq \bar{f}(\mathbf{x})$, we have $\hat{p}_i \neq \bar{f}|_{\mathbf{x},\mathbf{r}_i}$. Since $i \in S$, we have $\Delta(f|_{\mathbf{x},\mathbf{r}_i}, \bar{f}|_{\mathbf{x},\mathbf{r}_i}) \leq 1/4$. We also have $\Delta(\hat{p}_i, \bar{f}|_{\mathbf{x},\mathbf{r}_i}) \geq 1 - D/|\mathbb{F}|$. Thus, by triangle inequality we have:

$$\Delta(\hat{p}_i, f|_{\mathbf{x},\mathbf{r}_i}) > 3/4 - D/|\mathbb{F}|$$

and thus $\Pr_{s_i}[\,\hat{p}_i(s_i) = f(\mathbf{x} + s_i \cdot \mathbf{r}_i)\,] \leq 1/4 + D/|\mathbb{F}|$. Thus, the overall probability is bounded by $(1/4 + D/|\mathbb{F}|)^{t_1}$ which is $2^{-\lambda}$ when $t_1 = c\lambda$ for a sufficiently large constant $c > 0$. This completes the proof. □

4.2 Multivariate PCS from KU Preprocessing

We now construct a multivariate polynomial commitment scheme using the IOPs in Figures 4 and 5. In our construction, we specialize Definition 3.6 to the uniform individual-degree bound $\mathbf{d} = (d, \ldots, d)$ and use $D = d \cdot m$ as a strict upper bound on the total degree.

The Construction. The multivariate polynomial commitment scheme MPCS is described by the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, q, m, d, \delta) \to \mathsf{pp}$: Outputs $\mathsf{pp} = (\mathbb{F}_q, m, d, \delta)$. We assume $\delta = 1/8$ and $D = d \cdot m$ throughout.
- $\mathsf{Commit}(\mathsf{pp}, \bar{f}) \to [\![f]\!]$: Given m-variate polynomial $\bar{f}$ with degree $< d$ in each variable, $\mathcal{P}$ outputs a function oracle $[\![f]\!]$ for $f : \mathbb{F}^m \to \mathbb{F}$ given by $f(\mathbf{x}) = \bar{f}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{F}^m$.
- $\mathsf{Open}(\mathsf{pp}, f, \bar{f}, \delta)$: Output 1 iff $\Delta(f, \bar{f}) \leq \delta$ and $\bar{f}$ has total degree at most D.
- $\mathsf{CheckProximity}\langle \mathcal{P}(\bar{f}), \mathcal{V} \rangle(\mathsf{pp}, [\![f]\!], \delta) \to b$: Executes the IOPP in Figure 4 with $\mathcal{P}(\mathsf{pp}, f, \bar{f})$ and $\mathcal{V}(\mathsf{pp}, [\![f]\!])$. Outputs b as $\mathcal{V}$'s output in the IOPP.
- $\mathsf{Eval}\langle \mathcal{P}(\bar{f}), \mathcal{V} \rangle(\mathsf{pp}, [\![f]\!], \mathbf{x}, v) \to b$: Executes the IOP in Figure 5 with $\mathcal{P}(\mathsf{pp}, f, \mathbf{x}, \mathbf{v}, \bar{f})$ and $\mathcal{V}(\mathsf{pp}, [\![f]\!], \mathbf{x}, \mathbf{v})$. Outputs b as $\mathcal{V}$'s output in the IOP.

Theorem 4.1 (Our Multivariate PCS)**.** *The scheme* $\mathsf{MPCS} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{CheckProximity}, \mathsf{Eval})$ *satisfies* completeness*,* δ-binding*, and* knowledge-soundness*. Additionally, the evaluation IOP satisfies* round-by-round knowledge soundness*. When polynomials are pre-processed using techniques of Kedlaya-Umans [KU08, Uma08], we obtain commitment time of* $N^{1+\epsilon}$*, and evaluation time of* $O(\text{polylog}(N))$ *for both the prover and verifier, where* $N = d^m$*.*

Proof. Completeness, knowledge soundness and round-by-round knowledge soundness follow from the corresponding properties of the evaluation IOP in Figure 5, which are established in Lemma 4.2. We show the remaining δ-binding below. Formally, for every adversary $\mathcal{A}$, we show that

$$\Pr\left[\begin{array}{l|l} (b=1) \wedge \bar{f}_0 \neq \bar{f}_1 & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ \wedge \mathsf{Open}(\mathsf{pp}, f, \bar{f}_0, \delta) = 1 & ([\![f]\!], \bar{f}_0, \bar{f}_1, \mathsf{aux}) \leftarrow \mathcal{A}_0(\mathsf{pp}); \\ \wedge \mathsf{Open}(\mathsf{pp}, f, \bar{f}_1, \delta) = 1 & b \leftarrow \mathsf{CheckProximity}\langle \mathcal{A}_1(\mathsf{aux}), \mathcal{V} \rangle(\mathsf{pp}, [\![f]\!], \delta) \end{array}\right] \leq \text{negl}(\lambda)$$

This follows from the definition of the Open relation. If $\mathsf{Open}(\mathsf{pp}, f, \bar{f}_b, \delta) = 1$, for $b \in \{0, 1\}$, then $\Delta(f, \bar{f}_b) \leq \delta$. By a union bound, $\Delta(\bar{f}_0, \bar{f}_1) \leq 2\delta$. But $2\delta < 1 - \frac{D}{q}$, the minimum distance of the Reed-Muller code when $\delta = 1/8$ and $D < q/2$. This proves the binding. The efficiency claims follow from the efficiency guarantees of KU preprocessing. This completes the proof of the theorem. □

Completeness-Soundness Gap. In MPCS, completeness is stated for polynomials with individual degrees $< d$, while knowledge-soundness holds with respect to polynomials with total degree $< D = d \cdot m$, i.e, the honest prover is restricted to a smaller set of polynomials than what the verifier checks. We adopt this gap, as KU preprocessing does not work with general m-variate polynomials of total degree $< D$. The gap does not impact our applications of the PCS in this paper. One could avoid this gap at the cost of making evaluation $\widetilde{O}(N)$ for the prover.

4.3 Univariate PCS from KU Preprocessing

We can obtain a univariate PCS in a black-box manner from the multivariate PCS MPCS from Section 4.2, exploiting the isomorphism $\Psi_{d,m}$ from $\mathbb{F}^{<N}[X]$ to $\mathbb{F}^{<(d,\ldots,d)}[X_0, \ldots, X_{m-1}]$ as defined in Section 3.4, where $N = d^m$. We define the resulting univariate PCS below.

The Construction. The univariate polynomial commitment scheme UPCS is described by the following PPT algorithms.

- $\mathsf{Setup}(1^\lambda, q, N) \to \mathsf{pp}$: Choose d, m such that $N = d^m$, $\delta = 1/8$. Outputs $\mathsf{pp} \leftarrow \mathsf{MPCS.Setup}(1^\lambda, q, m, d, \delta)$.
- $\mathsf{Commit}(\mathsf{pp}, \hat{f}) \to [\![f]\!]$: Given univariate polynomial $\hat{f}$ with degree $< N$, $\mathcal{P}$ constructs $\bar{f} = \Psi_{d,m}(\hat{f})$ and outputs $[\![f]\!] \leftarrow \mathsf{MPCS.Commit}(\mathsf{pp}, \bar{f})$.
- $\mathsf{Open}(\mathsf{pp}, f, \hat{f}, \delta)$: Output 1 iff $\mathsf{MPCS.Open}(\mathsf{pp}, f, \Psi_{d,m}(\hat{f}), \delta) = 1$, where $\Psi_{d,m}(\hat{f})$ outputs the multivariate representation of the univariate polynomial $\hat{f}$.
- $\mathsf{CheckProximity}\langle \mathcal{P}(\hat{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \delta) \to b$: Outputs

$$b \leftarrow \mathsf{MPCS.CheckProximity}\langle \mathcal{P}(\bar{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!])$$

  where $\bar{f} = \Psi_{d,m}(\hat{f})$.
- $\mathsf{Eval}\langle \mathcal{P}(\hat{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], x, v) \to b$: Outputs

$$b \leftarrow \mathsf{MPCS.Eval}\langle \mathcal{P}(\bar{f}), \mathcal{V}\rangle(\mathsf{pp}, [\![f]\!], \mathbf{x}, v)$$

  where $\bar{f} = \Psi_{d,m}(\hat{f})$ and $\mathbf{x} = (x, x^d, \ldots, x^{d^{m-1}})$.

The univariate polynomial commitment scheme UPCS also satisfies completeness, binding, knowledge-soundness, and round-by-round knowledge soundness. It inherits the completeness-soundness gap from MPCS, where the honest prover is restricted to polynomials of degree $< N$ (corresponding to m-variate polynomials of degree $< d$ in each variable under $\Psi_{d,m}$), while knowledge-soundness enforces a degree bound of $N\,\mathrm{polylog}(N)$ (corresponding to m-variate polynomials with total degree bound $d \cdot m$ under the map $\Psi_{d,m}$).

5 IOP for Sublinear Static Lookup

We use the PCS from Section 4 to design the first lookup arguments for both *static* and *dynamic* tables that simultaneously: (i) achieve sublinear online prover, (ii) admit plausibly post-quantum realizations with transparent setup based purely on cryptographic hashing, and (iii) avoid the prohibitive costs of preprocessing certification inside a general-purpose SNARK incurred by prior work [CCC+25]. In this section, we present the *static* case.

Given integers N, k with $k \ll N$, and committed vectors $\mathbf{t} \in \mathbb{F}^N$ and $\mathbf{v} \in \mathbb{F}^k$, prove that $\mathbf{v}$ is a *sub-vector* of $\mathbf{t}$, i.e, there exist a_i such that $v_i = t_{a_i}$ for all $i \in [k]$. We denote the same by $\mathbf{v} \preceq \mathbf{t}$. Let $\mathbb{L}$ and $\mathbb{V}$ denote the set of N^{th} and k^{th} roots of unity in $\mathbb{F}$, and fix integers $d, m = \mathrm{polylog}(N)$ with $N = d^m$. Using polynomials to interpolate the vectors over respective domains leads to the following oracle relation for lookup.

Definition 5.1 (Subvector Relation)**.** *The oracle relation $\mathcal{R}_{\text{subvec}}$ consists of $(\mathbb{x}, \mathbb{y}, \mathbb{w})$ where $\mathbb{x} = (\mathbb{F}, \mathbb{L}, \mathbb{V}, d, m)$, $\mathbb{y} = ([\![t]\!], [\![v]\!])$, $\mathbb{w} = (\bar{t}, \hat{v})$ and $\mathbf{v} \preceq \mathbf{t}$. Here $\mathbf{t}$ is the vector of evaluations of $\hat{t}$ over $\mathbb{L}$, where $\hat{t}(X) = \bar{t}(X, X^d, \ldots, X^{d^{m-1}})$, and $\mathbf{v}$ is the vector of evaluations of $\hat{v}$ over $\mathbb{V}$.*

The IOP. The IOP for the relation $\mathcal{R}_{\text{subvec}}$ is described formally in Figure 6. The table oracle is $[\![t]\!] \leftarrow \mathsf{UPCS.Commit}(\mathsf{pp}, \hat{t})$, an oracle for $\bar{t} = \Psi_{d,m}(\hat{t})$. Its evaluations are proved with $\mathsf{MPCS.Eval}$. All other polynomials, including $\hat{v}$, have degree less than some $\ell = O(dmk) = O(k\,\mathrm{polylog}(N))$. They need no KU preprocessing, so we commit to them with any univariate PCS PC (Definition 3.6), such as Basefold [ZCF24] or WHIR [ACFY25]. Throughout, pp contains public parameters for every scheme and every polynomial size we use, and each invocation implicitly uses the appropriate part of pp. We only require that $\mathsf{PC.Eval}$ and $\mathsf{MPCS.Eval}$ are *sound* for $\mathcal{R}_{Ev}$, with a proximity parameter δ within the unique decoding radius of the corresponding code. We also assume that $(dm\ell + k)/|\mathbb{F}| = \mathrm{negl}(\lambda)$. Besides Equations (1) and (2), the prover shows $\hat{b}^d_{m-1} \equiv 1 \pmod{\hat{Z}_{\mathbb{V}}}$, i.e., $\hat{a}^N \equiv 1$, so that each $\hat{a}(\nu^j)$ lies in $\mathbb{L}$.

The formal protocol then implements the identities from Section 2.2: the prover commits to the address polynomial, its successively reduced powers, and the corresponding quotient polynomials, after which the

Protocol: SubvectorIOP for $\mathcal{R}_{\text{subvec}}$

Input: $\mathcal{P}$ gets $(\mathbb{x}, \mathbb{y}, \mathbb{w})$ and $\mathcal{V}$ gets $(\mathbb{x}, \mathbb{y})$, where $\mathbb{x} = (\mathbb{F}, \mathbb{L}, \mathbb{V}, d, m)$, $\mathbb{y} = (\llbracket t \rrbracket, \llbracket v \rrbracket)$ and $\mathbb{w} = (\bar{t}, \hat{v})$. Both parties hold pp.

1. **Prover's first message.** $\mathcal{P}$ computes the following.

   (a) An index vector $\mathbf{a} = (a_0, \ldots, a_{k-1}) \in [0, N)^k$ with $v_j = t_{a_j}$ for all $j \in [0, k)$, and the polynomial $\hat{a} \in \mathbb{F}_{<k}[X]$ with $\hat{a}(\nu^j) = \omega^{a_j}$ for all $j \in [0, k)$. Set $\hat{b}_0 = \hat{a}$.

   (b) For $i = 1, \ldots, m-1$, the polynomial $\hat{b}_i \in \mathbb{F}_{<k}[X]$ given by $\hat{b}_i(X) = \hat{b}_{i-1}(X)^d \bmod \hat{Z}_{\mathbb{V}}(X)$.

   (c) The quotient polynomials $\hat{q}$ and $\hat{q}_1, \ldots, \hat{q}_m$ satisfying

$$\bar{t}\big(\hat{b}_0(X), \ldots, \hat{b}_{m-1}(X)\big) - \hat{v}(X) = \hat{q}(X)\, \hat{Z}_{\mathbb{V}}(X), \tag{12}$$

$$\hat{b}_i(X) - \hat{b}_{i-1}(X)^d = \hat{q}_i(X)\, \hat{Z}_{\mathbb{V}}(X), \qquad 1 \le i < m, \tag{13}$$

$$\hat{b}_{m-1}(X)^d - 1 = \hat{q}_m(X)\, \hat{Z}_{\mathbb{V}}(X). \tag{14}$$

   $\mathcal{P}$ sends the oracles

$$\llbracket a \rrbracket \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \hat{a}), \quad \llbracket b_i \rrbracket \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \hat{b}_i),\ 1 \le i < m,$$

$$\llbracket q \rrbracket \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \hat{q}), \quad \llbracket q_i \rrbracket \leftarrow \mathsf{PC.Commit}(\mathsf{pp}, \hat{q}_i),\ 1 \le i \le m.$$

   We write $\llbracket b_0 \rrbracket = \llbracket a \rrbracket$.

2. **Verifier's challenge.** $\mathcal{V}$ samples $x \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.

3. **Prover's claimed evaluations.** $\mathcal{P}$ sends

$$y_i = \hat{b}_i(x)\ (0 \le i < m), \quad y_v = \hat{v}(x), \quad y_q = \hat{q}(x), \quad y_{q_i} = \hat{q}_i(x)\ (1 \le i \le m),$$

   and $y_t = \bar{t}(y_0, \ldots, y_{m-1})$.

4. **Evaluation IOPs.** $\mathcal{P}$ and $\mathcal{V}$ run the following protocols, and let $\beta_1, \ldots, \beta_{2m+2}$ be the bits output by $\mathcal{V}$:

   - $\mathsf{PC.Eval}\langle \mathcal{P}(\hat{b}_i), \mathcal{V} \rangle(\mathsf{pp}, \llbracket b_i \rrbracket, x, y_i)$ for $0 \le i < m$;
   - $\mathsf{PC.Eval}\langle \mathcal{P}(\hat{v}), \mathcal{V} \rangle(\mathsf{pp}, \llbracket v \rrbracket, x, y_v)$ and $\mathsf{PC.Eval}\langle \mathcal{P}(\hat{q}), \mathcal{V} \rangle(\mathsf{pp}, \llbracket q \rrbracket, x, y_q)$;
   - $\mathsf{PC.Eval}\langle \mathcal{P}(\hat{q}_i), \mathcal{V} \rangle(\mathsf{pp}, \llbracket q_i \rrbracket, x, y_{q_i})$ for $1 \le i \le m$;
   - $\mathsf{MPCS.Eval}\langle \mathcal{P}(\bar{t}), \mathcal{V} \rangle\big(\mathsf{pp}, \llbracket t \rrbracket, (y_0, \ldots, y_{m-1}), y_t\big)$.

5. $\mathcal{V}$ accepts iff all evaluation IOPs in Step 4 accept and

$$y_t - y_v = y_q \cdot \hat{Z}_{\mathbb{V}}(x), \qquad \text{(Equation (12) at } x\text{)}$$

$$y_i - y_{i-1}^d = y_{q_i} \cdot \hat{Z}_{\mathbb{V}}(x) \quad (1 \le i < m), \qquad \text{(Equation (13) at } x\text{)}$$

$$y_{m-1}^d - 1 = y_{q_m} \cdot \hat{Z}_{\mathbb{V}}(x). \qquad \text{(Equation (14) at } x\text{)}$$

Figure 6: IOP for the subvector relation $\mathcal{R}_{\text{subvec}}$.

verifier samples a single random evaluation point. Evaluation protocols for the constituent commitments reduce all remaining checks to scalar identities at this point. In addition to the composition identity establishing the queried values, the protocol checks the recurrence between successive address powers and the final root-of-unity condition; the latter guarantees that every encoded address indeed corresponds to an entry of the committed table.

Theorem 5.1. *Suppose* PC.Eval *and* MPCS.Eval *are sound for* $\mathcal{R}_{Ev}$*. Then the IOP in Figure 6 is knowledge-sound (Definition 3.2) for* $\mathcal{R}_{\text{subvec}}$ *with knowledge error* $\text{negl}(\lambda)$*. The extractor* $\mathcal{E}$ *outputs the* δ*-close decodings* $(\bar{t}^*, \hat{v}^*)$ *of* $[\![t]\!]$ *and* $[\![v]\!]$*.*

Proof. Let $[\![g]\!]$ be an oracle in the protocol, i.e., one of $[\![t]\!], [\![v]\!]$ or an oracle sent by $\mathcal{P}$, and let pp be its public parameters. A δ*-close decoding* of $[\![g]\!]$ is a polynomial g^* in the corresponding code with $\mathsf{Open}(\mathsf{pp}, g, g^*, \delta) = 1$. For the oracles committed with PC, this is a univariate polynomial of degree $< \ell$. For $[\![t]\!]$, it is an m-variate polynomial $\bar{t}^*$ of total degree $< dm$. Since δ is within the unique decoding radius, each oracle has at most one such decoding. Moreover, the corresponding codes are efficiently uniquely decodable within distance δ. The extractor $\mathcal{E}$ therefore obtains the unique δ-close decodings of $[\![t]\!]$ and $[\![v]\!]$, and outputs these as $(\bar{t}^*, \hat{v}^*)$. In the analysis below, we also denote the decodings of auxiliary oracles sent by the prover by $\hat{a}^* = \hat{b}_0^*$, $\hat{b}_i^*$, $\hat{q}^*$, and $\hat{q}_i^*$. First, we identify events that prevent $\mathcal{E}$ from producing a valid witness, and bound their probability.

Not all oracles decode. If some $[\![g]\!]$ has no δ-close decoding, then its evaluation instance in Step 4 is not in $\mathcal{R}_{Ev}$. By soundness of PC.Eval and MPCS.Eval,

$$\Pr\left[\mathcal{V} \text{ accepts} \wedge \text{some } [\![g]\!] \text{ has no } \delta\text{-close decoding}\right] \leq \text{negl}(\lambda).$$

Decodings do not match evaluations. Here, we assume every oracle has a decoding. Then an evaluation instance is in $\mathcal{R}_{Ev}$ exactly when the decoding satisfies the claimed evaluation. Again by soundness,

$$\Pr\left[\mathcal{V} \text{ accepts} \wedge \text{some } g^* \text{ does not satisfy its evaluation claim}\right] \leq \text{negl}(\lambda).$$

Outside these two events, $\hat{b}_i^*(x) = y_i$, $\hat{v}^*(x) = y_v$, $\hat{q}^*(x) = y_q$, $\hat{q}_i^*(x) = y_{q_i}$ and $\bar{t}^*(y_0, \ldots, y_{m-1}) = y_t$. The verifier's checks in Step 5 then say that $P_0(x) = \cdots = P_m(x) = 0$, where

$$\begin{aligned} P_0 &= \bar{t}^*(\hat{b}_0^*, \ldots, \hat{b}_{m-1}^*) - \hat{v}^* - \hat{q}^* \hat{Z}_{\mathbb{V}}, \\ P_i &= \hat{b}_i^* - (\hat{b}_{i-1}^*)^d - \hat{q}_i^* \hat{Z}_{\mathbb{V}} \quad (1 \leq i < m), \\ P_m &= (\hat{b}_{m-1}^*)^d - 1 - \hat{q}_m^* \hat{Z}_{\mathbb{V}}. \end{aligned}$$

Polynomial identities fail. All oracles are sent before $\mathcal{V}$ samples x. Hence their decodings, and therefore each P_i, are fixed before x is sampled. Each P_i has degree less than $dm\ell + k$. If some $P_i \not\equiv 0$, then $P_i(x) = 0$ with probability at most $(dm\ell + k)/|\mathbb{F}| = \text{negl}(\lambda)$. Hence, except with negligible probability, all P_i are identically zero and therefore vanish on $\mathbb{V}$.

Witness extraction. Fix $j \in [0, k)$. The identities $P_1, \ldots, P_m$ give $\hat{b}_0^*(\nu^j)^{d^m} = \hat{b}_0^*(\nu^j)^N = 1$. Hence $\hat{b}_0^*(\nu^j) = \omega^{a_j}$ for some $a_j \in [0, N)$, and $\hat{b}_i^*(\nu^j) = \omega^{a_j d^i}$ for all i. Let $\hat{t}^*(X) := \bar{t}^*(X, X^d, \ldots, X^{d^{m-1}})$, and let $\mathbf{t}, \mathbf{v}$ be the evaluations of $\hat{t}^*$ and $\hat{v}^*$ over $\mathbb{L}$ and $\mathbb{V}$, as in Definition 5.1. Evaluating P_0 at ν^j gives

$$v_j = \hat{v}^*(\nu^j) = \bar{t}^*(\omega^{a_j}, \omega^{a_j d}, \ldots, \omega^{a_j d^{m-1}}) = \hat{t}^*(\omega^{a_j}) = t_{a_j}.$$

Therefore $\mathbf{v} \preceq \mathbf{t}$, i.e., $(\mathbb{x}, \mathbb{y}, (\bar{t}^*, \hat{v}^*)) \in \mathcal{R}_{\text{subvec}}$. Summing the above failure probabilities gives

$$\Pr\left[\mathcal{V} \text{ accepts} \wedge (\mathbb{x}, \mathbb{y}, \mathcal{E}(\mathbb{x}, \mathbb{y})) \notin \mathcal{R}_{\text{subvec}}\right] \leq \text{negl}(\lambda).$$

This completes the proof. □

Since all auxiliary polynomials have total size $O(k \,\text{polylog}(N))$, their construction and commitment also cost $O(k \,\text{polylog}(N))$. Computing the reduced composition with the table requires $O(k \,\text{polylog}(N))$ evaluations of its multivariate representation; after preprocessing, each such evaluation costs $\text{polylog}(N)$, which is absorbed into the same $O(k \,\text{polylog}(N))$ notation. Finally, the verifier requests only polylogarithmically many evaluations of the resulting commitments, whose opening costs are likewise polylogarithmic. Thus the entire online prover cost is $O(k \,\text{polylog}(N))$, as claimed in Informal Theorem 1.2.

6 IOP for Sublinear Updatable Lookup

We now provide a detailed description of the IOP for sublinear *updatable* lookup. Beyond the amortization strategy and sparse representation described in Sections 2.3 to 2.6, we formalize the state maintained across updates and the consistency conditions that must be certified when this state changes. Concretely, we define a *leveled table oracle* (Definition 6.1) whose non-empty levels consist of commitments to the sparse rational representation $(\hat{p}_j, \hat{q}_j, \hat{q}'_j)$. We also define a well-formedness predicate (Definition 6.2) ensuring that $\hat{q}_j$ divides the table vanishing polynomial and that $\hat{q}'_j$ is its formal derivative; a well-formed collection of levels therefore provides query access to a well-defined table polynomial. We then formalize a batch update as an oracle relation between the source and destination leveled tables (Definition 6.3). Besides establishing the old and new values at the queried addresses, this relation requires the newly created level to be well-formed and the two tables to agree everywhere outside the updated addresses.

The update IOP (described formally in Figure 7) certifies these conditions through several modular sub-protocols. It first enforces the successive address-power identities and the root-of-unity condition used already in the static construction. For each relevant level, it then proves the sparse-table lookup identities, including the partition according to whether the denominator polynomial vanishes at a queried address. The new level additionally requires certifying divisibility of its denominator by the table vanishing polynomial and consistency of the committed derivative; the latter is handled through a virtual derivative oracle and its evaluation protocol. Finally, agreement of the source and destination tables away from the updated positions is delegated to the $\mathsf{AIdentical}$ sub-protocol of Section 6.3.1. The soundness proof follows the same decoding-and-random-evaluation template as the static argument, but combines these checks to establish all conditions of the update relation.

The subprotocols $\mathsf{AIdentical}, \mathsf{DerivEval}$, and $\mathsf{ProdEval}$ (which are derivative and divisibility checks) are deferred to Section 6.3.

6.1 Definitions

Throughout, k is the batch size and $N = k2^\ell = d^M$, where k and d are powers of two and $d, M = \mathrm{polylog}(N)$. As before, $\mathbb{L}$ and $\mathbb{V}$ are the groups of N^{th} and k^{th} roots of unity, generated by ω and ν. For $j \in \{0, \ldots, \ell\}$, let $s_j = k2^j$, and let m_j be the least integer with $d^{m_j} > s_j$. Note that $m_j \le M + 1$. For a univariate polynomial $\hat{q}$, we write $\partial\hat{q}$ for its formal derivative.

As in Section 5, only the table levels use MPCS with KU preprocessing. Every other polynomial, including those behind $[\![v]\!]$, $[\![a]\!]$ and $[\![v']\!]$, is committed with a generic univariate PCS PC (Definition 3.6). As before, pp contains public parameters for all schemes and polynomial sizes, and each invocation uses the appropriate part implicitly. We require only that $\mathsf{PC.Eval}$, $\mathsf{UPCS.Eval}$ and $\mathsf{MPCS.Eval}$ are sound for $\mathcal{R}_{Ev}$, with proximity within the unique decoding radius.

Leveled Tables. As in Section 2.4, the table is split into levels, and each level is given in the sparse representation of Section 2.5.

Definition 6.1 (Leveled Representation)**.** *A* leveled table oracle *is a tuple* $C = ([\![C_0]\!], \ldots, [\![C_\ell]\!])$ *of* level oracles*. Each level oracle is either the constant oracle* $[\![C_j]\!] = 0$ *(an empty level) or a triple* $[\![C_j]\!] = ([\![p_j]\!], [\![q_j]\!], [\![q'_j]\!])$*. In the latter case, the components are* UPCS *commitments, with parameters* (d, m_j)*, to univariate polynomials* $\hat{p}_j, \hat{q}_j, \hat{q}'_j$*, respectively, where* $\deg \hat{q}_j \le s_j$ *and* $\deg \hat{p}_j, \deg \hat{q}'_j < s_j$ *(see Equation* (4)*). We write* $\bar{p}_j = \Psi_{d,m_j}(\hat{p}_j)$*, and similarly* $\bar{q}_j$ *and* $\bar{q}'_j$*.*

- *A* query *to* $[\![C_j]\!]$ *at* $y \in \mathbb{F}$ *returns* $[\![C_j]\!](y) = 0$ *if* $[\![C_j]\!] = 0$*. Otherwise it returns*

$$[\![C_j]\!](y) = \begin{cases} \hat{Z}_{\mathbb{L}}(y)\,\hat{p}_j(y)/\hat{q}_j(y) & \text{if } \hat{q}_j(y) \neq 0, \\ N y^{-1} \hat{p}_j(y)/\hat{q}'_j(y) & \text{if } \hat{q}_j(y) = 0. \end{cases}$$

  This uses one query to each of $[\![p_j]\!], [\![q_j]\!], [\![q'_j]\!]$*, as described in Section 2.5.*

- *A* query *to* C *at* $y \in \mathbb{F}$ *returns* $[\![C]\!](y) = \sum_{j=0}^{\ell} [\![C_j]\!](y)$*.*

Definition 6.2 (Well-formed Level)**.** *The predicate* WellFormed *on level oracles is defined as follows. An empty level is well-formed. A level oracle* $[\![C_j]\!] \neq 0$*, with underlying polynomials* $\hat{p}_j, \hat{q}_j, \hat{q}'_j$*, is well-formed if and only if* $\hat{q}_j \neq 0$*,* $\hat{q}_j \mid \hat{Z}_{\mathbb{L}}$*, and* $\hat{q}'_j = \partial \hat{q}_j$*.*

We note that a well formed level $[\![C_j]\!]$ simulates query access to a polynomial of degree $< N$, explicitly given by $\hat{t}_j(X) = \hat{Z}_{\mathbb{L}}(X)\hat{p}_j(X)/\hat{q}_j(X)$, i.e, $[\![C_j]\!](y) = \hat{t}_j(y)$ for every $y \in \mathbb{F}$. This is clear when $\hat{q}_j(y) \neq 0$. When $\hat{q}_j(y) = 0$, $[\![C_j]\!](y) = \hat{t}_j(y)$ follows from Lemma 2.2. For an empty level we set $\hat{t}_j = 0$. Hence, if every level of C is well-formed, then $[\![C]\!](y) = \sum_j \hat{t}_j(y)$ for every $y \in \mathbb{F}$. In other words, $[\![C]\!]$ simulates query access to polynomial $\hat{t} = \sum_j \hat{t}_j$. The *table* encoded by C is the function $T : [0, N) \to \mathbb{F}$ given by $T(x) = \hat{t}(\omega^x)$.

The Update Relation. An update writes new values at k positions. In the hierarchical strategy of Section 2.6, only level μ receives a new oracle. Levels below μ are *zeroed*, and levels above μ are unchanged. The relation below captures a *correct update* from a source table C to a destination table C^*. It assumes that the source table is well-formed, i.e., $\mathsf{WellFormed}([\![C_j]\!]) = 1$ for every j. In an update sequence, source levels were created by earlier updates, which would have established their well-formedness at that time. Formally, $\mathcal{R}_{\text{upd}}$ is defined only over instances whose source levels are well-formed, and the IOP checks well-formedness only for the new level. We note that this is slight weakening of the setting in [DGP+24], where update relation allowed adversarial source tables, and both well-formed source and destination tables could be extracted from a successful prover.

Definition 6.3 (Update Relation)**.** *The oracle relation* $\mathcal{R}_{\text{upd}}$ *consists of* $(\mathtt{x}, \mathtt{y}, \mathtt{w})$ *as follows.*

- $\mathtt{x} = (\mathbb{F}, \mathbb{L}, \mathbb{V}, d, M, \ell, \mu)$ *for a level* $\mu \in \{0, \ldots, \ell\}$*.*
- $\mathtt{y} = (C, [\![v]\!], [\![a]\!], [\![v']\!], C^*)$*. Here* $C = ([\![C_0]\!], \ldots, [\![C_\ell]\!])$ *is a well-formed source table, and* $C^* = ([\![C_0^*]\!], \ldots, [\![C_\ell^*]\!])$ *is the destination table. Its level* $[\![C_\mu^*]\!] = ([\![p_*]\!], [\![q_*]\!], [\![q'_*]\!])$ *is a new level oracle, and* $[\![C_j^*]\!] \in \{[\![C_j]\!], 0\}$ *for every* $j \neq \mu$*.*
- $\mathtt{w}$ *consists of the polynomials behind every oracle in* $\mathtt{y}$*, namely* $\hat{v}, \hat{a}, \hat{v}'$*, the level polynomials of* C*, and* $(\hat{p}_*, \hat{q}_*, \hat{q}'_*)$*.*

Let $T, T^* : [0, N) \to \mathbb{F}$ *be the tables encoded by* C *and* C^**. The following conditions hold for* $(\mathtt{x}, \mathtt{y}, \mathtt{w}) \in \mathcal{R}_{\text{upd}}$*.*

1. $\mathsf{WellFormed}([\![C_\mu^*]\!]) = 1$*;*
2. $\hat{a}(\nu^i) \in \mathbb{L}$ *for all* $i \in [0, k)$*; we write* $\hat{a}(\nu^i) = \omega^{a_i}$*;*
3. $\hat{v}(\nu^i) = T(a_i)$ *and* $\hat{v}'(\nu^i) = T^*(a_i)$ *for all* $i \in [0, k)$*;*
4. $T^*(x) = T(x)$ *for every* $x \in [0, N) \setminus \{a_0, \ldots, a_{k-1}\}$*.*

Apart from emptied levels, at most one destination level differs from the source. Conditions 3 and 4 say that C^* is obtained from C by writing $\mathbf{v}'$ at the positions $\mathbf{a}$, and that $\mathbf{v}$ is the old content at those positions. Repeated addresses are allowed; condition 3 forces $a_i = a_{i'} \Rightarrow v'_i = v'_{i'}$.

6.2 The IOP

We write $\hat{t}$ and $\hat{t}^*$ for the polynomials to which C and C^* simulate query access, so that $T(x) = \hat{t}(\omega^x)$ and $T^*(x) = \hat{t}^*(\omega^x)$. We now describe the main steps of the IOP for proving correctness of update, building on the blueprint in Section 2.3. The protocol is described formally in Figure 7.

Enforcing Address Powers. As in Section 5, let $\hat{b}_0 = \hat{a}$ and $\hat{b}_i = \hat{b}_{i-1}^d \bmod \hat{Z}_{\mathbb{V}}$ for $1 \leq i < M$. The prover shows

$$\hat{b}_i - \hat{b}_{i-1}^d \equiv 0 \ (1 \leq i < M), \qquad \hat{b}_{M-1}^d - 1 \equiv 0 \pmod{\hat{Z}_{\mathbb{V}}}. \tag{15}$$

We set $\hat{b}_M = 1$. For level j, let $\mathbf{B}_j = (\hat{b}_0, \ldots, \hat{b}_{m_j - 1})$. If Equation (15) holds, then $\hat{b}_i(\eta) = \hat{a}(\eta)^{d^i}$ for all $\eta \in \mathbb{V}$ and $i \leq M$. Hence $\bar{p}_j(\mathbf{B}_j(\eta)) = \hat{p}_j(\hat{a}(\eta))$, and likewise for $\bar{q}_j$ and $\bar{q}'_j$.

Protocol: UpdateIOP for $\mathcal{R}_{\text{upd}}$

Input: $\mathcal{P}$ gets $(\mathbb{x}, \mathbb{y}, \mathbb{w})$ and $\mathcal{V}$ gets $(\mathbb{x}, \mathbb{y})$. Both parties hold pp, and every new oracle is committed with PC. Let Λ be the set of distinct non-empty level oracles among C and C^*. For $\lambda \in \Lambda$ at level j, let $m_\lambda = m_j$. The messages of $\mathsf{AIdentical}$ refer to Figure 8.

1. **Round 1.** $\mathcal{P}$ sends $[\![b_i]\!]$ $(1 \le i < M)$, the lookup oracles $[\![v_\lambda]\!], [\![g_\lambda]\!], [\![h_\lambda]\!], [\![\beta_\lambda]\!]$ for $\lambda \in \Lambda$, the oracles $[\![e_i]\!]$ $(1 \le i < M)$ of Equation (17) for $\hat{G} = \hat{q}_*$, and the quotients of the congruences in Equations (15) to (17). It also sends the Round 1 oracles of $\mathsf{AIdentical}$.
   $\mathcal{V}$ sends $z \xleftarrow{\$} \mathbb{F}$.
2. **Claims.** $\mathcal{P}$ sends:
   - the claimed evaluations at z of every univariate oracle above and of $[\![v]\!], [\![v']\!], [\![a]\!], [\![q_*]\!]$, with y_{q_*} the value for $[\![q_*]\!]$;
   - for each $\lambda \in \Lambda$, the claimed values of $\bar{p}_\lambda, \bar{q}_\lambda, \bar{q}'_\lambda$ at $\mathbf{B}_\lambda(z)$, the vector of claimed values of $\hat{b}_0, \ldots, \hat{b}_{m_\lambda - 1}$ at z (with $\hat{b}_M(z) = 1$);
   - a value $y_{q'_*}$, claimed both as $\hat{q}'_*(z)$ and as $\partial \hat{q}_*(z)$;
   - the claims of $\mathsf{AIdentical}$.
3. **Evaluations.** $\mathcal{P}$ and $\mathcal{V}$ run:
   - $\mathsf{PC.Eval}$ or $\mathsf{UPCS.Eval}$ for every claimed evaluation at z, including $y_{q'_*}$ for $[\![q'_*]\!]$;
   - $\mathsf{MPCS.Eval}(\mathsf{pp}, \cdot, \mathbf{B}_\lambda(z), \cdot)$ on $[\![p_\lambda]\!], [\![q_\lambda]\!], [\![q'_\lambda]\!]$ for each $\lambda \in \Lambda$;
   - $\mathsf{DerivEval}([\![q_*]\!], z, y_{q'_*})$ (Section 6.3.2);
   - the evaluation IOPs of $\mathsf{AIdentical}$.
4. **Decision.** $\mathcal{V}$ accepts iff all evaluation IOPs accept, the checks of $\mathsf{AIdentical}$ pass, and the following hold at $X = z$ with the claimed values in place of the evaluations:
   - Equations (15) to (17), each congruence in its quotient form;
   - $\hat{v}(z) = \sum_\lambda \hat{v}_\lambda(z)$ and $\hat{v}'(z) = \sum_{\lambda'} \hat{v}_{\lambda'}(z)$, where λ and λ' range over the non-empty levels of C and C^*, respectively.

Figure 7: IOP for the update relation $\mathcal{R}_{\text{upd}}$.

Enforcing Level Lookups. Following Section 2.5, the lookup $\hat{v}_j(\eta) = \hat{t}_j(\hat{a}(\eta))$ for $\eta \in \mathbb{V}$ uses a partition $\hat{g}_j \hat{h}_j = \hat{Z}_{\mathbb{V}}$. Here $\hat{h}_j = \gcd(\hat{q}_j(\hat{a}), \hat{Z}_{\mathbb{V}})$ vanishes exactly where $\hat{a}(\eta)$ is a root of $\hat{q}_j$. The prover also sends a Bezout polynomial $\hat{\beta}_j$, i.e., an inverse of $\bar{q}_j(\mathbf{B}_j)$ modulo $\hat{g}_j$. The identities are

$$\begin{aligned} \hat{g}_j \hat{h}_j &= \hat{Z}_{\mathbb{V}}, & \hat{v}_j &\equiv 0 \pmod{\hat{g}_j}, \\ \bar{q}_j(\mathbf{B}_j) &\equiv 0 \pmod{\hat{h}_j}, & \bar{q}_j(\mathbf{B}_j)\,\hat{\beta}_j &\equiv 1 \pmod{\hat{g}_j}, \\ N\bar{p}_j(\mathbf{B}_j) - \hat{b}_0 \hat{v}_j\, \bar{q}'_j(\mathbf{B}_j) &\equiv 0 \pmod{\hat{h}_j}. \end{aligned} \tag{16}$$

The last identity is Equation (7) in Section 2.5, multiplied by $\hat{b}_0 = \hat{a}$. Since $\hat{a}^{N-1} \equiv \hat{a}^{-1}$ on $\mathbb{V}$, this avoids the polynomial $\hat{R}$. As in Section 5, the compositions are used directly, so their quotients have degree $O(dMk)$. The following lemma shows correctness of the checks for level lookups.

Lemma 6.1. *Suppose that* $\mathsf{WellFormed}([\![C_j]\!]) = 1$ *and Equations* (15) *and* (16) *hold. Then* $\hat{v}_j(\eta) = \hat{t}_j(\hat{a}(\eta))$ *for all* $\eta \in \mathbb{V}$.

Proof. Fix $\eta \in \mathbb{V}$ and let $y = \hat{a}(\eta)$. Then $y \in \mathbb{L}$ by Equation (15). Since $\hat{g}_j \hat{h}_j = \hat{Z}_{\mathbb{V}}$, exactly one of $\hat{g}_j(\eta)$ and $\hat{h}_j(\eta)$ is zero. If $\hat{g}_j(\eta) = 0$, then the Bezout identity gives $\hat{q}_j(y) \neq 0$. Since $\hat{Z}_{\mathbb{L}}(y) = 0$, this gives $\hat{v}_j(\eta) = 0 = \hat{t}_j(y)$. If $\hat{h}_j(\eta) = 0$, then $\hat{q}_j(y) = 0$. Since $\hat{q}_j$ has distinct roots, $\hat{q}'_j(y) \neq 0$, and the last identity gives $\hat{v}_j(\eta) = Ny^{-1}\hat{p}_j(y)/\hat{q}'_j(y) = \hat{t}_j(y)$. This completes the proof. □

Prove Well-formedness of New Level. This check requires showing that the level polynomial $\hat{q}_\mu \neq 0$, $\hat{q}_\mu$ divides $Z_{\mathbb{L}}(X)$ and $\hat{q}'_\mu = \partial \hat{q}_\mu$. To show that a committed polynomial $\hat{G}$ divides $\hat{Z}_{\mathbb{L}}$, the prover sends $\hat{e}_i$

with $\hat{e}_i \equiv X^{d^i} \pmod{\hat{G}}$ and $\deg \hat{e}_i < \deg \hat{G}$, for $1 \le i < M$, and shows

$$\hat{e}_1 - X^d \equiv 0, \qquad \hat{e}_i - \hat{e}_{i-1}^d \equiv 0 \ (2 \le i < M), \qquad \hat{e}_{M-1}^d - 1 \equiv 0 \pmod{\hat{G}}. \tag{17}$$

These give $X^N \equiv 1 \pmod{\hat{G}}$, i.e., $\hat{G} \mid \hat{Z}_{\mathbb{L}}$. They also force $\hat{G} \neq 0$, since $X^N \neq 1$. We use this check for $\hat{G} = \hat{q}_*$ here, and for $\hat{G} = \hat{Z}_I$ below. For the derivative, the prover claims a single value $y_{q'_*}$ twice after z is sampled: as $\hat{q}'_*(z)$, and as $\partial\hat{q}_*(z)$. The first claim is proved by $\mathsf{UPCS.Eval}$, and the second by $\mathsf{DerivEval}([\![q_*]\!], z, y_{q'_*})$. The latter is the IOP of Section 6.3.2 for the claim $\partial\hat{q}_*(z) = y_{q'_*}$, given $[\![q_*]\!]$, and is based on the Taylor expansion of $\hat{q}_*$ at z. Together, the two claims test the local identity $\hat{q}'_* - \partial\hat{q}_* = 0$ at z. Both polynomials are fixed before z is sampled, so, except with negligible probability, $\hat{q}'_*$ is the derivative of $\hat{q}_*$. Neither check depends on the degree of $\hat{q}_*$.

a-identical Tables. Condition 4 is proved by the sub-protocol $\mathsf{AIdentical}(C, C^*, [\![a]\!])$ of Section 6.3.1 (Figure 8), which adapts [DGP+24, Section 6.2]. The prover commits to the vanishing polynomial $\hat{Z}_I$ of the distinct addresses and shows three things: $\hat{Z}_I \mid \hat{Z}_{\mathbb{L}}$; $\hat{Z}_I$ divides $\prod_{\eta\in\mathbb{V}}(X - \hat{a}(\eta))$; and $\hat{Z}_I(\hat{t} - \hat{t}^*) \equiv 0 \pmod{\hat{Z}_{\mathbb{L}}}$. The last identity is checked at z by querying the levels that C and C^* do not share. The quotient has degree less than k, and its values on the addresses are determined by $\mathbf{v} - \mathbf{v}'$. Hence the prover runs in time $O(k\,\mathrm{polylog}(N))$. The sub-protocol has a single round. Its oracles are sent in Round 1 of Figure 7, and it uses the same challenge z. It uses $\mathsf{ProdEval}$ of Section 6.3.2, which checks a claimed value of $\prod_{\eta\in\mathbb{V}}(z - \hat{a}(\eta))$ given $[\![a]\!]$.

Soundness Sketch. The argument follows Theorem 5.1. By soundness of the final IOPs, including $\mathsf{DerivEval}$ and $\mathsf{ProdEval}$, except with negligible probability every oracle has a δ-close decoding that agrees with its claimed evaluations. Every identity checked at z involves only Round 1 oracles and inputs, so it is fixed before z is sampled. Hence, by the Schwartz–Zippel lemma, all these identities hold as polynomial identities. Then:

- Equation (15) gives condition 2.
- Equation (17) and the two claims of $y_{q'_*}$ give $\mathsf{WellFormed}([\![C^*_\mu]\!]) = 1$, which is condition 1.
- Lemma 6.1 gives $\hat{v}_\lambda = \hat{t}_\lambda(\hat{a})$ on $\mathbb{V}$ for every $\lambda \in \Lambda$. Since $\hat{t}$ and $\hat{t}^*$ are the sums of the level polynomials of C and C^*, the two sum identities give condition 3.
- The checks of $\mathsf{AIdentical}$ give condition 4 (Section 6.3.1).

The prover's work on polynomials of degree $O(dMk)$ is $O(k\,\mathrm{polylog}(N))$ per level, so $O(k\,\mathrm{polylog}(N))$ in total. The work on the polynomials tied to $\hat{q}_*$ and the re-computation of $[\![C^*_\mu]\!]$ cost $O(s_\mu^{1+\epsilon})$ and occur only for the mutated level. This matches the accounting of $\mathcal{C}_{\mathsf{pp}}(\mu)$ and $\mathcal{C}_{\mathsf{lk}}$ in Section 2.4.

Round-by-Round Knowledge Soundness. In Section 7, we give a general blueprint for establishing round-by-round knowledge soundness of our lookup IOPs, and show that both the static and updatable constructions fit this blueprint. At a high level, we associate each partial transcript with a state that captures whether successful extraction is already guaranteed. We then analyze how this state evolves as the constituent subprotocols are executed, showing that problematic state transitions occur only with negligible probability. This yields the desired round-by-round knowledge soundness guarantee for the composed lookup protocols.

6.3 Auxiliary IOPs

We now describe the missing auxiliary IOPs $\mathsf{AIdentical}$, $\mathsf{DerivEval}$, and $\mathsf{ProdEval}$.

6.3.1 IOP for a-Identical Tables

We give the sub-protocol $\mathsf{AIdentical}$ used in Figure 7. It adapts the **a**-identical argument of [DGP+24, Section 6.2] to leveled table oracles (Definition 6.1). We use the notation of Section 6.

Protocol: $\mathsf{AIdentical}(C, C^*, [\![a]\!])$ for $\mathcal{R}_{\text{aid}}$

Both parties hold pp, and every new oracle is committed with PC. Let $\Lambda_\triangle$ be the set of non-empty level oracles that appear in exactly one of C and C^*.

1. **Round 1.** $\mathcal{P}$ sends $[\![Z_I]\!], [\![D]\!], [\![K]\!]$ and $[\![K/Z_I]\!]$. It also sends the oracles $\hat{e}_i \equiv X^{d^i} \pmod{\hat{Z}_I}$ $(1 \le i < M)$ of Equation (17) for $\hat{G} = \hat{Z}_I$, together with their quotients.
   $\mathcal{V}$ sends $z \xleftarrow{\$} \mathbb{F}$.
2. **Claims.** $\mathcal{P}$ sends the claimed evaluations at z of every Round 1 oracle. In particular, y_Z, y_D and y_K denote the claimed values of $[\![Z_I]\!]$, $[\![D]\!]$ and $[\![K]\!]$. The value y_K is also claimed as $\prod_{\eta\in\mathbb{V}}(z - \hat{a}(\eta))$. For each $\lambda \in \Lambda_\triangle$, $\mathcal{P}$ sends $y_{p_\lambda}, y_{q_\lambda}, y_{q'_\lambda}$, the claimed values of the components of λ at z.
3. **Evaluations.** $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{PC.Eval}$ for the claims on Round 1 oracles, $\mathsf{ProdEval}([\![a]\!], z, y_K)$ (Section 6.3.2), and $\mathsf{UPCS.Eval}$ for the component claims.
4. **Decision.** For a level λ, let $c_\lambda = \hat{Z}_{\mathbb{L}}(z)\, y_{p_\lambda}/y_{q_\lambda}$ if $y_{q_\lambda} \neq 0$, and $c_\lambda = Nz^{-1} y_{p_\lambda}/y_{q'_\lambda}$ otherwise (Definition 6.1). $\mathcal{V}$ accepts iff all evaluation IOPs accept, Equation (17) for $\hat{G} = \hat{Z}_I$ and $\hat{K} = (\hat{K}/\hat{Z}_I)\cdot\hat{Z}_I$ hold at $X = z$ with the claimed values, and
$$y_Z\Big(\sum_{\lambda\in C} c_\lambda - \sum_{\lambda\in C^*} c_\lambda\Big) = y_D\, \hat{Z}_{\mathbb{L}}(z). \tag{18}$$
   Here $\lambda \in C$ ranges over the levels of C in $\Lambda_\triangle$, and likewise for C^*.

Figure 8: IOP for **a**-identical tables, adapted from [DGP+24, Figure 3].

Relation. Let C and C^* be well-formed leveled table oracles, and let $\hat{t}, \hat{t}^*$ be the polynomials to which they simulate query access. Let $[\![a]\!]$ be an oracle for $\hat{a} \in \mathbb{F}_{<k}[X]$, and let $I = \{x \in [0, N) : \omega^x \in \hat{a}(\mathbb{V})\}$. The relation $\mathcal{R}_{\text{aid}}$ consists of instances $\mathbb{x} = (\mathbb{F}, \mathbb{L}, \mathbb{V}, d, M)$ and $\mathbb{y} = (C, C^*, [\![a]\!])$ such that

$$\hat{t}(\omega^x) = \hat{t}^*(\omega^x) \qquad \text{for every } x \in [0, N) \setminus I.$$

The witness consists of the polynomials behind the oracles. As in Definition 6.3, well-formedness of C and C^* is a promise. Write $\hat{Z}_I(X) = \prod_{x\in I}(X - \omega^x)$. The relation holds if and only if

$$\hat{Z}_I \cdot (\hat{t} - \hat{t}^*) \equiv 0 \pmod{\hat{Z}_{\mathbb{L}}}. \tag{19}$$

Honest prover. Let $\Delta_x = \hat{t}(\omega^x) - \hat{t}^*(\omega^x)$. By [DGP+24, Equation (1)], the quotient of Equation (19) is

$$\hat{D} = \frac{\hat{Z}_I(\hat{t} - \hat{t}^*)}{\hat{Z}_{\mathbb{L}}} = \sum_{x\in I} \frac{\Delta_x\, \hat{Z}'_I(\omega^x)}{\hat{Z}'_{\mathbb{L}}(\omega^x)}\, \hat{\mu}^I_x,$$

where $\{\hat{\mu}^I_x\}_{x\in I}$ is the Lagrange basis over $\{\omega^x\}_{x\in I}$. Hence $\deg \hat{D} < |I| \le k$. Since $\hat{Z}'_{\mathbb{L}}(\omega^x) = N\omega^{-x}$, the prover interpolates $\hat{D}$ in $O(k \log^2 k)$ field operations from the values Δ_x with $x \in I$. In Figure 7, these values are $v_i - v'_i$, so the tables are not accessed. The prover also uses $\hat{K}(X) = \prod_{\eta\in\mathbb{V}}(X - \hat{a}(\eta))$, which is divisible by $\hat{Z}_I$.

Soundness Sketch. As in Theorem 5.1, we decode every oracle, and by soundness of the evaluation IOPs all claims, including the one checked by $\mathsf{ProdEval}$, agree with the decodings. All identities checked at z involve only Round 1 oracles and inputs, so they are fixed before z is sampled. Hence, except with negligible probability, they hold as polynomial identities. Then:

- Equation (17) gives $\hat{Z}_I \neq 0$ and $\hat{Z}_I \mid \hat{Z}_{\mathbb{L}}$. Hence the roots of $\hat{Z}_I$ are distinct and lie in $\mathbb{L}$.
- The two claims of y_K give $\hat{K}(z) = \prod_{\eta\in\mathbb{V}}(z - \hat{a}(\eta))$. So the polynomial $\hat{K} - \prod_{\eta\in\mathbb{V}}(X - \hat{a}(\eta))$ vanishes at z, and hence is identically zero. Together with $\hat{Z}_I \mid \hat{K}$, all roots of $\hat{Z}_I$ lie in $\{\omega^x\}_{x\in I}$.
- Levels shared by C and C^* cancel in $\hat{t} - \hat{t}^*$. So Equation (18) says that $\hat{Z}_I(z)(\hat{t} - \hat{t}^*)(z) = \hat{D}(z)\hat{Z}_{\mathbb{L}}(z)$, which gives Equation (19).

- For $x \notin I$, the previous items give $\hat{Z}_I(\omega^x) \neq 0$, so $\hat{t}(\omega^x) = \hat{t}^*(\omega^x)$.

All committed polynomials have degree $O(dk)$, and the verifier queries $O(\ell)$ level components at z. The prover therefore runs in time $O(k \operatorname{polylog}(N))$, and the verifier in time $O(\operatorname{polylog}(N))$.

6.3.2 IOPs for Derivative and Product Relations

Most identities in our IOPs are *local*: each one is a polynomial expression in the evaluations of committed oracles at a single point. Two quantities used in Section 6 are not of this form:

- the derivative of a committed polynomial at a point, and
- the product, over $\mathbb{V}$, of a committed polynomial shifted by a point.

We capture them by two relations, which serve as final relations in the template of Section 7. As in Section 6, $\partial\hat{q}$ denotes the formal derivative of a univariate polynomial $\hat{q}$. Let $[\![q]\!]$ and $[\![a]\!]$ be oracles, committed with UPCS or PC, with decodings $\hat{q}^*$ and $\hat{a}^*$. The two relations are

$$\mathcal{R}_{\partial Ev} = \big\{([\![q]\!], z, v) : \ \partial\hat{q}^*(z) = v\big\},$$
$$\mathcal{R}_{\Pi Ev} = \big\{([\![a]\!], z, y) : \ \textstyle\prod_{\eta\in\mathbb{V}}(z - \hat{a}^*(\eta)) = y\big\},$$

where, in both relations, the oracle must also have a δ-close decoding. Both IOPs below follow the template with $r = 1$, and all their final relations are $\mathcal{R}_{Ev}$. We show PIOP soundness for each. By Lemma 7.1, both are then round-by-round sound, as the template requires of its final IOPs.

Derivative Evaluation. The IOP $\mathsf{DerivEval}([\![q]\!], z, v)$ proves $\partial\hat{q}(z) = v$ from $[\![q]\!]$, using the Taylor expansion of $\hat{q}$ at z.

1. The prover sends a value y_0, claimed to be $\hat{q}(z)$, and $[\![\tau]\!]$, committed with PC, where

   $$\hat{\tau} = (\hat{q} - y_0 - (X - z)v)/(X - z)^2.$$

   The verifier sends $x \xleftarrow{\$} \mathbb{F}$.

2. The prover claims the value $\hat{\tau}(x)$, and the parties run the evaluation IOPs of $[\![q]\!]$ at z and x, and of $[\![\tau]\!]$ at x.

3. The verifier accepts iff these evaluation IOPs accept and

   $$\hat{q}(x) - y_0 - (x - z)\, v = (x - z)^2\, \hat{\tau}(x). \tag{20}$$

For PIOP soundness, the only identity is $\varphi = \hat{q}^* - y_0 - (X - z)v - (X - z)^2\hat{\tau}^*$. If the evaluation claims are true, the verifier's check passes, and Lucky_1 does not occur, then $\varphi \equiv 0$. In that case $(X - z)^2$ divides $\hat{q}^* - y_0 - (X - z)v$, which holds if and only if $y_0 = \hat{q}^*(z)$ and $v = \partial\hat{q}^*(z)$. This check does not depend on the degree of $\hat{q}^*$. The prover runs in time linear in $\deg\hat{q}$.

Product Evaluation. The IOP $\mathsf{ProdEval}([\![a]\!], z, y)$ is the grand-product argument of [DGP+24, Equation (2)]. The verifier rejects if $y = 0$. When z is random, as in our applications, this affects completeness only with negligible probability. Let $\beta = y - 1$.

1. The prover sends $[\![u]\!]$, committed with PC. The polynomial $\hat{u} \in \mathbb{F}_{<k}[X]$ satisfies $\hat{u}(\nu^i) = \prod_{i'<i}(z - \hat{a}(\nu^{i'}))/(1 + \beta\hat{\mu}_0^{\mathbb{V}}(\nu^{i'}))$. The prover also sends the quotients of

   $$\hat{\mu}_0^{\mathbb{V}}\,(\hat{u} - 1) \equiv 0, \quad \hat{u}(\nu X)\,(1 + \beta\hat{\mu}_0^{\mathbb{V}}) - \hat{u}\cdot(z - \hat{a}) \equiv 0 \pmod{\hat{Z}_{\mathbb{V}}}. \tag{21}$$

   The verifier sends $x \xleftarrow{\$} \mathbb{F}$.

2. The prover claims the values of $\hat{u}$ at x and νx, of $\hat{a}$ at x, and of the quotients at x. The parties run the corresponding evaluation IOPs.

3. The verifier accepts iff these evaluation IOPs accept and Equation (21) holds at $X = x$, in quotient form.

For PIOP soundness: if the evaluation claims are true, the verifier's check passes, and Lucky_1 does not occur, then the identities of Equation (21) hold identically. Hence

$$y = 1 + \beta = \prod_{\eta \in \mathbb{V}} (z - \hat{a}^*(\eta)),$$

by [DGP+24, Lemma 4]. The prover runs in time $O(k \log k)$.

7 Blueprint for Round-by-Round Soundness

In this section, we show that IOPs in Sections 5 and 6 satisfy the stronger round-by-round knowledge soundness property. We prove the property for a more general blueprint, which fits both the IOPs in Figures 6 and 7. We describe the blueprint below.

The Blueprint Let $\mathcal{R} = (\mathbb{x}, \mathbb{y}, \mathbb{w})$ be an oracle relation whose witness $\mathbb{w}$ consists of the polynomials behind the input oracles $\mathbb{y}$. An IOP $(\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ follows the blueprint if it proceeds as follows.

1. *PIOP rounds.* Initially $\mathcal{P}$ and $\mathcal{V}$ execute an r-round public-coin PIOP where $\mathcal{P}$ sends polynomial oracles and field elements, while $\mathcal{V}$ in round $\rho \in [r]$ replies with $c_\rho \xleftarrow{\$} \mathbb{F}$. Each round ρ defines a set Φ_ρ of polynomial identities, all tested at c_ρ. We write $\mathsf{Checks} = \Phi_1 \cup \cdots \cup \Phi_r$ for the set of all these identities.

2. *Final claims.* At the end of the PIOP rounds, $\mathcal{P}$ and $\mathcal{V}$ define instances $\mathsf{inst}_1, \ldots, \mathsf{inst}_n$ of oracle relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$. Each instance involves oracles sent so far, including the input oracles and field elements. Typically, most of these are polynomial evaluation claims in $\mathcal{R}_{Ev}$, but they may be arbitrary in general. Examples are $\mathcal{R}_{\partial Ev}$ and $\mathcal{R}_{\Pi Ev}$ of Section 6.3.2.

3. *Terminal IOPs.* For each i, the parties run an IOP Π_i for $\mathcal{R}_i$ on inst_i. The IOPs run in parallel, padded with dummy messages so that all of them have the same number of rounds.

4. *Decision.* Let $\mathcal{V}^{\text{piop}}$ be the predicate that outputs 1 iff every identity $\varphi \in \Phi_\rho$ vanishes at c_ρ, for every ρ. The verifier evaluates each identity from the field elements in the final claims. It accepts iff $\mathcal{V}^{\text{piop}} = 1$ and every Π_i accepts.

Admissible Blueprint. We say that an IOP $(\mathcal{P}, \mathcal{V})$ satisfying the above blueprint is *admissible* for proximity parameter δ, if every oracle $[\![g]\!]$, whether an input or sent by the prover, has at most one δ-close polynomial g^*, which we call its *decoding*. The relations $\mathcal{R}_i$ are stated in terms of these decoded polynomials, and an instance can only belong to the relation if its oracles decode. Let $\mathbb{y}^*$ be the tuple of decodings of the input oracles, and let Valid be the event that every input oracle decodes and $(\mathbb{x}, \mathbb{y}, \mathbb{y}^*) \in \mathcal{R}$, i.e, Valid implies witness extraction. We note that this event depends only on the instance.

Written in terms of the decodings, each $\varphi \in \Phi_\rho$ is a *univariate* polynomial in X, of degree at most ${D_\rho}$[4]. A polynomial identity in Φ may be defined in terms oracles, field elements sent by the prover and verifier challenges sent across several rounds. It belongs to Φ_ρ once all of its ingredients are sent before c_ρ; we then say it is *active* in round ρ. Thus $\varphi \in \Phi_\rho$ is fixed before c_ρ is sampled. Let Lucky_ρ be the event that all oracles sent before c_ρ decode, and that some $\rho' \le \rho$ and some $\varphi \in \Phi_{\rho'}$ satisfy $\varphi \not\equiv 0$ and $\varphi(c_{\rho'}) = 0$. We require two properties to say that the blueprint is *admissible*, namely consistency and PIOP soundness, as outlined below.

[4]This makes sense even when multivariate oracles are involved, as long as φ replaces each variable by a univariate polynomial, e.g., $\bar{t}^*(\hat{b}_0^*(X), \ldots, \hat{b}_{m-1}^*(X))$

Consistency. The field elements that $\mathcal{V}^{\text{piop}}$ uses are the claimed values appearing in $\mathsf{inst}_1, \ldots, \mathsf{inst}_n$. Typically, these values are claimed evaluations of decoded polynomials at one of the challenges, and are enforced by defining a $\mathcal{R}_i$ to be $\mathcal{R}_{Ev}$ for the corresponding PCS. More generally, each such value χ is a claimed evaluation of a polynomial P_χ of degree at most D_ρ, determined by the decodings, such that membership of the corresponding instance in its relation forces $\chi = P_\chi(c_\rho)$. Each $\varphi \in \Phi_\rho$ is obtained from the verifier's check by substituting P_χ for χ. Hence, if $\mathsf{inst}_i \in \mathcal{R}_i$ for every i, then $\mathcal{V}^{\text{piop}} = 1$ if and only if $\varphi(c_\rho) = 0$ for every ρ and every $\varphi \in \Phi_\rho$. See subsequent paragraph for further discussion on consistency.

PIOP Soundness. If $\mathsf{inst}_i \in \mathcal{R}_i$ for every i and every identity in Checks is identically zero, then Valid holds.

For an evaluation claim on $[\![g]\!]$ we have $P_\chi = g^*$, for a claim in $\mathcal{R}_{\partial Ev}$ we have $P_\chi = \partial\hat{q}^*$, and for one in $\mathcal{R}_{\Pi Ev}$ we have $P_\chi = \prod_{\eta\in\mathbb{V}}(X - \hat{a}^*(\eta))$. This is where the low degree of the identities comes from, and hence the applicability of the Schwartz-Zippel lemma. The second property is the soundness of the underlying PIOP: polynomials that satisfy all the identities of the protocol yield a witness. The first ties the verifier's checks to it. Finally, we have the following claim on admissible IOPs.

Lemma 7.1 (Round by Round Soundness for PIOPs)**.** *Suppose that an IOP $(\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R}$ is admissible, and that each terminal IOP Π_i is round-by-round sound for $\mathcal{R}_i$ (Definition 3.5) with error at most κ_{fin}. Then $(\mathcal{P}, \mathcal{V})$ is round-by-round knowledge sound with knowledge error*

$$\kappa^{\text{rbr}} \le \max\Big(\max_{\rho} \frac{|\Phi_\rho|\, D_\rho}{|\mathbb{F}|},\ n\,\kappa_{\text{fin}}\Big),$$

and the round-by-round extractor outputs $\mathbb{y}^*$.

Proof. Let State_i be the state function of Π_i, and let tr_i be its part of the transcript. By monotonicity (Definition 3.5), State_i is 1 throughout whenever $\mathsf{inst}_i \in \mathcal{R}_i$, and 0 until the first verifier message of Π_i otherwise. We define State as a function of the transcript up to its last verifier message, so prover messages never change it:

- before c_1, $\mathsf{State} = 0$;
- after c_ρ, for $\rho \le r$, and until the first verifier message of the terminal IOPs, $\mathsf{State} = \mathsf{Valid} \vee \mathsf{Lucky}_\rho$. In particular, the final claims do not change it;
- after each verifier message of the terminal IOPs,

$$\mathsf{State} = \big[\mathcal{V}^{\text{piop}} = 1\big] \wedge \forall i :\ \mathsf{State}_i(\mathsf{tr}_i) = 1.$$

  Here $\mathcal{V}^{\text{piop}}$ is determined by the final claims, so it does not change in these rounds.

We check the required properties.

- *Empty and full transcripts.* The empty transcript has state 0. For a full transcript, the state is the verifier's decision bit: it is 0 exactly when $\mathcal{V}^{\text{piop}} = 0$ or some $\mathsf{inst}_i \notin \mathcal{R}_i$ ends with $\mathsf{State}_i = 0$, in which case Π_i rejects.
- *Prover moves.* Immediate, since State does not depend on the prover messages that follow the last verifier message.
- *Transition at c_ρ.* If Valid holds, the extractor succeeds. Otherwise, a $0 \to 1$ transition means that some nonzero $\varphi \in \Phi_\rho$ vanishes at c_ρ. Each such φ is fixed before c_ρ is sampled, so this has probability at most $|\Phi_\rho| D_\rho / |\mathbb{F}|$.
- *Transition at the first verifier message of the terminal IOPs.* Here the state changes form, from $\mathsf{Valid} \vee \mathsf{Lucky}_r$ to the decision expression. If Valid holds, the extractor succeeds. Otherwise the state before the message is 0, so Lucky_r does not occur. Suppose the new state is 1, i.e., $\mathcal{V}^{\text{piop}} = 1$ and $\mathsf{State}_i = 1$ for every i. Assume for contradiction that $\mathsf{inst}_i \in R_i$ for every i. By consistency, $\mathcal{V}^{\text{piop}} = 1$ then says precisely that $\varphi(c_\rho) = 0$ for every ρ and every $\varphi \in \Phi_\rho$. Since Lucky_r does not occur, each such φ is identically zero, and PIOP soundness gives Valid, a contradiction. Hence some inst_i is false, and its State_i moved from 0 to 1 at this message. The verifier's message is a tuple of messages to the Π_i, so by round-by-round soundness and a union bound, this has probability at most $n\,\kappa_{\text{fin}}$.

- *Transition at a later verifier message of the terminal IOPs.* Now the state has the same form before and after the message, and $\mathcal{V}^{\text{piop}}$ is fixed by the final claims. So the state changes only through the component state functions. Before the message the state is 0, and $\mathcal{V}^{\text{piop}} = 1$, since otherwise the state stays 0. Hence $\mathsf{State}_i = 0$ for some i, and after the message $\mathsf{State}_i = 1$ for every i. Any such inst_i is false, because the state of a true instance is 1 on every transcript. So some false instance moved from 0 to 1 at this message, which by round-by-round soundness and a union bound has probability at most $n\,\kappa_{\text{fin}}$.

This concludes the proof. We note that no extraction from the terminal IOPs is needed. □

Instantiations. The evaluation IOPs of MPCS and UPCS are round-by-round knowledge sound by Lemma 4.2, and hence round-by-round sound. For PC, we assume round-by-round soundness. In the IOPs below, $n = \text{polylog}(N)$, $D_\rho = \text{poly}(N)$, and we assume $D_\rho/|\mathbb{F}| = \text{negl}(\lambda)$.

- *Static lookup (Figure 6).* Here $r = 1$ and $c_1 = x$, and all final relations are $\mathcal{R}_{Ev}$. The identities are $\Phi_1 = \{P_0, \ldots, P_m\}$ from the proof of Theorem 5.1, of degree less than $dm\ell + k$. PIOP soundness is Steps 2-4 of that proof.
- *Derivative and product (Section 6.3.2).* DerivEval and ProdEval follow the template with $r = 1$, and all their final relations are $\mathcal{R}_{Ev}$. PIOP soundness is shown in Section 6.3.2. By Lemma 7.1, they are round-by-round knowledge sound, and hence round-by-round sound for $\mathcal{R}_{\partial Ev}$ and $\mathcal{R}_{\Pi Ev}$.
- *Update (Figure 7).* Here too $r = 1$, with $c_1 = z$. The final relations are $\mathcal{R}_{Ev}$, together with $\mathcal{R}_{\partial Ev}$ for $[\![q_*]\!]$ and $\mathcal{R}_{\Pi Ev}$ for $[\![a]\!]$. Φ_1 contains the identities of Equations (15) to (17), the two sum identities, and the checks of AIdentical. It also contains two identities that compare final claims:
  - $\hat{q}'_* - \partial\hat{q}_*$, whose value at z is the difference between the claims for $[\![q'_*]\!]$ and for $\mathcal{R}_{\partial Ev}$;
  - $\hat{K} - \prod_{\eta\in\mathbb{V}}(X - \hat{a}(\eta))$, whose value at z is the difference between the claims for $[\![K]\!]$ and for $\mathcal{R}_{\Pi Ev}$.

  PIOP soundness follows from the arguments in Section 6.

Statement on Generative AI Usage

All ideas, constructions, and proofs were conceived by the authors, without any use of generative AI. We used Claude Opus 4.8 to help with writing and generating protocol boxes based on the textual descriptions of the protocols. In the case of Section 7, a substantial portion of the text was generated by Claude Opus 4.8, which was given the blueprint for the overall idea, followed by substantial interaction to improve the correctness and readability of the AI outputs. The generated content was rigorously reviewed and significantly revised by the authors.

References

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 380–413, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-solomon proximity testing with super-fast verification. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 214–243, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *33rd FOCS*, pages 14–23, Pittsburgh, PA, USA, October 24–27, 1992. IEEE Computer Society Press.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd FOCS*, pages 2–13, Pittsburgh, PA, USA, October 24–27, 1992. IEEE Computer Society Press.

[AST24] Arasu Arun, Srinath T. V. Setty, and Justin Thaler. Jolt: SNARKs for virtual machines via lookups. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 3–33, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 326–349, 2012.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, Berkeley, CA, USA, May 18–21, 2014. IEEE Computer Society Press.

[BCG+18] Jonathan Bootle, Andrea Cerulli, Jens Groth, Sune K. Jakobsen, and Mary Maller. Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 595–626, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Cham, Switzerland.

[BCHO22] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 427–457, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333, Tokyo, Japan, March 3–6, 2013. Springer Berlin Heidelberg, Germany.

[BCJ+24] Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. zkLogin: Privacy-preserving blockchain authentication with existing credentials. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3182–3196, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 781–796, San Diego, CA, USA, August 20–22, 2014. USENIX Association.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *23rd ACM STOC*, pages 21–31, New Orleans, LA, USA, May 6–8, 1991. ACM Press.

[BFR+13] Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. Verifying computations with state (extended version). Cryptology ePrint Archive, Report 2013/356, 2013.

[BS80] Jon L. Bentley and James B. Saxe. Decomposable searching problems i: Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 499–530, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

[CCC+25] Matteo Campanelli, Mario Carrillo, Ignacio Cascudo, Dario Fiore, Danilo Francati, and Rosario Gennaro. On the power of polynomial preprocessing: Proving computations in sublinear time, and more. Cryptology ePrint Archive, Report 2025/238, 2025.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090, Phoenix, AZ, USA, June 23–26, 2019. ACM Press.

[CFH+22] Matteo Campanelli, Dario Fiore, Semin Han, Jihye Kim, Dimitris Kolonelos, and Hyunok Oh. Succinct zero-knowledge batch proofs for set accumulators. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 455–469, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

[CGG+24] Arka Rai Choudhuri, Sanjam Garg, Aarushi Goel, Sruthi Sekar, and Rohit Sinha. SublonK: Sublinear prover PlonK. *PoPETs*, 2024(3):314–335, July 2024.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

[DGP+24] Moumita Dutta, Chaya Ganesh, Sikhar Patranabis, Shubh Prakash, and Nitin Singh. Batching-efficient RAM using updatable lookup arguments. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 4077–4091, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. Garuda and pari: Faster and smaller snarks via equifficient polynomial commitments. *IACR Cryptol. ePrint Arch.*, page 1245, 2024.

[DXNT24] Zijing Di, Lucas Xia, Wilson D. Nguyen, and Nirvan Tyagi. MuxProofs: Succinct arguments for machine computation from vector lookups. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part V*, volume 15488 of *LNCS*, pages 236–265, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[EFG22] Liam Eagen, Dario Fiore, and Ariel Gabizon. cq: Cached quotients for fast lookups. Cryptology ePrint Archive, Report 2022/1763, 2022.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *32nd FOCS*, pages 2–12, San Juan, Puerto Rico, October 1–4, 1991. IEEE Computer Society Press.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany.

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *ISTCS 1995*, pages 190–198, Tel Aviv, Israel, January 4–6, 1995. IEEE Computer Society.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany.

[GLS+23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[GPPS24] Chaya Ganesh, Sikhar Patranabis, Shubh Prakash, and Nitin Singh. GAPP: Generic aggregation of polynomial protocols. Cryptology ePrint Archive, Report 2024/1685, 2024.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

[GRR18] Tom Gur, Govind Ramnarayan, and Ron D. Rothblum. Relaxed locally correctable codes. In Anna R. Karlin, editor, *ITCS 2018*, volume 94, pages 27:1–27:11, Cambridge, MA, USA, January 11–14, 2018. LIPIcs.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, 1992.

[GW20] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive, Report 2020/315, 2020.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019.

[Hab22] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Report 2022/1530, 2022.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732, 1992.

[KU08] Kiran S. Kedlaya and Christopher Umans. Fast modular composition in any characteristic. In *49th FOCS*, pages 146–155, Philadelphia, PA, USA, October 25–28, 2008. IEEE Computer Society Press.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

[Lab17] Protocol Labs. Filecoin: A decentralized storage network. `https://filecoin.io/filecoin.pdf`, 2017.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189, Taormina, Sicily, Italy, March 19–21, 2012. Springer Berlin Heidelberg, Germany.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60, Bengalore, India, December 1–5, 2013. Springer Berlin Heidelberg, Germany.

[LKKO20] Seunghwa Lee, Hankyung Ko, Jihye Kim, and Hyunok Oh. vCNN: Verifiable convolutional neural network. Cryptology ePrint Archive, Report 2020/584, 2020.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128, London, UK, November 11–15, 2019. ACM Press.

[Mic94] Silvio Micali. Cs proofs. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE, 1994.

[OWWB20] Alex Ozdemir, Riad S. Wahby, Barry Whitehat, and Dan Boneh. Scaling verifiable computation using efficient set accumulators. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2075–2092. USENIX Association, August 12–14, 2020.

[PCP26] Christodoulos Pappas, Zhuo Cai, and Dimitrios Papadopoulos. Rogue: Updatable matrix lookup arguments and applications to verifiable databases. Cryptology ePrint Archive, Paper 2026/1890, 2026.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252, Berkeley, CA, USA, May 19–22, 2013. IEEE Computer Society Press.

[rol21] An incomplete guide to rollups. `https://vitalik.eth.limo/general/2021/01/05/rollup.html`, 2021.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62, Cambridge, MA, USA, June 18–21, 2016. ACM Press.

[SDP22] Nitin Singh, Pankaj Dayama, and Vinayaka Pandit. Zero knowledge proofs towards verifiable decentralized AI pipelines. In Ittay Eyal and Juan A. Garay, editors, *FC 2022*, volume 13411 of *LNCS*, pages 248–275, Grenada, May 2–6, 2022. Springer, Cham, Switzerland.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275, 2020.

[SLZ24] Haochen Sun, Jason Li, and Hongyang Zhang. zkLLM: Zero knowledge proofs for large language models. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 4405–4419, Salt Lake City, UT, USA, October 14–18, 2024. ACM Press.

[ST25] Srinath Setty and Justin Thaler. Twist and shout: Faster memory checking arguments via one-hot addressing and increments. Cryptology ePrint Archive, Report 2025/105, 2025.

[STW24] Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Unlocking the lookup singularity with Lasso. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 180–209, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[Uma08] Christopher Umans. Fast polynomial factorization and modular composition in small characteristic. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 481–490, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.

[XZZ+19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 733–764, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[ZBK+22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. Caulk: Lookup arguments in sublinear time. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 3121–3134, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 138–169, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[ZFZS20] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2039–2053, Virtual Event, USA, November 9–13, 2020. ACM Press.

[ZGK+25] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. Baloo: Algebraic lookup arguments. *CiC*, 2(2):1, 2025.