Relaxed Modular PCS from Arbitrary PCS and Applications to SNARKs for Integers

Alireza Shirzad[1][0009−0009−7161−4978], Sriram Sridhar[2][0000−0002−2489−6935], Dimitrios Papadopoulos[3][0000−0003−2621−3015] and Charalampos Papamanthou[4,5][0000−0002−1905−9403]

[1] University of Pennsylvania
alrshir@seas.upenn.edu
[2] University of California, Berkeley
srirams@berkeley.edu
[3] Hong Kong University of Science and Technology
dipapado@cse.ust.hk
[4] Lagrange Labs*
[5] Yale University
charalampos.papamanthou@yale.edu

Abstract. *Modular Polynomial Commitment Schemes (Mod-PCS)* extend standard PCSs by enabling provable evaluation of integer polynomials modulo a random modulus, providing a natural foundation for SNARKs that operate directly over large integers without emulating arithmetic in finite fields. Only two Mod-PCS constructions are known. The first (Campanelli and Hall-Andersen, IACR ePrint 2024) is not post-quantum secure due to its reliance on groups of unknown order. The second (Garreta et al., CRYPTO 2025) introduces the weaker notion of *relaxed* Mod-PCS, but is not fully succinct: committing to a multilinear polynomial with N terms and B-bit coefficients requires $O(\sqrt{N}B)$ proof size and verification time.

We present the first black-box information theoretic transformation that builds relaxed Mod-PCS from any standard PCS, enabling new constructions. Instantiating our transformation with a tensor-code PCS yields the first relaxed Mod-PCS with $O(\log(N + B))$ proof size and verifier time, which is transparent and plausibly post-quantum secure. Using this scheme within the framework of Garreta et al., we obtain the first fully succinct SNARK for the Customizable Constraint System over $\mathbb{Z}_B$, achieving $O(BN \log(BN))$ prover time and $O(\log(N + B))$ verifier time and proof size.

Our approach relies on a novel modulus-switching technique for integer polynomials and a new batched integer commitment scheme from any PCS. We further introduce improved arguments for integer addition and multiplication, a convolution check, and general Diophantine relations over committed integers which can be of independent interest.

Keywords: SNARK, Integer, Polynomial commitment scheme.

* Work partially done while the first two authors were at Lagrange Labs.

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) are cryptographic proofs that enable an untrusted prover $\mathcal{P}$ to efficiently convince a computationally weak verifier $\mathcal{V}$ of claims of the form "Given a program P and public input x, I know a private input w such that $P(\mathsf{x}, \mathsf{w}) = 1$".

Arithmetization. Most efficient SNARKs today are designed for NP-complete languages that operate over *finite-field* arithmetic. These include R1CS [58,22], CCS [59], GR1CS [28], Plonkish systems [32,21], AIR [4], QAP, QSP [36], SSP [27], SAP [40], etc. Such languages are amenable to succinct and probabilistic checking and admit synthesis from higher-level computation models, such as circuit satisfiability and RAM programs.

Performance Blowups. While these languages are NP-complete and theoretically as expressive as general computation models, the compilation process often incurs substantial overhead when the underlying computation is not naturally expressed over finite fields. This overhead can range from moderate slowdowns—on the order of $2^5\times$ for basic 32-bit CPU arithmetic—to extreme blowups exceeding $2^{12}\times$ for cryptographic computations such as RSA [18,35]. These costs are particularly pronounced for computations that inherently rely on non-finite-field arithmetic, including (i) integer computations, such as fixed–point arithmetic in real-time systems, digital signal processing, and machine learning; (ii) computations over integer rings, such as k-bit CPU arithmetic over $\mathbb{Z}/2^k\mathbb{Z}$, RSA operations over $\mathbb{Z}/N\mathbb{Z}$, and fully homomorphic computations; and (iii) computations over rational numbers or over fields that are non-native to the proving system.

Soundness Pitfalls. Beyond performance overhead, the arithmetization process is complex and error-prone, particularly for non-native computations such as fixed-width integer arithmetic, comparisons, and bit-level operations. These operations must be simulated using auxiliary constraints (e.g., range checks, bit decompositions, and carry relations), and subtle omissions can lead to underconstrained circuits which are the dominant vulnerability class in deployed ZK systems [20,55]. In particular, missing range constraints or improperly enforced fixed-width arithmetic can introduce modular wraparound or aliasing behaviors that compromise soundness [5,53].

Ideally, we would like SNARKs that can operate directly over integers or rationals, without requiring expensive arithmetic *emulation* within finite fields.

Challenges of building SNARKs over Integers. Building SNARKs over the integers has been significantly less explored than their finite-field counterparts, largely due to the difficulty of constructing suitable underlying building blocks. These challenges include (a) the absence of multiplicative inverses in the integers, which complicates witness extraction; (b) the unbounded magnitude of integers, which hinders efficiency and succinctness; and (c) the scarcity of commitment schemes for integers. Most existing approaches rely on groups of unknown order (GUO), which are neither post-quantum secure nor practical at the moment.

Blueprint: Mod-PIOP + Mod-PC → SNARK. Following the widely used "PIOP+PC → SNARK" framework for constructing SNARKs over finite fields [32,22], Campanelli et al. [18] recently proposed a blueprint for constructing SNARKs over integer arithmetic by combining two novel primitives: *modular polynomial interactive oracle proofs* (Mod-PIOPs) and *modular polynomial commitments* (mod-PCs). Mod-PIOPs extend the standard interactive oracle proof model by allowing polynomial oracles to answer queries modulo a randomly chosen modulus. Mod-PCs extend standard polynomial commitment schemes by enabling commitments to integer polynomials that can later be opened modulo a random modulus.

On the Difficulty of Constructing Mod-PCs. Constructing Mod-PIOPs is relatively straightforward, as existing finite-field PIOPs can often be adapted to the modular setting with only modest modifications [18,35]. In contrast, building efficient mod-PCs, the core cryptographic component, remains a major technical challenge. Evidence for this gap is striking: while the literature contains numerous standard polynomial commitment schemes [44,11,3,9,29], there is, to the best of our knowledge, only a single mod-PC construction, due to Campanelli et al. [18]. Their scheme builds on the approach of Block et al. [7], extending DARK [13] and operating over groups of unknown order (GUO). As a result, it is neither post-quantum secure nor practical in concrete settings. In particular, generating a single proof for moderately sized instances can take on the order of tens of hours [52], and proof sizes are typically several megabytes. This undermines the primary motivation for using SNARKs over integers, namely, enabling *concretely* fast provers by reducing arithmetization overhead.

Relaxed Mod-PC. Subsequently, Garreta et al. [35] introduced a relaxed notion of mod-PC. This relaxation occurs along two dimensions. First, they fix the maximum bit length supported by the mod-PC (and, consequently, by the resulting SNARK); we denote this bound by $B \in \mathbb{N}$ throughout the paper, and write $\mathbb{Z}_B$ for the ring of integers of bit length at most B. Second, they relax the extractability guarantee to allow rational extractability. Concretely, in honest executions the primitive provides the same API as a mod-PC; however, against a malicious prover, the knowledge extractor is only guaranteed to recover a polynomial with rational—rather than integer—coefficients. We refer to this weaker notion as a relaxed mod-PC.

Garreta et al. further show that this relaxation remains useful in practice: the gap between rational and integer extractability can be compensated at the PIOP level using a lookup argument that enforces the integrality of the coefficients. While this approach enjoys several attractive properties—being transparent, hash-based, post-quantum secure, and supporting a fast prover—it suffers from two main drawbacks in terms of succinctness: 1. The reliance on a Brakedown-style argument [38] leads to large proof sizes and high verification costs. In particular, for multilinear polynomials over a boolean hypercube of size $N = 2^\mu$, both the proof size and the verification time scale as $O(\sqrt{N})$ asymptotically, and in practice exceed one megabyte in proof size and nearly one second in verification time. 2. Both the proof size and the verification time grow linearly with

B, the maximum bit length of the integers. This contrasts with the mod-PC of Campanelli et al. [18], which is *fully succinct*, i.e., succinct both in the size of the polynomial and in the magnitude of its coefficients. This naturally leads to the following open question:

RQ1: Can we build a transparent, post-quantum secure, and fully succinct relaxed mod-PC?

Lack of general frameworks for relaxed mod-PCs. The landscape of standard polynomial commitment schemes is remarkably rich, spanning a wide range of cryptographic regimes, including elliptic-curve–based, lattice-based, and hash-based constructions. Each regime offers distinct trade-offs: elliptic-curve–based PCs typically achieve very small proof sizes [44,10], lattice-based constructions provide the only post-quantum options with useful homomorphic properties [54,62,24], and hash-based schemes offer transparency and strong post-quantum security. Even within the hash-based setting, different code choices lead to different performance profiles—for example, Reed–Solomon–based constructions often yield smaller proofs and extremely fast verification [1], while RAA-code–based approaches typically enable faster provers [9].

In contrast, the mod-PC landscape lacks this diversity. At present, there is a single integer-extractable construction in the groups-of-unknown-order setting [18] and a single rationally extractable construction in the hash-based (JEA code) regime [52]. Achieving a comparable level of diversity would require designing a new mod-PC separately for each cryptographic regime, a task that demands substantial research effort. This raises a natural research question:

RQ2: Can we design a generic framework to bootstrap any polynomial commitment scheme into a mod-PC?

1.1 Our Contributions

We organize our contributions into main and additional contributions. The main contributions, presented in Section 1.1.1, focus on designing a family of SNARKs over integer arithmetic by addressing Research question RQ2 (and automatically resolving RQ1 by instantiating the framework with an appropriate PCS). These results are enabled by a collection of technical innovations, each of which is an additional contribution and of independent interest and is detailed in Section 1.1.2.

1.1.1 Main Contribution

In this work, we introduce a new family of SNARKs for integer arithmetic by proposing a general framework for constructing relaxed mod-PCs. Specifically, we present a bootstrapping compiler, $\mathbb{Q}\bar{a}ren$[6], which is the first generic transformation that turns any standard polynomial commitment scheme over $\mathbb{F}_p$ into a relaxed mod-PC. The transformation places *no structural requirement* on the

[6] A mythical and righteous Persian gladiator, son of Kāveh the Blacksmith.

underlying PCS: it works with any multilinear PCS over a field of characteristic greater than B (the maximum integer bit-length), requiring no roots of unity, smooth subgroups, or FFT-friendliness. In particular, it is compatible with FFT-unfriendly and large-characteristic extension fields; the sole exclusion is small-characteristic fields (such as binary fields), where the carry-based integer arithmetic would no longer be faithful. See Section 2.2 and Section 5 for informal and formal descriptions of the compiler, respectively.

Theorem 1 (informal). *Let $p > B$ be a prime, and* PCS *a multilinear polynomial commitment scheme over $\mathbb{F}_p$. Then there exists a relaxed mod-PC that uses* PCS *as a black box and inherits its transparency and security assumptions.*

Corollary 1. *Let $p > B$ be a prime, and let* PCS *be a multilinear PCS over $\mathbb{F}_p$. Then there exists a SNARK for the Customizable Constraint System (CCS) language over $\mathbb{Z}_B$ (as defined in [35]), where $B \in \mathbb{N}$ denotes the maximum bit length of the witness vector. Moreover, the resulting SNARK inherits the transparency and security assumptions of the underlying* PCS.

1.1.2 Additional Contributions

To achieve our main contribution—constructing a relaxed mod-PC—we reduce the task of *evaluating an integer polynomial with N coefficients of bit length at most B* to *committing to a constant number of large integers of size $\tilde{O}(NB)$ and proving simpler relations over them*, such as integer addition, multiplication, and Diophantine satisfiability. We now outline these contributions:

Batched Integer Commitment Scheme (ICS). We construct a commitment scheme for bounded-size integers by interpreting each integer as a multilinear polynomial and committing to it using a multilinear PCS over $\mathbb{F}_p$. We extend the construction to support batch commitment to t bounded integers. The resulting ICS inherits the commitment time (for an MLE of size tB), transparency and security assumptions of the underlying PCS. See Section 2.2 and Section 4.1 for informal and formal descriptions of the construction, respectively.

Then, we develop a suite of special-purpose sumcheck-based SNARKs for this family of integer commitment schemes. Leveraging the structure of our ICS, we design dedicated PIOPs over $\mathbb{F}_p$ and compile them into full-fledged arguments:

Integer Addition and Multiplication. We develop efficient succinct arguments (Sections 4.2.1 and 4.2.4) for proving addition and multiplication relations over committed *large* integers (of bit-length at least 2^{10}). Concretely, given commitments cm_a, cm_b, and cm_c to integers a, b, and c, the protocol proves that either $a+b=c$ or $a \cdot b = c$. Further, we extend these protocols to batched settings where we commit to $3t$ integers $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{Z}_B^t$ and prove either $\boldsymbol{a}[i] + \boldsymbol{b}[i] = \boldsymbol{c}[i]$ or $\boldsymbol{a}[i] \times \boldsymbol{b}[i] = \boldsymbol{c}[i]$ for $i = 0, \ldots, t-1$. The addition protocol has $O(tB)$ prover time and the multiplication protocol has $O(tB \log B)$ prover time; both have $O(\log(tB))$ verifier time and proof size. See Section 2.3 and Section 4.2.4 for informal and formal descriptions of the multiplication protocol, respectively. Since the addition protocol is not directly used as a building block in the main construction, we only provide a formal description in Section 4.2.1.

Convolution Check. Our integer multiplication argument reduces, at its core, to checking that a committed vector is the convolution of two others. Since convolution is exactly univariate polynomial multiplication, we verify it with a single random evaluation point via the univariate Schwartz–Zippel lemma, even though our vectors are committed in multilinear form. To bridge this mismatch, we introduce an information-theoretic technique that transforms *any* multilinear PCS into a univariate PCS. This approach can be viewed as the conceptual inverse of the well-studied paradigm that compiles multilinear PCS from univariate PCS [63,34,29,8], and may be of independent interest. See Section 2.3.1 and Section 4.2.3 for informal and formal descriptions of the protocol, respectively.

Arguments for Diophantine Equations. A more versatile building block developed in this work is a succinct argument for proving Diophantine relations [48] over committed large (but bounded) integers. More specifically, given a batch commitment $\mathsf{cm}_{\boldsymbol{a}}$ to integers $\boldsymbol{a} = [a_1, \ldots, a_t]$, the protocol proves that these integers satisfy a relation of the form $G(a_1, \ldots, a_t) = 0$, where $G \in \mathbb{Z}[X_1, \ldots, X_t]$ represents an arithmetic circuit over integers of size $s \in \mathbb{N}$. Our protocol has $O(t\ B \log B)$ prover time and $O(t\lambda + \log Bt)$ proof size and verifier time. See Section 2.2.2 and Section 4.2.6 for informal and formal descriptions of the protocol, respectively.

Definitional contributions. We introduce a few new definitions that help us with the presentation and security analysis of our protocols.

- *Generalized Polynomial Interactive Oracle Reduction* (Gen-PIOR): a unifying definition that captures all of our idealized protocols. It extends the standard notion of PIOPs to PIORs—analogous to how IORs generalize IOPs in [14]—and further generalizes both standard and modular oracle protocols, where oracle queries are answered modulo a random modulus. See Section 3.2.1 for a formal definition. We further extend the definition of round-by-round soundness (originally introduced in [19]) and knowledge soundness to the Gen-PIOR setting in Definitions 6 and 7.
- *Generalized Polynomial Commitment Scheme* (Gen-PCS): a generalization of the standard notion of polynomial commitment schemes that captures both standard and modular commitment schemes as special cases. See Section 3.5 for a formal definition.

1.2 Related Works

1.2.1 SNARKs over the integers

A growing line of work studies proof systems for computations over the integers and more general rings. Ganesh et al. [33] introduced Rinocchio, the first SNARK for arithmetic circuits over rings; however, the scheme relies on new "linear-only" assumptions, requires a circuit-specific trusted setup, and is not succinct in the bit-length of ring elements. More recently, [61,42] proposed SNARKs for statements over Galois rings via polynomial commitment schemes defined over such rings. While these constructions are hash-based, post-quantum secure, and offer fast provers, they incur $\sqrt{N}$-size proofs and verification time, making them

Scheme	Properties				Prover	Verifier	Proof
	PQ	IE	Tr	Assumpt.			
Campanelli et al. [18]	✗	✓	✓	GUO	$O(BN \log N \operatorname{poly}(\lambda))$	$O(\operatorname{poly}(\lambda)\mu \log^2 B)$	$O(\operatorname{poly}(\lambda) \log^2 B)$
Zip [35]	✓	✗	✓	RO	$N(B+\lambda) + \lambda\sqrt{N}\, B + \lambda^2\sqrt{N} \log N$	$\sqrt{N}B + \lambda^2\sqrt{N} + \lambda^2 \log N$	$\sqrt{N}(B+\lambda) + \lambda^2 \log N$
$\mathbb{Q}\bar{a}ren_{\mathsf{TS}}$ (Instantiated with [11])	✓	✗	✓	RO	$BN \log(BN)$	$\log(B+N)$	$\log(B+N)$
$\mathbb{Q}\bar{a}ren_{\mathsf{MC}}$ (Instantiated with [29])	✗	✗	✗	Q-DLOG	$BN \log(BN)$	$\log(B+N)$	$\log(B+N)$

PQ: Post-Quantum Security, IE: Integer Extractability, Tr: Transparent
N: The size of the multilinear polynomial
B: The maximum bit-length of the coefficients of the polynomial

Table 1. Comparison of *all* relaxed mod-PC schemes

impractical for many applications. The closest works to ours, as discussed earlier, are those of Campanelli et al. [18] and Garreta et al. [35]; we provide a detailed comparison in Table 1.

1.2.2 Arguments for large Integer Addition and Multiplication

Proving arithmetic relations over large committed integers has received substantial attention in the literature, with constructions based on groups of unknown order (GUO) [25,26,48] as well as lattice-based techniques [47,50,46]. Such arguments serve as fundamental building blocks for a wide range of cryptographic applications, including range proofs, order relations, set non-membership proofs, and zero-knowledge lists, which in turn underpin systems such as anonymous credentials [16], privacy-preserving certificate transparency [30], and secure electronic voting [39]. We compare our protocols with the state of the art in Table 2.

1.2.3 Arguments for Diophantine equations over the integers

Proving Diophantine equations over committed integers was first studied in early works such as [31,26,48], which relied on groups of unknown order (GUOs). More recently, Towa and Vergnaud [60] gave the first succinct argument for satisfiability of Diophantine equations with proof size logarithmic in the size of the polynomial. The work most closely related to ours is the PoKEMath protocol of [15], where the proof size, verifier time, and communication grow logarithmically in the magnitude of the solution, while the verifier time scales linearly in the size of the equation and the proof size scales linearly in the number of variables. However, PoKEMath also relies on GUOs, limiting practicality and post-quantum security. Our protocol, Masdak, supports only bounded-size integers, which restricts generality, but can be viewed as a practical version of PoKEMath with similar asymptotic efficiency, while being instantiable from standard polynomial commitment schemes and therefore amenable to post-quantum.

Scheme	Properties				Int Addition			Int Multiplication		
	PQ	Tr	ZK	Assumpt.	$T(\mathcal{P})$	$T(\mathcal{V})$	$\lvert\pi\rvert$	$T(\mathcal{P})$	$T(\mathcal{V})$	$\lvert\pi\rvert$
Libert et al. [47]	✓	✓	✓	SIVP	tB	tB	tB	$tB^{\log_2 3}$	$tB^{\log_2 3}$	$tB^{\log_2 3}$
Lyubashevsky et al. [50]	✓	✓	✓	R-SIS R-LWE	tB	tB	tB	$tB \log B$	tB	tB
Kuchta et al. [46]	✓	✓	✓	M-SIS M-LWE	tB	tB	tB	$tB \log B$	$tB \log B$	$tB \log B$
This work (TS)	✓	✓	✗	RO	tB	$\log tB$	$\log tB$	$tB \log B$	$\log tB$	$\log tB$
This work (MC)	✗	✗	✗	Q-DLOG	tB	$\log tB$	$\log tB$	$tB \log B$	$\log tB$	$\log tB$

PQ: Is Post-Quantum Secure, ZK: Is Zero-Knowledge, Tr: Is Transparent
t: The number of addition/multiplication instances being proved
B: The maximum bit-length of the coefficients of the polynomial

Table 2. Comparison of Integer Addition and Multiplication Arguments

2 Techniques

We now outline the techniques underlying our contributions, making several simplifying assumptions for clarity of exposition. We explicitly indicate whenever such a simplification is made.

2.1 Prelude

Before presenting our contributions, we briefly review SNARKs over the integers and summarize the key techniques and concepts drawn from prior work. As in Zinc [35], our goal is to construct a SNARK for the Customizable Constraint System (CCS) [59] over bounded integers $\mathbb{Z}_B$, where $B \in \mathbb{N}$ denotes the bit-length bound.

Mod-PIOP + Mod-PC → SNARK over Integers. We obtain our SNARK by instantiating the framework of ℤinc [35], which builds upon the framework introduced by Campanelli et al. [18]. At a high level, this framework compiles an information-theoretic component—called a Modular Interactive Oracle Proof (Mod-PIOP)—with a cryptographic component that we refer to as a relaxed Modular Polynomial Commitment (mod-PC). In a (multivariate) Mod-PIOP, in the commitment phase, the prover $\mathcal{P}$ first sends oracles to multilinear polynomials defined over $\mathbb{Z}_B$ to the verifier. Then, in the querying phase, the verifier $\mathcal{V}$ samples a random prime $q \in \mathbb{N}$ and queries these oracles only modulo q. Finally, in the decision phase, the verifier checks the validity of the proof using the answers to its queries. We adopt the same PIOP as ℤinc, consisting of a PIOP for CCS augmented with a lookup PIOP.

Mod-PC. To realize an idealized Mod-PIOP as a concrete argument, oracle access to polynomials over $\mathbb{Z}_B$ is instantiated using a mod-PC. Concretely, in the commitment phase, the prover sends *commitments* to μ-variate multilinear

Scheme	Properties					Prover	Verifier	Proof
	PQ	ZK	Tr	∞	Assumpt.			
Towa et al. [60]	✗	✓	✓	✓	*GUO**	$sB \log B$	$B + \log s + \lambda$	$B + \log s + \lambda$
PoKEMath Bünz et al.[15]	✓	✗	✓	✓	*GUO**	$tB \log B$	$\lambda(t + s \log \lambda)$	$t\lambda$
$\mathbb{Q}\bar{a}ren_{\mathsf{TS}}$ (Instantiated with [11])	✓	✗	✓	✗	RO	$tB \log B$	$t\lambda + \log tB + \lambda \log \lambda$	$t\lambda + \log tB$
$\mathbb{Q}\bar{a}ren_{\mathsf{MC}}$ (Instantiated with [29])	✗	✗	✗	✗	Q-DLOG	$tB \log B$	$t\lambda + \log tB + \lambda \log \lambda$	$t\lambda + \log tB$

PQ: Is Post-Quantum Secure, ZK: Is Zero-Knowledge
∞: Supports unbounded integers, Tr: Is Transparent
t: The number of addition/multiplication instances being proved
B: The maximum bit-length of the coefficients of the polynomial

Table 3. Comparison of arguments for Diophantine equations

polynomials defined over $\mathbb{Z}_B$. Later, in the evaluation phase, the prover can open these commitments at verifier-chosen random points *modulo a randomly sampled prime* q, providing a proof π that the revealed value z satisfies $z \equiv p(\boldsymbol{y}) \pmod q$.

Observe that this querying/opening step mirrors that of a standard PCS over $\mathbb{F}_q$, where evaluations are verified modulo a fixed prime q. The key difference is that, in our setting, the prime is sampled during the query phase, *after* the polynomial has already been committed. This "commit-then-choose-q" requirement is therefore incompatible with many mainstream PCS constructions, such as those based on elliptic curves [44,10].

Why Field-Agnostic PCSs Are Not Enough. There exists a class of PCSs known as *field-agnostic*, which can operate over arbitrary fields [6,38,57,2,13]. While these schemes offer considerable flexibility, they still require the underlying field to be fixed before the commitment is created. In contrast, in a mod-PC the verifier samples the prime field $\mathbb{F}_q$ only after the commitments are produced, making the construction of a mod-PC strictly more challenging.

Relaxed mod-PC vs. Mod-PC. As in $\mathbb{Z}$inc, which introduces $\mathbb{Z}$ip, we focus on constructing a *relaxed* variant of a mod-PC. In an honest-prover execution, a relaxed mod-PC behaves identically to a standard mod-PC; the distinction arises only in the extractability guarantees against malicious provers. In a mod-PC, the extractor is guaranteed to recover polynomials with *integer* coefficients (or evaluations), whereas in a relaxed mod-PC the extractor is only guaranteed to recover polynomials with *rational* coefficients (or evaluations).

2.2 Warmup: A PCS with Known Modulus

Before presenting our framework for constructing a relaxed mod-PC, we begin with a simplified version that yields a standard PCS in the setting where the

prime q is known in advance. In Section 2.4, we then lift this construction to obtain a relaxed mod-PC. For clarity of exposition, we focus in this technical overview on commitments to univariate polynomials $f \in \mathbb{Z}_B^{\leq N}[X]$; the framework extends to multilinear polynomials in a straightforward manner, as described in Section 5.

Before committing to integer-valued polynomials, we first construct a much simpler primitive: a commitment scheme for a single bounded integer.

An Integer Commitment Scheme. To commit to an integer $a \in \mathbb{Z}_B$, assuming $B = 2^\beta$, we first decompose it into its bit representation $\boldsymbol{a} = [a_0, a_1, \ldots, a_{B-1}]$, where $a = \sum_{i=0}^{B-1} a_i 2^i$. We then interpret the vector $\boldsymbol{a}$ as the evaluation vector of a β-variate multilinear polynomial $\widetilde{a}$ and commit to a by committing to $\widetilde{a}$ using a multilinear PCS over $\mathbb{F}_p$. We denote the resulting integer commitment by cm_a. Notably, this ICS is generic with respect to the choice of the underlying multilinear PCS over $\mathbb{F}_p$, and more importantly, p is completely independent of q and can be different. Also, we extend this ICS to a batched setting which allows us to commit to multiple integers simultaneously (see Section 4.1 for more details). Later, we will see that this choice of integer commitment allows us to treat the verifier as having oracle access to the binary representation of the integer, enabling the reuse and construction of sumcheck-based PIOPs.

We now describe how to commit to an integer-valued polynomial using the integer commitment primitive introduced above. These techniques are also used in prior work [2,15].

PCS Commitment Phase. We commit to a univariate polynomial $f \in \mathbb{Z}_B^{\leq N}[X]$ as follows:

1. We evaluate the polynomial at a publicly known, sufficiently large integer point $\alpha \in \mathbb{Z}$ such that $\alpha \gg B$. This can be viewed as an *encoding* or *Injective map* of the polynomial f into the single very large integer value $f(\alpha) \in \mathbb{Z}$.
2. We then commit to the integer value $f(\alpha)$ using the integer commitment scheme described above.

The binding of the PCS follows from the binding of the integer commitment: if a commitment could be opened to two polynomials $f, g \in \mathbb{Z}_B^{\leq N}[X]$, then we would have $f(\alpha) = g(\alpha)$ over the integers, which is impossible due to the injectivity of the encoding map $f \mapsto f(\alpha)$ for sufficiently large α.

Evaluation Phase. For simplicity, we focus on a non-adaptive evaluation point, i.e., $z \in \mathbb{Z}_B$ sampled uniformly at random, which suffices for our SNARK applications. In Section 5, we extend the construction to adaptive evaluation via a simple repetition of the protocol. To prove that $f(z) = y \bmod q$, we begin with the following observation: Defining $g \in \mathbb{Z}_B^{\leq N}[X]$ as $g(X) := 1 + zX + z^2X^2 + \cdots + z^{N-1}X^{N-1}$, and its reverse polynomial $g^{\mathsf{rev}}(X) \in \mathbb{Z}_B^{\leq N}[X]$ as $g^{\mathsf{rev}}(X) := z^{N-1} + z^{N-2}X + \cdots + X^{N-1}$. Observe that multiplying the integer encodings of f and g^{rev} yields a highly structured result:

$$f(\alpha) \cdot g^{\mathsf{rev}}(\alpha) = \left(\sum_{i=0}^{N-1} f_i \alpha^i\right)\left(\sum_{i=0}^{N-1} g_{N-1-i}\alpha^i\right)$$

$$= \left(\sum_{i=0}^{N-2} c_i \alpha^i\right) + \langle \boldsymbol{f}, \boldsymbol{g} \rangle \cdot \alpha^{N-1} + \left(\sum_{i=N}^{2N-2} c_i \alpha^i\right)$$

which shows that the coefficient of α^{N-1} is the inner product of the coefficient vectors of f and g which is $\langle \boldsymbol{f}, \boldsymbol{g} \rangle = f(z) = y$, hence

$$f(\alpha) \cdot g^{\mathsf{rev}}(\alpha) = l + m \cdot \alpha^{N-1} + r \tag{1}$$

We denote by l, m, and r the left, middle, and right parts of the above equation, respectively. The protocol then proceeds by having the prover convince the verifier that (a) Eq. (1) holds over the integers, i.e., the prover knows integers l, m, r satisfying the equation, (b) y is the remainder of m modulo q, i.e., the prover knows an integer h such that $m = y + h \cdot q$, (c) l, m, and r are bounded integers. Since integers l, m, and r satisfying Eq. (1) are large, the prover cannot directly send them to the verifier. Instead, the prover commits to these constant number of integers using the ICS described before (see Section 4.1 for more details) and then proves the required relations as follows: First of all, statement (c) corresponds to a range proof over committed integers, which we will explain next:

2.2.1 Range Proof

The goal of the range proof is to show that a committed integer $a \in \mathbb{Z}_B$, with $B = 2^\beta$, satisfies $a < M$. As we will see in Section 5, we deliberately choose the bound to be a power of two, i.e., $M = 2^m$, which makes the range proof particularly simple. For non-power-of-two bounds, see Section 4.2.2 for a more general construction.

For a power-of-two bound $M = 2^m$, it suffices to verify that the binary decomposition of a, which we denote by $\boldsymbol{a} \in \{0,1\}^B$, has its $B - M$ most significant bits equal to zero. By the choice of the ICS, the verifier has oracle access to $\widetilde{a}$, the multilinear extension of $\boldsymbol{a}$. The verifier samples $\boldsymbol{r} \in \mathbb{F}_q^m$ and queries $\widetilde{a}(0^{\beta-m} \,\|\, \boldsymbol{r})$. If the result is zero, the verifier accepts; otherwise, it rejects.

Moreover, the verifier must ensure that the provided oracle corresponds to a valid binary decomposition, i.e., that its values are boolean. To this end, the prover and verifier run an instance of the standard multilinear ZeroCheck (see appendix A.5.5) on the polynomial $\widetilde{a}(1 - \widetilde{a})$.

Continuing with the argument for Eq. (1), Statements (a) and (b) are purely algebraic and require the prover to demonstrate knowledge of integers satisfying certain Diophantine equations, which is more complex than a range proof. To handle this, we introduce a new tool called *MaSDAK*[7], described next.

2.2.2 MaSDAK

Magnitude Succinct Diophantine Argument of Knowledge (MaSDAK, see the detailed description of the protocol in Section 4.2.6) proves knowledge of large integers $u_1, \ldots, u_t \in \mathbb{Z}_B$ (where this B is specific to MaSDAK) satisfying a Diophantine equation $f(u_1, \ldots, u_t) = 0$ for an integer polynomial f. The core idea

[7] named after Mazdak (d. c. 524 CE), a Persian reformer associated with ideals of justice and equity

of MaSDAK is to verify the relation modulo a random prime rather than directly over the integers. Concretely, the verifier samples a random λ-bit prime ℓ and asks the prover for the remainders $r_1, \ldots, r_t \in \mathbb{Z}_\ell$ of $u_1, \ldots, u_t$. The verifier then checks that $f(r_1, \ldots, r_t) \equiv 0 \pmod{\ell}$. The integers $u_1, \ldots, u_t$ need not all reside in the same commitment: MaSDAK accepts a tuple of commitments and reconstructs the u_i from their concatenation, so a single Diophantine equation can relate values bound by different commitments (this is what our evaluation argument needs, as it links the input polynomial commitment to a freshly committed auxiliary vector).

It remains to prove consistency between $r_1, \ldots, r_t$ and $u_1, \ldots, u_t$, i.e., that there exist integers $h_1, \ldots, h_t \in \mathbb{Z}_B$ such that $u_i = r_i + h_i \cdot \ell$ for all $i \in [t]$. We introduce a tool called *Division Check* to prove this efficiently.

2.3 Division and Multiplication Check

Before describing the Division Check, we first present a key building block: an argument for integer multiplication. We will later show that a simple modification of this multiplication argument yields the Division Check.

2.3.1 Integer Multiplication Argument

Here, the prover aims to convince the verifier that the committed integers $a, b, c \in \mathbb{Z}_B$ satisfy the multiplication relation $c = a \cdot b$. Recall that, by our ICS design, the verifier receives commitments to the multilinear polynomials $\widetilde{a}, \widetilde{b}, \widetilde{c}$ corresponding to the bit decompositions $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \{0,1\}^B$ of a, b, c, respectively. For simplicity, we treat these commitments as oracles and refer to openings as oracle queries.

As in [50,46], we view integer multiplication as two steps on the bit representations: first form the schoolbook convolution, then ripple the carries to obtain the product bits.

The key observation for the first step is that the schoolbook convolution is exactly univariate polynomial multiplication: writing $f_a(Y) = \sum_i a_i Y^i$, $f_b(Y) = \sum_j b_j Y^j$, and $f_t(Y) = \sum_\ell t_\ell Y^\ell$ for the polynomials whose coefficient vectors are $\boldsymbol{a}$, $\boldsymbol{b}$, and the schoolbook digit vector $\boldsymbol{t}$, we have $\boldsymbol{t} = \boldsymbol{a} \star \boldsymbol{b}$ if and only if $f_t = f_a \cdot f_b$. By the univariate Schwartz–Zippel lemma (Lemma 1), this polynomial identity can be checked at a *single* random point $r \in \mathbb{F}_p$: it suffices that $f_a(r) \cdot f_b(r) = f_t(r)$. To make this succinct over the committed multilinear polynomials, note that each evaluation $f_a(r) = \sum_i a_i r^i = \sum_{\boldsymbol{x}} \widetilde{a}(\boldsymbol{x})\, \widetilde{\mathsf{mon}}_r(\boldsymbol{x})$ is an inner product with the monomial vector $\mathbf{mon}_r = [1, r, \ldots, r^{B-1}]$, whose MLE $\widetilde{\mathsf{mon}}_r(\boldsymbol{x}) = \prod_j \left((1-x_j)+x_j r^{2^j}\right)$ the verifier can evaluate succinctly. So the prover sends the three scalar evaluations $f_a(r), f_b(r), f_t(r)$; a sumcheck binds each of them to the corresponding committed vector $\widetilde{a}, \widetilde{b}, \widetilde{t}$, and the verifier checks the single product $f_t(r) = f_a(r) \cdot f_b(r)$. This is the convolution check of Section 4.2.3.

To check the second step, the prover exhibits a carry witness $\boldsymbol{k}$ and proves a ripple-carry equation of the form $t[j]+k[j] = c[j]+2k[j+1]$, where $t[j]$ is the j-th convolution digit (the same object that the convolution check ties to $\boldsymbol{a}$ and $\boldsymbol{b}$) and $\boldsymbol{c}$ is the committed product decomposition. A subtlety distinguishes this from integer addition: there the column sums are at most 3, so the carry is a single bit,

whereas here the convolution digits $t[j]$ can be as large as $B/2$, so the carries are *not* binary—they range over $[0, B/2)$ (see Lemma 3). Consequently the prover does not booleanity-check $\boldsymbol{k}$ itself; instead it commits the $w = \beta - 1$ binary bit-decomposition vectors of $\boldsymbol{k}$ (the vector $\kappa^{(t)}$ holds the t-th bit of each carry), booleanity-checks those, and proves the reconstruction $k[j] = \sum_{t=0}^{w-1} 2^t \kappa^{(t)}[j]$, which certifies $0 \leq k[j] < B/2$.

Summary. In summary, the prover sends the schoolbook digit vector $\boldsymbol{t}$, the evaluations $f_a(r), f_b(r), f_t(r)$ at the random point r, the carry $\boldsymbol{k}$, and its bit-decomposition vectors $\kappa^{(0)}, \ldots, \kappa^{(w-1)}$, and proves the following (together with the committed $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$):

- **Booleanity**: $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and the carry bit-decomposition vectors $\kappa^{(0)}, \ldots, \kappa^{(w-1)}$ are bit vectors, as in the range proof (Section 2.2.1); the carry $\boldsymbol{k}$ itself is *not* a bit vector but is bounded to $[0, B/2)$ via its bit-decomposition reconstruction;
- **Overflow**: no overflow in ab, reduced (as in Section 2.2.1) to checking that a and b use at most $B/2$ bits;
- **Convolution**: $\boldsymbol{t} = \boldsymbol{a} \star \boldsymbol{b}$, verified by the convolution check—evaluating f_a, f_b, f_t at a random point r via three sumchecks and checking the single product $f_t(r) = f_a(r) \cdot f_b(r)$ (Section 4.2.3);
- **Ripple carry**: $\boldsymbol{k}$ satisfies the ripple-carry equation linking the convolution digits $t[j]$ to the product bits $c[j]$, in analogy to integer addition (Section 4.2.1), except that $\boldsymbol{k}$ is range-bounded (via its bit-decomposition vectors) rather than boolean, since the convolution digits exceed one bit.

The full PIOR is given in Section 4.2.4.

Division Check. The Division Check verifies a relation of the form $u = \ell h + r$ (e.g. u and h committed, ℓ a λ-bit prime, r bounded). As with multiplication, one cannot certify $u = \ell h + r$ by a single pointwise identity: integer multiplication requires convolution plus carries, and the subsequent add of r requires its own carry chain. The formal PIOR (PIOR 5) therefore reduces to an integer multiplication argument on (h, ℓ) (producing an intermediate product commitment) and an integer addition argument that links that product to u with remainder r.

Conclusion of the Warmup. By combining the above techniques, we can construct a standard field-agnostic PCS over $\mathbb{F}_q$ where the prime $q \in \mathbb{N}$ is known beforehand. But, it's still not clear how to extend this construction to a relaxed mod-PC where q is only known during evaluation. We address this in the next section using the commitment switching technique.

2.4 From PC over $\mathbb{F}_p$ to a relaxed mod-PC

At a first glance, it may seem that the prime q has not been used anywhere at the time of commitment, so the above construction should already yield a relaxed mod-PC. However, this is not the case.

Recall (as described in Section 2.2) that to commit to a polynomial $f \in \mathbb{Z}_B^{\leq N}[X]$, the prover computes the integer $\hat{f}(\alpha)$ for a sufficiently large public integer α. However, α cannot be chosen arbitrarily – it must also satisfy $q \mid \alpha$.

This is crucial during evaluation, where the prover shows that $\hat{f}(z) = y \pmod{q}$ by proving the existence of an integer h such that $\hat{f}(z) = y + h \cdot q \mod \alpha$. As shown in DewTwo [15], to conclude $\hat{f}(z) = y \pmod{q}$, it is necessary that $\alpha \equiv 0 \pmod{q}$. To ensure both this and the largeness property of α, DewTwo [15] sets $\alpha = q^{7\mu}$.

In our setting, however, the modulus q is not known at commitment time, so α cannot be fixed in advance and the commitment cannot be computed directly. This constitutes our main technical challenge, which we address using the *Commitment Switching* technique. Throughout this paper, we use the terms "modulus switching" and "commitment switching" interchangeably.

Commitment Switching. To address the above roadblock, we propose a technique called commitment switching. This concept is closely related to mechanisms used in several modern PC schemes: it appears implicitly in KZH [43], and explicitly via code switching in Blaze [9] and FICS [3].

The rough idea of commitment switching is to first commit to a polynomial f under a certain commitment scheme (called the source commitment scheme) and get $\mathsf{cm}_{\mathsf{source}}$, and later commit to the same polynomial under a different commitment scheme (called the target commitment scheme) to get $\mathsf{cm}_{\mathsf{target}}$, then argue that both $\mathsf{cm}_{\mathsf{source}}$ and $\mathsf{cm}_{\mathsf{target}}$ correspond to the same polynomial f. There are different reasons to perform commitment switching in different schemes, but in our case, the reason is to first commit to the polynomial f using α_{source} that only satisfies the "largeness" requirement and not the divisibility requirement, because q is not known yet; then, after receiving the challenge q, switch to a commitment α_{target} that satisfies both "largeness" and "divisibility" requirements. Hence, in the commitment phase, the prover sends $\mathsf{cm}_{\mathsf{source}} = \mathsf{ICS.Commit}(f(\alpha_{\mathsf{source}}))$. Then, after knowing q at the time of evaluation, the prover sends a *new* commitment with $\alpha_{\mathsf{target}} = q\alpha_{\mathsf{source}}$ that satisfies both the largeness and divisibility requirements. Then we continue with proving the evaluation of the polynomial $f \bmod q$ using $\mathsf{cm}_{\mathsf{target}}$.

Now it remains to prove that both commitments $\mathsf{cm}_{\mathsf{source}}$ and $\mathsf{cm}_{\mathsf{target}}$ correspond to the same polynomial f. To do this, we also run the evaluation protocol for $\mathsf{cm}_{\mathsf{source}}$; although we know that it doesn't guarantee anything about the evaluation mod q, it still guarantees the evaluation mod α_{source}. In fact, during the extraction, we extract different polynomials f_{source} and f_{target} from each commitment separately, such that

$$f_{\mathsf{source}}(z) = y \mod \alpha_{\mathsf{source}} \quad \text{and} \quad f_{\mathsf{target}}(z) = y \mod \alpha_{\mathsf{target}}$$

since $\alpha_{\mathsf{target}} = q\alpha_{\mathsf{source}}$, we have that $f_{\mathsf{target}}(z) = y \mod \alpha_{\mathsf{source}}$. Hence, $f_{\mathsf{source}}(z) = f_{\mathsf{target}}(z) \mod \alpha_{\mathsf{source}}$. Since f_{source} and f_{target} were chosen before z is known, by the Schwartz–Zippel lemma (either assuming that α_{source} only has large prime factors or by using the multilinear Schwartz–Zippel lemma for composite moduli [12]), we have that with high probability over the choice of z,

$$f_{\mathsf{source}}(X) = f_{\mathsf{target}}(X) \mod \alpha_{\mathsf{source}} \tag{2}$$

Note that although the above argument shows that the two polynomials are equal mod α_{source}, the prover could still cheat by adding multiples of α_{source} to the coefficients of f_{target}. We show that this is impossible because during the extraction process, we get the guarantee that the coefficients of f_{target} are bounded rationals, hence Eq. (2) can be lifted to an equality over the rationals: $f_{\mathsf{source}}(X) = f_{\mathsf{target}}(X)$ over the rationals. This concludes the argument for commitment switching and the construction of a relaxed mod-PC.

3 Preliminaries

In this section, we review the basic definitions, notations, and tools that will be used throughout the paper.

Notation. Throughout the paper, we adopt the following notational conventions: ring (and field) elements and indices are denoted by lowercase letters (e.g., a, b, c). Vectors are denoted with lowercase bold letters (e.g., $\boldsymbol{a} \in \mathbb{Z}^n$), while matrices are denoted by uppercase bold letters (e.g., $\mathbf{A}, \mathbf{B} \in \mathbb{Z}^{m \times n}$).

Indexing and Ranges. For indexing, we write $\boldsymbol{x}[i]$ to denote the i-th entry of a vector $\boldsymbol{x}$, and $\mathbf{A}[i, j]$ to denote the entry in the i-th row and j-th column of a matrix A. Unless stated otherwise, indices are assumed to start from 0. For any integer $a \geq 1$, we define $(a] := \{1, 2, \ldots, a\}$ and $[a) := \{0, 1, \ldots, a-1\}$.

Algebra. We denote by $\mathbb{R}$ an arbitrary ring; by $\mathbb{Z}$ the set of integers; by $\mathbb{Z}_B$ the set of B-bit positive integers. We further denote by $\mathbb{Q}$ the set of rational numbers, and by $\mathbb{Q}_B := \{a/b \mid a, b \in \mathbb{Z}_B,\ b \neq 0\}$ the set of bounded rational numbers. Finally, we denote by $\mathbb{F}_p$ the finite field of prime order p, and by $\mathbb{F}_p^\times$ the multiplicative subgroup of non-zero elements of $\mathbb{F}_p$. Throughout this paper, all finite fields are prime fields. For an integer $z \in \mathbb{Z}_B$, we denote by $\mathsf{Bits}(z) \in \{0,1\}^B$ the big-endian bit-decomposition of z; i.e. MSB bits appear first. Likewise, for a vector $\boldsymbol{z} \in \{0,1\}^B$, we denote by $\mathsf{Int}(\boldsymbol{z}) \in \mathbb{Z}_B$ the integer whose big-endian bit-decomposition is $\boldsymbol{z}$.

3.1 Polynomials

We write $\mathbb{R}_\mu^{\mathsf{tot} \leq d}$ (resp., $\mathbb{R}_\mu^{\mathsf{ind} \leq d}$) for the set of μ-variate polynomials over $\mathbb{R}$ of total (resp., individual) degree $\leq d$, and $\mathbb{R}^{\leq d}$ for the set of univariate polynomials over $\mathbb{F}_p$ of degree $\leq d$.

Multilinear Extension (MLE). Let $N = 2^\mu$ and $\boldsymbol{u} \in \mathbb{F}_p^N$. We denote by $\widetilde{u} \in \mathbb{F}_{p,\mu}^{\mathsf{ind} \leq 1}$ the multilinear extension (MLE) of $\boldsymbol{u}$ defined as follows:

$$\widetilde{u}(X_1, \ldots, X_\mu) := \sum_{\boldsymbol{b} \in \{0,1\}^\mu} \boldsymbol{u}\left[\mathsf{Int}\left(\boldsymbol{b}\right)\right] \cdot \mathsf{eq}_{\boldsymbol{b}}\left(X_1, \ldots, X_\mu\right)$$

Lemma 1 (Schwartz–Zippel[56,64]). *Let $\mathbb{F}$ be a field, let $S \subseteq \mathbb{F}$ be a finite set, and let $f \in \mathbb{F}_{p,\mu}^{\mathsf{tot} \leq d}$ be a nonzero polynomial (for any $\mu \geq 1$). If*

$\mathbf{r} = (r_1, \ldots, r_\mu)$ *is sampled uniformly at random from* S^μ*, then*

$$\Pr_{\mathbf{r} \leftarrow S^\mu}\left[f(\mathbf{r}) = 0\right] \;\leq\; \frac{d}{|S|}.$$

In particular, for $\mu = 1$ *(the univariate case), this specializes to* $\Pr_{r \leftarrow S}[f(r) = 0] \leq d/|S|$.

3.2 Idealized Oracle Relations and Proof Systems

In this section, we introduce the idealized algebraic objects and proof systems used throughout the paper. The relationships among these objects are summarized in Fig. 1. Since our focus is exclusively on polynomials, all oracles and relations considered are polynomial oracles and polynomial oracle relations.

Polynomial Oracle. Let $\mathbb{R}$ be the ambient ring, $\mathbb{S}$ be evaluation ring, and $\varphi : \mathbb{R} \to \mathbb{S}$ be an evaluation map. We denote by $[\![f]\!]_\varphi$ an oracle to $f \in \mathbb{R}_\mu^{\mathsf{ind} \leq d}$ that on input $\boldsymbol{x} \in \mathbb{R}^\mu$ outputs $\varphi(f(\boldsymbol{x}))$. Whenever $\mathbb{R} = \mathbb{S}$ and φ is the identity map, we omit the subscript and simply write $[\![f]\!]$.

Notation. For ease of presentation, we use colored text to denote index-related components that are present in the general definition but typically omitted in the relations considered in this paper.

Definition 1 (Indexed Polynomial Oracle Relations). *An indexed polynomial oracle relation* $\mathfrak{R}$ *is a set of triples* $(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{w})$*, where* i *is the index,* x *is the explicit instance,* y *is the implicit instance which includes a set of polynomial oracles, and* w *is the witness.*

Definition 2 (Cartesian Product of Relations). *Consider ternary relations* $\mathfrak{R}_1$ *and* $\mathfrak{R}_2$ *over instance, oracle-instance, witness tuples. We define*

$$\mathfrak{R}_1 \times \mathfrak{R}_2 = \left\{ ((\mathsf{i}_1, \mathsf{i}_2), (\mathsf{x}_1, \mathsf{x}_2), (\mathsf{y}_1, \mathsf{y}_2), (\mathsf{w}_1, \mathsf{w}_2)) \;\middle|\; \begin{matrix} (\mathsf{i}_1, \mathsf{x}_1, \mathsf{y}_1, \mathsf{w}_1) \in \mathfrak{R}_1 \\ (\mathsf{i}_2, \mathsf{x}_2, \mathsf{y}_2, \mathsf{w}_2) \in \mathfrak{R}_2 \end{matrix} \right\}.$$

We let $\mathfrak{R}^\ell$ *denote* $\mathfrak{R} \times \cdots \times \mathfrak{R}$ *for* ℓ *times.*

3.2.1 Generalized Polynomial Interactive Oracle Reduction

Here, we define Generalized Polynomial Interactive Oracle Reductions (GenPIORs). Our definition is inspired by the interactive oracle reductions (IORs) of Bünz et al. [14], and extends that framework in one key dimension while restricting it in another. Specifically: (a) We generalize IORs by allowing the ambient ring of a polynomial oracle to differ from its evaluation ring, capturing settings in which polynomials are defined over one ring but evaluated over another via an evaluation map. (b) We restrict attention to polynomial oracles and polynomial oracle relations, rather than arbitrary oracles and relations. This is a deliberate scoping choice—tailored to the needs of our work—rather than a technical limitation of the framework.

Definition 3 (Generalized Polynomial Interactive Oracle Reduction (Gen-PIOR)). *Let $\mathfrak{R}$ and $\mathfrak{R}'$ be indexed oracle relations. A (public-coin, holographic) interactive oracle reduction from $\mathfrak{R}$ to $\mathfrak{R}'$ is a tuple of the form* $\mathsf{Gen\text{-}PIOR} = (\mathsf{Algs}, \mathsf{Eval\text{-}Conf}, \mathsf{Metrics})$ *defined as follows:*

- Algs *consists of a tuple of polynomial time algorithms $(\mathcal{I}, \mathcal{P}, \mathcal{V})$ participating in the protocol, namely an indexer $\mathcal{I}$, a prover $\mathcal{P}$, and a verifier $\mathcal{V}$. At the beginning of the protocol, the indexer $\mathcal{I}$ receives as input an index i (for a relation $\mathfrak{R}$) and outputs a short index ι, an implicit index I, and a new index i' (for a relation $\mathfrak{R}'$). The prover $\mathcal{P}$ and verifier $\mathcal{V}$ are interactive algorithms. The prover receives as input the index i, an explicit instance x, an implicit instance y, and a witness w, while the verifier receives as input the short index ι, the explicit instance x, oracle access to the implicit index I, and oracle access to the implicit instance y. The prover and verifier engage in (potentially multiple) rounds of interaction. At the end of the protocol, the prover outputs a new witness w', and the verifier outputs a new explicit instance x' and a new implicit instance y'.*
- $\mathsf{Eval\text{-}Conf}$ *is a tuple $(\mathbb{R}, \mathbb{S}, \Phi)$ defining the evaluation configuration of the oracles in the protocol, where $\mathbb{R}$ denotes the ambient ring, $\mathbb{S}$ denotes the evaluation ring, and Φ is a set of evaluation mappings of the form $\varphi : \mathbb{R} \to \mathbb{S}$.*
- $\mathsf{Metrics}$ *is a tuple $(k, n_\Pi, n_\varrho, s, v)$ defining the efficiency metrics of the protocol, where k denotes the number of rounds, $n_\Pi : (k] \to \mathbb{N}$ specifies the number of polynomials sent by the prover in each round, $n_\varrho : (k] \to \mathbb{N}$ specifies the number of challenges sent by the verifier in each round, $s : \mathbb{N} \to \mathbb{N}$ specifies the number of oracle queries made by the verifier, and $v : (k] \times \mathbb{N} \to \mathbb{N}$ specifies the number of variables of the polynomials sent by the prover in each round.*

We describe the indexing phase in more detail below.

Indexer in the 0-th round. The indexer $\mathcal{I}$ runs on input i and outputs a short index ι, an implicit index I, and a new index i'. Concretely, it computes the round-0 oracles to multilinear polynomials

$$\Pi_0 = \{\, p_{0,j} \,\}_{j=1}^{n_\Pi(0)} \subseteq \mathbb{R}^{\mathsf{ind}\leq 1}_{v(0,j)},$$

and encodes them as an oracle description in the implicit index I. The short index ι contains any public auxiliary data needed by the verifier. The indexer outputs $(\iota, \mathsf{I}, \mathsf{i}')$.

We describe the prover-verifier interaction in more detail below.

$\mathcal{P}$ in i-th round. The prover $\mathcal{P}$ sends a collection of polynomial oracles $\Pi_i \in \{\, p_{i,j} \,\}_{j=1}^{n_\Pi(i)}$. Each polynomial $p_{i,j} \in \mathbb{R}^{\mathsf{ind}\leq 1}_{v(i,j)}$ is a multilinear polynomial over $\mathbb{R}$ in $v(i,j)$ variables.

$\mathcal{V}$ in i-th round. $\mathcal{V}$ receives evaluation oracle access to polynomials in Π_i, then replies with a collection of uniformly random challenges $\varrho_i \in \{\, \rho_{i,j} \,\}_{j=1}^{n_\varrho(i)}$. Each challenge $\rho_{i,j} \in$ is a ring element in $\mathbb{R}$.

Verifier phases. Without loss of generality, the verifier is split into two phases. In the interaction phase, it samples challenges and sends them to the prover. In

the query phase, first it samples an evaluation map $\varphi \in \Phi$, then it makes s queries to the polynomial oracles in the implicit index, implicit instance, and prover's messages. All oracle queries are evaluated via evaluation map $\varphi : \mathbb{R} \to \mathbb{S}$ where $\mathbb{S}$ is the evaluation ring. The verifier's output is a deterministic function of its input and the transcript, which we denote

$$(\mathsf{x}', \mathsf{y}') := \mathcal{V}^{\boldsymbol{\Pi},\mathsf{I},\mathsf{y}}(\iota, \mathsf{x}, \boldsymbol{\varrho})$$

Now, we define the security properties of Gen-PIORs:

Definition 4 (Completeness). Gen-PIOR *is complete if the following holds. For any* $(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{w}) \in \mathfrak{R}$*,*

$$\Pr\left[(\mathsf{i}',\mathsf{x}',\mathsf{y}',\mathsf{w}') \in \mathfrak{R}' \;\middle|\; \begin{array}{l} (\iota, \mathsf{I}, \mathsf{i}') \leftarrow \mathcal{I}(\mathsf{i}) \\ (\mathsf{w}', (\mathsf{x}', \mathsf{y}')) \leftarrow \langle \mathcal{P}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{w}), \mathcal{V}^{\mathsf{I},\mathsf{y}}(\iota,\mathsf{x}) \rangle \end{array} \right] = 1 \ .$$

We define the notion of round-by-round soundness and knowledge-soundness for our PIORs, which are stronger than the standard notions of soundness and knowledge-soundness, and imply soundness and knowledge-soundness in the non-interactive setting after the Fiat–Shamir transformation [19,23]. We first define the state function for PIORs as follows:

Definition 5 (State function for PIORs). *A* state function *for* Gen-PIOR *is a function* State *for which the following holds:*

- Empty transcript: $\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\emptyset) = 0$ *unconditionally, where* $\emptyset$ *is the empty transcript.*
- Prover moves: *If* $\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$ *for a partial transcript* $\mathsf{tr} = (\Pi_1, \varrho_1, \ldots, \Pi_{i-1}, \varrho_i)$*, then for every possible next prover message* Π_{i+1}*, where the prover is about to move, then for any prover message* Π_i*,* $\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}||\Pi_i) = 0$
- Full transcript: *If* $\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$ *for a full transcript* $\mathsf{tr} = (\Pi_1, \varrho_1, \ldots, \Pi_k, \varrho_k)$*, then the verifier outputs* $(\mathsf{x}', \mathsf{y}') := \mathcal{V}^{\boldsymbol{\Pi},\mathsf{I},\mathsf{y}}(\iota,\mathsf{x},\boldsymbol{\varrho})$ *such that* $(\mathsf{i}',\mathsf{x}',\mathsf{y}') \notin \mathfrak{L}(\mathfrak{R}')$ *(where* $(\iota, \mathsf{I}, \mathsf{i}') := \mathcal{I}(\mathsf{i})$*).*

Definition 6 (Round-by-round Soundness). Gen-PIOR *has round-by-round (rbr) soundness error* ϵ_{RBR} *if there exists a state function* State *such that the following holds. If* $\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$ *for* $(\mathsf{i},\mathsf{x},\mathsf{y}) \notin \mathfrak{L}(\mathfrak{R})$ *and a partial transcript* $\mathsf{tr} = (\Pi_1, \varrho, \ldots, \Pi_i)$*,* $i \in (k]$*, where the verifier is about to move, then*

$$\Pr_{\varrho_i \leftarrow \mathbb{F}^*}[\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}||\varrho_i) = 1] \leq \epsilon_{\mathsf{RBR}}(\mathsf{i},\mathsf{x}) \tag{3}$$

Definition 7 (Round-by-round Knowledge Soundness). Gen-PIOR *has round-by-round knowledge error* κ_{RBR} *if there exists a state function State and polynomial-time extractor* $\mathcal{E}$ *such that the following holds. If* $\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$ *for a partial transcript* $\mathsf{tr} = (\Pi_1, \varrho_1, \ldots, \Pi_i)$*,* $i \in (k]$ *where the verifier is about to move, then*

$$\Pr_{\varrho_i \leftarrow \mathbb{F}^*}[\mathsf{State}(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{tr}||\varrho_i) = 1] > \kappa_{\mathsf{RBR}}(\mathsf{i},\mathsf{x})$$

implies the following. For any transcript continuation $(\Pi_{i+1}, \ldots, \Pi_k)$*, the extractor outputs* $\mathsf{w} \leftarrow \mathcal{E}(\mathsf{i},\mathsf{x},\mathsf{y},\boldsymbol{\Pi},\mathsf{w}')$ *such that* $(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{w}) \in \mathfrak{R}$*.*

3.2.2 Gen-PIOR Composition

Theorem 2 (Product Composition of Gen-PIORs). *Let* $\mathfrak{R}$ *and* $\{\mathfrak{R}_i\}_{i=1}^m$ *be indexed oracle relations such that*

$$\mathfrak{R} = \bigcap_{i=1}^{m} \mathfrak{R}_i \quad \textit{i.e.,} \quad (\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}) \in \mathfrak{R} \iff (\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}) \in \mathfrak{R}_i \quad \textit{for all } i \in [m].$$

For each $i \in [m]$*, let* $\mathsf{Gen\text{-}PIOR}_i : \mathfrak{R}_i \to \mathfrak{R}'_i$ *be a Gen-PIOR with round-by-round soundness error* $\epsilon^{(i)}_{\mathsf{RBR}}$*. Define* $\mathfrak{R}' := \mathfrak{R}'_1 \times \cdots \times \mathfrak{R}'_m$ *to be the* Cartesian product of relations *as in Definition 2. Let* $\mathsf{PIOR} : \mathfrak{R} \to \mathfrak{R}'$ *be the protocol that executes the* PIOR_i *on the same input index-instance–witness tuple and outputs the resulting tuple of index-instance–oracle–witness triples. Then* PIOR *is a PIOR with round-by-round soundness error*

$$\epsilon_{\mathsf{RBR}} \leq \max_{i \in [m]} \epsilon^{(i)}_{\mathsf{RBR}}.$$

Proof. See Appendix A.1. □

3.2.3 Generalized Polynomial interactive Oracle Proofs

Generalized Polynomial Interactive Oracle Proofs (Gen-PIOPs) are special cases of Gen-PIORs that reduce arbitrary relations to decision relations defined as follows.

Definition 8 (Decision Relation). *An indexed oracle relation* $\mathfrak{R}$ *is a decision relation* $\mathfrak{R}_{\mathsf{Decision}}$ *is the set of triples of the form* $(\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}) = (\varnothing, \mathsf{accept}, \varnothing, \varnothing)$*.*

Compared to standard PIOPs as defined in [32,22], Gen-PIOPs generalize the model by allowing polynomial oracles to be defined over one ring and evaluated over another, and thereby subsume Mod-PIOPs as introduced in [18].

Definition 9 (Generalized Polynomial Interactive Oracle Proof (Gen-PIOP)). *We denote by* $\mathsf{Gen\text{-}PIOP} = (\mathsf{Algs}, \mathsf{Eval\text{-}Conf}, \mathsf{Metrics})$ *a polynomial interactive oracle proof for a relation* $\mathfrak{R}$ *if* $\mathsf{Gen\text{-}PIOR} = (\mathsf{Algs}, \mathsf{Eval\text{-}Conf}, \mathsf{Metrics})$ *is a Gen-PIOR of the form* $\mathsf{Gen\text{-}PIOR} : \mathfrak{R} \to \mathfrak{R}_{\mathsf{Decision}}$*.*

3.2.4 Special Cases of Gen-PIORs and Gen-PIOPs

Here, we define several special cases of Gen-PIORs and Gen-PIOPs that are commonly used in practice.

Definition 10 (Standard PIOR/PIOP). *A standard PIOR (resp., PIOP) is a Gen-PIOR (resp., Gen-PIOP) where the ambient ring and evaluation ring are the same and equal to a field* $\mathbb{F}$*, that is* $\mathbb{R} = \mathbb{S} = \mathbb{F}$*. In this case, the evaluation map is the identity map, i.e.* $\varphi : \mathbb{F} \to \mathbb{F},\ \varphi(x) = x$*.*

Next, we define modular PIORs and PIOPs over the integers. The notion of a modular PIOP (Mod-PIOP) was first introduced by Campanelli et al. [18, Definition 10].

Definition 11 (Modular PIOR/PIOP [18]). *A modular PIOR (resp., PIOP) is a Gen-PIOR (resp., Gen-PIOP) where the ambient ring is the integers and the evaluation ring is a finite field $\mathbb{F}_p$, that is $\mathbb{R} = \mathbb{Z}$ and $\mathbb{S} = \mathbb{F}_p$. In this case, the evaluation map is the natural modulo map $\varphi : \mathbb{Z} \to \mathbb{F}_p$, $\varphi(x) = x \bmod p$.*

Definition 12 (Relaxed Modular PIOR/PIOP). *A relaxed modular PIOR (resp., PIOP) is a mod-PIOR (resp., Mod-PIOP) in which the extractor is allowed to output a rational witness $\mathsf{w} \in \mathbb{Q}_B$.*

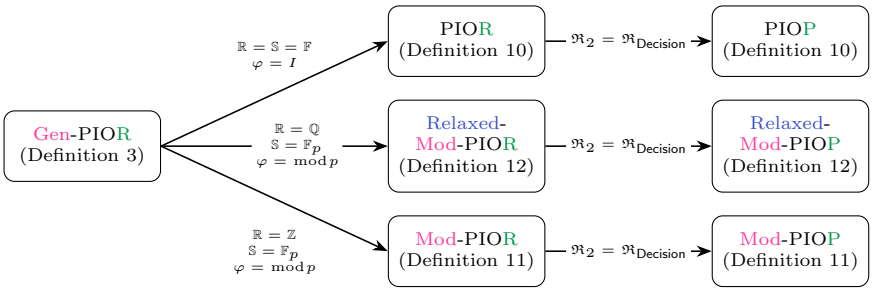

Fig. 1. Relationship between the idealized proof systems defined in Section 3.2.

3.3 NP Relations and Arguments Systems

Definition 13 (Indexed Relation). *An indexed relation $\mathfrak{R}$ is a set of triples $(\mathsf{i},\mathsf{x},\mathsf{y},\mathsf{w})$, where i is the index, x is the explicit instance, y is the implicit instance, and w is the witness.*

The explicit and implicit instance separation is useful for decoupling the committed and non-committed information in the argument system, as was proposed in Campanelli et al.[17] with a different terminology.

3.3.1 Reduction of Knowledge

Since we invoke arguments in a nested manner, we prefer to work with the notion of reduction of knowledge (RoK) as defined by Kothapalli et al. [45], which generalizes the notion of argument of knowledge (AoK).

Definition 14 (Reduction of Knowledge(RoK, [45])). *Consider indexed relations $\mathfrak{R}$ and $\mathfrak{R}'$. A reduction of knowledge (RoK) from $\mathfrak{R}$ to $\mathfrak{R}'$ is defined by PPT algorithms $\mathsf{ROK} = (\mathcal{S}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ defined as follows:*

- *$\mathcal{S}(1^\lambda, N) \to \mathsf{pp}$: The setup algorithm takes as input a security parameter 1^λ and a maximum size parameter N, and outputs public parameters pp.*
- *$\mathcal{I}(\mathsf{pp}, \mathsf{i}) \to (\mathsf{pk}, \mathsf{vk}, \mathsf{i}')$: The indexer algorithm takes as input the public parameters pp and an index i for relation $\mathfrak{R}$, and outputs a proving key pk, a verification key vk, and a new index i' for relation $\mathfrak{R}'$.*

– $\langle \mathcal{P}(\mathsf{pk}, \mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}), \mathcal{V}(\mathsf{vk}, \mathsf{x}, \mathsf{y}, \mathsf{w}') \rangle \to (\mathsf{w}', \mathsf{x}', \mathsf{y})$*: The prover and verifier are interactive algorithms. The prover takes as input the proving key* pk*, an index* i*, an explicit instance* x*, an implicit instance* y*, and a witness* w*, while the verifier takes as input the verification key* vk*, the explicit instance* x*, and a new explicit instance* x'*. At the end of the protocol, the prover outputs a new witness* w'*, and the verifier outputs a new instance* x'*.*

Definition 15 (Completeness). *For all size bounds* $N \in \mathbb{N}$ *and efficient adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{c|l} (\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}) \in \mathfrak{R} & \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda, N) \\ \vee & (\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{i}', \mathsf{x}', \mathsf{y}', \mathsf{w}') \notin \mathfrak{R}' & (\mathsf{pk}, \mathsf{vk}, \mathsf{i}') \leftarrow \mathcal{I}(\mathsf{pp}, \mathsf{i}) \end{array}\right] = 1$$

Definition 16 (Adaptive Knowledge Soundness). *We say* ROK *is an adaptive knowledge-sound reduction of knowledge, if for all size bounds* $N \in \mathbb{N}$ *and for all expected poly-time stateful adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *there exists an expected poly-time extractor* $\mathcal{E}$ *such that*

$$\Pr\left[\begin{array}{c|l} (\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}^*) \notin \mathfrak{R} & \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda, N) \\ \wedge & \mathsf{i} \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ (\mathsf{i}', \mathsf{x}', \mathsf{y}', \mathsf{w}') \in \mathfrak{R}' & (\mathsf{pk}, \mathsf{vk}, \mathsf{i}') \leftarrow \mathcal{I}(\mathsf{pp}, \mathsf{i}) \\ & (\mathsf{x}, \mathsf{y}) \leftarrow \mathcal{A}_2(\mathsf{pk}, \mathsf{vk}, \mathsf{i}) \\ & \mathsf{w}^* \leftarrow \mathcal{E}(\mathsf{pk}, \mathsf{vk}, \mathsf{i}, \mathsf{x}, \mathsf{y}) \\ \hline & (\mathsf{w}', \mathsf{x}', \mathsf{y}) \leftarrow \langle \mathcal{A}_3(\mathsf{pk}, \mathsf{x}, \mathsf{y}), \mathcal{V}(\mathsf{vk}, \mathsf{x}, \mathsf{y}) \rangle \end{array}\right] \leq negl(\lambda)$$

Definition 17 (Semi-Adaptive knowledge soundness). *Given a partitioning of the explicit instance* $\mathsf{x} = (\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3)$*, we say* ROK *satisfies semi-adaptive knowledge soundness if for every polynomial-time public-coin sampler* `Sampler`*, and all expected poly-time adversaries* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4)$*, there exists an expected poly-time extractor* $\mathcal{E}$ *such that:*

$$\Pr\left[\begin{array}{c|l} (\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w}^*) \notin \mathfrak{R} & \mathsf{pp} \leftarrow \mathcal{S}(1^\lambda, N) \\ \wedge & \mathsf{i} \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ (\mathsf{i}', \mathsf{x}', \mathsf{y}', \mathsf{w}') \in \mathfrak{R}' & (\mathsf{pk}, \mathsf{vk}, \mathsf{i}') \leftarrow \mathcal{I}(\mathsf{pp}, \mathsf{i}) \\ & \mathsf{x}_1 \leftarrow \mathcal{A}_2(\mathsf{pp}) \\ & \mathsf{x}_2 \leftarrow \mathtt{Sampler} \\ & \mathsf{x}_3 \leftarrow \mathcal{A}_3(\mathsf{pk}, \mathsf{vk}, \mathsf{i}, \mathsf{x}_1, \mathsf{x}_2) \\ \hline & \mathsf{x} := (\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3) \\ & \mathsf{w}^* \leftarrow \mathcal{E}(\mathsf{pk}, \mathsf{vk}, \mathsf{i}, \mathsf{x}, \mathsf{y}) \\ \hline & (\mathsf{w}', \mathsf{x}', \mathsf{y}) \leftarrow \langle \mathcal{A}_4(\mathsf{pk}, \mathsf{x}, \mathsf{y}), \mathcal{V}(\mathsf{vk}, \mathsf{x}, \mathsf{y}) \rangle \end{array}\right] \leq negl(\lambda)$$

Definition 18. *We say* ROK *is succinct if the size of the communication transcript between the prover and the verifier is* $\mathrm{poly}(\log(|\mathsf{w}|), \lambda)$ *and the running time of the verifier is* $\mathrm{poly}(\log(|\mathsf{w}|), |\mathsf{x}|, \lambda)$*.*

3.3.2 Arguments of Knowledge

Definition 19 (Argument of Knowledge (ARK)). *We denote by* $\mathsf{ARK} = (\mathcal{S}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ *an argument of knowledge for a relation* $\mathfrak{R}$ *if* $\mathsf{ROK} = (\mathcal{S}, \mathcal{I}, \mathcal{P}, \mathcal{V})$ *is a reduction of knowledge from* $\mathfrak{R}$ *to the decision relation (Definition 8).*

3.4 Commitment Schemes

A commitment scheme CS over a message space $\mathfrak{D}$ is a tuple of PPT algorithms $\mathsf{CS} = (\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$ defined as follows:

- $\mathtt{Setup}(1^\lambda, \mathfrak{D}) \to \mathsf{pp}$: On input a security parameter 1^λ and a description $\mathfrak{D}$ of the message space generates the public parameters pp.
- $\mathtt{Commit}(\mathsf{pp}; m) \to (\mathsf{cm}; \mathsf{opn})$: On input the public parameters pp and a message $m \in \mathfrak{D}$, outputs a commitment cm and an opening opn.
- $\mathtt{Open}(\mathsf{pp}, \mathsf{cm}, m, \mathsf{opn}) \to \{0, 1\}$: On input the public parameter pp , a commitment cm, a message m and an opening opn , verifies the opening of cm to m using the opening opn.

Binding. A commitment scheme is binding if for all PPT adversaries $\mathcal{A}$,

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

3.5 Generalized Polynomial Commitment Scheme

We now define a Generalized Polynomial Commitment Scheme (Gen-PCS) for multilinear polynomials, extending standard polynomial commitment schemes [44] and modular polynomial commitment schemes (Mod-PC) [18]. The key distinction is that a Gen-PCS permits the evaluation ring to differ from the polynomial's ambient ring, thereby subsuming both standard PCS (Definition 24) and Mod-PC as special cases.

Definition 20. *A Generalized Polynomial Commitment Scheme (Gen-PCS) for multilinear polynomials is a tuple of the form* $\mathsf{Gen\text{-}PCS} = (\mathsf{Eval\text{-}Conf}, \mathtt{Setup}, \mathtt{Commit}, \mathtt{Open}, \mathsf{ROK}_{Eval})$ *where*

- $\mathsf{Eval\text{-}Conf}$ *is a tuple* $(\mathbb{R}, \mathbb{S}, \varphi)$ *defining the evaluation configuration of the oracles in the protocol, where* $\mathbb{R}$ *denotes the ambient ring,* $\mathbb{S}$ *denotes the evaluation ring, and* φ *is an evaluation map of the form* $\varphi : \mathbb{R} \to \mathbb{S}$.
- *the tuple* $\mathsf{CS} = (\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$ *separately constitutes a binding commitment scheme (see Section 3.4) for multilinear polynomials over* $\mathbb{R}$, *i.e. the message space* $\mathfrak{D}_{\mathsf{CS}} = \mathbb{R}_\mu^{\mathsf{ind} \leq 1}$.
- ROK_{Eval} *is an interactive argument system for the relations* $\mathfrak{R}_{\mathtt{Eval}}$ *(Def. 21).*

Following the approach in [13,15] we express the evaluation correctness of the Gen-PCS via an NP relation defined as follows. For maximum generality, we define the evaluation relation

Definition 21 (Evaluation relation). *Given a commitment scheme* $\mathsf{CS} = (\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$*, we define the NP relation* $\mathfrak{R}_{\mathtt{Eval}}$ *to be the set of triples*

$$\begin{pmatrix} \mathsf{i}, \\ \mathsf{x}, \\ \mathsf{w} \end{pmatrix} = \begin{pmatrix} (\mathsf{ck}, \mathsf{vk}) \\ (\ell, \varphi, [\mathsf{cm}_i, \boldsymbol{y}_i, z_i]_{i\in(\ell)}), \\ ([p_i, \mathsf{opn}_i]_{i\in(\ell)}) \end{pmatrix}$$

where $(\mathsf{ck}, \mathsf{vk})$ *is a valid committing and verification key for* CS *and contains a description of the message domain* $\mathfrak{D}_{\mathsf{CS}} = \mathbb{R}_{\mu}^{\mathsf{ind}\leq 1}$*, and* $\varphi : \mathbb{R} \to \mathbb{S}$ *is a mapping from the coefficient ring to the evaluation ring, such that for all* $i \in [N]$*,*

$$z_i = \varphi(p_i(\boldsymbol{y}_i)) \quad \textit{and} \quad \mathsf{PCS}.\mathtt{Open}(\mathsf{ck}, \mathsf{cm}_i, p_i, \mathsf{opn}_i) = 1$$

Special Cases of Gen-PCS. Depending on whether adversaries are permitted to adaptively select the evaluation point $\mathbf{y}$, we distinguish between two notions of extractability: non-adaptive and adaptive extractability.

Definition 22 (Adaptive Extractability). *A generalized polynomial commitment scheme* $\mathsf{Gen\text{-}PCS} = (\mathsf{Eval\text{-}Conf}, \mathtt{Setup}, \mathtt{Commit}, \mathtt{Open}, \mathsf{ROK}_{Eval})$ *has adaptive extractability if* ROK_{Eval} *is a succinct* semi-adaptive *argument of knowledge (see Section 3.3.2) for the relation* $\mathfrak{R}_{\mathtt{Eval}}$ *(see Definition 21) with respect to the following instance partitioning:*

$$\mathsf{x}_1 = (\ell, [\mathsf{cm}]_{i\in(\ell)}), \quad \mathsf{x}_2 = \varphi, \quad \mathsf{x}_3 = ([\boldsymbol{y}_i, z_i]_{i\in(\ell)})$$

Definition 23 (Non-Adaptive Extractability). *A polynomial commitment scheme* $\mathsf{Gen\text{-}PCS} = (\mathsf{Eval\text{-}Conf}, \mathtt{Setup}, \mathtt{Commit}, \mathtt{Open}, \mathsf{ROK}_{Eval})$ *has adaptive extractability if Eval is a succinct* semi-adaptive *argument of knowledge (see Section 3.3.2) for the relation* $\mathfrak{R}_{\mathtt{Eval}}$ *(see Definition 21).*

$$\mathsf{x}_1 = (\ell, [\mathsf{cm}]_{i\in(\ell)}), \quad \mathsf{x}_2 = ([\varphi, \boldsymbol{y}]_{i\in(\ell)}), \quad \mathsf{x}_3 = ([z]_{i\in(\ell)})$$

Definition 24 (Standard Polynomial Commitment Scheme [44]). *A standard polynomial commitment scheme for multilinear polynomials is a Gen-PCS in which the ambient ring and the evaluation ring coincide and are equal to a field* $\mathbb{F}$*, that is,* $\mathbb{R} = \mathbb{S} = \mathbb{F}$*. In this case, the evaluation map* φ *is the identity.*

Definition 25 (Relaxed Modular Polynomial Commitment Scheme). *A relaxed modular polynomial commitment scheme (Mod-PC) for multilinear polynomials is a mod-PC in which the extractor is allowed to output a rational witness polynomials* $p_i \in \mathbb{Q}_{\mu}^{\mathsf{ind}\leq 1}$*.*

4 Toolbox: Integer commitment and arguments

In this section, we define integer commitment schemes (ICS) in Section 4.1 and present a generic construction based on polynomial commitment schemes (PCS). Building on this ICS, we then develop a collection of useful arguments over integers in Section 4.2. These arguments serve as key building blocks for our relaxed mod-PC (see Section 5) construction and may be of independent interest.

4.1 Integer Commitment Schemes (ICS)

We begin by defining ICS, which are commitment schemes whose message space consists of t positive B-bit integers, i.e., $\mathfrak{M} = \mathbb{Z}_B^{+t}$. We then present a construction of a family of ICS based on polynomial commitment schemes (PCS). For concreteness and comparison, we instantiate our construction using the state-of-the-art PCS TensorSwitch (Blaze) [11]; however, the construction is compatible with any standard PCS.

Definition 26. *An integer commitment scheme* ICS *is a tuple of PPT algorithms* $\mathsf{ICS} = (\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$

1. $\mathtt{Setup}(1^\lambda, B, t) \to \mathsf{pp}$ *: On input a security parameter* 1^λ*, arity* $t := 2^\tau$ *and a the maximum bit length supported* $B := 2^\beta$ *generates public parameters* pp*.*
2. $\mathtt{Commit}(\mathsf{pp}; \boldsymbol{z}) \to (\mathsf{cm}; \mathsf{opn})$ *: On input the public parameters* pp *and a message* $\boldsymbol{z} \in \mathbb{Z}_B^{+t}$*, outputs a commitment* cm *and an opening* opn*.*
3. $\mathtt{Open}(\mathsf{pp}, \mathsf{cm}, \boldsymbol{z}, \mathsf{opn}) \to \{0,1\}$ *: On input the public parameter* pp *, a commitment* cm*, a message* $\boldsymbol{z}$ *and an opening* opn *, verifies the opening of* cm *to* $\boldsymbol{z}$ *using the opening* opn*.*

Next, we present our construction of ICS based on PCS. The main idea is to represent the integer bits as evaluations of a multilinear polynomial on the boolean hypercube and then commit to the polynomial using a multilinear PCS.

Construction 1: Integer Commitment

$\mathsf{ICS}.\mathtt{Setup}(1^\lambda, t, B) \to (\mathsf{ck}, \mathsf{vk})$

1. Update B and t to their next powers of two, i.e., $B := 2^\beta$ and $t := 2^\tau$.
2. Invoke $(\mathsf{ck}_{\mathsf{PCS}}, \mathsf{vk}_{\mathsf{PCS}}) := \mathsf{PCS}.\mathtt{Setup}(1^\lambda, Bt)$.
3. Output $\mathsf{ck}_{\mathsf{ICS}} := (B, t, \mathsf{ck}_{\mathsf{PCS}})$ and $\mathsf{vk}_{\mathsf{ICS}} := (B, t, \mathsf{vk}_{\mathsf{PCS}})$.

$\mathsf{ICS}.\mathtt{Commit}(\boldsymbol{z}, \mathsf{ck}) \to (\mathsf{cm}; \mathsf{opn})$

1. Parse ck as $\mathsf{ck}_{\mathsf{PCS}}$ and extract $\mathbb{F}_p$ from $\mathsf{ck}_{\mathsf{PCS}}$.
2. For $i \in (t]$, Expand the integer $\boldsymbol{z}[i]$ into its binary representation $\hat{\boldsymbol{z}}_i = [\hat{\boldsymbol{z}}_{i,0}, \hat{\boldsymbol{z}}_{i,1}, \ldots, \hat{\boldsymbol{z}}_{i,B-1}] \in \{0,1\}^B$.
3. Concatenate the binary representations $\hat{\boldsymbol{z}} := [\hat{\boldsymbol{z}}_1 || \hat{\boldsymbol{z}}_2 || \ldots, || \hat{\boldsymbol{z}}_t] \in \{0,1\}^{tB}$.
4. Compute the multilinear extension $\widetilde{\hat{z}} \in \mathbb{F}_{p,\beta+\tau}^{\mathsf{ind}\leq 1}$ of the vector $\hat{\boldsymbol{z}}$.
5. Compute the polynomial commitment $(\mathsf{cm}_{\mathsf{PCS}}; \mathsf{opn}_{\mathsf{PCS}}) := \mathsf{PCS}.\mathtt{Commit}(\mathsf{ck}_{\mathsf{PCS}}, \widetilde{\hat{z}})$.
6. Set $\mathsf{cm} := \mathsf{cm}_{\mathsf{PCS}}$ and $\mathsf{opn} := \mathsf{opn}_{\mathsf{PCS}}$, then output $(\mathsf{cm}; \mathsf{opn})$.

$\mathsf{ICS}.\mathtt{Open}(\mathsf{ck}, \mathsf{cm}, \boldsymbol{z}, \mathsf{opn}) \to b$

1. Parse ck as $\mathsf{ck}_{\mathsf{PCS}}$, opn as $\mathsf{opn}_{\mathsf{PCS}}$, and cm as $\mathsf{cm}_{\mathsf{PCS}}$.
2. For $i \in (t]$, Expand the integer $\boldsymbol{z}[i]$ into its binary representation $\hat{\boldsymbol{z}}_i = [\hat{\boldsymbol{z}}_{i,0}, \hat{\boldsymbol{z}}_{i,1}, \ldots, \hat{\boldsymbol{z}}_{i,B-1}] \in \{0,1\}^B$.
3. Concatenate the binary representations $\hat{\boldsymbol{z}} := [\hat{\boldsymbol{z}}_1 || \hat{\boldsymbol{z}}_2 || \ldots, || \hat{\boldsymbol{z}}_t] \in \{0,1\}^{tB}$.
4. Compute the multilinear extension $\widetilde{\hat{z}} \in \mathbb{F}_{p,\beta+\tau}^{\mathsf{ind}\leq 1}$ of the vector $\hat{\boldsymbol{z}}$.
5. Invoke $b_{\mathsf{PCS}} := \mathsf{PCS}.\mathtt{Open}(\mathsf{ck}_{\mathsf{PCS}}, \mathsf{cm}_{\mathsf{PCS}}, \widetilde{\hat{z}}, \mathsf{opn}_{\mathsf{PCS}})$ and output b_{PCS}.

Theorem 3. *Let* PCS *be a standard polynomial commitment scheme (see Definition 24), then Construction 1 is an integer commitment scheme according to Definition 26 with the following properties:*

Setup Time	*Ck Size*	*Vk Size*	*Com. Time*	*Com. Size*	*Opn. Time*
$T(\mathsf{PCS.Setup})$	$\lvert\mathsf{ck}_{\mathsf{PCS}}\rvert$	$\lvert\mathsf{vk}_{\mathsf{PCS}}\rvert$	$T(\mathsf{PCS.Commit})$	$\lvert\mathsf{cm}_{\mathsf{PCS}}\rvert$	$T(\mathsf{PCS.Open})$

Moreover, instantiating Construction 1 with the state-of-the-art PCS Tensor-Switch (Blaze) [11] yields a transparent integer commitment scheme under collision-resistant hashing with the following optimal properties:

Setup Time	*Ck Size*	*Vk Size*	*Com. Time*	*Com. Size*	*Opn. Time*
$O(1)$	$O(1)$	$O(1)$	$O(Bt)$	$O(1)$	$O(Bt)$

4.2 A Suite of Primitive Integer Arguments

In this section, we develop a suite of arguments over integers that serve as building blocks for our relaxed mod-PC (Section 5) construction.

Idealized presentation of the protocols. Our protocols are tailored to the family of integer commitment schemes defined in Construction 1, which represent committed integers using multilinear polynomials. For clarity and ease of exposition, rather than presenting the protocols as full arguments of knowledge over committed integers for a relation $\mathfrak{R}_X$, we describe them for a dual relation $\mathfrak{R}_{X\text{-}Oracle}$ defined as follows:

Definition 27 (Dual Integer Indexed Relations). *Let* ICS *be the integer commitment scheme for* $\mathbb{Z}_B$ *as defined in Construction 1. Let* $\mathfrak{R}_X^{\mathsf{ICS}}$ *be an indexed relation (defined in Definition 13) consisting of tuples* $(\mathsf{i}, \mathsf{x}, \mathsf{y}, \mathsf{w})$ *and* $\mathfrak{R}_{X\text{-}Oracle}^{\mathsf{ICS}}$ *be an indexed oracle relation (defined in Definition 1) consisting of tuples* $(\mathsf{i}_o, \mathsf{x}_o, \mathsf{y}_o, \mathsf{w}_o)$*. We say that* $\mathfrak{R}_{X\text{-}Oracle}^{\mathsf{ICS}}$ *and* $\mathfrak{R}_X^{\mathsf{ICS}}$ *are* dual *relations if*

- $\mathsf{i} = \emptyset$ *and* $\mathsf{i}_o = (\mathsf{ck}, \mathsf{vk})$.
- *For all* $x \in \mathbb{Z}_B$*, if* $x \in \mathsf{x}$ *(respectively,* w*), then* $\widetilde{x} \in \mathsf{x}_o$ *(respectively,* w_o*) where* $\widetilde{x}$ *is the multilinear extension of* $\boldsymbol{x}$*, the binary decomposition of* x*.*
- *For all* $x \in \mathbb{Z}_B$*, let* $(\mathsf{cm}_x; \mathsf{opn}_x) = \mathsf{ICS.Commit}(x, \mathsf{ck})$*, then if* $\mathsf{cm}_x \in \mathsf{y}$*, then* $[\![\widetilde{x}]\!] \in \mathsf{y}_o$ *where* $\widetilde{x}$ *is the multilinear extension of* $\boldsymbol{x}$*, the binary decomposition of* x*.*
- *For all* $x \notin \mathbb{Z}_B$*, if* $x \in \mathsf{x}$ *(respectively,* y *and* w*), then* $x \in \mathsf{x}_o$ *(respectively,* y_o *and* w_o*).*

in an idealized model where the prover has oracle access to the multilinear polynomials representing the committed integers. These idealized protocols can be compiled into arguments of knowledge for the committed integers using the general recipe described in Construction 7.

Protocols as reductions. Since our protocols invoke several subprotocols, we present them as reductions to other relations. Concretely, instead of using Polynomial Interactive Oracle Proofs (PIOPs, see Definition 9), we work with Polynomial Interactive Oracle Reductions (PIORs, see Definition 3).

Purpose of the protocols. Ultimately, we require two main protocols from this section: (a) the range-check protocol (Section 4.2.2), which verifies that a committed integer lies within a specified range, and (b) the integer division-check protocol (Section 4.2.5), which verifies the correctness of the quotient and remainder for a committed dividend and divisor. To construct these protocols, we introduce several intermediate building blocks.

Batched Representation. We present all our protocols in a batched form, where the prover proves the correctness of t instances of the relation in parallel. For example, in the integer addition protocol, the prover proves the correctness of t addition instances in parallel.

4.2.1 Integer Addition

Definition 28 (Integer Addition Oracle Relation). *The oracle relation* $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$ *is the set of all triples of the form* $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (\varnothing, (\llbracket\widetilde{a}\rrbracket, \llbracket\widetilde{b}\rrbracket, \llbracket\widetilde{c}\rrbracket), (\widetilde{a}, \widetilde{b}, \widetilde{c}))$, *where* $\widetilde{a}, \widetilde{b}, \widetilde{c} \in \mathbb{F}_{p,\beta+\tau}^{\mathsf{ind}\leq 1}$ *have evaluation vectors* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \{0,1\}^{Bt}$ *such that* $B = 2^\beta$ *and* $t = 2^\tau$, *and they are of the form*

$$\boldsymbol{a} = [\boldsymbol{a}_0 || \ldots || \boldsymbol{a}_{t-1}], \quad \boldsymbol{b} = [\boldsymbol{b}_0 || \ldots || \boldsymbol{b}_{t-1}], \quad \boldsymbol{c} = [\boldsymbol{c}_0 || \ldots || \boldsymbol{c}_{t-1}]$$

Moreover, for $i \in [t-1)$, *each of the vectors* $\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i \in \{0,1\}^{2B}$ *represent the bit decomposition of integers* a_i, b_i, c_i *respectively as*

$$a_i = \sum_{j\in[B)} \boldsymbol{a}_i[j]2^j, \quad b_i = \sum_{j\in[B)} \boldsymbol{b}_i[j]2^j, \quad c_i = \sum_{j\in[2B)} \boldsymbol{c}_i[j]2^j$$

and they satisfy the addition relation; i.e. $c_i = a_i + b_i$

Moreover, $\mathfrak{R}_{\mathsf{Add}}$ *denotes the dual relation of* $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$ *according to Definition 27.*

Next, we define the corresponding PIOR for the integer addition relation. The key idea is that if $c_i = a_i + b_i$, then there exists a binary vector $\boldsymbol{k}_i$ (the carry vector) such that the bit decompositions of $\boldsymbol{a}_i$, $\boldsymbol{b}_i$, and $\boldsymbol{c}_i$ satisfy the relation $a_i[j] + b_i[j] + k_i[j] = c_i[j] + 2k_i[j+1]$ over $\mathbb{F}_p$. Let $\mu := \beta + \tau$ and $N := 2^\mu$. The prover supplies an oracle for the carry oracle $\llbracket\widetilde{k}\rrbracket$ and a *separate* oracle $\llbracket\widetilde{k'}\rrbracket$ for the MLE of the evaluation vector obtained by cyclically shifting the carry hypercube vector by one index in lexicographic order on $\{0,1\}^\mu$ (the same shift used by the ripple-carry recurrence). The verifier tests the ripple identity at a single random point using $\widetilde{k}(\boldsymbol{r})$ and $\widetilde{k'}(\boldsymbol{r})$, and the protocol also outputs an instance–witness pair for the cyclic rotation relation (Definition 38), to be enforced by the rotation-check PIOR (PIOR 8) once that subroutine is specified.

PIOR 1: $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}} \to \mathfrak{R}_{\mathsf{Decision}} \times \mathfrak{R}_{\mathsf{Bin}}^5 \times \mathfrak{R}_{\mathsf{rot}-\mathsf{Oracle}}$

Parse: $\mathsf{x} = \varnothing$, $\mathsf{y} = (\llbracket\widetilde{a}\rrbracket, \llbracket\widetilde{b}\rrbracket, \llbracket\widetilde{c}\rrbracket)$, $\mathsf{w} = (\widetilde{a}, \widetilde{b}, \widetilde{c})$

1. $\mathcal{P}$ computes the carry vector $\boldsymbol{k}$ which is the concatenatation of $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_t$ where each $\boldsymbol{k}_i$ is the carry vector for the i-th addition instance. Concretely, for $i \in (t]$ and $j \in [n-1)$, the carry bit $\boldsymbol{k}_i[j]$ is defined by the following

recursive relation:

$$2\,k_i[j+1] = a_i[j] + b_i[j] + k_i[j] - c_i[j]$$

where we set $k_i[0] = 0$ for all $i \in (t]$. Let $\tilde{\boldsymbol{k}}$ be the length-N vector obtained by cyclically shifting the evaluation vector of $\widetilde{k}$ by one position in lexicographic hypercube order. $\mathcal{P}$ sends the oracles $[\![\widetilde{k}]\!]$ and $[\![\widetilde{k'}]\!]$ to $\mathcal{V}$.

2. $\mathcal{V}$ samples a random vector $\boldsymbol{r} \in \mathbb{F}_p^\mu$ and queries the oracles $[\![\widetilde{a}]\!]$, $[\![\widetilde{b}]\!]$, $[\![\widetilde{c}]\!]$, $[\![\widetilde{k}]\!]$, and $[\![\widetilde{k'}]\!]$ at the point $\boldsymbol{r}$ to obtain the values $\widetilde{a}(\boldsymbol{r})$, $\widetilde{b}(\boldsymbol{r})$, $\widetilde{c}(\boldsymbol{r})$, $\widetilde{k}(\boldsymbol{r})$, and $\widetilde{k'}(\boldsymbol{r})$.
3. To check if the addition relation holds for the queried point, $\mathcal{V}$ checks:

$$\widetilde{a}(\boldsymbol{r}) + \widetilde{b}(\boldsymbol{r}) + \widetilde{k}(\boldsymbol{r}) \stackrel{?}{=} \widetilde{c}(\boldsymbol{r}) + 2 \cdot \widetilde{k'}(\boldsymbol{r})$$

$\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Decision}} := \varnothing$ and $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Decision}}, \mathsf{y}'_{\mathsf{Decision}}) := (\mathsf{accept}, \varnothing)$ for the decision relation $\mathfrak{R}_{\mathsf{Decision}}$ which checks the above polynomial identity.

4. To verify that the committed vectors are binary, $\mathcal{P}$ outputs the following witnesses for the binary relation (Definition 39): $\mathsf{w}'_{\mathsf{Bin},a} := \widetilde{a}$, $\mathsf{w}'_{\mathsf{Bin},b} := \widetilde{b}$, $\mathsf{w}'_{\mathsf{Bin},c} := \widetilde{c}$, $\mathsf{w}'_{\mathsf{Bin},k} := \widetilde{k}$, and $\mathsf{w}'_{\mathsf{Bin},k'} := \widetilde{k'}$. Correspondingly, $\mathcal{V}$ outputs the associated instances and oracle instances $\mathsf{x}'_{\mathsf{Bin},a} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},a} := [\![\widetilde{a}]\!]$; $\mathsf{x}'_{\mathsf{Bin},b} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},b} := [\![\widetilde{b}]\!]$; $\mathsf{x}'_{\mathsf{Bin},c} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},c} := [\![\widetilde{c}]\!]$; $\mathsf{x}'_{\mathsf{Bin},k} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},k} := [\![\widetilde{k}]\!]$; and $\mathsf{x}'_{\mathsf{Bin},k'} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},k'} := [\![\widetilde{k'}]\!]$.
5. To verify that $\widetilde{k'}$ is the cyclic rotation of $\widetilde{k}$ by one position, $\mathcal{P}$ outputs the witness $\mathsf{w}'_{\mathsf{rot}} := (\widetilde{k}, \widetilde{k'})$ for the rotation oracle relation $\mathfrak{R}_{\mathsf{rot}-\mathsf{Oracle}}$ (Definition 38) with $\sigma = 1$. Correspondingly, $\mathcal{V}$ outputs the associated instance and oracle instance $\mathsf{x}'_{\mathsf{rot}} := 1$ and $\mathsf{y}'_{\mathsf{rot}} := ([\![\widetilde{k}]\!], [\![\widetilde{k'}]\!])$.

Theorem 4. *PIOR 1 has perfect completeness and round-by-round soundness error at most μ/p with the following efficiency measures:*

rounds	*queries*	*oracle size*	*prover time*	*verifier time*
1	5	2^μ	$O(2^\mu)$	$O(\mu)$

Proof. Deferred to appendix A.4.1. □

4.2.2 Integer Range

Definition 29 (Integer Range Oracle Relation). *The oracle relation $\mathfrak{R}_{\mathsf{Range\text{-}Oracle}}$ is the set of all triples of the form $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = ((m), ([\![\widetilde{a}]\!]), (\widetilde{a}))$, where $m \in \mathbb{Z}_B^+$ is the range upper bound and $\widetilde{a} \in \mathbb{F}_{p,\beta+\tau}^{\mathsf{ind}\leq 1}$ has an evaluation vector $\boldsymbol{a} \in \{0,1\}^{Bt}$ such that $B = 2^\beta$ and $t = 2^\tau$, and it's of the form $\boldsymbol{a} = [\boldsymbol{a}_0 || \ldots || \boldsymbol{a}_{t-1}]$. Moreover, for $i \in [t-1)$, the vector $\boldsymbol{a}_i \in \{0,1\}^{2B}$ represents the bit decomposition of an integer a_i as $a_i = \sum_{j \in [B)} \boldsymbol{a}_i[j] 2^j$ and it satisfied the range bound; i.e. $0 \leq a_i \leq m$.*

Moreover, $\mathfrak{R}_{\mathsf{Range}}$ denotes the dual relation of $\mathfrak{R}_{\mathsf{Range\text{-}Oracle}}$ according to Definition 27.

PIOR 2: $\mathfrak{R}_{\mathsf{Range\text{-}Oracle}} \rightarrow \mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$

Parse: $\mathsf{x} = m$, $\mathsf{y} = [\![\widetilde{a}]\!]$, $\mathsf{w} = \widetilde{a}$

1. $\mathcal{P}$ computes $\widetilde{b}$ such that the underlying integers satisfy $a_i + b_i = m$ for all i, and sends the oracle $[\![\widetilde{b}]\!]$ to $\mathcal{V}$.
2. $\mathcal{P}$ outputs a witness for $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$ corresponding to the tuple (a, b, c) with $c_i = m$ for all i, namely $\mathsf{w}'_{\mathrm{Add}} := (\widetilde{a}, \widetilde{b}, \widetilde{c})$, where $\widetilde{c}$ is the multilinear extension of the evaluation vector obtained by concatenating, for each batch index i, the $2B$-bit binary expansion of m (least significant bit first, padded with leading zeros), matching the layout of $\boldsymbol{a}$ in Definition 29.
3. $\mathcal{V}$ outputs the corresponding instance and oracle instance for $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$, namely $(\mathsf{x}'_{\mathrm{Add}}, \mathsf{y}'_{\mathrm{Add}}) := (\varnothing, ([\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{c}]\!]))$, using the received oracle $[\![\widetilde{b}]\!]$.

Theorem 5. *PIOR 2 has perfect completeness and round-by-round soundness error* 0 *with the following efficiency measures:*

rounds	*queries*	*oracle size*	*prover time*	*verifier time*
1	5	2	$O(2^\mu)$	$O(1)$

Proof. Deferred to appendix A.4.2. □

Next, we present the Convolution-Check PIOR, the key building block for the integer multiplication argument: it verifies that a committed vector is the convolution of two other committed vectors.

4.2.3 Convolution Check

In this section, we formalize the convolution relation between three committed vectors and present a PIOR that reduces it to the Sumcheck relation (Definition 42) and the Hadamard relation (Definition 40). We will use this Convolution-Check PIOR as a key building block for the integer multiplication (see PIOR 4) and division check (see PIOR 5) arguments.

Remark 1 (From an NTT check to a convolution check). An earlier version of this work verified the schoolbook convolution through a *number-theoretic transform* (NTT): the prover committed the NTTs of the two operands and of the digit vector and proved a Hadamard (pointwise-product) identity in the evaluation domain, which in turn required an auxiliary Lagrange check together with three NTT checks. We thank an anonymous reviewer for pointing out that this machinery is unnecessary. Convolution is exactly univariate polynomial multiplication, so it can be verified by a *single* random-point identity $f_s = f_a \cdot f_b$ via the Schwartz–Zippel lemma (Lemma 1). The resulting convolution check replaces the NTT check, the Lagrange check, and the Hadamard step at once; it commits asymptotically smaller oracles, drops the Lagrange machinery entirely, and—because it evaluates at a single field point rather than over a root-of-unity domain—removes the requirement that $\mathbb{F}_p$ contain a primitive B-th root of unity (a 2-smooth subgroup). The only remaining field requirement is on the *characteristic*: we need $p > B$ so that the carry-based integer arithmetic (carries and convolution digits are bounded by $O(B)$) is represented faithfully, and so that the Schwartz–Zippel degree bound applies. This is a condition on the characteristic alone—no smoothness or FFT-friendliness is needed—and it is met by any

large-characteristic field while excluding small-characteristic ones such as binary fields.

Definition 30 (Convolution Relation). *The oracle relation* $\mathfrak{R}_{\mathsf{Conv\text{-}Oracle}}$ *is the set of all triples of the form* $(\mathsf{x},\mathsf{y},\mathsf{w}) = (\varnothing, ([\![\widetilde{a}]\!],[\![\widetilde{b}]\!],[\![\widetilde{s}]\!]), (\widetilde{a},\widetilde{b},\widetilde{s}))$ *where* $\widetilde{a},\widetilde{b},\widetilde{s} \in \mathbb{F}_{p,\beta+\tau}^{\mathsf{ind}\leq 1}$ *(with* $B = 2^\beta$, $t = 2^\tau$*) have evaluation vectors of the batched form* $\boldsymbol{a} = [\boldsymbol{a}_0 \| \ldots \| \boldsymbol{a}_{t-1}]$, $\boldsymbol{b} = [\boldsymbol{b}_0 \| \ldots \| \boldsymbol{b}_{t-1}]$, *and* $\boldsymbol{s} = [\boldsymbol{s}_0 \| \ldots \| \boldsymbol{s}_{t-1}]$ *with* $\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{s}_i \in \mathbb{F}_p^B$, *and, writing* $f_{a_i}(Y) := \sum_{k=0}^{B-1} \boldsymbol{a}_i[k]\, Y^k$ *and analogously* f_{b_i}, f_{s_i}, *satisfy the convolution relation per batch:*

$$f_{s_i} = f_{a_i} \cdot f_{b_i} \qquad \text{for all } i \in [t-1),$$

equivalently $\boldsymbol{s}_i[\ell] = \sum_{i'+j'=\ell} \boldsymbol{a}_i[i']\, \boldsymbol{b}_i[j']$ *for all* $\ell \in [B-1)$. *(This requires* $\deg f_{a_i} + \deg f_{b_i} < B$, *i.e. the convolution fits in* B *coefficients, which holds for the overflow-bounded operands of PIOR 4.)*

Moreover, $\mathfrak{R}_{\mathsf{Conv}}$ *denotes the dual relation of* $\mathfrak{R}_{\mathsf{Conv\text{-}Oracle}}$ *according to Definition 27.*

The core idea is that the per-batch convolution $\boldsymbol{s}_i = \boldsymbol{a}_i \star \boldsymbol{b}_i$ is equivalent to the univariate polynomial identity $f_{s_i} = f_{a_i} \cdot f_{b_i}$, which by the Schwartz–Zippel lemma can be checked at a single random point r: the prover commits the per-batch evaluations of the three polynomials at r, the sumcheck relation binds these evaluations to the committed vectors, and the Hadamard relation checks the per-batch products $f_{s_i}(r) = f_{a_i}(r) \cdot f_{b_i}(r)$.

PIOR 3: $\mathfrak{R}_{\mathsf{Conv\text{-}Oracle}} \to \mathfrak{R}_{\mathsf{sum}}^3 \times \mathfrak{R}_{\mathsf{Had}}$

Parse: $\mathsf{x} = \varnothing$, $\mathsf{y} = ([\![\widetilde{a}]\!],[\![\widetilde{b}]\!],[\![\widetilde{s}]\!])$, $\mathsf{w} = (\widetilde{a},\widetilde{b},\widetilde{s})$

1. $\mathcal{V}$ samples a random element $r \xleftarrow{\$} \mathbb{F}_p$ and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ and $\mathcal{V}$ define the succinctly representable MLE of the monomial vector $\mathbf{mon}_r = [r^0, r^1, \ldots, r^{B-1}]$,

$$\widetilde{u}(x_0,\ldots,x_{\beta-1}) = \prod_{j=0}^{\beta-1} \left((1-x_j) + x_j r^{2^j}\right) \in p_\beta^{\mathsf{ind}\leq 1}.$$

3. $\mathcal{P}$ computes the per-batch evaluations of the three polynomials at r, i.e. the vectors $\hat{\boldsymbol{a}}, \hat{\boldsymbol{b}}, \hat{\boldsymbol{s}} \in \mathbb{F}_p^t$ with $\hat{a}[i] = f_{a_i}(r)$, $\hat{b}[i] = f_{b_i}(r)$, and $\hat{s}[i] = f_{s_i}(r)$, and sends the oracles $[\![\widetilde{\hat{a}}]\!], [\![\widetilde{\hat{b}}]\!], [\![\widetilde{\hat{s}}]\!]$ to $\mathcal{V}$, where $\widetilde{\hat{a}}, \widetilde{\hat{b}}, \widetilde{\hat{s}} \in \mathbb{F}_{p,\tau}^{\mathsf{ind}\leq 1}$ are the multilinear extensions of these vectors.
4. $\mathcal{V}$ samples $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{F}_p^\tau$ and queries $[\![\widetilde{\hat{a}}]\!], [\![\widetilde{\hat{b}}]\!], [\![\widetilde{\hat{s}}]\!]$ at $\boldsymbol{\rho}$ to obtain $\widetilde{\hat{a}}(\boldsymbol{\rho}), \widetilde{\hat{b}}(\boldsymbol{\rho}), \widetilde{\hat{s}}(\boldsymbol{\rho})$. To verify that these oracles hold the true evaluations at r, $\mathcal{P}$ outputs three witnesses for the Sumcheck relation (Definition 42), $\mathsf{w}'_{\mathsf{Sum},a} := g_a$, $\mathsf{w}'_{\mathsf{Sum},b} := g_b$, $\mathsf{w}'_{\mathsf{Sum},s} := g_s$, while $\mathcal{V}$ outputs the corresponding instances $(\mathsf{x}'_{\mathsf{Sum},a}, \mathsf{y}'_{\mathsf{Sum},a}) := (\widetilde{\hat{a}}(\boldsymbol{\rho}), [\![g_a]\!])$ and analogously for b, s, where the virtual polynomials over $\boldsymbol{x} \in \{0,1\}^\beta$ are

$$g_a(\boldsymbol{x}) := \widetilde{u}(\boldsymbol{x})\, \widetilde{a}(\boldsymbol{x},\boldsymbol{\rho}), \quad g_b(\boldsymbol{x}) := \widetilde{u}(\boldsymbol{x})\, \widetilde{b}(\boldsymbol{x},\boldsymbol{\rho}), \quad g_s(\boldsymbol{x}) := \widetilde{u}(\boldsymbol{x})\, \widetilde{s}(\boldsymbol{x},\boldsymbol{\rho}).$$

Indeed $\sum_{\boldsymbol{x}\in\{0,1\}^\beta} g_a(\boldsymbol{x}) = \widetilde{a}(\boldsymbol{\rho})$ whenever $\widetilde{a}$ holds the true evaluations, and analogously for b, s.

5. To verify the per-batch products, $\mathcal{P}$ outputs a witness for the Hadamard relation (Definition 40), $\mathsf{w}'_{\mathsf{Had}} := (\widetilde{\hat{a}}, \widetilde{\hat{b}}, \widetilde{\hat{s}}, 0)$ with zero linear term, while $\mathcal{V}$ outputs the corresponding instance $(\mathsf{x}'_{\mathsf{Had}}, \mathsf{y}'_{\mathsf{Had}}) := (\varnothing, ([\![\widetilde{\hat{a}}]\!], [\![\widetilde{\hat{b}}]\!], [\![\widetilde{\hat{s}}]\!], [\![0]\!]))$, which checks $\hat{\boldsymbol{s}} = \hat{\boldsymbol{a}} \circ \hat{\boldsymbol{b}}$, i.e. $\hat{s}[i] = \hat{a}[i] \cdot \hat{b}[i]$ for every batch i.

Theorem 6. *PIOR 3 has perfect completeness and round-by-round soundness error at most* $\max\{tB/p,\ \mu/p\}$ *for* $\mu = \beta + \tau$*, with the following efficiency measures:*

rounds	*queries*	*oracle size*	*prover time*	*verifier time*
2	3	$3t$	$O(Bt)$	$O(\mu)$

Proof. Deferred to appendix A.4.3. □

4.2.4 Integer Multiplication

In this section, we formalize the integer multiplication relation, which checks if three committed integers satisfy the multiplication relation, and then present a PIOR for this relation. We will use this Integer Multiplication PIOR as a building block for the Division-Check PIOP (see PIOR 5) and Masdak (see PIOR 6).

Definition 31 (Integer Multiplication Oracle Relation). *The oracle relation* $\mathfrak{R}_{\mathsf{Mul\text{-}Oracle}}$ *is the set of all triples of the form* $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (\varnothing, ([\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{c}]\!]), (\widetilde{a}, \widetilde{b}, \widetilde{c}))$ *where* $\widetilde{a}, \widetilde{b}, \widetilde{c} \in \mathbb{F}^{\mathsf{ind}\le 1}_{p,\beta+\tau}$ *have evaluation vectors* $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \{0,1\}^{Bt}$ *such that* $B = 2^\beta$ *and* $t = 2^\tau$*, and they are of the form*

$$\boldsymbol{a} = [\boldsymbol{a}_0 || \dots || \boldsymbol{a}_{t-1}], \quad \boldsymbol{b} = [\boldsymbol{b}_0 || \dots || \boldsymbol{b}_{t-1}], \quad \boldsymbol{c} = [\boldsymbol{c}_0 || \dots || \boldsymbol{c}_{t-1}]$$

Moreover, for $i \in [t-1)$*, each of the vectors* $\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i \in \{0,1\}^{2B}$ *represent the bit decomposition of integers* a_i, b_i, c_i *respectively as*

$$a_i = \sum_{j\in[B)} \boldsymbol{a}_i[j]2^j, \quad b_i = \sum_{j\in[B)} \boldsymbol{b}_i[j]2^j, \quad c_i = \sum_{j\in[2B)} \boldsymbol{c}_i[j]2^j$$

and they satisfy the multiplication relation; i.e. $c_i = a_i b_i$

Moreover, $\mathfrak{R}_{\mathsf{Mul}}$ *denotes the dual relation of* $\mathfrak{R}_{\mathsf{Mul\text{-}Oracle}}$ *according to Definition 27.*

A PIOR for the batched integer multiplication oracle relation is presented below. Refer to Section 2.3.1 for an informal overview of the main ideas.

PIOR 4: $\mathfrak{R}_{\mathsf{Mul\text{-}Oracle}} \to \mathfrak{R}^2_{\mathsf{Decision}} \times \mathfrak{R}_{\mathsf{Conv\text{-}Oracle}} \times \mathfrak{R}^{3+w}_{\mathsf{Bin}} \times \mathfrak{R}_{\mathsf{Zero}} \times \mathfrak{R}_{\mathsf{Zero}} \times \mathfrak{R}_{\mathsf{rot}-\mathsf{Oracle}}$

Parse: $\mathsf{x} = \varnothing$, $\mathsf{y} = ([\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{c}]\!])$, $\mathsf{w} = (\widetilde{a}, \widetilde{b}, \widetilde{c})$

1. $\mathcal{P}$ and $\mathcal{V}$ define a mask polynomial $\widetilde{m} \in p^{\mathsf{ind}\le 1}_{\beta+\tau}$ with succinct representation that has an evaluation vector of the form $\boldsymbol{m} = [\boldsymbol{m}' || \dots || \boldsymbol{m}']$ where $\boldsymbol{m}' = [1^{B/2} || 0^{B/2}]$ as follows:

$$\widetilde{m}(x_0, \dots, x_{\beta-1}, y_0, \dots, y_{\tau-1}) = 1 - x_{\beta-1}$$

2. To verify that there is no overflow in the multiplication; i.e. the integers encoded in $\boldsymbol{a}$ and $\boldsymbol{b}$ have bit-length at most $B/2$, $\mathcal{P}$ outputs two witnesses for the Zero Check relation (Definition 41) as $\mathsf{w}'_{\mathsf{Zero},a} := f_a$, $\mathsf{w}'_{\mathsf{Zero},b} := f_b$ while $\mathcal{V}$ outputs the corresponding explicit and implicit instances for the Zero Check relation as $(\mathsf{x}'_{\mathsf{Zero},a}, \mathsf{y}'_{\mathsf{Zero},a}) := (\emptyset, [\![f_a]\!])$ and $(\mathsf{x}'_{\mathsf{Zero},b}, \mathsf{y}'_{\mathsf{Zero},b}) := (\emptyset, [\![f_b]\!])$ where $f_a \in \mathbb{F}_{p^{\beta+\tau}}^{\mathsf{ind}\leq 2}$ and $f_b \in \mathbb{F}_{p^{\beta+\tau}}^{\mathsf{ind}\leq 2}$ are virtual polynomials defined as

$$f_a(\boldsymbol{x}, \boldsymbol{y}) := \widetilde{m}(\boldsymbol{x}, \boldsymbol{y}) \cdot \widetilde{a}(\boldsymbol{x}, \boldsymbol{y}), \quad f_b(\boldsymbol{x}, \boldsymbol{y}) := \widetilde{m}(\boldsymbol{x}, \boldsymbol{y}) \cdot \widetilde{b}(\boldsymbol{x}, \boldsymbol{y})$$

3. $\mathcal{P}$ computes the schoolbook digit vector $\boldsymbol{s}$ from $\boldsymbol{a}$ and $\boldsymbol{b}$ (the convolution of the two bit vectors, indexed consistently with $\boldsymbol{c}$), and the carry vector $\boldsymbol{k}$ by the same integer recurrence as in PIOR 1, i.e. for each batch index $i \in (t]$ and each relevant j,
$$k_i[j+1] = \frac{s_i[j] + k_i[j] - c_i[j]}{2}$$
over $\mathbb{Z}$, with $k_i[0] = 0$. Setting $w := \beta - 1$, the prover also computes the binary decomposition of the carry into w *bit-decomposition vectors*: for $t \in [w-1)$ the vector $\kappa^{(t)} \in \{0,1\}^{Bt}$ holds the t-th bit of each carry, so that $k_i[j] = \sum_{t=0}^{w-1} 2^t \, \kappa_i^{(t)}[j]$ for all i, j. Let $\tilde{\boldsymbol{k}}$ be the length-2^μ vector obtained by cyclically shifting the evaluation vector of $\widetilde{k}$ by one position in lexicographic hypercube order on $\{0,1\}^\mu$ for $\mu = \beta + \tau$. $\mathcal{P}$ sends the oracles $[\![\widetilde{s}]\!]$, $[\![\widetilde{k}]\!]$, $[\![\widetilde{k'}]\!]$, and $\{[\![\widetilde{\kappa^{(t)}}]\!]\}_{t=0}^{w-1}$ to $\mathcal{V}$.
4. To verify that the schoolbook digit vector $\boldsymbol{s}$ is the convolution of the operands $\boldsymbol{a}$ and $\boldsymbol{b}$, $\mathcal{P}$ outputs a witness for the Convolution relation (Definition 30), $\mathsf{w}'_{\circledast} := (\widetilde{a}, \widetilde{b}, \widetilde{s})$, while $\mathcal{V}$ outputs the corresponding instance and oracle instance
$$(\mathsf{x}'_{\circledast}, \mathsf{y}'_{\circledast}) := (\varnothing, ([\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{s}]\!])).$$
By the overflow check of Item 2, the operands have bit-length at most $B/2$, so the convolution fits in B coefficients and $\boldsymbol{s}$ is a genuine (non-cyclic) convolution.
5. Let $\mu := \beta + \tau$. $\mathcal{V}$ samples a random vector $\boldsymbol{r} \in \mathbb{F}_p^\mu$ and queries the oracles $[\![\widetilde{s}]\!]$, $[\![\widetilde{c}]\!]$, $[\![\widetilde{k}]\!]$, and $[\![\widetilde{k'}]\!]$ at the point $\boldsymbol{r}$ to obtain $\widetilde{s}(\boldsymbol{r})$, $\widetilde{c}(\boldsymbol{r})$, $\widetilde{k}(\boldsymbol{r})$, and $\widetilde{k'}(\boldsymbol{r})$. To check the multiplication carry relation at the queried point, $\mathcal{V}$ checks
$$\widetilde{s}(\boldsymbol{r}) + \widetilde{k}(\boldsymbol{r}) \stackrel{?}{=} \widetilde{c}(\boldsymbol{r}) + 2 \cdot \widetilde{k'}(\boldsymbol{r}).$$
$\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Decision}} := \varnothing$ and $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Decision}}, \mathsf{y}'_{\mathsf{Decision}}) := (\mathsf{accept}, \varnothing)$ for the decision relation $\mathfrak{R}_{\mathsf{Decision}}$ which checks the above polynomial identity.
6. To check that the carry oracle is consistent with its committed bit-decomposition vectors, $\mathcal{V}$ queries the oracles $\{[\![\widetilde{\kappa^{(t)}}]\!]\}_{t=0}^{w-1}$ at $\boldsymbol{r}$ and checks the reconstruction identity
$$\widetilde{k}(\boldsymbol{r}) \stackrel{?}{=} \sum_{t=0}^{w-1} 2^t \, \widetilde{\kappa^{(t)}}(\boldsymbol{r}).$$
$\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Decision},k} := \varnothing$ and $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Decision},k}, \mathsf{y}'_{\mathsf{Decision},k}) := (\mathsf{accept}, \varnothing)$ for the decision relation $\mathfrak{R}_{\mathsf{Decision}}$ which checks this polynomial identity.

7. To verify that the committed vectors are binary and therefore valid bit decompositions of integers, $\mathcal{P}$ outputs the following witnesses for the binary relation (Definition 39): $\mathsf{w}'_{\mathsf{Bin},a} := \widetilde{a}$, $\mathsf{w}'_{\mathsf{Bin},b} := \widetilde{b}$, $\mathsf{w}'_{\mathsf{Bin},c} := \widetilde{c}$, and, for each $t \in [w-1)$, the carry bit-decomposition vector $\mathsf{w}'_{\mathsf{Bin},\kappa^{(t)}} := \widetilde{\kappa^{(t)}}$. Correspondingly, $\mathcal{V}$ outputs the associated instances and oracle instances $\mathsf{x}'_{\mathsf{Bin},a} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},a} := [\![\widetilde{a}]\!]$; $\mathsf{x}'_{\mathsf{Bin},b} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},b} := [\![\widetilde{b}]\!]$; $\mathsf{x}'_{\mathsf{Bin},c} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},c} := [\![\widetilde{c}]\!]$; and, for each $t \in [w-1)$, $\mathsf{x}'_{\mathsf{Bin},\kappa^{(t)}} := \varnothing$, $\mathsf{y}'_{\mathsf{Bin},\kappa^{(t)}} := [\![\widetilde{\kappa^{(t)}}]\!]$. Note that the carry value oracle $[\![\widetilde{k}]\!]$ and its rotation $[\![\widetilde{k'}]\!]$ are *not* binary-checked; their entries are bounded via the bit-decomposition reconstruction of the previous step, and $[\![\widetilde{k'}]\!]$ inherits the bound from $[\![\widetilde{k}]\!]$ through the cyclic-rotation relation below.
8. $\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{rot},\mathsf{mul}} := (\widetilde{k}, \widetilde{k'})$. $\mathcal{V}$ sets $\mathsf{x}'_{\mathsf{rot},\mathsf{mul}} := 1$; for $\mathfrak{R}_{\mathsf{rot}-\mathsf{Oracle}}$ (Definition 38) with $\sigma = 1$, it sets $\mathsf{y}'_{\mathsf{rot},\mathsf{mul}}$ to the ordered pair of $[\![\widetilde{k}]\!]$ and $[\![\widetilde{k'}]\!]$.

Unlike the addition carry of PIOR 1, the schoolbook digits $s_i[j]$ are not binary, so the carries are not binary either. The overflow constraint (Item 2) keeps each digit in $[0, B/2]$, and a short induction (Lemma 3, proved together with the round-by-round soundness analysis in appendix A.4.4) shows the carries remain in $[0, B/2) = [0, 2^w)$ with $w = \beta - 1$. This is exactly why the protocol bounds the carry by committing and binary-checking its w bit-decomposition vectors (steps above), rather than checking that the carry is a single bit as in integer addition.

Theorem 7. *PIOR 4 has perfect completeness and round-by-round soundness error at most* $\max\{\mu/p, tB/p\}$*, where* $\mu = \beta + \tau$*, with the following efficiency measures:*

rounds	*queries*	*oracle size*	*prover time*	*verifier time*
1	$5 + w$	$O(w\, 2^{\mu})$	$O(Bt \log B)$	$O(\mu + w)$

Here $w = \beta - 1$*, so the additional carry bit-decomposition vectors contribute only a logarithmic (in* B*) overhead over the operand oracles. Replacing the earlier NTT/Hadamard convolution check by the single convolution check of PIOR 3 removes the three NTT oracles (each of size* 2^{μ}*) in favour of three evaluation oracles of size* t*, and drops the field requirement that* $\mathbb{F}_p$ *admit a primitive* B*-th root of unity.*

Proof. Deferred to appendix A.4.4. □

4.2.5 Division Check

In this section, we define the integer division relation and present a PIOR that reduces it to integer multiplication (PIOR 4) and integer addition (PIOR 1).

Definition 32 (Division Oracle Relation). *The oracle relation* $\mathfrak{R}_{\mathsf{Div}\text{-}\mathsf{Oracle}}$ *is the set of all tuples of the form* $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (\gamma, ([\![\widetilde{v}]\!], [\![\widetilde{q}]\!], [\![\widetilde{\ell}]\!], [\![\widetilde{r}]\!]), (\widetilde{v}, \widetilde{q}, \widetilde{\ell}, \widetilde{r}))$ *where* $\widetilde{v}, \widetilde{q}, \widetilde{\ell}, \widetilde{r} \in \mathbb{F}_{p,\beta+\tau}^{\mathsf{ind}\leq 1}$ *(*$B = 2^{\beta}$*,* $t = 2^{\tau}$*) have evaluation vectors* $\boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\ell}, \boldsymbol{r}$ *of the form*

$$\boldsymbol{v} = [\boldsymbol{v}_0 || \ldots || \boldsymbol{v}_{t-1}],\ \boldsymbol{q} = [\boldsymbol{q}_0 || \ldots || \boldsymbol{q}_{t-1}],$$

$$\boldsymbol{\ell} = [\boldsymbol{\ell}_0 || \dots || \boldsymbol{\ell}_{t-1}],\ \boldsymbol{r} = [\boldsymbol{r}_0 || \dots || \boldsymbol{r}_{t-1}]$$

where $\gamma \in \mathbb{N}$ *is the number of trailing nonzero entries of each divisor and remainder,* $\boldsymbol{\ell}_i = [0^{\beta-\gamma} || \boldsymbol{\ell}'_i]$ *and* $\boldsymbol{r}_i = [0^{\beta-\gamma} || \boldsymbol{r}'_i]$ *for all* $i \in [t-1)$*. Moreover, for* $i \in [t-1)$*, the vectors* $\boldsymbol{v}_i, \boldsymbol{q}_i, \boldsymbol{\ell}_i, \boldsymbol{r}_i$ *are binary and encode integers via*

$$v_i = \sum_{j \in [B)} \boldsymbol{v}_i[j] 2^j, \qquad q_i = \sum_{j \in [B)} \boldsymbol{q}_i[j] 2^j,$$
$$\ell_i = \sum_{j \in [2B)} \boldsymbol{\ell}_i[j] 2^j, \qquad r_i = \sum_{j \in [2B)} \boldsymbol{r}_i[j] 2^j,$$

and satisfy the division relation $v_i = q_i \cdot \ell_i + r_i$*.*

Moreover, $\mathfrak{R}_{\mathsf{Div}}$ *denotes the dual relation of* $\mathfrak{R}_{\mathsf{Div\text{-}Oracle}}$ *according to Definition 27.*

PIOR 5: $\mathfrak{R}_{\mathsf{Div\text{-}Oracle}} \to \mathfrak{R}_{\mathsf{Mul\text{-}Oracle}} \times \mathfrak{R}_{\mathsf{Add\text{-}Oracle}} \times \mathfrak{R}_{\mathsf{Zero}} \times \mathfrak{R}_{\mathsf{Zero}}$

Parse: $\mathsf{x} = \gamma$, $\mathsf{y} = ([\![\widetilde{v}]\!], [\![\widetilde{q}]\!], [\![\widetilde{\ell}]\!], [\![\widetilde{r}]\!])$, $\mathsf{w} = (\widetilde{v}, \widetilde{q}, \widetilde{\ell}, \widetilde{r})$

1. $\mathcal{P}$ and $\mathcal{V}$ define a mask polynomial $\widetilde{m} \in p^{\mathsf{ind} \le 1}_{\beta+\tau}$ with succinct representation that has an evaluation vector of the form $\boldsymbol{m} = [\boldsymbol{m}' || \dots || \boldsymbol{m}']$ where $\boldsymbol{m'} = [1^{B/2} || 0^{B/2}]$ as follows:

$$\widetilde{m}(x_0, \dots, x_{\beta-1}, y_0, \dots, y_{\tau-1}) = 1 - x_{\beta-1}$$

2. $\mathcal{P}$ computes $\widetilde{p}$ whose evaluation vector $\boldsymbol{p} \in \{0,1\}^{2Bt}$ is the bit decomposition of the products $p_i := q_i \cdot \ell_i$ for $i \in [t-1)$, and sends the oracle $[\![\widetilde{p}]\!]$ to $\mathcal{V}$.
3. To verify that the integers encoded in $\boldsymbol{v}$ and $\boldsymbol{q}$ have bit-length at most $B/2$, $\mathcal{P}$ outputs two witnesses for the Zero Check relation (Definition 41) as $\mathsf{w}'_{\mathsf{Zero},v} := f_v$, $\mathsf{w}'_{\mathsf{Zero},q} := f_q$ while $\mathcal{V}$ outputs the corresponding explicit and implicit instances for the Zero Check relation as $(\mathsf{x}'_{\mathsf{Zero},v}, \mathsf{y}'_{\mathsf{Zero},v}) := (\emptyset, [\![f_v]\!])$ and $(\mathsf{x}'_{\mathsf{Zero},q}, \mathsf{y}'_{\mathsf{Zero},q}) := (\emptyset, [\![f_q]\!])$ where $f_v \in \mathbb{F}^{\mathsf{ind} \le 2}_{p_{\beta+\tau}}$ and $f_q \in \mathbb{F}^{\mathsf{ind} \le 2}_{p_{\beta+\tau}}$ are virtual polynomials defined as

$$f_v(\boldsymbol{x}, \boldsymbol{y}) := \widetilde{m}(\boldsymbol{x}, \boldsymbol{y}) \cdot \widetilde{v}(\boldsymbol{x}, \boldsymbol{y}), \quad f_q(\boldsymbol{x}, \boldsymbol{y}) := \widetilde{m}(\boldsymbol{x}, \boldsymbol{y}) \cdot \widetilde{q}(\boldsymbol{x}, \boldsymbol{y})$$

4. To enforce $q_i \cdot \ell_i = p_i$ for all i, $\mathcal{P}$ outputs the full witness tuple for $\mathfrak{R}_{\mathsf{Mul\text{-}Oracle}}$ obtained from PIOR 4 by identifying $\widetilde{a} := \widetilde{q}$, $\widetilde{b} := \widetilde{\ell}$, $\widetilde{c} := \widetilde{p}$ (using the same batched indexing / padding conventions as Definition 31) and including the auxiliary schoolbook digit, carry, and convolution-side oracles prescribed there. $\mathcal{V}$ outputs the explicit and implicit instances for $\mathfrak{R}_{\mathsf{Mul\text{-}Oracle}}$ produced by that reduction.
5. To enforce $p_i + r_i = v_i$ for all i, $\mathcal{P}$ outputs the full witness tuple for $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$ obtained from PIOR 1 by identifying $\widetilde{a} := \widetilde{p}$, $\widetilde{b} := \widetilde{r}$, $\widetilde{c} := \widetilde{v}$ (embedding the dividend's bit vector into the $2B$-wide sum slot of Definition 28 by zero-padding if the division interface uses fewer significant bits) and including the carry and rotated-carry oracles $[\![\widetilde{k}]\!], [\![\widetilde{k'}]\!]$ and rotation instance prescribed in

PIOR 1. $\mathcal{V}$ outputs the explicit and implicit instances for $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$ produced by that reduction.

Theorem 8. *PIOR 5 has perfect completeness and round-by-round soundness error at most* $\max\{\mu/p, tB/p\}$*, where* $\mu = \beta + \tau$*, with the following efficiency measures:*

rounds	*queries*	*oracle size*	*prover time*	*verifier time*
1	10	2^μ	$O(Bt)$	$O(\mu)$

Proof. Deferred to appendix A.4.5. □

4.2.6 MaSDAK: Magnitude-Succinct Diophantine Argument of Knowledge

We present Masdak, a public-coin argument of knowledge for proving knowledge of bounded, positive integer solutions to Diophantine equations. This protocol serves as a core component of our relaxed mod-PC and mod-PC constructions. Masdak is magnitude-succinct, meaning that the proof size and verification time scale sublinearly in the bit-length of the integer solutions being proven. While Masdak is not succinct in the length of the full witness, we only invoke it for Diophantine equations of constant arity, ensuring that the overall proof size and verification time remain sublinear in the bit-length of the integer solutions.

Definition 33. *We define* $\mathfrak{R}_{\mathsf{Masdak\text{-}Oracle}}$ *as the set of all triples of the form* $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = ((t, f), (\llbracket \widetilde{u}_1 \rrbracket, \ldots, \llbracket \widetilde{u}_\kappa \rrbracket), (\widetilde{u}_1, \ldots, \widetilde{u}_\kappa))$ *where* $f : \mathbb{Z}_B^t \to \mathbb{Z}_B$ *is a Diophantine function of arity* $t \in \mathbb{N}$*, given by an arithmetic circuit of size* s*. The witness contains* κ *multilinear polynomials* $\widetilde{u}_1, \ldots, \widetilde{u}_\kappa$*, committed under the* κ *commitments* $\mathsf{cm}_1, \ldots, \mathsf{cm}_\kappa$*, whose binary evaluation vectors concatenate to a single vector* $\boldsymbol{u} \in \{0,1\}^{tB}$ *such that* $f(\boldsymbol{v}) = 0$*, where each integer input is reconstructed from its bit decomposition as* $\boldsymbol{v}[i] = \sum_{j=0}^{n} 2^j \boldsymbol{u}[in + j]$*. I.e., the relation asserts that the reconstructed integer vector* $\boldsymbol{v}$*, whose entries may be drawn from several distinct commitments, satisfies the Diophantine equation* $f(\boldsymbol{v}) = 0$*. The single-commitment case (*$\kappa = 1$*) recovers the usual setting where all inputs to* f *reside in one commitment.*

Moreover, $\mathfrak{R}_{\mathsf{Masdak}}$ *denotes the dual relation of* $\mathfrak{R}_{\mathsf{Masdak\text{-}Oracle}}$ *according to Definition 27.*

PIOR 6: $\mathfrak{R}_{\mathsf{Masdak\text{-}Oracle}} \to \mathfrak{R}_{\mathsf{Div\text{-}Oracle}} \times \mathfrak{R}_{\mathsf{Decision}}$

Parse: $\mathsf{x} = (t, f)$, $\mathsf{y} = (\llbracket \widetilde{u}_1 \rrbracket, \ldots, \llbracket \widetilde{u}_\kappa \rrbracket)$, $\mathsf{w} = (\widetilde{u}_1, \ldots, \widetilde{u}_\kappa)$ $\quad \langle\ \mathcal{P}(\mathsf{i}, \mathsf{x}, \mathsf{w})\ ,\ \mathcal{V}(\mathsf{x})\ \rangle$

1. $\mathcal{V}$ samples a random prime modulus $\ell \in \mathsf{Primes}[\gamma]$ and sends it to the prover.
2. $\mathcal{P}$ computes the quotients $\boldsymbol{q} \in \mathbb{Z}_{B-\lambda}^t$ and remainders $\boldsymbol{r} \in \mathbb{Z}_\lambda^t$ s.t. the division equations hold; i.e. $\boldsymbol{u} = \ell \cdot \boldsymbol{q} + \boldsymbol{r}$.
3. $\mathcal{P}$ sends the plaintext $\boldsymbol{r}$ and the oracle $\llbracket \boldsymbol{q} \rrbracket$ to $\mathcal{V}$.
4. $\mathcal{V}$ checks if the Diophantine equation holds mod ℓ; i.e. $f(\boldsymbol{r}) \stackrel{?}{=} 0 \bmod \ell$.

5. $\mathcal{P}$ outputs a witness for the decision relation (Definition 8) as $\mathsf{w}'_{\mathsf{Decision}} := \varnothing$ while $\mathcal{V}$ outputs the corresponding instance for the decision relation as $(\mathsf{x}'_{\mathsf{Decision}}, \mathsf{y}'_{\mathsf{Decision}}) := (b, \varnothing)$ where b is accept if the check in Step 4 passes and b is reject otherwise.
6. Finally, $\mathcal{P}$ outputs the witness for the division relation as $\mathsf{w}'_{\mathsf{Div}} = (\widetilde{v}, \widetilde{q}, \widetilde{\ell}, \widetilde{r})$ while $\mathcal{V}$ outputs the corresponding instance and oracle instance for the division relation as $(\mathsf{x}'_{\mathsf{Div}}, \mathsf{y}'_{\mathsf{Div}}) := (\gamma, (\llbracket\widetilde{v}\rrbracket, \llbracket\widetilde{q}\rrbracket, \llbracket\widetilde{\ell}\rrbracket, \llbracket\widetilde{r}\rrbracket))$.

Theorem 9. *Let γ be the bit-length of the prime modulus sampled by the verifier, $\mu = \beta + \tau$, and $A := s\,n\,t$. Assume $\gamma \geq 2$ and $|f(\boldsymbol{v})| < 2^A$ on all accepting transcripts (in particular, this holds for natural circuit models of f). PIOR 6 has perfect completeness and round-by-round soundness error at most $\max\{\frac{A}{(\gamma-1)|\mathsf{Primes}[\gamma]|}, \mu/p, tB/p\}$. Moreover, let $\mathtt{M}(B)$ be the complexity of performing B-bit integer multiplication, then PIOR 6 has the following efficiency measures:*

rounds	*Message size*	*queries*	*oracle size*	*prover time*	*verifier time*
1	$t\gamma$	0	$2Bt$	$O(t\,\mathtt{M}(B))$	$O(t\gamma)$

Proof. Deferred to appendix A.4.6. □

Remark 2. Masdak is closely related in spirit to the PoKEMath protocol introduced in DewTwo [15] and can be viewed as a generalization of DewTwo: the latter operates in groups of unknown order and supports arbitrary-length integer solutions, whereas Masdak works over a generic ICS and leverages the division checks supported by that ICS.

Remark 3 (Diophantine equations spanning several commitments). An earlier version of Masdak required all integers appearing in the Diophantine equation to reside in a single commitment. We thank a reviewer for pointing out that our evaluation argument (Construction 3) evaluates a Diophantine equation whose variables are split between the input polynomial commitment and a freshly committed auxiliary vector. We therefore state Masdak directly for a tuple of commitments $(\mathsf{cm}_1, \ldots, \mathsf{cm}_\kappa)$: the verifier reconstructs the integer vector $\boldsymbol{v}$ from the concatenation of the committed bit-decompositions and runs the (unchanged) decision and division checks on it. Since the ICS supports batched openings, this incurs no additional cost, and the soundness and efficiency bounds of Theorem 9 depend only on the total arity t and are unaffected by how the inputs are partitioned across commitments.

4.3 Summary of the costs

In this section, we assemble the efficiency measures of the end-to-end PIOPs and compiled arguments for all the PIORs presented in this section. To demonstrate the efficiency of the final SNARK constructions, we instantiate the costs with two PCS: 1. TensorSwitch [11] a state-of-the-art hash-based PCS with $O(\lambda N)$ prover time, $O(\lambda \log N)$ proof size, and $O(\lambda \log N)$ verifier time, and, 2. Mercury [29], the state of the art elliptic-curve based PCS with $O(N)$ prover time, $O(1)$ proof size, and $O(\log N)$ verifier time (Table 4).

SNARK	proof size	prover time	verifier time
Addition	$\beta + \tau$	Bt	$\beta + \tau$
Range	$\beta + \tau$	Bt	$\beta + \tau$
Int. Mul.	$\beta + \tau$	$tB \log B$	$\beta + \tau$
MaSDAK	$t\lambda + \beta + \tau$	$tB \log B$	$t\lambda + \beta + \tau + \lambda \log \lambda$

λ: The security parameter, $B = 2^{\beta}$: The maximum bit-length of the integers
$t = 2^{\tau}$: The number of integers in the witness (assumption $\tau \geq 2$)

Table 4. Summary of the efficiency of the assembled SNARKs obtained by instantiating the PIORs from Section 4 using the compiler of Construction 7. Big-O notation is omitted for clarity.

5 $\mathbb{Q}\bar{a}$ren: A framework for building relaxed mod-pc

In this section, we present $\mathbb{Q}\bar{a}ren$, our generic framework for building relaxed mod-pcs (defined in Definition 25). For an intuitive overview of the construction, see Section 2.2.

Some notations. We denote by $\mathrm{rev}(\boldsymbol{v})$ the vector obtained by reversing the elements of $\boldsymbol{v}$. We also define $\lceil x \rceil_2$ for an integer x which represents the closest power of 2 larger than or equal to x (we use this as our range proofs support bounds that are powers of 2).

5.1 Commiting to an integer multilinear polynomial

According to Definition 20, we first need to construct a commitment scheme for integer multilinear polynomials. Our commitment scheme relies on encoding a μ-variate polynomial $p \in \mathbb{Z}_B[X_1, \ldots, X_\mu]$ as an integer $e_{\boldsymbol{p},\alpha} \in \mathbb{Z}$ which is described next:

Algorithm 1: Integer Encoding and Decoding

IntegerEncode$(p, \alpha) \to e_{\boldsymbol{p},\alpha}$ where $p \in \mathbb{Z}_B[X_1, \ldots, X_\mu]$, $e_{\boldsymbol{p},\alpha} \in \mathbb{Z}$ and $\alpha \in \mathbb{Z}^+$

1. Convert the polynomial p into a vector of coefficients $\boldsymbol{p} = [p_0, \ldots, p_{N-1}] \in \mathbb{Z}_B^N$.
2. Assert $\|\boldsymbol{p}\|_\infty < \alpha$ to ensure injectivity of the encoding.
3. Output $e_{\boldsymbol{p},\alpha} := \sum_{i=0}^{N-1} p_i \alpha^i$.

IntegerDecode$(e_{\boldsymbol{p},\alpha}, \alpha) \to p$ where $p \in \mathbb{Z}_B[X_1, \ldots, X_\mu]$, $e_{\boldsymbol{p},\alpha} \in \mathbb{Z}$ and $\alpha \in \mathbb{Z}^+$

1. Initialise an empty vector $\boldsymbol{p}$ to hold the digits.
2. Loop while $e_{\boldsymbol{p},\alpha} \neq 0$:
   (a) Compute a pair (q, r) such that $e_{\boldsymbol{p},\alpha} = q\alpha + r$ and $0 \leq r < |\alpha|$.
   (b) Append r to $\boldsymbol{p}$ and update $e_{\boldsymbol{p},\alpha} := q$.
3. Output $\boldsymbol{p}$.

Remark 4 (Decoding vs. extraction). Alg 1 defines IntegerDecode as the inverse of radix expansion: it recovers *integer* coefficients in $\mathbb{Z}_B^N$ from $e_{\boldsymbol{p},\alpha}$. The relaxed mod-PC extractor in Appendix B (Alg 4) applies IntegerDecode only as an intermediate step; it then runs the rational-recovery loop (Alg 4, steps 3–4) to obtain opening polynomials with coefficients in $\mathbb{Q}$, matching rational extractability (Definition 25).

Next, We describe our commitment scheme for integer multilinear polynomials which first encodes the polynomial as an integer using the above encoding and then commits to the resulting integer using an integer commitment scheme ICS (see Definition 26).

Construction 2: $\mathbb{Q}\bar{a}ren$ commitment scheme

$\mathtt{Setup}(1^\lambda, N, B) \to (\mathsf{ck}, \mathsf{vk})$

1. Invoke $(\mathsf{ck}_{\mathsf{ICS}}, \mathsf{vk}_{\mathsf{ICS}}) := \mathsf{ICS}.\mathtt{Setup}(1^\lambda, B, t)$
2. Let $\widehat{B} := \max(B, \lambda)$ denote the challenge bit-length (see Remark 9). Set the integer encoding base α to be an integer with no prime factors smaller than 2^B that satisfies
$$\alpha > \max\left\{ 2^{7\widehat{B}\mu},\ 2^{B+1} \cdot N^{8\mu}(2\mu)^\lambda \right\} \tag{4}$$
(see Remark 10).
3. Output $\mathsf{ck} := (\mathsf{ck}_{\mathsf{ICS}}, \alpha)$ and $\mathsf{vk} := (\mathsf{vk}_{\mathsf{ICS}}, \alpha)$

$\mathtt{Commit}(p, \mathsf{ck}) \to (\mathsf{cm}; \mathsf{opn})$

1. Compute the integer encoding of the polynomial as $e := \text{IntegerEncode}(p, \alpha)$.
2. Compute the integer commitment $(\mathsf{cm}_{\mathsf{ICS}}; \mathsf{opn}_{\mathsf{ICS}}) := \mathsf{ICS}.\mathtt{Commit}(\mathsf{ck}_{\mathsf{ICS}}, e)$.
3. Output $\mathsf{cm}_{\mathsf{PCS}} = \mathsf{cm}_{\mathsf{ICS}}$ and $\mathsf{opn}_{\mathsf{PCS}} = \mathsf{opn}_{\mathsf{ICS}}$.

$\mathtt{Open}(\mathsf{ck}, \mathsf{cm}, p, \mathsf{opn}) \to b$

1. Parse ck as $\mathsf{ck}_{\mathsf{ICS}}$, cm as $\mathsf{cm}_{\mathsf{ICS}}$, and opn as $\mathsf{opn}_{\mathsf{ICS}}$.
2. Regenerate the integer encoding of the polynomial $e := \text{IntegerEncode}(p)$.
3. Perform the opening for ICS and output $b := \mathsf{ICS}.\mathtt{Open}(\mathsf{ck}_{\mathsf{ICS}}, \mathsf{cm}_{\mathsf{ICS}}, e, \mathsf{opn}_{\mathsf{ICS}})$.

Remark 5 (Comparison with KZG). Construction 2 is reminiscent of the classic approach of KZG [44], where a polynomial is first evaluated at a point and the resulting value is then committed using a standard commitment scheme. Nevertheless, there are several key differences: (a) In KZG, all computations take place over a finite field, whereas our construction operates over bounded integers $\mathbb{Z}_B$, which form a finite ring. (b) In KZG, the scalar encoding of a polynomial is not injective. In contrast, we ensure injectivity by choosing the encoding base α to exceed the bound on the coefficients. (c) In KZG, the encoding base is secret and must be discarded as *toxic waste*. In our setting, the encoding base is public and forms part of the commitment key.

Next, we lift the commitment scheme to relaxed modular polynomial commitment scheme by augmenting it with an evaluation argument which we describe next.

5.2 A None-Adaptive Relaxed Mod-PC

Before presenting the constructions for evaluation argument, which is an argument for the evaluation relation defined in Definition 21, we introduce two helper algorithms that we use in the argument: First, Alg 2 which expands a μ-variate evaluation point $\boldsymbol{y} \in \mathbb{F}_q^\mu$ into a N-dimensional vector $\mathbf{y} \in \mathbb{Z}_B^N$ representing all possible monomials of a multilinear polynomial evaluated at this point.

Algorithm 2: Point to Coefficient Vector

PointToCoefficients$(\boldsymbol{y}) \to \mathbf{y}$

1. For all $i \in \{0, \ldots, N-1\}$: compute $\mathbf{y}_i = \prod_{j=0}^{\mu-1} \boldsymbol{y}_j^{i_j}$ where i_j is the j-th bit of the binary representation of i and the product is over integers.
2. Output the monomial expansion $\mathbf{y} = [\mathbf{y}_i]_{i=0}^{N-1}$.

Second, Alg 3, which splits an integer w at index N into three integers l, m, r in base α.

Algorithm 3: Three-way Decomposition

ThreeWayDecomp$(w, \alpha, N) \to (l, m, r)$

1. Let $\mathbf{c} = (c_0, \ldots, c_m)$ represent w in base α where $m := \lceil \log_\alpha |w| \rceil$.
2. Output (l, m, r) where

$$w = \sum_{j=0}^{m} c_j \alpha^j = \sum_{j=0}^{N-2} c_j \alpha^j + c_{N-1}\alpha^{N-1} + \sum_{j=N}^{m} c_j \alpha^j = l + m\alpha^{N-1} + r\alpha^N$$

Our evaluation argument, both for adaptive and non-adaptive extractability, relies on repeating a core sub-protocol. Intuitively, this core protocol only guarantees that the extracted polynomial satisfies the evaluation relation modulo α which is the encoding base. Obviously, this guarantee is not sufficient for a relaxed mod-PC. Later, we show that by running the core protocol twice with different encoding bases, we can ensure that the extracted polynomial satisfies the evaluation relation over the random modulus q, which is sufficient for a relaxed mod-PC.

Moreover, we present the construction only for a single evaluation claim (i.e., $\ell = 1$), and the evaluation map φ set to mod q for a randomly sampled modulo q and therefore simplify the notation accordingly compared to Definition 21.

Definition 34. *We define $\mathfrak{R}_{CoreEval}$ as the set of all tuples of the form* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) = (\mathsf{ck}, \mathsf{vk}, (q, \mathsf{cm}, \boldsymbol{y}, z), (p, \mathsf{opn}))$*, where* ck, vk *contains commitment and verification keys for the PCS, and* x, w *satisfy* $z = p(\boldsymbol{y}) \mod \alpha$ *and* $\mathsf{PCS.Open}(\mathsf{ck}, \mathsf{cm}, p, \mathsf{opn}) = 1$.

Construction 3: $\mathsf{ROK}_{CoreEval} : \mathfrak{R}_{CoreEval} \to \mathfrak{R}_{\mathsf{Masdak}} \times \mathfrak{R}_{\mathsf{Range}}$

Parse: $\mathsf{i} = (\mathsf{ck}, \mathsf{vk})$, $\mathsf{x} = (q, \mathsf{cm}, \boldsymbol{y}, z)$, $\mathsf{w} = (p, \mathsf{opn})$

1. The prover parses $\mathsf{pk} := \mathsf{ck}_{\mathsf{PCS}}$ where $\mathsf{ck}_{\mathsf{PCS}} := (\mathsf{ck}_{\mathsf{ICS}}, \alpha)$, $\mathsf{w} := (p, \mathsf{opn})$, verifier parses $\mathsf{vk} := \mathsf{vk}_{\mathsf{PCS}}$ where $\mathsf{vk}_{\mathsf{PCS}} := (\mathsf{vk}_{\mathsf{ICS}}, \alpha)$, and both parse $\mathsf{x} := (q, \mathsf{cm}, \boldsymbol{y}, z)$.
2. $\mathcal{P}$ computes $\mathbf{y} := \text{PointToCoefficients}(\boldsymbol{y})$ where $\mathbf{y} \in \mathbb{Z}_B^N$. The verifier does *not* materialize $\mathbf{y}$; see Remark 6.
3. $\mathcal{P}$ computes the integer encoding of the polynomial and the coefficient vector as follows

$$e_{\boldsymbol{p},\alpha} := \text{IntegerEncode}(\boldsymbol{p}, \alpha), \quad e_{\boldsymbol{y}} := \text{IntegerEncode}(\text{rev}(\mathbf{y}), \alpha) = \prod_{k=0}^{\mu-1} \left(\alpha^{2^k} + \boldsymbol{y}_k\right)$$

   and computes the integer $w := e_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}$.
4. $\mathcal{P}$ decomposes the base-$e_{\boldsymbol{y}}$ representation of integer w into left, middle and right integers at the index N; i.e.

$$(l, m, r) := \text{ThreeWayDecomp}(w, \alpha, N)$$

   and writes the middle integer as $m = z + hq$ with $z := z < q$, so that reducing m modulo q recovers the claimed evaluation z. Then it forms the vector of integers $\boldsymbol{u} = [l, m, r, h]$ which we prove some properties about in the next steps.
5. $\mathcal{P}$ commits to the integers $\boldsymbol{u}$ by computing $\mathsf{cm}_{\boldsymbol{u}} = \mathsf{ICS.Commit}(\boldsymbol{u}, \mathsf{ck}_{\mathsf{ICS}})$.
6. To check that the committed integers satisfy the Diophantine equations f_1 and f_2 which are defined as follows

$$f_1(e_{\boldsymbol{p},\alpha}, l, m, r, h) = l + m\alpha^{N-1} + r\alpha^N - e_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}$$
$$f_2(e_{\boldsymbol{p},\alpha}, l, m, r, h) = m - z - hq$$

   Here the evaluation-point encoding $e_{\boldsymbol{y}}$ and the values $z := z$ and q are public, whereas the polynomial encoding $e_{\boldsymbol{p},\alpha}$ is a *secret* integer bound by the input commitment cm, viewed as an ICS commitment to the integer encoding of p in base α (recall $e_{\boldsymbol{p},\alpha} := \text{IntegerEncode}(\boldsymbol{p}, \alpha)$). Since $e_{\boldsymbol{p},\alpha}$ lives in cm while $\boldsymbol{u}$ lives in $\mathsf{cm}_{\boldsymbol{u}}$, the Diophantine system spans two commitments, and we invoke the multi-commitment form of Masdak (Definition 33 and Remark 3). $\mathcal{P}$ outputs the witness $\mathsf{w}'_{\mathsf{Masdak}} := (p, \boldsymbol{u})$ and $\mathcal{V}$ outputs the instance $(\mathsf{x}, \mathsf{y}) = ((5, [f_1, f_2]), (\mathsf{cm}, \mathsf{cm}_{\boldsymbol{u}}))$.
7. To check that the committed integers are in the appropriate ranges, $\mathcal{P}$ outputs the witness $\mathsf{w}'_{\mathsf{Range}} := \boldsymbol{u}$ and $\mathcal{V}$ outputs the instance

$$(\mathsf{x}'_{\mathsf{Range}}, \mathsf{y}'_{\mathsf{Range}}) = ((\lceil N 2^{\widehat{B}(\mu+1)} \alpha^{N-2} \rceil_2, \lceil N 2^{\widehat{B}(\mu+1)} \rceil_2, \lceil 2^{\widehat{B}(\mu+1)} \alpha^{N-2} \rceil_2, \perp), \mathsf{cm}_{\boldsymbol{u}})$$

   where $\widehat{B} = \max(B, \lambda)$ is the challenge bit-length.

Remark 6 (Succinct verifier evaluation of $e_{\boldsymbol{y}}$). Although PointToCoefficients (Alg 2) produces a vector of length $N = 2^\mu$, the verifier never materializes it. The evaluation-point encoding admits the closed product form $e_{\boldsymbol{y}} = \text{IntegerEncode}(\text{rev}(\mathbf{y}), \alpha) = \prod_{k=0}^{\mu-1}(\alpha^{2^k} + \boldsymbol{y}_k)$, obtained by expanding the reversed monomial

basis. The verifier only ever uses $e_{\boldsymbol{y}}$ as a public coefficient of the Diophantine equation f_1, where it is reduced modulo the prime sampled inside Masdak; it therefore evaluates the product form under that modulus in $O(\mu) = O(\log N)$ modular multiplications, keeping the verifier sublinear as claimed in Theorem 10. Only the prover, which needs $e_{\boldsymbol{y}}$ over $\mathbb{Z}$ for the three-way decomposition, computes the full encoding. This is the same observation made in DewTwo [15].

Next, we construct the full non-adaptive evaluation argument which relies on "Commitment Switching" technique as intuitively described in Section 2.4. The construction simply invokes the core evaluation argument twice, first with the original commitment and encoding base α, and then with a switched commitment and encoding base β which is a random multiple of α.

We define the non-adaptive evaluation relation first, and reduce it to two instances of the core evaluation relation. Note that q and $\boldsymbol{y}$ are randomly sampled inside the construction (steps 2 and 4), causing the non-adaptive nature of the argument.

Definition 35. *We define* $\mathfrak{R}_{EvalNonAdaptive}$ *as the set of all tuples of the form* $(\mathsf{i}, \mathsf{x}, \mathsf{w}) = (\mathsf{ck}, \mathsf{vk}, (\mathsf{cm}), (p, \mathsf{opn}))$*, where* ck, vk *contains commitment and verification keys for the PCS, and* x, w *satisfy* $\mathsf{PCS.Open}(\mathsf{ck}, \mathsf{cm}, p, \mathsf{opn}) = 1$.

Construction 4: $\mathsf{ROK}_{EvalNonAdaptive} : \mathfrak{R}_{EvalNonAdaptive} \to {\mathfrak{R}_{CoreEval}}^2$

Parse: $\mathsf{i} = (\mathsf{ck}, \mathsf{vk})$, $\mathsf{x} = (\mathsf{cm})$, $\mathsf{w} = (p, \mathsf{opn})$

1. The prover parses $\mathsf{pk} := \mathsf{ck}_{\mathsf{PCS}}$ where $\mathsf{ck}_{\mathsf{PCS}} := (\mathsf{ck}_{\mathsf{ICS}}, \alpha)$ and the verifier parses $\mathsf{vk} := \mathsf{vk}_{\mathsf{PCS}}$ where $\mathsf{vk}_{\mathsf{PCS}} := (\mathsf{vk}_{\mathsf{ICS}}, \alpha)$.
2. $\mathcal{P}$ and $\mathcal{V}$ sample a random prime $q > 2^{\widehat{B}}$ (where $\widehat{B} = \max(B, \lambda)$ is the challenge bit-length) and compute $\beta := q\alpha$ and $\mathsf{pk}' := (\mathsf{ck}_{\mathsf{ICS}}, \beta)$ and $\mathsf{vk}' := (\mathsf{vk}_{\mathsf{ICS}}, \beta)$.
3. $\mathcal{P}$ computes $\mathsf{cm}' := \mathsf{PCS.Commit}(p, \mathsf{pk}')$.
4. $\mathcal{P}$ and $\mathcal{V}$ sample a random $\boldsymbol{y} \in \mathbb{Z}_{\widehat{B}}^{\mu}$.
5. $\mathcal{P}$ computes $z := p(\boldsymbol{y}) \bmod q$.
6. $\mathcal{P}$ outputs the witness $\mathsf{w}'_\alpha := (p, \mathsf{opn})$ and $\mathcal{V}$ outputs the instance $(\mathsf{x}'_\alpha, \mathsf{y}'_\alpha) = ((q, \mathsf{cm}, \boldsymbol{y}, z), \perp)$ for $\mathfrak{R}_{CoreEval}$ with keys $(\mathsf{pk}, \mathsf{vk})$.
7. $\mathcal{P}$ outputs the witness $\mathsf{w}'_\beta := (p, \mathsf{opn})$ and $\mathcal{V}$ outputs the instance $(\mathsf{x}'_\beta, \mathsf{y}'_\beta) = ((q, \mathsf{cm}', \boldsymbol{y}, z), \perp)$ for $\mathfrak{R}_{CoreEval}$ with keys $(\mathsf{pk}', \mathsf{vk}')$.

Theorem 10. *Construction 4 is a reduction of knowledge from* $\mathfrak{R}_{EvalNonAdaptive}$ *to* ${\mathfrak{R}_{CoreEval}}^2$*. Consequently, composing it with* $\mathsf{ROK}_{CoreEval}$ *(Construction 3) yields a relaxed mod-PC (see Def 25) with non-adaptive extractability (see Def. 23). See Section 5.3 for strengthening this to adaptive extractability. Additionally, we have the following asymptotic efficiency measures (as a function of the sub-protocols used);*

- *Prover time is bounded as* $O(N) + T_P(\mathsf{ICS.Commit}(N2^{\widehat{B}(\mu+1)}\alpha^{N-2})) + T_P(\mathsf{Masdak}(5, [f_1, f_2], (\mathsf{cm}, \mathsf{cm}_{\boldsymbol{u}}))) + T_P(\mathsf{Range}(N2^{\widehat{B}(\mu+1)}\alpha^{N-2}))$

– *Verifier time is* $O(\mu)+T_V(\mathsf{Masdak}(5,[f_1,f_2],(\mathsf{cm},\mathsf{cm}_{\boldsymbol{u}})))+T_V(\mathsf{Range}(N2^{\widehat{B}(\mu+1)}\alpha^{N-2}))$,
– *Proof size is* $O(1)+\pi(\mathsf{Masdak}(5,[f_1,f_2],(\mathsf{cm},\mathsf{cm}_{\boldsymbol{u}})))+\pi(\mathsf{Range}(N2^{\widehat{B}(\mu+1)}\alpha^{N-2}))$

where $T_P(\mathsf{ICS.Commit}(n))$ *denotes the time taken to commit to an integer of size* n *via the integer commitment scheme* ICS, $T_P(\mathsf{Masdak}(t,\boldsymbol{f},\mathsf{cm}))$ *denotes the prover time for the* Masdak *protocol (see Theorem 9) on* t *integers with Diophantine equations* $\boldsymbol{f}$ *and commitment* cm, *and* $T_P(\mathsf{Range}(t,n))$ *denotes the time taken to produce a range proof of* t *integers of size at most* n. *The other quantities are similarly defined with the subscript* V *denoting verifier time and* $\pi(\cdot)$ *denoting proof sizes.*

We defer the proof to Appendix B. By instantiating the polynomial commitment schemes with (Tensorswitch) TS and (Mercury) MC, we get the following concrete efficiency metrics for our construction.

Variant	Prover time	Verifier time	Proof size
$\mathsf{ROK}_{Eval\mathsf{TS}}$	$\widehat{B}N\log(\widehat{B}N)$	$\log(\widehat{B}+N)$	$\log(\widehat{B}+N)$
$\mathsf{ROK}_{Eval\mathsf{MC}}$	$\widehat{B}N\log(\widehat{B}N)$	$\log(\widehat{B}+N)$	$\log(\widehat{B}+N)$

Fig. 2. Efficiency metrics for ROK_{Eval} when instantiated with different polynomial commitment schemes. Here $\widehat{B}=\max(B,\lambda)$ is the challenge bit-length; in the intended regime $B\geq\lambda$ it equals the coefficient bit-length B, and these reduce to $BN\log(BN)$ and $\log(B+N)$.

5.3 Lifting to Adaptive Extractability

Obtaining an adaptive PCS from a non-adaptive PCS just requires running the protocol again with an arbitrary evaluation point, as described in [2,15]. This is because the non-adaptive PCS constructed in Theorem 10 proves that the underlying committed polynomial satisfies some structure, i.e., all the coefficients are bounded rationals. Now, the third call to the core protocol with the externally provided evaluation point will prove that the underlying polynomial evaluates to the claimed value at that point (without any soundness issues, since the first two calls already established the structural properties of the polynomial).

We define the adaptive evaluation relation, which additionally takes an externally chosen evaluation point $\hat{y}$ and claimed evaluation $\hat{z}$ as part of the instance.

Definition 36. *We define* $\mathfrak{R}_{EvalAdaptive}$ *as the set of all tuples of the form* $(\mathsf{i},\mathsf{x},\mathsf{w})=(\mathsf{ck},\mathsf{vk},(\mathsf{cm},\hat{y},\hat{z}),(p,\mathsf{opn}))$, *where* ck,vk *contains commitment and verification keys for the PCS, and* x,w *satisfy* $\hat{z}=p(\hat{y}) \mod q$ *and* $\mathsf{PCS.Open}(\mathsf{ck},\mathsf{cm},p,\mathsf{opn})=1$.

Construction 5: $\mathsf{ROK}_{EvalAdaptive} : \mathfrak{R}_{EvalAdaptive} \to {\mathfrak{R}_{CoreEval}}^3$

Parse: $\mathsf{i} = (\mathsf{ck}, \mathsf{vk})$, $\mathsf{x} = (\mathsf{cm}, \hat{y}, \hat{z})$, $\mathsf{w} = (p, \mathsf{opn})$

1. The prover parses $\mathsf{pk} := \mathsf{ck}_{\mathsf{PCS}}$ where $\mathsf{ck}_{\mathsf{PCS}} := (\mathsf{ck}_{\mathsf{ICS}}, \alpha)$ and the verifier parses $\mathsf{vk} := \mathsf{vk}_{\mathsf{PCS}}$ where $\mathsf{vk}_{\mathsf{PCS}} := (\mathsf{vk}_{\mathsf{ICS}}, \alpha)$.
2. $\mathcal{P}$ and $\mathcal{V}$ sample a random prime $q > 2^{\widehat{B}}$ (where $\widehat{B} = \max(B, \lambda)$ is the challenge bit-length) and compute $\beta := q\alpha$ and $\mathsf{pk}' := (\mathsf{ck}_{\mathsf{ICS}}, \beta)$ and $\mathsf{vk}' := (\mathsf{vk}_{\mathsf{ICS}}, \beta)$.
3. $\mathcal{P}$ computes $\mathsf{cm}' := \mathsf{PCS.Commit}(p, \mathsf{pk}')$.
4. $\mathcal{P}$ and $\mathcal{V}$ sample a random $\boldsymbol{y} \in \mathbb{Z}_{\widehat{B}}^{\mu}$.
5. $\mathcal{P}$ computes $z := p(\boldsymbol{y}) \bmod q$.
6. $\mathcal{P}$ outputs the witness $\mathsf{w}'_\alpha := (p, \mathsf{opn})$ and $\mathcal{V}$ outputs the instance $(\mathsf{x}'_\alpha, \mathsf{y}'_\alpha) = ((q, \mathsf{cm}, \boldsymbol{y}, z), \bot)$ for $\mathfrak{R}_{CoreEval}$ with keys $(\mathsf{pk}, \mathsf{vk})$.
7. $\mathcal{P}$ outputs the witness $\mathsf{w}'_\beta := (p, \mathsf{opn})$ and $\mathcal{V}$ outputs the instance $(\mathsf{x}'_\beta, \mathsf{y}'_\beta) = ((q, \mathsf{cm}', \boldsymbol{y}, z), \bot)$ for $\mathfrak{R}_{CoreEval}$ with keys $(\mathsf{pk}', \mathsf{vk}')$.
8. $\mathcal{P}$ outputs the witness $\mathsf{w}'_{\hat{y}} := (p, \mathsf{opn})$ and $\mathcal{V}$ outputs the instance $(\mathsf{x}'_{\hat{y}}, \mathsf{y}'_{\hat{y}}) = ((q, \mathsf{cm}, \hat{y}, \hat{z}), \bot)$ for $\mathfrak{R}_{CoreEval}$ with keys $(\mathsf{pk}, \mathsf{vk})$.

Corollary 2. *Construction 5 is a reduction of knowledge from* $\mathfrak{R}_{EvalAdaptive}$ *to* ${\mathfrak{R}_{CoreEval}}^3$*. Consequently, composing it with* $\mathsf{ROK}_{CoreEval}$ *(Construction 3) yields a relaxed mod-PC (see Def 25) with adaptive extractability. The asymptotic efficiency measures are the same as in Theorem 10.*

The proof of this corollary follows from the same arguments as those in Theorem 5.7 in [15].

6 Acknowledgements

We thank Pratyush Mishra, Benedikt Bünz, Shravan Srinivasan, Christodoulos Pappas, Tushar Mopuri, and Anubhav Baweja for helpful discussions. We also thank an anonymous reviewer for an observation that led to a simplification of our integer-multiplication argument (see Remark 1).

References

1. Arnon, G., Chiesa, A., Fenzi, G., Yogev, E.: WHIR: reed-solomon proximity testing with super-fast verification. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 15604, pp. 214–243. Springer (2025). https://doi.org/10.1007/978-3-031-91134-7_8, https://doi.org/10.1007/978-3-031-91134-7_8
2. Arun, A., Ganesh, C., Lokam, S.V., Mopuri, T., Sridhar, S.: Dew: A transparent constant-sized polynomial commitment scheme. In: Boldyreva, A., Kolesnikov, V. (eds.) Public-Key Cryptography - PKC 2023 - 26th IACR International Conference

on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13941, pp. 542–571. Springer (2023). https://doi.org/10.1007/978-3-031-31371-4_19, https://doi.org/10.1007/978-3-031-31371-4_19

3. Baweja, A., Mishra, P., Mopuri, T., Shtepel, M.: FICS and FACS: fast iopps and accumulation via code-switching. IACR Cryptol. ePrint Arch. p. 737 (2025), https://eprint.iacr.org/2025/737
4. Ben-Sasson, E., Carmon, D., Kopparty, S., Levit, D.: Scalable and transparent proofs over all large fields, via elliptic curves - (ECFFT part II). In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13747, pp. 467–496. Springer (2022). https://doi.org/10.1007/978-3-031-22318-1_17, https://doi.org/10.1007/978-3-031-22318-1_17
5. Besier, M.: Common circom pitfalls and how to dodge them — part 1. Blog post, zkSecurity (2025), https://blog.zksecurity.xyz/posts/circom-pitfalls-1/, discusses common pitfalls such as aliasing bugs and under-constrained signals in Circom circuits
6. Block, A.R., Fang, Z., Katz, J., Thaler, J., Waldner, H., Zhang, Y.: Field-agnostic snarks from expand-accumulate codes. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X. Lecture Notes in Computer Science, vol. 14929, pp. 276–307. Springer (2024). https://doi.org/10.1007/978-3-031-68403-6_9, https://doi.org/10.1007/978-3-031-68403-6_9
7. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Time- and space-efficient arguments from groups of unknown order. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 12828, pp. 123–152. Springer (2021). https://doi.org/10.1007/978-3-030-84259-8_5, https://doi.org/10.1007/978-3-030-84259-8_5
8. Bootle, J., Chiesa, A., Hu, Y., Orrù, M.: Gemini: Elastic snarks for diverse environments. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13276, pp. 427–457. Springer (2022). https://doi.org/10.1007/978-3-031-07085-3_15, https://doi.org/10.1007/978-3-031-07085-3_15
9. Brehm, M., Chen, B., Fisch, B., Resch, N., Rothblum, R.D., Zeilberger, H.: Blaze: Fast snarks from interleaved RAA codes. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 15604, pp. 123–152. Springer (2025). https://doi.org/10.1007/978-3-031-91134-7_5, https://doi.org/10.1007/978-3-031-91134-7_5
10. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco,

California, USA. pp. 315–334. IEEE Computer Society (2018). https://doi.org/10.1109/SP.2018.00020, https://doi.org/10.1109/SP.2018.00020
11. Bünz, B., Fenzi, G., Rothblum, R.D., Wang, W.: Tensorswitch: Nearly optimal polynomial commitments from tensor codes. IACR Cryptol. ePrint Arch. p. 2065 (2025), https://eprint.iacr.org/2025/2065
12. Bünz, B., Fisch, B.: Multilinear schwartz-zippel mod N and lattice-based succinct arguments. In: Rothblum, G.N., Wee, H. (eds.) Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14371, pp. 394–423. Springer (2023). https://doi.org/10.1007/978-3-031-48621-0_14, https://doi.org/10.1007/978-3-031-48621-0_14
13. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 677–706. Springer (2020). https://doi.org/10.1007/978-3-030-45721-1_24, https://doi.org/10.1007/978-3-030-45721-1_24
14. Bünz, B., Mishra, P., Nguyen, W., Wang, W.: Arc: Accumulation for reed-solomon codes. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 16006, pp. 128–160. Springer (2025). https://doi.org/10.1007/978-3-032-01907-3_5, https://doi.org/10.1007/978-3-032-01907-3_5
15. Bünz, B., Mopuri, T., Shirzad, A., Sridhar, S.: Dewtwo: A transparent pcs with quasi-linear prover, logarithmic verifier and 4.5kb proofs from falsifiable assumptions. In: Tauman Kalai, Y., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025. pp. 549–583. Springer Nature Switzerland, Cham (2025)
16. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Computer Science, vol. 2045, pp. 93–118. Springer (2001). https://doi.org/10.1007/3-540-44987-6_7, https://doi.org/10.1007/3-540-44987-6_7
17. Campanelli, M., Fiore, D., Querol, A.: Legosnark: Modular design and composition of succinct zero-knowledge proofs. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 2075–2092. ACM (2019). https://doi.org/10.1145/3319535.3339820, https://doi.org/10.1145/3319535.3339820
18. Campanelli, M., Hall-Andersen, M.: Fully-succinct arguments over the integers from first principles. IACR Cryptol. ePrint Arch. p. 1548 (2024), https://eprint.iacr.org/2024/1548
19. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019. pp. 1082–1090. ACM (2019). https://doi.org/10.1145/3313276.3316380, https://doi.org/10.1145/3313276.3316380

20. Chaliasos, S., Ernstberger, J., Theodore, D., Wong, D., Jahanara, M., Livshits, B.: Sok: what don't we know? understanding security vulnerabilities in snarks. In: Proceedings of the 33rd USENIX Conference on Security Symposium. SEC '24, USENIX Association, USA (2024)

21. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 499–530. Springer (2023). https://doi.org/10.1007/978-3-031-30617-4_17, https://doi.org/10.1007/978-3-031-30617-4_17

22. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zksnarks with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 738–768. Springer (2020). https://doi.org/10.1007/978-3-030-45721-1_26, https://doi.org/10.1007/978-3-030-45721-1_26

23. Chiesa, A., Yogev, E.: Building cryptographic proofs from hash functions. Online book (2024), https://github.com/hash-based-snargs-book, available at https://snargsbook.org/

24. Cini, V., Malavolta, G., Nguyen, N.K., Wee, H.: Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X. Lecture Notes in Computer Science, vol. 14929, pp. 207–242. Springer (2024). https://doi.org/10.1007/978-3-031-68403-6_7, https://doi.org/10.1007/978-3-031-68403-6_7

25. Couteau, G., Peters, T., Pointcheval, D.: Removing the strong RSA assumption from arguments over the integers. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10211, pp. 321–350 (2017). https://doi.org/10.1007/978-3-319-56614-6_11, https://doi.org/10.1007/978-3-319-56614-6_11

26. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) Advances in Cryptology — ASIACRYPT 2002. pp. 125–142. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

27. Danezis, G., Fournet, C., Groth, J., Kohlweiss, M.: Square span programs with applications to succinct NIZK arguments. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 532–550. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_28, https://doi.org/10.1007/978-3-662-45611-8_28

28. Dellepere, M., Mishra, P., Shirzad, A.: Garuda and pari: Faster and smaller snarks via equifficient polynomial commitments. IACR Cryptol. ePrint Arch. p. 1245 (2024), https://eprint.iacr.org/2024/1245

29. Eagen, L., Gabizon, A.: MERCURY: A multilinear polynomial commitment scheme with constant proof size and no prover FFTs. Cryptology ePrint Archive, Paper 2025/385 (2025), https://eprint.iacr.org/2025/385
30. Eskandarian, S., Messeri, E., Bonneau, J., Boneh, D.: Certificate transparency with privacy. Proc. Priv. Enhancing Technol. **2017**(4), 329–344 (2017). https://doi.org/10.1515/POPETS-2017-0052, https://doi.org/10.1515/popets-2017-0052
31. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Jr., B.S.K. (ed.) Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1294, pp. 16–30. Springer (1997). https://doi.org/10.1007/BFB0052225, https://doi.org/10.1007/BFb0052225
32. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. p. 953 (2019), https://eprint.iacr.org/2019/953
33. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: Snarks for ring arithmetic. J. Cryptol. **36**(4), 41 (2023). https://doi.org/10.1007/S00145-023-09481-3, https://doi.org/10.1007/s00145-023-09481-3
34. Ganesh, C., Patranabis, S., Singh, N.: Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. IACR Cryptol. ePrint Arch. p. 419 (2025), https://eprint.iacr.org/2025/419
35. Garreta, A., Waldner, H., Vlasov, I., Hristova, K., Dall'Ava, L., Čupić, M., Klein, M.: Zinc: Succinct arguments with small arithmetization overheads from iops of proximity to the integers. In: Tauman Kalai, Y., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025. pp. 259–291. Springer Nature Switzerland, Cham (2025)
36. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 626–645. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_37, https://doi.org/10.1007/978-3-642-38348-9_37
37. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for Muggles. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 113–122. STOC '08, ACM (2008). https://doi.org/10.1145/1374376.1374396, https://doi.org/10.1145/1374376.1374396
38. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14082, pp. 193–226. Springer (2023). https://doi.org/10.1007/978-3-031-38545-2_7, https://doi.org/10.1007/978-3-031-38545-2_7
39. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3531, pp. 467–482

(2005). https://doi.org/10.1007/11496137_32, https://doi.org/10.1007/11496137_32

40. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 581–612. Springer (2017). https://doi.org/10.1007/978-3-319-63715-0_20, https://doi.org/10.1007/978-3-319-63715-0_20
41. Haböck, U.: Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Paper 2022/1530 (2022), https://eprint.iacr.org/2022/1530
42. Huang, M.M., Mao, X., Zhang, J.: Sublinear proofs over polynomial rings. IACR Cryptol. ePrint Arch. p. 199 (2025), https://eprint.iacr.org/2025/199
43. Kadianakis, G., Zapico, A., Hafezi, H., Bünz, B.: Kzh-fold: Accountable voting from sublinear accumulation. In: Huang, C., Chen, J., Shieh, S., Lie, D., Cortier, V. (eds.) Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025. pp. 409–422. ACM (2025). https://doi.org/10.1145/3719027.3744796, https://doi.org/10.1145/3719027.3744796
44. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (2010). https://doi.org/10.1007/978-3-642-17373-8_11, https://doi.org/10.1007/978-3-642-17373-8_11
45. Kothapalli, A., Parno, B.: Algebraic reductions of knowledge. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14084, pp. 669–701. Springer (2023). https://doi.org/10.1007/978-3-031-38551-3_21, https://doi.org/10.1007/978-3-031-38551-3_21
46. Kuchta, V., Sakzad, A., Steinfeld, R., Liu, J.K.: Lattice-based zero-knowledge arguments for additive and multiplicative relations. Des. Codes Cryptogr. **89**(5), 925–963 (2021). https://doi.org/10.1007/S10623-021-00851-1, https://doi.org/10.1007/s10623-021-00851-1
47. Libert, B., Ling, S., Nguyen, K., Wang, H.: Lattice-based zero-knowledge arguments for integer relations. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 700–732. Springer (2018). https://doi.org/10.1007/978-3-319-96881-0_24, https://doi.org/10.1007/978-3-319-96881-0_24
48. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Laih, C. (ed.) Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2894, pp. 398–415. Springer (2003). https://doi.org/10.1007/978-3-540-40061-5_26, https://doi.org/10.1007/978-3-540-40061-5_26

49. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM **39**(4), 859–868 (1992). https://doi.org/10.1145/146585.146605, https://doi.org/10.1145/146585.146605
50. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Practical lattice-based zero-knowledge proofs for integer relations. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020. pp. 1051–1070. ACM (2020). https://doi.org/10.1145/3372297.3417894, https://doi.org/10.1145/3372297.3417894
51. Namboothiry, B., Shirzad, A., Solit, S., Marcus, R., Mishra, P.: TruthTable: A verifiable query engine. Cryptology ePrint Archive, Paper 2026/1260 (2026), https://eprint.iacr.org/2026/1260
52. Nethermind: Zinc concrete: Costs and experimental results. Online Notion page (2025), https://www.notion.so/nethermind/Zinc-Concrete-costs-and-experimental-results-1d3360fc38d08069b583d00c8f2cd5e6, accessed: 2025-01-03
53. Nethermind Security Audits Team: Zk circuit security: A guide for engineers and architects. Nethermind Blog (2025), https://www.nethermind.io/blog/zk-circuit-security-a-guide-for-engineers-and-architects, accessed 2026-03-12
54. Nguyen, N.K., Seiler, G.: Greyhound: Fast polynomial commitments from lattices. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X. Lecture Notes in Computer Science, vol. 14929, pp. 243–275. Springer (2024). https://doi.org/10.1007/978-3-031-68403-6_8, https://doi.org/10.1007/978-3-031-68403-6_8
55. Pailoor, S., Chen, Y., Wang, F., Rodríguez, C., Van Geffen, J., Morton, J., Chu, M., Gu, B., Feng, Y., Dillig, I.: Automated detection of under-constrained circuits in zero-knowledge proofs. Proc. ACM Program. Lang. **7**(PLDI) (Jun 2023). https://doi.org/10.1145/3591282, https://doi.org/10.1145/3591282
56. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM **27**(4), 701–717 (1980)
57. Seres, I.A., Burcsi, P.: Behemoth: Transparent polynomial commitment scheme with constant opening proof size and verifier time. In: Nitaj, A., Petkova-Nikova, S., Rijmen, V. (eds.) Progress in Cryptology - AFRICACRYPT 2025 - 16th International Conference on Cryptology in Africa, Rabat, Morocco, July 21-23, 2025, Proceedings. Lecture Notes in Computer Science, vol. 15651, pp. 343–369. Springer (2025). https://doi.org/10.1007/978-3-031-97260-7_16, https://doi.org/10.1007/978-3-031-97260-7_16
58. Setty, S.T.V.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 704–737. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_25, https://doi.org/10.1007/978-3-030-56877-1_25
59. Setty, S.T.V., Thaler, J., Wahby, R.S.: Customizable constraint systems for succinct arguments. IACR Cryptol. ePrint Arch. p. 552 (2023), https://eprint.iacr.org/2023/552
60. Towa, P., Vergnaud, D.: Succinct diophantine-satisfiability arguments. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part III.

Lecture Notes in Computer Science, vol. 12493, pp. 774–804. Springer (2020). https://doi.org/10.1007/978-3-030-64840-4_26, https://doi.org/10.1007/978-3-030-64840-4_26
61. Wei, Y., Zhang, X., Deng, Y.: Transparent snarks over galois rings. In: Jager, T., Pan, J. (eds.) Public-Key Cryptography – PKC 2025. pp. 418–451. Springer Nature Switzerland, Cham (2025)
62. Zhang, L., Gao, S., Xiao, B.: Hyperwolf: Efficient polynomial commitment schemes from lattices. IACR Cryptol. ePrint Arch. p. 922 (2025), https://eprint.iacr.org/2025/922
63. Zhao, J., Setty, S.T.V., Cui, W., Zaverucha, G.: Micronova: Folding-based arguments with efficient (on-chain) verification. In: Blanton, M., Enck, W., Nita-Rotaru, C. (eds.) IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025. pp. 1964–1982. IEEE (2025). https://doi.org/10.1109/SP61157.2025.00168, https://doi.org/10.1109/SP61157.2025.00168
64. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Proceedings of the 1979 International Symposium on Symbolic and Algebraic Computation. pp. 216–226. EUROSAM '79 (1979)

A PIOP Preliminaries

A.1 PIOR Product Composition

Here we provide the deferred proof for Theorem 2.

Proof.

$$\mathsf{State}(\mathsf{x},\mathsf{y},\mathsf{tr}) = \begin{cases} 0 & \text{if } \mathsf{tr} = \varnothing \\ \mathsf{State}_1(\mathsf{x},\mathsf{y},\mathsf{tr}) & \text{if } |\mathsf{tr}| \le k_1 \\ \mathsf{State}_j(\mathsf{x},\mathsf{y},\mathsf{tr}) & \text{if } \sum_{k=1}^{j-1} k_k < |\mathsf{tr}| \le \sum_{k=1}^{j} k_k \end{cases}$$

First of all note that the above state function is a valid state function according to Definition 5, because (a) It is zero on empty transcript unconditionally. (b) The prover moves condition holds because each sub-PIOR's state function satisfies the prover moves condition. (c) The full transcript condition holds. We prove this by contradiction. Assume that the full transcript condition does not hold. Then, there exists a full transcript tr such that $\mathsf{State}(\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$ but the verifier outputs $(\mathsf{x}',\mathsf{y}')$ such that $(\mathsf{x}',\mathsf{y}') \in \mathfrak{L}(\mathfrak{R}')$. This means that for all $i \in (m]$, $(\mathsf{x}',\mathsf{y}') \in \mathfrak{L}(\mathfrak{R}'_i)$. This means that for all $i \in (m]$, $\mathsf{State}_i(\mathsf{x},\mathsf{y},\mathsf{tr}_i) = 1$ where tr_i is the sub-transcript corresponding to the i-th sub-PIOR. This contradicts the assumption that $\mathsf{State}(\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$.

Now, we analyze the round-by-round soundness error: If $\mathsf{State}(\mathsf{x},\mathsf{y},\mathsf{tr}) = 0$ for $(\mathsf{x},\mathsf{y}) \notin \mathfrak{L}(\mathfrak{R})$ and a partial transcript $\mathsf{tr} = (\Pi_1, \varrho, \ldots, \Pi_i)$, $i \in (k]$, where $k = \sum_{i\in(m_i]} k_i$ the verifier is about to move, then

$$\Pr_{\varrho_i \leftarrow \mathbb{F}^*}[\mathsf{State}(\mathsf{x},\mathsf{y},\mathsf{tr}||\varrho_i) = 1] = \begin{cases} \epsilon^{(1)}_{\mathsf{RBR}}(\mathsf{x}) & \text{if } i \le k_1 \\ \epsilon^{(j)}_{\mathsf{RBR}}(\mathsf{x}) & \text{if } \sum_{k=1}^{j-1} k_k < i \le \sum_{k=1}^{j} k_k \end{cases}$$

$$\leq \max_{i \in (m]} \epsilon^{(i)}_{\mathsf{RBR}}(\mathsf{x})$$

□

A.2 PIOR trivial extractor

We now state a useful lemma, adapted from [21, Lemma 2.3], establishing that for all protocols considered in this paper, round-by-round knowledge soundness follows directly from round-by-round soundness.

Lemma 2. *Let* PIOR *be a generalized polynomial interactive oracle reduction (PIOR) for an indexed oracle relation* $\mathfrak{R}$ *with round-by-round soundness error* ϵ_{RBR} *and an identity evaluation map* φ*. Assume that for every* $(\mathsf{x}, \mathsf{y}, \mathsf{w}) \in \mathfrak{R}$*, the witness* w *consists solely of polynomials and that the oracle instance* y *provides oracle access to these polynomials. Then* PIOR *satisfies round-by-round knowledge soundness with* $\kappa_{\mathsf{RBR}} = \epsilon_{\mathsf{RBR}}$ *and there exists an extractor that runs in time* $O(|\mathsf{w}|)$*. We refer to relations satisfying the above property as* trivially extractable relations*.*

Proof. We follow the same extraction idea as in the proof of [21, Lemma 2.3], adapted to the round-by-round notion. [21] (see also the proof sketch in [21, L10–L31]).

Extractor. Define an extractor $\mathcal{E}$ that, on input $(\mathsf{x}, \mathsf{y}, \boldsymbol{\Pi}, \mathsf{w}')$, ignores the transcript and reconstructs the witness polynomials directly from the oracle instance y. Concretely, for each witness polynomial f (which is promised to have an associated evaluation oracle $[\![f]\!]$ inside y), $\mathcal{E}$ queries $[\![f]\!]$ on sufficiently many points to recover a full description of f (e.g., its coefficient table in the standard monomial basis). Since φ is the identity, oracle answers are exact field evaluations of f.

If f is μ-variate with individual degree at most d, then f is uniquely determined by its evaluations on any set of $(d+1)^\mu$ points in $\mathbb{F}^\mu$ that forms a full interpolation grid (e.g., a Cartesian product of $(d+1)$ distinct values per coordinate). Thus the extractor can recover each f by multivariate interpolation from $(d+1)^\mu$ oracle queries. In particular, for multilinear f (the case used throughout this paper), it suffices to query all Boolean points, i.e., 2^μ points. Overall, the extractor runs in time linear in the size of the explicit representation of the witness polynomials, i.e., $O(|\mathsf{w}|)$.

Why knowledge error equals soundness error. Fix any round i and any partial transcript tr ending right before the verifier samples ϱ_i, such that $\mathsf{State}(\mathsf{x}, \mathsf{y}, \mathsf{tr}) = 0$. Suppose

$$\Pr_{\varrho_i}\left[\mathsf{State}(\mathsf{x}, \mathsf{y}, \mathsf{tr} \| \varrho_i) = 1\right] > \epsilon_{\mathsf{RBR}}(\mathsf{x}).$$

By definition of round-by-round soundness, the above strict inequality can only occur when $(\mathsf{x}, \mathsf{y}) \in \mathfrak{L}(\mathfrak{R})$: otherwise it would contradict the soundness bound at this point.

Now assume $(\mathsf{x}, \mathsf{y}) \in \mathfrak{L}(\mathfrak{R})$. By the lemma's hypothesis, *every* valid witness for $\mathfrak{R}$ consists only of polynomials that already appear as oracles in y. Hence the extractor described above can reconstruct a candidate witness $\mathsf{w}^\star$ *solely from* y, independently of how the transcript is continued. By construction, $\mathsf{w}^\star$ is the unique tuple of polynomials consistent with the oracles in y (two distinct μ-variate individual-degree-d polynomials cannot agree on a full $(d+1)^\mu$ interpolation grid). Therefore, whenever the instance is in the language, $\mathsf{w}^\star$ is a valid witness and satisfies $(\mathsf{x}, \mathsf{y}, \mathsf{w}^\star) \in \mathfrak{R}$.

This exactly matches the round-by-round knowledge-soundness requirement: whenever a prover can drive the state to 1 with probability exceeding $\epsilon_{\mathsf{RBR}}(\mathsf{x})$ at some verifier move, the extractor outputs a valid witness for the original relation, and it does so for *any* transcript continuation. Thus $\kappa_{\mathsf{RBR}} = \epsilon_{\mathsf{RBR}}$.

Finally, we call relations satisfying the lemma's hypothesis *trivially extractable relations*. □

A.3 From PIOP to Arguments of Knowledge

Construction 6: Argument of Knowledge from PIOP + PC

$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathtt{Setup}(1^\lambda, \mathsf{i})$

1. Infer the maximum degree N of the polynomial oracles sent by the prover in the PIOP from i.
2. Sample the public parameters for the $(\mathsf{ck}_{\mathsf{PCS}}, \mathsf{vk}_{\mathsf{PCS}}) \leftarrow \mathsf{PCS}.\mathtt{Setup}(1^\lambda, N)$.
3. Output $\mathsf{pk} := \mathsf{ck}_{\mathsf{PCS}}$ and $\mathsf{vk} := \mathsf{vk}_{\mathsf{PCS}}$.

$b \leftarrow \langle\ \mathcal{P}(\mathsf{pk}, \mathsf{x}, \mathsf{w})\ ,\ \mathcal{V}(\mathsf{vk}, \mathsf{x})\ \rangle$

1. $\mathcal{P}$ parses the proving key $\mathsf{pk} = \mathsf{ck}_{\mathsf{PCS}}$ and $\mathcal{V}$ parses the verification key $\mathsf{vk} = \mathsf{vk}_{\mathsf{PCS}}$.
2. For each round $i \in k(\mathsf{i})$ of the interactive phase of PIOP :
   (a) $\mathcal{V}$ invokes $\mathcal{V}$ and receives public random challenges ϱ_i, then forwards them to $\mathcal{P}$.
   (b) $\mathcal{P}$ forwards ϱ_i to $\mathcal{P}$ and receives oracle messages Π_i.
   (c) $\mathcal{P}$ commits to polynomials in Π_i using the PCS:

   $$\mathsf{cm}_{\Pi_i} \leftarrow \mathsf{PCS}.\mathtt{Commit}(\mathsf{ck}_{\mathsf{PCS}}, \Pi_i)$$

   and sends it to $\mathcal{V}$.
3. For the querying phase of PIOP:
   (a) $\mathcal{V}$ invokes $\mathcal{V}$ and receives query randomness ϱ_{k+1}, then forwards it to $\mathcal{P}$.
   (b) $\mathcal{P}$ uses the query algorithm of $\mathcal{V}$ to compute the query set $Q := \mathcal{V}_Q(\mathsf{x}; \varrho_1, \ldots, \varrho_{k+1})$
   (c) $\mathcal{P}$ replies to the queries by the evaluation $\nu := \Pi(Q)$.
   (d) $\mathcal{P}$ and $\mathcal{V}$ invoke the PCS evaluation argument for the relation

A.4 Deferred Proofs

A.4.1 Addition PIOR

Proof. PIOR 1 outputs seven instance–witness pairs: one decision check, five binary checks, and one cyclic rotation instance (Definition 38) whose dedicated PIOR is deferred (PIOR 8). We analyze each induced PIOR and then apply product composition.

Decision check. Let

$$g(\boldsymbol{x}) := \widetilde{a}(\boldsymbol{x}) + \widetilde{b}(\boldsymbol{x}) + \widetilde{k}(\boldsymbol{x}) - \widetilde{c}(\boldsymbol{x}) - 2\widetilde{k'}(\boldsymbol{x})$$

be the tested polynomial. Define the one-round state function

$$\mathsf{State}_{\mathsf{Decision}}(\mathsf{tr}) = \begin{cases} 0 & \text{if } \mathsf{tr} = \varnothing \\ 1 & \text{if } \mathsf{tr} = (\boldsymbol{r}) \text{ and } g(\boldsymbol{r}) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Validity is immediate, and the escape probability equals $\Pr[g(\boldsymbol{r}) = 0]$. If the addition relation is false, then g is a nonzero multilinear polynomial in μ variables, hence $\Pr[g(\boldsymbol{r}) = 0] \le \mu/p$ by Lemma 1.

Binary checks. Each binary check is the (non-interactive) PIOR of Theorem 13; the state functions are trivial and the escape probability is 0.

Rotation check. Until PIOR 8 is instantiated, we treat this output as contributing round-by-round error 0 (no verifier randomness is used in the reduction).

Product composition. By Theorem 2, the overall round-by-round soundness error is the maximum of the above components, i.e. μ/p. □

A.4.2 Range PIOR

Proof. PIOR 2 is deterministic and outputs a fixed $\mathfrak{R}_{\mathsf{Add\text{-}Oracle}}$ instance. There is no verifier randomness in the reduction, so a false range instance always maps to a false addition instance. Hence the state function is trivial and the probability of escaping the doomed state is 0. □

A.4.3 Convolution PIOR

Proof of Theorem 6. PIOR 3 outputs four instance–witness pairs: three for the SumCheck relation (binding the committed evaluation oracles $\widetilde{\hat{a}}, \widetilde{\hat{b}}, \widetilde{\hat{s}}$ to the operand and digit polynomials at the random point r) and one for the Hadamard relation (checking the per-batch products). We analyze each and then apply product composition.

Perfect completeness. On an honest execution, for every batch i we have $\hat{a}[i] = f_{a_i}(r)$, $\hat{b}[i] = f_{b_i}(r)$, and $\hat{s}[i] = f_{s_i}(r)$, and $\sum_{\boldsymbol{x}} \widetilde{u}(\boldsymbol{x})\widetilde{a}(\boldsymbol{x}, \boldsymbol{\rho}) = \widetilde{\hat{a}}(\boldsymbol{\rho})$ holds for every $\boldsymbol{\rho}$ (and analogously for b, s), because $\widetilde{u}(\boldsymbol{x}) = r^{\mathrm{int}(\boldsymbol{x})}$ on the hypercube and both sides are multilinear in the batch variables. Since $f_{s_i} = f_{a_i} \cdot f_{b_i}$, we

have $\hat{s}[i] = \hat{a}[i] \cdot \hat{b}[i]$ for all i, so the Hadamard instance is satisfied. Hence all four sub-instances accept and completeness is perfect.

SumCheck reductions. Each of the three sumchecks binds an evaluation oracle to the committed vector: the identity $\sum_{\boldsymbol{x}\in\{0,1\}^\beta} \widetilde{u}(\boldsymbol{x})\widetilde{a}(\boldsymbol{x},\boldsymbol{\rho}) = \widetilde{\hat{a}}(\boldsymbol{\rho})$ (and analogously) is a sumcheck instance whose summand has individual degree at most 2; by Theorem 16 each contributes escape probability at most β/p. The auxiliary randomness $\boldsymbol{\rho}$ over which the two multilinear sides are compared contributes at most τ/p by Lemma 1. Thus each sumcheck component contributes at most μ/p for $\mu = \beta + \tau$.

Hadamard reduction. The Hadamard reduction is analyzed as in Theorem 14 and contributes escape probability at most τ/p.

Soundness. Suppose the convolution relation is false, i.e. $f_{s_i} \neq f_{a_i} \cdot f_{b_i}$ for some batch i. Both sides are univariate polynomials of degree $< B$, so $f_{s_i}(r) \neq f_{a_i}(r) \cdot f_{b_i}(r)$ except with probability at most B/p over r by Lemma 1; a union bound over the t batches gives tB/p. Conditioned on this event, the committed evaluation oracles (bound to the true evaluations by the sumchecks) violate the Hadamard identity, so the Hadamard instance is false. By Theorem 2, the overall round-by-round soundness error is at most $\max\{tB/p,\ \mu/p\}$. □

A.4.4 Integer Multiplication PIOR

We first record the bound on the multiplication carries, which underlies both the completeness and the soundness of PIOR 4.

Lemma 3 (Multiplication carry bound). *Fix a batch index i and suppose the operands encoded by $\boldsymbol{a}_i, \boldsymbol{b}_i \in \{0,1\}^B$ have bit-length at most $B/2$ (as enforced by Item 2). Then the schoolbook digits satisfy $0 \le s_i[j] \le B/2$ and the carries produced by the recurrence of PIOR 4 with $k_i[0] = 0$ satisfy*

$$0 \le k_i[j] \le B/2 - 1 < 2^w, \qquad w = \beta - 1,$$

for every index j. In particular each carry admits a w-bit binary decomposition.

Proof. Each digit $s_i[j] = \sum_{i'+j'=j} a_i[i']\, b_i[j']$ is a sum of products of bits; since both operands have at most $B/2$ nonzero positions, at most $B/2$ terms are nonzero, so $0 \le s_i[j] \le B/2$. Write $M := B/2$. We show $k_i[j] \le M - 1$ by induction on j. The base case $k_i[0] = 0 \le M - 1$ holds. Assuming $k_i[j] \le M - 1$, and using $c_i[j] \in \{0,1\}$ so that the numerator is nonnegative,

$$k_i[j+1] = \left\lfloor \tfrac{s_i[j] + k_i[j]}{2} \right\rfloor \le \left\lfloor \tfrac{M + (M-1)}{2} \right\rfloor = \left\lfloor \tfrac{2M-1}{2} \right\rfloor = M - 1.$$

Hence $0 \le k_i[j] \le M - 1 = B/2 - 1 < 2^w$ for all j. □

Proof of Theorem 7. PIOR 4 outputs instance–witness pairs for two zero checks, one convolution check, two decision checks (the ripple-carry identity and the carry bit-decomposition reconstruction identity), $3 + w$ binary checks (on the operands, the product, and the $w = \beta - 1$ carry bit-decomposition vectors), and one cyclic rotation instance (deferred as in appendix A.4.1).

Perfect completeness. On an honest execution the schoolbook digits, product bits, and carries are computed by the integer recurrence, so the ripple-carry decision identity $\widetilde{s}(\boldsymbol{r}) + \widetilde{k}(\boldsymbol{r}) = \widetilde{c}(\boldsymbol{r}) + 2\,\widetilde{k'}(\boldsymbol{r})$ holds at every point. By Lemma 3 each honest carry lies in $[0, B/2) = [0, 2^w)$, so its w-bit binary decomposition exists; the resulting bit-decomposition vectors $\kappa^{(t)}$ are genuinely $\{0,1\}$-valued (hence pass their binary checks) and satisfy the reconstruction identity $\widetilde{k}(\boldsymbol{x}) = \sum_t 2^t \widetilde{\kappa^{(t)}}(\boldsymbol{x})$ on the whole hypercube (hence pass the second decision check). The remaining components accept exactly as before, so completeness is perfect. This corrects the earlier attempt to binary-check the carry directly: unlike the addition carry, the multiplication carry is not binary, and only its w-bit bounded decomposition is.

Round-by-round soundness. The zero reductions are deterministic, so those components contribute round-by-round soundness error 0 to the product composition. Both decision checks are analyzed as in appendix A.4.1, each contributing escape probability at most μ/p for $\mu = \beta + \tau$. Each binary check contributes 0 as in Theorem 13; the bit-decomposition-vector binary checks together with the reconstruction decision check bound every carry entry to $[0, 2^w)$, and the cyclic-rotation relation transfers this bound to the rotated carry, which is what the ripple-carry identity needs to force a faithful integer normalization. The convolution check certifies that the schoolbook digit vector is the convolution of the two operands; by Theorem 6 its round-by-round soundness error is at most $\max\{tB/p,\, \mu/p\}$. By Theorem 2, the overall round-by-round soundness error is at most $\max\{\mu/p,\, tB/p\}$. □

A.4.5 Integer Division PIOR

Proof. PIOR 5 composes two zero checks with the reductions underlying PIOR 4 and PIOR 1. The zero checks are deterministic with round-by-round error 0. The multiplication and addition components inherit the analyses of appendix A.4.4 and appendix A.4.1, respectively. By Theorem 2, the overall round-by-round soundness error is at most the maximum of the two composed bounds, i.e. $\max\{\mu/p,\, tB/p\}$ for $\mu = \beta + \tau$. □

A.4.6 MaSDAK

Proof. **Round-by-round soundness.** PIOR 6 outputs two instance–witness pairs: one for the decision relation (the modular equation check) and one for the division relation. We analyze each induced PIOR and then apply product composition.

Decision check. Let $v := f(\boldsymbol{v}) \in \mathbb{Z}$ be the value of the Diophantine function on the true integer witness. If the statement is false then $v \neq 0$. The verifier samples a random prime $\ell \in \mathsf{Primes}[\gamma]$ and accepts the decision instance iff $v \equiv 0 \pmod{\ell}$. For nonzero $v \in \mathbb{Z}$, each prime divisor $p \mid v$ with $p \geq 2^{\gamma-1}$ contributes at least $\gamma - 1$ to $\log_2 |v|$, so v has at most $\lfloor \log_2 |v| / (\gamma - 1) \rfloor$ distinct such prime divisors. Hence

$$\Pr_{\ell \leftarrow \mathsf{Primes}[\gamma]}[v \equiv 0 \bmod \ell] = \frac{\left|\{p \in \mathsf{Primes}[\gamma] : p \mid v\}\right|}{|\mathsf{Primes}[\gamma]|} \leq \frac{\lfloor \log_2(1 + |v|)/(\gamma - 1) \rfloor}{|\mathsf{Primes}[\gamma]|}.$$

Under the theorem's assumption $|v| < 2^A$ with $A := s\,n\,t$, this is at most $\frac{A}{(\gamma-1)|\mathsf{Primes}[\gamma]|}$.

Division check. This component reduces to PIOR 5, which by appendix A.4.5 has round-by-round soundness error at most $\max\{\mu/p,\, tB/p\}$ for $\mu = \beta + \tau$.

Product composition. By Theorem 2, the overall round-by-round soundness error is the maximum of the two errors, i.e. $\max\{\frac{A}{(\gamma-1)|\mathsf{Primes}[\gamma]|},\, \mu/p,\, tB/p\}$ for A as in Theorem 9. □

A.5 PIOP Toolbox

A.5.1 LogUp Lookup Check

We adopt the multivariate logarithmic-derivative lookup protocol of Haböck [41] (commonly called *LogUp*) and rephrase it in our PIOR framework as a reduction to zero-checks and sumchecks. Lookup is the main subprotocol behind our cyclic rotation check (appendix A.5.2).

Definition 37 (Lookup oracle relation). *Fix $\mu_f, \mu_t \in \mathbb{N}$ and let $N_f = 2^{\mu_f}$, $N_t = 2^{\mu_t}$. The lookup oracle relation $\mathfrak{R}_{\mathsf{Lookup}}$ is the set of all triples of the form $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = ((\mu_f, \mu_t), ([\![f]\!], [\![t]\!]), m)$ where $f \in \mathbb{F}^{\mathsf{ind}\le 1}_{p,\mu_f}$ and $t, m \in \mathbb{F}^{\mathsf{ind}\le 1}_{p,\mu_t}$ such that, as rational functions in the formal variable X,*

$$\sum_{x\in\{0,1\}^{\mu_f}} \frac{1}{X - f(x)} \;=\; \sum_{y\in\{0,1\}^{\mu_t}} \frac{m(y)}{X - t(y)}.$$

By [41, Lemma 5], the relation holds if and only if the value-multiset of f on $\{0,1\}^{\mu_f}$ is contained in the value-multiset of t on $\{0,1\}^{\mu_t}$, with $m(y)$ recording, for each y, the multiplicity with which the value $t(y)$ is matched by entries of f. Setting $m \equiv 1$ recovers the multiset-equality case used by our rotation reduction.

PIOR 7: $\mathfrak{R}_{\mathsf{Lookup}} \to \mathfrak{R}^2_{\mathsf{Zero}} \times \mathfrak{R}^2_{\mathsf{Sum}}$

Parse: $\mathsf{x} = (\mu_f, \mu_t)$, $\mathsf{y} = ([\![f]\!], [\![t]\!])$, $\mathsf{w} = m$

1. $\mathcal{V}$ samples a random $\alpha \in \mathbb{F}_p$ and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ constructs the helper multilinear polynomials $h_f \in \mathbb{F}^{\mathsf{ind}\le 1}_{p,\mu_f}$ and $h_t \in \mathbb{F}^{\mathsf{ind}\le 1}_{p,\mu_t}$ defined by
$$h_f(x) := \frac{1}{\alpha - f(x)}, \quad h_t(y) := \frac{m(y)}{\alpha - t(y)},$$
and sends the oracles $[\![h_f]\!]$ and $[\![h_t]\!]$ to $\mathcal{V}$. $\mathcal{P}$ additionally sends $\mathcal{V}$ the field element $s := \sum_{x\in\{0,1\}^{\mu_f}} h_f(x)$.
3. To enforce $h_f(x)\cdot(\alpha - f(x)) - 1 = 0$ on $\{0,1\}^{\mu_f}$, $\mathcal{P}$ outputs the witness for the zero relation (Definition 41) as $\mathsf{w}'_{\mathsf{Zero},f} := p_f$, where $p_f(\boldsymbol{x}) := h_f(\boldsymbol{x})\cdot(\alpha - f(\boldsymbol{x})) - 1$, while $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Zero},f}, \mathsf{y}'_{\mathsf{Zero},f}) := (\emptyset, [\![p_f]\!])$.
4. To enforce $h_t(y)\cdot(\alpha - t(y)) - m(y) = 0$ on $\{0,1\}^{\mu_t}$, $\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Zero},t} := p_t$, where $p_t(\boldsymbol{y}) := h_t(\boldsymbol{y})\cdot(\alpha - t(\boldsymbol{y})) - m(\boldsymbol{y})$, while $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Zero},t}, \mathsf{y}'_{\mathsf{Zero},t}) := (\emptyset, [\![p_t]\!])$.

5. To enforce $\sum_x h_f(x) = s$, $\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Sum},f} := h_f$ while $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Sum},f}, \mathsf{y}'_{\mathsf{Sum},f}) := (s, [\![h_f]\!])$.
6. To enforce $\sum_y h_t(y) = s$, $\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Sum},t} := h_t$ while $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Sum},t}, \mathsf{y}'_{\mathsf{Sum},t}) := (s, [\![h_t]\!])$.

Theorem 11. *PIOR 7 from $\mathfrak{R}_{\mathsf{Lookup}}$ to $\mathfrak{R}^2_{\mathsf{Zero}} \times \mathfrak{R}^2_{\mathsf{Sum}}$ has perfect completeness and round-by-round soundness error at most $(N_f + N_t)/p$, with the following efficiency measures:*

rounds	*message size*	*queries*	*oracle size*	*prover time*	*verifier time*
1	1	0	2	$O(N_f + N_t)$	$O(\mu_f + \mu_t)$

Proof. Completeness and the efficiency measures follow by construction: an honest prover sets h_f and h_t to their canonical values, after which p_f and p_t vanish identically on their hypercubes, and the two sums equal the common value s guaranteed by the lookup relation evaluated at $X = \alpha$.

Round-by-round soundness. The only verifier randomness in the reduction is α. Suppose the relation does not hold; that is, the rational function $\Delta(X) := \sum_x \frac{1}{X - f(x)} - \sum_y \frac{m(y)}{X - t(y)}$ is not the zero rational function. Clearing denominators yields a univariate polynomial of degree at most $N_f + N_t - 1$ which is not identically zero, and by Schwartz–Zippel (Lemma 1) it vanishes at a uniform $\alpha \in \mathbb{F}_p$ with probability at most $(N_f + N_t)/p$. Whenever $\Delta(\alpha) \neq 0$, any choice of h_f, h_t, s that satisfies both zero-checks must also fail at least one of the two sumcheck instances, so at least one output sub-instance is false. The reduction is therefore round-by-round sound with error at most $(N_f + N_t)/p$. □

A.5.2 Rotation check

Definition 38 (Cyclic rotation oracle relation). *Fix $\mu \in \mathbb{N}$ and let $N = 2^\mu$. The oracle relation $\mathfrak{R}_{\mathsf{rot-Oracle}}$ is the set of all triples $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (\sigma, ([\![\widetilde{u}]\!], [\![\widetilde{v}]\!]), (\widetilde{u}, \widetilde{v}))$ such that $\sigma \in \{0, 1, \ldots, N-1\}$, the polynomials $\widetilde{u}, \widetilde{v} \in \mathbb{F}^{\mathsf{ind} \leq 1}_{p,\mu}$ have evaluation vectors $(u_0, \ldots, u_{N-1})$ and $(v_0, \ldots, v_{N-1})$ on $\{0,1\}^\mu$ in lexicographic order, and*

$$v_i = u_{(i+\sigma) \bmod N} \qquad \forall i \in \{0, \ldots, N-1\}.$$

The rotation relation reduces to a single lookup instance via a standard tagging trick. Let $\mathcal{I} \in \mathbb{F}^{\mathsf{ind} \leq 1}_{p,\mu}$ be the index multilinear polynomial $\mathcal{I}(\boldsymbol{x}) := \sum_{i=1}^{\mu} 2^{i-1} x_i$, which maps each $\boldsymbol{x} \in \{0,1\}^\mu$ to its lexicographic index in $\{0, \ldots, N-1\}$. Observe that the rotation condition $v_i = u_{(i+\sigma) \bmod N}$ is equivalent to the equality of the two multisets of tagged pairs over $\{0, \ldots, N-1\} \times \mathbb{F}_p$

$$\big\{ (i, v_i) \,:\, i \in \{0, \ldots, N-1\} \big\} \;=\; \big\{ ((j-\sigma) \bmod N,\, u_j) \,:\, j \in \{0, \ldots, N-1\} \big\},$$

since the substitution $j = (i + \sigma) \bmod N$ is a bijection on $\{0, \ldots, N-1\}$ that aligns the two sides whenever the per-index identity holds. We fingerprint each tagged pair into a single field element using a random coefficient $\gamma \in \mathbb{F}_p$ supplied by the verifier, reducing multiset equality of tagged pairs to multiset equality of fingerprints, which is exactly a lookup with constant-1 multiplicities.

PIOR 8: $\mathfrak{R}_{\mathsf{rot-Oracle}} \to \mathfrak{R}_{\mathsf{Lookup}}$[51]

Parse: $\mathsf{x} = \sigma$, $\mathsf{y} = ([\![\widetilde{u}]\!], [\![\widetilde{v}]\!])$, $\mathsf{w} = (\widetilde{u}, \widetilde{v})$

1. $\mathcal{V}$ samples a random $\gamma \in \mathbb{F}_p$ and sends it to $\mathcal{P}$.
2. Both parties locally define the fingerprinted polynomials

$$f(\boldsymbol{x}) := \gamma \cdot \mathcal{I}(\boldsymbol{x}) + \widetilde{v}(\boldsymbol{x}), \quad t(\boldsymbol{x}) := \gamma \cdot \big((\mathcal{I}(\boldsymbol{x}) - \sigma) \bmod N\big) + \widetilde{u}(\boldsymbol{x}).$$

   Since $\mathcal{I}$ and the shifted-index expression have a succinct closed form, $\mathcal{V}$ can answer any query to $[\![f]\!]$ or $[\![t]\!]$ locally using $[\![\widetilde{u}]\!]$ and $[\![\widetilde{v}]\!]$.
3. To enforce the multiset-equality of the tagged pairs, $\mathcal{P}$ outputs the witness for the lookup relation (Definition 37) as $\mathsf{w}'_{\mathsf{Lookup}} := m \equiv 1$ (the constant-1 multiplicity polynomial), while $\mathcal{V}$ outputs the corresponding explicit and implicit instances $(\mathsf{x}'_{\mathsf{Lookup}}, \mathsf{y}'_{\mathsf{Lookup}}) := \big((\mu, \mu), ([\![f]\!], [\![t]\!])\big)$.

Theorem 12. *PIOR 8 from $\mathfrak{R}_{\mathsf{rot-Oracle}}$ to $\mathfrak{R}_{\mathsf{Lookup}}$ has perfect completeness and round-by-round soundness error at most N/p, with the following efficiency measures:*

rounds	*message size*	*queries*	*oracle size*	*prover time*	*verifier time*
1	0	0	0	$O(1)$	$O(\mu)$

Proof. Completeness and efficiency follow by construction: when the rotation relation holds, the multisets of tagged pairs match exactly, so the fingerprinted multisets agree as well, and the lookup instance with $m \equiv 1$ is a true instance of $\mathfrak{R}_{\mathsf{Lookup}}$. The prover sends no oracles in this step, and the verifier only stores the description of the virtual oracles $[\![f]\!]$ and $[\![t]\!]$.

Round-by-round soundness. The only verifier randomness in the reduction is γ. Suppose the rotation relation does not hold; then the two multisets of tagged pairs differ, and there exists at least one tagged pair appearing in one multiset but not in the other (with multiplicities counted). Treating $f(\boldsymbol{x}) - \gamma\mathcal{I}(\boldsymbol{x}) - \widetilde{v}(\boldsymbol{x})$ and the analogous expression on the right as univariate polynomials in γ, a standard Reed–Solomon fingerprinting argument shows that the value-multisets of f and t over $\{0,1\}^\mu$ differ as multisets except with probability at most N/p over a uniform $\gamma \in \mathbb{F}_p$. Whenever the fingerprinted multisets differ, the output lookup instance with $m \equiv 1$ is false. □

A.5.3 BinaryCheck

Definition 39. *The oracle relation $\mathfrak{R}_{\mathsf{Bin}}$ is the set of all triples of the form $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (\emptyset, [\![f]\!], f)$ where $f \in \mathbb{F}_{p,\mu}^{\mathsf{ind} \le d}[\boldsymbol{X}]$ such that for every $\boldsymbol{x} \in \{0,1\}^\mu$, $f(\boldsymbol{x}) \in \{0,1\}$.*

PIOR 9: $\mathfrak{R}_{\mathsf{Bin}} \to \mathfrak{R}_{\mathsf{Zero}}$[51]

Parse: $\mathsf{x} = \emptyset$, $\mathsf{y} = [\![f]\!]$, $\mathsf{w} = f$

1. $\mathcal{P}$ computes $f' := f^2 - f$ where $f' \in \mathbb{F}_{p,\mu}^{\mathsf{ind}\leq 2}$ and $\mathcal{V}$ virtually computes the oracle $[\![f']\!]$.
2. $\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Zero}} := f'$ and $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Zero}}, \mathsf{y}'_{\mathsf{Zero}}) := (\emptyset, [\![f']\!])$.

Theorem 13. *PIOR 9 from $\mathfrak{R}_{\mathsf{Bin}}$ (defined in Definition 39) to $\mathfrak{R}_{\mathsf{Zero}}$ (defined in Definition 41) has perfect completeness and round-by-round soundness error 0 with the following efficiency measures:*

rounds	*message size*	*queries*	*oracle size*	*prover time*	*verifier time*
0	0	0	1	$O(N)$	$O(1)$

Proof. The perfect completeness and efficiency measures follow directly from the construction. We now prove the round-by-round soundness for the PIOR.

Round-by-round soundness. The reduction in PIOR 9 is non-interactive: it deterministically maps f to $f' := f^2 - f$ and outputs the ZeroCheck instance $(\emptyset, [\![f']\!])$. Therefore, for any false binary claim, the produced ZeroCheck instance is false and there is no verifier randomness that could turn it into a true instance. Hence the round-by-round soundness error is 0. □

A.5.4 HadamardCheck

Definition 40. *The oracle relation $\mathfrak{R}_{\mathsf{Had}}$ is the set of all triples of the form $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (\varnothing, ([\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{c}]\!], [\![\widetilde{d}]\!]), (\widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{d}))$ where $\widetilde{a}, \widetilde{b}, \widetilde{d}, \widetilde{c} \in \mathbb{F}_{p,\mu}^{\mathsf{ind}\leq 1}$, such that*

$$\boldsymbol{d} = \boldsymbol{a} \circ \boldsymbol{b} + \boldsymbol{c}$$

where $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{d}$, and $\boldsymbol{c}$ are the evaluation of $\widetilde{a}$, $\widetilde{b}$, $\widetilde{d}$, and $\widetilde{c}$ on the boolean hypercube $\{0,1\}^\mu$ respectively.

PIOR 10: $\mathfrak{R}_{\mathsf{Had}} \to \mathfrak{R}_{\mathsf{Decision}}$[58]

Parse: $\mathsf{x} = \varnothing$, $\mathsf{y} = [\![\widetilde{a}]\!], [\![\widetilde{b}]\!], [\![\widetilde{c}]\!], [\![\widetilde{d}]\!]$, $\mathsf{w} = \widetilde{a}, \widetilde{b}, \widetilde{c}, \widetilde{d}$

1. $\mathcal{V}$ samples a random $\boldsymbol{r} \in \mathbb{F}_p^\mu$ and queries the oracles $[\![\widetilde{a}]\!]$, $[\![\widetilde{b}]\!]$, $[\![\widetilde{c}]\!]$, and $[\![\widetilde{d}]\!]$ at $\boldsymbol{r}$ to obtain $a := \widetilde{a}(\boldsymbol{r})$, $b := \widetilde{b}(\boldsymbol{r})$, $c := \widetilde{c}(\boldsymbol{r})$, and $d := \widetilde{d}(\boldsymbol{r})$.
2. $\mathcal{P}$ outputs $\mathsf{w}'_{\mathsf{Decision}} := \varnothing$. If $a \cdot b + c = d$, then $\mathcal{V}$ outputs $(\mathsf{x}'_{\mathsf{Decision}}, \mathsf{y}'_{\mathsf{Decision}}) := (\mathsf{accept}, \varnothing)$; otherwise it outputs $(\mathsf{reject}, \varnothing)$.

Theorem 14. *PIOR 10 has perfect completeness and round-by-round soundness error at most μ/p with the following efficiency measures:*

rounds	*#queries*	*oracle size*	*prover time*	*verifier time*
1	4	0	$O(1)$	$O(\mu)$

Proof. The perfect completeness and efficiency measures follow directly from the construction. We now prove the round-by-round soundness.

Round-by-round soundness. Let $h(\boldsymbol{x}) := \widetilde{a}(\boldsymbol{x})\widetilde{b}(\boldsymbol{x}) + \widetilde{c}(\boldsymbol{x}) - \widetilde{d}(\boldsymbol{x})$. If the relation is false, then h is a nonzero multilinear polynomial, so $\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^\mu}[h(\boldsymbol{r}) = 0] \leq \mu/p$ by Lemma 1. □

A.5.5 ZeroCheck

For completeness, we restate the zerocheck relation, protocol, and its properties here. The zerocheck protocol is a fundamental building block for all of our protocols.

Definition 41. *The oracle relation* $\mathfrak{R}_{\mathsf{zero}}$ *is the set of all tuples of the form* $(\mathsf{x}, \mathsf{w}) = (\llbracket p \rrbracket, p)$ *where* $p \in \mathbb{F}_{p,\mu}^{\mathsf{ind} \leq d}[\boldsymbol{X}]$ *such that* $\forall \boldsymbol{x} \in \{0,1\}^\mu, p(\boldsymbol{x}) = 0$.

PIOR 11: $\mathfrak{R}_{\mathsf{Zero}} \to \mathfrak{R}_{\mathsf{sum}}$**[58]**

Parse: $\mathsf{x} = \emptyset$, $\mathsf{y} = \llbracket f \rrbracket$, $\mathsf{w} = f$

1. $\mathcal{V}$ samples a random point $\boldsymbol{r} \in \mathbb{F}_p^\mu$ and sends it to $\mathcal{P}$.
2. To verify that p is zero on the boolean hypercube, $\mathcal{P}$ outputs the witness for the sumcheck relation (Definition 42) as $\mathsf{w}'_{\mathsf{Sum}} := f$ while $\mathcal{V}$ outputs the corresponding explicit and implicit instance as $(\mathsf{x}'_{\mathsf{Sum}}, \mathsf{y}'_{\mathsf{Sum}}) := (0, \llbracket f \rrbracket)$, where f is the virtual polynomial defined as $f(\boldsymbol{x}) := p(\boldsymbol{x}) \cdot \mathsf{eq}(\boldsymbol{x}, \boldsymbol{r})$.

Theorem 15. *PIOR 11 has perfect completeness and round-by-round soundness error at most* $\mu/|\mathbb{F}|$ *with the following efficiency measures:*

rounds	*queries*	*oracle size*	*prover time*	*verifier time*
1	0	0	$O(N)$	$O(\mu)$

Proof. See [21]. Since PIOR 11 is a PIOR *to* SumCheck, we only analyze the verifier round that samples $\boldsymbol{r}$ and produces a SumCheck instance. To fit the round-by-round definition, we consider a functionally equivalent variant with an initial empty prover message $\Pi_1 = \varnothing$. We define the state function as

$$\mathsf{State}(\mathsf{x} = \llbracket p \rrbracket, \mathsf{y} = \emptyset, \mathsf{tr}) = \begin{cases} 0 & \mathsf{tr} = \varnothing \\ 0 & \mathsf{tr} = (\Pi_1) \\ 0 & \mathsf{tr} = (\Pi_1, \varrho_1 = \{\boldsymbol{r}\}) \wedge \sum\limits_{\boldsymbol{x} \in \{0,1\}^\mu} p(\boldsymbol{x}) \mathsf{eq}(\boldsymbol{x}, \boldsymbol{r}) \neq 0 \\ 1 & \mathsf{tr} = (\Pi_1, \varrho_1 = \{\boldsymbol{r}\}) \wedge \sum\limits_{\boldsymbol{x} \in \{0,1\}^\mu} p(\boldsymbol{x}) \mathsf{eq}(\boldsymbol{x}, \boldsymbol{r}) = 0. \end{cases}$$

This state function satisfies Definition 5. The round-by-round soundness error is the probability that a false zero-check claim is reduced to a true SumCheck instance:

$$\begin{aligned} \epsilon_{\mathsf{PIOP}}^{\mathsf{rbr}} &= \Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}_p^\mu} \left[\sum_{\boldsymbol{x} \in \{0,1\}^\mu} p(\boldsymbol{x}) \mathsf{eq}(\boldsymbol{x}, \boldsymbol{r}) = 0 \;\middle|\; \exists \boldsymbol{x} \in \{0,1\}^\mu : \; p(\boldsymbol{x}) \neq 0 \right] \\ &\leq \frac{\mu}{|\mathbb{F}|} \qquad \text{Schwartz-Zippel (Lemma 1)} \end{aligned}$$

□

Remark 7 (Batching ZeroCheck claims). To prove multiple ZeroCheck claims $\{(\llbracket p_i \rrbracket, p_i)\}_{i=1}^t$ where each p_i is a μ-variate polynomial with individual degree d, $\mathcal{V}$ can sample t random field elements $\{c_i\}_{i=1}^t$ and run a single ZeroCheck on the polynomial $p^* = \sum_{i=1}^t c_i p_i$.

A.5.6 SumCheck

For completeness, we restate the sumcheck relation, protocol, and it's properties here. The sumcheck protocol is a fundamental building block for all of our protocols.

Definition 42. *The oracle relation* $\mathfrak{R}_{\mathsf{sum}}$ *is the set of all tuples of the form* $(\mathsf{x}, \mathsf{y}, \mathsf{w}) = (v, [\![f]\!], f)$ *where* $v \in \mathbb{F}_p$ *is the sum and* $f \in \mathbb{F}_{p,\mu}^{\mathsf{ind} \leq d}[\boldsymbol{X}]$ *is such that* $\sum_{\boldsymbol{x} \in \{0,1\}^\mu} f(\boldsymbol{x}) = v$.

PIOR 12: $\mathfrak{R}_{\mathsf{Sum}} \to \varnothing$ **([49])**

Parse: $\mathsf{x} = v$, $\mathsf{y} = [\![f]\!]$, $\mathsf{w} = f$

1. $\mathcal{P}$ sends to $\mathcal{V}$ a univariate polynomial f_1 such that

$$f_1(X_1) = \sum_{\{x_2, \ldots, x_\mu\} \in \{0,1\}^{\mu-1}} f(X_1, x_2, \ldots, x_\mu)$$

2. $\mathcal{V}$ checks that the degree of $f_1(x_1)$ is at most the degree of f in x_1. It computes $v_1 = f_1(0) + f_1(1)$ and checks that $v_1 = v$. If not, $\mathcal{V}$ rejects. It then picks a random $r_1 \in \mathbb{F}$ and sends it to the Prover ($\mathcal{P}$).
3. For each i from 2 to μ (subsequent rounds):
   (a) $\mathcal{P}$ sends to $\mathcal{V}$ the univariate polynomial

$$f_i(x_i) = \sum_{\{x_{i+1}, \ldots, x_\mu\} \in \{0,1\}^{\mu-i}} f(r_1, \ldots, r_{i-1}, x_i, \ldots, x_\mu)$$

   (b) $\mathcal{V}$ checks that the degree of $f_i(x_i)$ is at most the degree of f in x_i. It computes $v_i = f_i(0) + f_i(1)$ and checks that $v_i = f_{i-1}(r_{i-1})$. If not, $\mathcal{V}$ rejects. It picks a random $r_i \in \mathbb{F}_p$ and sends it to $\mathcal{P}$.
4. $\mathcal{P}$ sends the value $f(r_1, r_2, \ldots, r_{\mu-1}, r_\mu)$ for the random values $r_1, r_2, \ldots, r_\mu$ chosen by $\mathcal{V}$.
5. $\mathcal{V}$ checks that this value is correct by querying $[\![f]\!]$ at $(r_1, r_2, \ldots, r_{\mu-1}, r_\mu)$.

Theorem 16. *Let* $d := \max_{i \in [\mu]} \deg_i(f)$ *be the maximum individual degree of* f*. PIOR 12 has perfect completeness and* $d/|\mathbb{F}|$ *round-by-round soundness error with the following efficiency measures:*

rounds	*#queries*	*oracle size*	*prover time*	*verifier time*
μ	$\mu + 1$	$o(d\mu)$	$O(Nd \log^2 d)$	$O(d\mu)$

Proof. See [21]. We follow the proof strategy of Canetti et al. [19] in their proof of round-by-round soundness for the GKR protocol [37].

We define a partial transcript as doomed if the initial claim for the corresponding step in the recursion is false. More formally, the state function $\mathsf{State}_{\mathsf{PIOP}}^{\mathsf{rbr}}$ is as follows:

$\mathsf{State}_{\mathsf{PIOP}}^{\mathsf{rbr}}(\mathbb{x} = ([\![p]\!], v), \mathsf{tr}) :=$

$$\begin{cases} 0 & (\mathsf{tr} = \varnothing \vee \mathsf{tr} = p_1) \wedge \sum_{x_1,\ldots,x_\mu \in \{0,1\}^\mu} p(x_1,\ldots,x_\mu) \neq v \\ 1 & (\mathsf{tr} = \varnothing \vee \mathsf{tr} = p_1) \wedge \sum_{x_1,\ldots,x_\mu \in \{0,1\}^\mu} p(x_1,\ldots,x_\mu) = v \\ 0 & \mathsf{tr} = (p_1,\rho_1,\ldots,p_i,\rho_i) \wedge \sum\limits_{\substack{x_{i+1},\ldots,x_\mu \\ \in \{0,1\}^{\mu-i}}} p(\rho_1,\ldots,\rho_i,x_{i+1},\ldots,x_\mu) \neq p_i(\rho_i) \\ 1 & \mathsf{tr} = (p_1,\rho_1,\ldots,p_i,\rho_i) \wedge \sum\limits_{\substack{x_{i+1},\ldots,x_\mu \\ \in \{0,1\}^{\mu-i}}} p(\rho_1,\ldots,\rho_i,x_{i+1},\ldots,x_\mu) = p_i(\rho_i) \end{cases}.$$

It is straightforward to see that the state function $\mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}$ satisfies the first two conditions Definition 5 by construction. For the third condition, i.e. the full transcript, if the full transcript is doomed, we have that $p(\rho_1,\ldots,\rho_\mu) \neq p_\mu(\rho_\mu)$, which, by definition, is the same as the condition for the sumcheck verifier to reject. Hence, $\mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}$ is a valid round by round state function. We are left to compute the round-by-round soundness error. We do this for round $i = 1$ and $i > 1$ separately:

Round 1. Let $\tilde{p}_i(X) := \sum_{x_{i+1},\ldots,x_\mu \in \{0,1\}^{\mu-i}} p(\rho_1,\ldots,\rho_{i-1},X,x_{i+1},\ldots,x_\mu)$. Using this notation, and upon inlining the first two cases of $\mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}$ definition into Eq. (3), allows us to rewrite Eq. (3) as follows:

$$\begin{aligned} \epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},1}(\mathbb{x}) &= \Pr\left[\begin{matrix} \mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}([\![p]\!],\varnothing) = 0 \\ \mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}([\![p]\!],(p_1,\rho_1)) = 1 \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \mathbb{F} \end{matrix}\right] \\ &= \Pr\left[\begin{matrix} \sum_{x_1,\ldots,x_\mu \in \{0,1\}^\mu} p(x_1,\ldots,x_\mu) \neq v \\ \sum_{x_2,\ldots,x_\mu \in \{0,1\}^{\mu-1}} p(\rho_1,x_2,\ldots,x_\mu) = p_1(\rho_1) \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \mathbb{F} \end{matrix}\right] \\ &= \Pr\left[\begin{matrix} \tilde{p}_1(0) + \tilde{p}_1(1) \neq v \\ \tilde{p}_1(\rho_1) = p_1(\rho_1) \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \{0,1\}^{\mathbb{F}^*} \end{matrix}\right] \end{aligned}$$

Now since the prover was able to convince the verifier in the first round, this implies that $p_1(0) + p_1(1) = v$, and so we have:

$$\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},1}(\mathbb{x}) = \Pr\left[\begin{matrix} \tilde{p}_1(0) + \tilde{p}_1(1) \neq p_1(0) + p_1(1) \\ \tilde{p}_1(\rho_1) = p_1(\rho_1) \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \{0,1\}^{\mathbb{F}^*} \end{matrix}\right]$$

By conditioning on event $p_1 \neq \tilde{p}_1$ we have:

$$\begin{aligned} \epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},1}(\mathbb{x}) &= \Pr\left[\begin{matrix} \tilde{p}_1(0) + \tilde{p}_1(1) \neq p_1(0) + p_1(1) \\ \tilde{p}_1(\rho_1) = p_1(\rho_1) \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \{0,1\}^{\mathbb{F}^*} \end{matrix}\right] \Pr\left[p \not= \tilde{p}_1\right] \\ &+ \Pr\left[\begin{matrix} \tilde{p}_1(0) + \tilde{p}_1(1) \neq p_1(0) + p_1(1) \\ \tilde{p}_1(\rho_1) = p_1(\rho_1) \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \{0,1\}^{\mathbb{F}^*} \end{matrix}\right] \Pr\left[p_1 \neq \tilde{p}_1\right] \\ &= \Pr\left[\begin{matrix} \tilde{p}_1(0) + \tilde{p}_1(1) \neq p_1(0) + p_1(1) \\ \tilde{p}_1(\rho_1) = p_1(\rho_1) \end{matrix} \;\middle|\; \begin{matrix} p_1 \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_1 \leftarrow \{0,1\}^{\mathbb{F}^*} \end{matrix}\right] \Pr\left[p_1 \neq \tilde{p}_1\right] \end{aligned}$$

Finally, applying the Lemma 1 gives that $\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},1}(\mathbb{x}) = \deg(p_1)/|\mathbb{F}|$.

Round $i > 1$. Repeating the steps for the case $i = 1$ and inlining the last two cases of $\mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}$ definition into Eq. (3), we can rewrite Eq. (3) as:

$$\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},i}(\mathbb{x}) = \Pr\left[\begin{matrix} \mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}([\![p]\!],(p_1,\rho_1,\ldots,p_i)) = 0 \\ \mathsf{State}^{\mathsf{rbr}}_{\mathsf{PIOP}}([\![p]\!],(p_1,\rho_1,\ldots,p_i,\rho_i)) = 1 \end{matrix} \;\middle|\; \begin{matrix} ([p_j]_{j=1}^{i},[\rho_j]_{j=1}^{i-1}) \\ \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_i \leftarrow \mathbb{F} \end{matrix}\right]$$

$$= \Pr\left[\begin{array}{l} \sum_{\substack{x_i,\ldots,x_\mu \\ \in\{0,1\}^\mu}} p(\rho_1,\ldots,\rho_{i-1},x_i,\ldots,x_\mu) \neq p_{i-1}(\rho_{i-1}) \\ \sum_{\substack{x_{i+1},\ldots,x_\mu \\ \in\{0,1\}^\mu}} p(\rho_1,\ldots,\rho_i,x_{i+1},\ldots,x_\mu) = p_i(\rho_i) \end{array} \;\middle|\; \begin{array}{l} ([p_j]_{j=1}^{i},[\rho_j]_{j=1}^{i-1}) \\ \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_i \leftarrow \mathbb{F} \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} \tilde{p}_i(0)+\tilde{p}_i(1) \neq p_{i-1}(\rho_{i-1}) \\ \tilde{p}_i(\rho_i) = p_i(\rho_i) \end{array} \;\middle|\; \begin{array}{l} ([p_j]_{j=1}^{i},[\rho_j]_{j=1}^{i-1}) \\ \leftarrow \tilde{\mathbf{P}}^{\mathsf{rbr}} \\ \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \end{array}\right]$$

Now since the prover was able to convince the verifier in the $i-1$-th round, this implies that $p_{i-1}(\rho_{i-1}) = p_i(0)+p_i(1)$, and so we have:

$$\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},i}(\mathbb{x}) = \Pr\left[\begin{array}{l} \tilde{p}_i(0)+\tilde{p}_i(1) \neq p_{i-1}(\rho_{i-1}) \\ \tilde{p}_i(\rho_i) = p_i(\rho_i) \end{array} \;\middle|\; \begin{array}{l} p_{i-1}(\rho_{i-1}) = p_i(0)+p_i(1) \\ \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \end{array}\right]$$

$$= \Pr\left[\begin{array}{l} \tilde{p}_i(0)+\tilde{p}_i(1) \neq p_i(0)+p_i(1) \\ \tilde{p}_i(\rho_i) = p_i(\rho_i) \end{array} \;\middle|\; \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \right]$$

By again conditioning on the event $p_i \neq \tilde{p}_i$, we have:

$$\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},i}(\mathbb{x}) = \Pr\left[\begin{array}{l} \tilde{p}_i(0)+\tilde{p}_i(1) \neq p_i(0)+p_i(1) \\ \tilde{p}_i(\rho_i) = p_i(\rho_i) \end{array} \;\middle|\; \begin{array}{l} p_i = \tilde{p}_i \\ \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \end{array}\right] \cdot \Pr\left[p_i = \tilde{p}_i\right]$$

$$+ \Pr\left[\begin{array}{l} \tilde{p}_i(0)+\tilde{p}_i(1) \neq p_i(0)+p_i(1) \\ \tilde{p}_i(\rho_i) = p_i(\rho_i) \end{array} \;\middle|\; \begin{array}{l} p_i \neq \tilde{p}_i \\ \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \end{array}\right] \cdot \Pr\left[p_i \neq \tilde{p}_i\right]$$

The first term is zero, and for the second term we can use the identity $\Pr[A \cap B] \leq \Pr[A|B]$ to obtain:

$$\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},i}(\mathbb{x}) = \Pr\left[\begin{array}{l} \tilde{p}_i(0)+\tilde{p}_i(1) \neq p_i(0)+p_i(1) \\ \tilde{p}_i(\rho_i) = p_i(\rho_i) \end{array} \;\middle|\; \begin{array}{l} p_i \neq \tilde{p}_i \\ \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \end{array}\right] \cdot \Pr\left[p_i \neq \tilde{p}_i\right]$$

$$\leq \Pr\left[\tilde{p}_i(\rho_i) = p_i(\rho_i) \;\middle|\; \begin{array}{l} p_i \neq \tilde{p}_i \\ \rho_i \leftarrow \{0,1\}^{\mathbb{F}^*} \end{array}\right] \cdot \Pr\left[p_i \neq \tilde{p}_i\right]$$

According to the Schwartz-Zippel lemma [64,56], we have $\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP},i}(\mathbb{x}) = \deg(p_i)/|\mathbb{F}|$. Hence, we have $\epsilon^{\mathsf{rbr}}_{\mathsf{PIOP}} = \max_{i\in[\mu]} \deg_i(p)/|\mathbb{F}|$. □

Remark 8 (Batching SumCheck claims). To prove multiple SumCheck claims $\{((s_i, [\![p_i]\!]), p_i)\}_{i=1}^t$ where each p_i is a μ-variate polynomial with individual degree d, $\mathcal{V}$ can sample t random field elements $\{c_i\}_{i=1}^t$ and run a single SumCheck on the polynomial $p^* = \sum_{i=1}^t c_i p_i$ and the sum $s^* = \sum_{i=1}^t c_i s_i$.

B Deferred proof of Theorem 10

Proof. We prove that Construction 4 satisfies the properties of a relaxed mod-PC scheme (see Definition 25), namely: 1. $(\mathtt{Setup}, \mathtt{Commit}, \mathtt{Open})$ is a binding commitment scheme for the message space $\mathbb{Z}_B[X_1,\ldots,X_\mu]$. 2. ROK_{Eval} is a succinct interactive public-coin argument system for the relation $\mathfrak{R}_{\mathtt{Eval}}$.

Binding. If an adversary outputs $(\mathsf{cm}, p, p', \mathsf{opn}, \mathsf{opn}')$ such that

$$\mathtt{Open}(\mathsf{pp}, \mathsf{cm}, p, \mathsf{opn}) = \mathtt{Open}(\mathsf{pp}, \mathsf{cm}, p', \mathsf{opn}') = 1$$

then by definition of the $\mathtt{Open}$ algorithm, we have

$$\begin{aligned}&\mathsf{CS}.\mathtt{Open}(\mathsf{pp}, \mathsf{cm}, \text{IntegerEncode}(p), \mathsf{opn}) = \\ &\mathsf{CS}.\mathtt{Open}(\mathsf{pp}, \mathsf{cm}, \text{IntegerEncode}(p'), \mathsf{opn}') = 1\end{aligned}$$

By the binding property of the commitment scheme CS, we have that with overwhelming probability, $\text{IntegerEncode}(p) = \text{IntegerEncode}(p')$. And by the injectivity of the integer encoding function IntegerEncode, we have that $p = p'$. Thus, the commitment scheme is binding.

Completeness. This follows from inspection; notice that the Masdak check $l + m\alpha^{N-1} + r\alpha^N = e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}$ will succeed by construction, and that when all coefficients of the initial polynomial are bounded by B and the evaluation-point components lie in $[0, 2^{\widehat{B}})$ (Remark 9), all the terms in the product are also bounded by the respective bounds above and by choice of α (Eq. (4), first term), there is no overflow of any of the coefficients of the product polynomial. The rest of the completeness proof follows from completeness of Masdak and Range.

Extractability. We show that for an expected poly-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4)$ if the verifier accepts in ROK_{Eval}, then there exists an extractor algorithm $\mathtt{Extract}$ that outputs a witness for $\mathfrak{R}_{\mathtt{Eval}}$, i.e. $\mathsf{w}^* = (p^*, \mathsf{opn}^*)$ such that

$$z = p^*(\boldsymbol{y}) \text{ over } \mathbb{Q} \quad \text{and} \quad \mathsf{PCS}.\mathtt{Open}(\mathsf{ck}, \mathsf{cm}, p^*, \mathsf{opn}^*) = 1$$

Note that we run the $\mathsf{ROK}_{Core\text{-}Eval}$ protocol twice in ROK_{Eval}. At a high level, the first part of the proof follows the proof of Theorem 5.5 in [15], and we extract valid openings for both commitments (under α and β) that satisfy the equations

$$\begin{aligned}p^*_\alpha(\boldsymbol{y}) &= z \bmod \alpha \\ p^*_\beta(\boldsymbol{y}) &= z \bmod \beta\end{aligned}$$

However, due to the choice of $\beta = \alpha \cdot q$, we have that

$$p^*_\beta(\boldsymbol{y}) = p^*_\alpha(\boldsymbol{y}) \bmod \alpha$$

As we are in the non-adaptive setting, the evaluation point $\boldsymbol{y}$ is randomly sampled after commitments to p^*_α and p^*_β are produced. Hence, applying the Schwartz-Zippel lemma and noticing that α has only large prime factors, we have that $p = p'$ as rational polynomials with overwhelming probability. (Although typically we only have guarantees $\bmod \alpha$, since all coefficients of p_α and p_β are guaranteed to be bounded rationals, we can extend the guarantees to the rationals.)

As both the polynomials are the same, we can use the second equation to obtain the necessary guarantee module q. Now, noticing that this is also a random modulus that we check our equation on, applying a standard fingerprinting argument shows that the evaluation z is correct over the integers with overwhelming probability.

Formally, consider the first call to the $\mathsf{ROK}_{Core\text{-}Eval}$ protocol. Since Construction 3 is a reduction of knowledge from $\mathfrak{R}_{CoreEval}$ to $\mathfrak{R}_{\mathsf{Masdak}} \times \mathfrak{R}_{\mathsf{Range}}$, we may assume that we have accepting witnesses for both the Masdak and Range relations.
These witnesses give us openings $e_{\boldsymbol{p},\alpha}$, l, m, r to the intermediate commitments $\mathsf{cm}_{\mathsf{CS}}, \mathsf{cm}_l, \mathsf{cm}_m, \mathsf{cm}_r$ such that

$$\begin{aligned} l + m\alpha^{N-1} + r\alpha^N = e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}} \\ 0 < l < 2 \cdot N 2^{\widehat{B}(\mu+1)} \alpha^{N-2}, \\ 0 < m < 2 \cdot N 2^{\widehat{B}(\mu+1)}, \\ 0 < r < 2 \cdot 2^{\widehat{B}(\mu+1)} \alpha^{N-2} \end{aligned}$$

These factors of 2 show up due to the range proof only admitting bounds that are powers of 2.

Algorithm 4: Extract

`Extract`

1. Obtain $e^*_{\boldsymbol{p},\alpha}$ from the accepting witness for the Masdak relation.
2. Compute $e^*_{\boldsymbol{p},\alpha}$ base α; i.e. compute $p^* := \text{IntegerDecode}(e^*_{\boldsymbol{p},\alpha}, \alpha)$.
3. To obtain the rational representation from this integer polynomial, do the following for each coefficient $i \in \{0, \ldots, N-1\}$:
   (a) Invoke the rational approximation algorithm to obtain $m_i/k_i \in \mathbb{Q}$ which is the closest fraction to p^*_i/α. If $k_i > N^{8\mu} \cdot (2\mu)^{\lambda}$, abort.
   (b) Compute $n_i = p^*_i k_i - m_i \alpha$.
   (c) Set $\tilde{p}_i = \frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}}$ where $m_{-1} = 0$ and $k_{-1} = 1$.
4. Compute $p = \tilde{p} \bmod q$ and output $(p, \tilde{p})$.

We require the following results from [12], adapted to our notation.

Lemma 4 (Corollary 1 from [12]). *For all n such that $\log n \geq 8\mu^2 + \log_2(2\mu) \cdot \lambda$, and any μ-linear polynomial f coprime with n,*

$$\Pr_{\boldsymbol{x} \leftarrow [0,m)^{\mu}} [f(\boldsymbol{x}) \equiv 0 \bmod n] \leq 2^{-\lambda} + \frac{\mu}{m}.$$

Corollary 3 (Local zero to global zero). *Under the conditions of Lemma 4, if $f(\boldsymbol{y}) \equiv 0 \bmod n$ for a random $\boldsymbol{y} \leftarrow [0,m)^{\mu}$, then f is the zero polynomial $\bmod\ n$ with probability $\geq 1 - 2^{-\lambda} - \mu/m$.*

Corollary 4 (Modulus upper bound). *If f is a μ-linear polynomial coprime with n, not identically zero, and $f(\boldsymbol{y}) \equiv 0 \bmod n$ for random $\boldsymbol{y} \leftarrow [0,m)^{\mu}$, then $n < N^{8\mu} \cdot (2\mu)^{\lambda}$.*

Lemma 5 (Lemma 1 from [12]: Evaluation bound). *For any μ-linear polynomial f and $m \geq 2$:*

$$\Pr_{\boldsymbol{x} \leftarrow [0,m)^\mu} \left[|f(\boldsymbol{x})| \leq \frac{1}{m^\mu} \cdot \|f\|_\infty \right] \leq \frac{3\mu}{m}.$$

Remark 9 (Size of the evaluation domain). The soundness of Corollaries 3 and 4 and Lemma 5 hinges on the sampling range m of the evaluation point through the error terms $2^{-\lambda} + \mu/m$ and $3\mu/m$; likewise, the fingerprinting reduction modulo q requires the prime to be drawn from a space of size at least 2^λ. To make both hold *unconditionally* in B and λ, our construction samples the evaluation point from $\mathbb{Z}_{\widehat{B}}^\mu$ and the prime as $q > 2^{\widehat{B}}$, where $\widehat{B} = \max(B, \lambda)$. Then $m = 2^{\widehat{B}} \geq 2^\lambda$, so μ/m and $3\mu/m$ are negligible, and the prime space exceeds 2^λ, regardless of how B compares to λ. This decoupling of the challenge domain from the coefficient bit-length mirrors the fingerprinting approach of [18], where challenges are drawn from a λ-bit prime field. In the intended regime $B \geq \lambda$ we have $\widehat{B} = B$, and both the sampling and the encoding base of Eq. (4) reduce to their B-only form; the price paid for supporting $B < \lambda$ is that the encoding base, and hence the committed integers, are sized by $\widehat{B}$ rather than B (see Fig. 2).

Remark 10 (Choosing the encoding base). The first bound in Eq. (4), $\alpha > 2^{7\widehat{B}\mu}$, guarantees injectivity of the integer encoding and rules out overflow in the products formed during evaluation; here $\widehat{B} = \max(B, \lambda)$ accounts for the fact that the evaluation-point components are drawn from a $\widehat{B}$-bit domain (Remark 9), so the digits of the product $e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}$ are governed by $\widehat{B}$ rather than by B alone. The second bound makes α dominate the hint-space denominator bound $N^{8\mu}(2\mu)^\lambda$ that governs extraction: for any honest coefficient of magnitude $c < 2^B$ we then have $c/\alpha < 1/\big(2\,N^{8\mu}(2\mu)^\lambda\big)$, so c/α is closer to 0 than to any nonzero fraction with denominator at most $N^{8\mu}(2\mu)^\lambda$. Hence the rational-recovery step of the extractor returns the honest integer coefficient rather than a spurious bounded rational. Both bounds are stated explicitly in terms of λ, so correctness requires *no* assumption relating B and λ; in the intended regime $B \geq \lambda$ we have $\widehat{B} = B$ and (with $\mu \geq 4$) the first bound already implies the second, recovering the simpler choice $\alpha = 2^{7B\mu}$.

Correctness of extracted opening. We show that the extracted coefficients p_i^* admit a bounded rational representation and that the evaluation identity holds. From the check in Masdak, we have that

$$e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}} = l_0^* + l_1^* \alpha^{N-2} + m^* \alpha^{N-1} + r^* \alpha^N$$

where $l_1^* < 2N2^{\widehat{B}(\mu+1)}$, $m^* < 2N2^{\widehat{B}(\mu+1)}$, and $r^* < 22^{\widehat{B}(\mu+1)}\alpha^{N-2}$.

We claim each p_i^* admits a representation

$$p_i^* = \frac{m_i \alpha + n_i}{k_i}, \quad m_i \geq 0,\ |n_i| < \alpha,\ k_i > 0,\ \gcd(k_i, \alpha) = 1. \tag{5}$$

(Such a representation exists trivially with $m_i = 0, n_i = p_i^*, k_i = 1$; we derive bounds on k_i, n_i.)

Step 1: Coefficient of α^{N-2}. Write $l^* = l_0^* + l_1^* \alpha^{N-2}$ where $l_1^* < 2N2^{\widehat{B}(\mu+1)}$. On the LHS, the coefficient of α^{N-2} is $\left\lfloor \frac{e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}}{\alpha^{N-2}} \right\rfloor \bmod \alpha$. Expanding:

$$\begin{aligned}\left\lfloor \frac{e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}}{\alpha^{N-2}} \right\rfloor \bmod \alpha &= \left\lfloor \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_i^* \cdot y_{N-j-1} \cdot \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor \bmod \alpha \\ &= \underbrace{\left\lfloor \frac{\sum_{0 \le i+j \le N-3} p_i^* \cdot y_{N-j-1} \cdot \alpha^{i+j}}{\alpha^{N-2}} \right\rfloor}_{:= \lfloor e \rfloor} + \sum_{i=0}^{N-2} p_i^* \cdot y_{i+1} \bmod \alpha \end{aligned} \tag{6}$$

where terms with $i + j > N - 2$ are absorbed into higher powers and vanish modulo α. The carry term e is bounded as

$$0 \le \lfloor e \rfloor \le \left\lfloor \frac{N^2 \cdot \widehat{B}^{\mu} \cdot \alpha^{N-2}}{\alpha^{N-2}} \right\rfloor = N^2 \widehat{B}^{\mu}.$$

Equating the coefficient of α^{N-2} on both sides gives $\sum_{i=0}^{N-2} p_i^* y_{i+1} = l_1^* - \lfloor e \rfloor \bmod \alpha$, so there exists $m \in \mathbb{Z}$ such that

$$\sum_{i=0}^{N-2} \frac{m_i \alpha + n_i}{k_i} y_{i+1} = (l_1^* - \lfloor e \rfloor) + m\alpha \tag{7}$$

for some $m \in \mathbb{Z}$. Setting $L := \operatorname{lcm}(k_0, \ldots, k_{N-2})$ and rearranging:

$$p_1(\boldsymbol{y}) := (l_1^* - \lfloor e \rfloor) L - \sum_{i=0}^{N-2} \frac{L\, n_i}{k_i} y_{i+1} \equiv 0 \bmod \alpha. \tag{8}$$

Step 2: Bounding k_i. Since $\gcd(k_i, \alpha) = 1$ for all i, p_1 is coprime with α. By the choice of encoding base (Eq. (4), second term), $\alpha > N^{8\mu} \cdot (2\mu)^{\lambda}$, so we apply Corollary 3: $p_1 \equiv 0 \bmod \alpha$ as a polynomial with overwhelming probability. Since all coefficients of p_1 are bounded by α, in fact $p_1 \equiv 0$ over $\mathbb{Z}$.

This implies $p_2(\boldsymbol{y}) := \sum_{i=0}^{N-2} \frac{L\, n_i}{k_i} y_{i+1} \equiv 0 \bmod L$. By Corollary 4, $L < N^{8\mu} \cdot (2\mu)^{\lambda}$, and hence each $k_i < N^{8\mu} \cdot (2\mu)^{\lambda}$.

Step 3: Bounding n_i. By Lemma 5 applied to $p_2(\boldsymbol{y}) = (l_1^* - \lfloor e \rfloor) L$, we get $|n_i| \le \widehat{B}^{2\mu+1} \cdot N \cdot k_i$ for $i \le N - 2$.

For $i = N - 1$, we analyze the coefficient of α^{2N-2} (the highest non-trivial power). Let u' denote the coefficient of α^{2N-2} on the RHS; by the bound on r^*, we have $u' < \widehat{B}^{\mu+1}$. On the LHS:

$$\left\lfloor \frac{e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}}{\alpha^{2N-2}} \right\rfloor \bmod \alpha = \underbrace{\left\lfloor \frac{\sum_{i+j<2N-2} p_i^* \cdot y_{N-j-1} \cdot \alpha^{i+j}}{\alpha^{2N-2}} \right\rfloor}_{:= \lfloor e' \rfloor} + p_{N-1}^* \cdot y_0 \bmod \alpha$$

where $\lfloor e' \rfloor \leq N^2 \widehat{B}^{\mu}$ (bounded analogously to $\lfloor e \rfloor$). Equating both sides and applying (5) with the constraint $m_{N-1} = 0$:

$$\frac{n_{N-1}}{k_{N-1}} \cdot y_0 = (u' - \lfloor e' \rfloor) \bmod \alpha$$

By Corollary 3 and the fact that $u' - \lfloor e' \rfloor < \alpha$, the polynomial $n_{N-1}y_0 - (u' - \lfloor e' \rfloor)k_{N-1}$ is identically zero over $\mathbb{Z}$. By Corollary 4, $k_{N-1} < N^{8\mu} \cdot (2\mu)^{\lambda}$. By Lemma 5, $|n_{N-1}| \leq N\widehat{B}^{2\mu+1}$.

Step 4: Valid opening. Since $\tilde{p}_i = n_i/k_i + m_{i-1}/k_{i-1}$ with $m_{-1} = 0, k_{-1} = 1$, the bounds imply $|\tilde{p}_i| \leq N\widehat{B}^{2\mu+1} + 1$ with denominator $\leq L < N^{8\mu} \cdot (2\mu)^{\lambda}$. By telescoping (splitting $p_i^* = n_i/k_i + m_i\alpha/k_i$, noting $m_{-1} = m_{N-1} = 0$), $\sum_i \tilde{p}_i \alpha^i = \sum_i p_i^* \alpha^i$, so the commitment is correctly opened.

Step 5: Coefficient of $\boldsymbol{\alpha^{N-1}}$ (correct evaluation). We now show the extracted polynomial evaluates correctly. On the LHS, the coefficient of α^{N-1} is:

$$\begin{aligned}
&\left\lfloor \frac{e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}}{\alpha^{N-1}} \right\rfloor \bmod \alpha \\
&= \left\lfloor \frac{\sum_{i+j<N-1} p_i^* \cdot y_{N-j-1} \cdot \alpha^{i+j}}{\alpha^{N-1}} + \sum_{i+j=N-1} p_i^* \cdot y_{N-j-1} \right\rfloor \bmod \alpha \\
&= \left\lfloor \frac{\sum_{i+j<N-2} p_i^* \cdot y_{N-j-1} \cdot \alpha^{i+j}}{\alpha^{N-1}} + \frac{\sum_{i=0}^{N-2} p_i^* \cdot y_{i+1}}{\alpha} \right\rfloor + \sum_{i=0}^{N-1} p_i^* \cdot y_i \bmod \alpha \\
&= \left\lfloor \frac{e + l_1^* - \lfloor e \rfloor}{\alpha} \right\rfloor + m + \sum_{i=0}^{N-1} p_i^* \cdot y_i \bmod \alpha
\end{aligned}$$

where the second-to-last equality separates the sum at $i+j = N-2$ (using (6)), and the last uses (7). Since $0 \leq e + l_1^* - \lfloor e \rfloor < \alpha$ (by the bound on l_1^*), the floor term vanishes. We then expand using the rational representation:

$$\begin{aligned}
\text{LHS coeff of } \alpha^{N-1} &= \sum_{i=0}^{N-1} p_i^* \cdot y_i + m \bmod \alpha \\
&= \sum_{i=0}^{N-1} \left(\frac{m_i\alpha + n_i}{k_i} \right) y_i + \sum_{i=0}^{N-2} \frac{m_i}{k_i} y_{i+1} \bmod \alpha \\
&= \sum_{i=0}^{N-1} \left(\frac{n_i}{k_i} + \frac{m_{i-1}}{k_{i-1}} \right) y_i \bmod \alpha \\
&= \sum_{i=0}^{N-1} \tilde{p}_i \cdot y_i \bmod \alpha
\end{aligned}$$

where we define $m_{-1} = 0, k_{-1} = 1$, and the penultimate equality follows from (7) (giving $m = \sum_{i=0}^{N-2} \frac{m_i}{k_i} y_{i+1} \bmod \alpha$). On the RHS, this coefficient equals m, so $\sum_{i=0}^{N-1} \tilde{p}_i \cdot y_i = m \bmod \alpha$.

By an identical argument, we also get the same extracted polynomial with β, implying that $p^*(\boldsymbol{y}) \equiv z \pmod{\beta}$. Since $q|\beta$, we have $p^*(\boldsymbol{y}) \equiv z \pmod{q}$. However, q was chosen after the polynomial was committed to, implying that $p^*(\boldsymbol{y}) = z$ over the rationals.

Efficiency considerations. We justify the running times for the prover, verifier and proof size as given in the theorem.

The prover time is dominated by the running time of the sub-protocols Masdak and Range which are called twice each. The other operations done by the prover include the three way decomposition as well as the integer encoding and multiplication of $e^*_{\boldsymbol{p},\alpha} \cdot e_{\boldsymbol{y}}$ – as these integers are of size α^N, these operations can cost up to $O(\log \alpha^N)$ bit operations, leading to a cost of $O(N \log N)$ bit operations. Group operation costs include committing to various integers (in step 5), which varies depending on the commitment scheme used.

Although in Construction 3, for simplicity, we specify that the verifier explicitly computes $e_{\boldsymbol{y}}$ from the $\boldsymbol{y}$, this would lead to a linear verifier. In the actual protocol, we only need to compute this value modulo a random prime used in Masdak – this can be done easily since the expansion has a specific form that allows the verifier to compute it in logarithmic time.
Specifically, we notice that $e_{\boldsymbol{y}} \mod \ell$ has a succinct representation

$$\prod_{i=1}^{\mu} \left(1 + \boldsymbol{y}_{\mu-i} \cdot \alpha^{2^{i-1}}\right) \mod \ell$$

allowing the verifier to compute this product of μ terms in $O(\mu)$ time, rather than $O(2^\mu)$. The rest of the verifier time is dominated by the Masdak and Range verifiers.

The proof size is a constant number of group elements followed by the sizes of the Masdak and Range proof.

□

C Reductions (and Arguments) of Knowledge from Gen-PIOR + Gen-PCS

In this section, we present a blueprint for constructing reductions of knowledge (RoK, Definition 14) for NP relations using a generalization of the popular PIOP + PC paradigm, which is Gen-PIOR (see Definition 3) + Gen-PCS (see Definition 20). Since arguments of knowledge (ARK, Definition 19) are a special case of RoKs, this compiler can also be used to build ARKs.

The high-level idea is that the RoK prover and verifier internally simulate the Gen-PIOR prover and verifier. They use a Gen-PCS to materialize the prover's oracle messages and then rely on the Gen-PCS evaluation argument to answer the verifier's queries. The key difference from the standard PIOP + PC paradigm is that, while the prover's messages are committed in the ambient ring $\mathbb{R}$, all subsequent messages and checks are then interpreted over the evaluation ring $\mathbb{S}$ via a randomly sampled evaluation map.

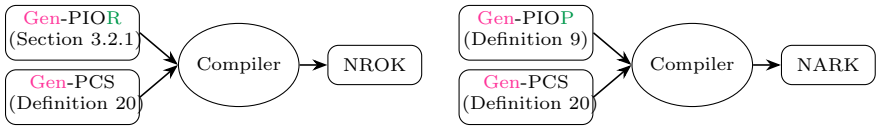

Fig. 3. Description of the reduction of knowledge from Gen-PIOR + Gen-PCS.

Construction 7: Reduction of Knowledge from Gen-PIOR + Gen-PCS

$\mathsf{pp} \leftarrow \mathcal{S}(1^\lambda, N)$

1. Parse the security parameter λ and the size bound N.
2. Use the size bound N to derive the size bound N' for the polynomial commitment scheme.
3. Invoke $(\mathsf{ck}_{\mathsf{PCS}}, \mathsf{vk}_{\mathsf{PCS}}) \leftarrow \mathsf{Gen\text{-}PCS.Setup}(1^\lambda, N)$ to obtain the commitment and verification key for the Gen-PCS.
4. Output $\mathsf{pp} := (\mathsf{ck}_{\mathsf{PCS}}, \mathsf{vk}_{\mathsf{PCS}})$.

$(\mathsf{pk}, \mathsf{vk}, \mathsf{i}') \leftarrow \mathcal{I}(\mathsf{pp}, \mathsf{i})$

1. Invoke the Gen-PIOR indexer $\mathcal{I}$ on i to obtain $(\iota, \mathsf{I}, \mathsf{i}')$ where i' is the derived index for the target relation, and ι and I are the short and implicit indices for the Gen-PIOR, respectively. More specifically, I includes the round-0 oracles: $\Pi_0 = \{ p_{0,j} \}_{j=1}^{n_\Pi(0)} \subseteq \mathbb{R}^{\mathsf{ind}\leq 1}_{v(0,j)}$
2. Next, commit to the index polynomials using the Gen-PCS:
$$\mathsf{cm}_0 \leftarrow \mathsf{Gen\text{-}PCS.Commit}(\mathsf{ck}_{\mathsf{PCS}}, \Pi_0).$$
3. Set $\mathsf{pk} := (\mathsf{ck}_{\mathsf{PCS}}, \mathsf{i})$ and $\mathsf{vk} := (\mathsf{vk}_{\mathsf{PCS}}, \iota, \mathsf{cm}_0)$, and output the derived index i' for the target relation.

$\langle\ \mathcal{P}(\mathsf{pk}, \mathsf{i}, \mathsf{x}, \mathsf{w})\ ,\ \mathcal{V}(\mathsf{vk}, \mathsf{x})\ \rangle$

1. **Commitment phase.** The RoK prover and verifier internally invoke the Gen-PIOR prover $\mathcal{P}$ on instance-witness pair (x, w) and Gen-PIOR verifier $\mathcal{V}$ on instance x. In the i-th round of the Gen-PIOR: The RoK verifier simulates $\mathcal{V}$ to obtain public random challenges ϱ_i, then forwards them to the RoK prover. The RoK prover forwards ϱ_i to the simulated $\mathcal{P}$ and obtains the polynomials $\Pi_i = \{ p_{i,j} \}_{j \in [n_\Pi(i)]} \subseteq \mathbb{R}^{\mathsf{ind}\leq 1}_{\mu}$. The RoK prover commits to all polynomials using the Gen-PCS:
$$\mathsf{cm}_i \leftarrow \mathsf{Gen\text{-}PCS.Commit}(\mathsf{ck}_{\mathsf{PCS}}, \Pi_i),$$
and sends this commitment to the verifier.
2. **Evaluation-map sampling.** After the interaction, the RoK verifier samples a random evaluation map $\varphi : \mathbb{R} \to \mathbb{S}$ from the prescribed family Φ and sends its description to the prover. From this point onward, all PIOR messages and checks are interpreted over $\mathbb{S}$ via φ.
3. **Query phase.**
   (a) The RoK verifier samples and sends query randomness ϱ_{k+1}.

(b) The RoK prover computes the query set Q using the Gen-PIOR verifier's query algorithm on $(\mathsf{x}; \varrho_1, \ldots, \varrho_{k+1})$.
(c) The RoK prover answers the queries by evaluating the committed polynomials and applying the evaluation map, obtaining $\nu := \varphi(\Pi(Q))$, and produces the corresponding Gen-PCS opening proof.

4. **Verification.** The RoK verifier checks the Gen-PCS opening proof. If it accepts, the RoK verifier feeds ν into the simulated $\mathcal{V}$ as oracle answers and accepts if and only if $\mathcal{V}$ accepts:

$$(\mathsf{x}', \bot) := \mathcal{V}^{\boldsymbol{\Pi}, \mathsf{I}, \mathsf{y}}(\iota, \mathsf{x}, \boldsymbol{\varrho})$$

5. **Outputs.** The prover outputs a new witness w' and the verifier outputs a new instance x' for the target relation.

Next, we provide a formal statement of the security and the asymptotics of Construction 7.

Theorem 17. *Let $\mathfrak{R}$ and $\mathfrak{R}'$ be indexed relations. Consider the following components:*

- *A generalized polynomial interactive oracle reduction (see Definition 3) of the form* $\mathsf{Gen\text{-}PIOR} = (\mathsf{Algs}, \mathsf{Eval\text{-}Conf}, \mathsf{Metrics})$ *for $\mathfrak{R}$ with round-by-round soundness error ϵ^{RBR}, where*
  - $\mathsf{Algs} = (\mathcal{I}, \mathcal{P}, \mathcal{V})$ *are the indexer, prover, and verifier algorithms.*
  - $\mathsf{Eval\text{-}Conf} = (\mathbb{R}, \mathbb{S}, \Phi)$ *specifies the ambient ring, evaluation ring, and evaluation map domain.*
  - $\mathsf{Metrics} = (k, n_\Pi, n_\varrho, s)$ *are the round and query complexity parameters where $n := \sum_{i=1}^{k} n_\Pi(i)$.*
- *A generalized non-adaptive polynomial commitment scheme (see Definition 20)* $\mathsf{Gen\text{-}PCS} = (\mathsf{Eval\text{-}Conf}, \mathtt{Setup}, \mathtt{Commit}, \mathtt{Open}, \mathsf{ROK}_{Eval})$ *with an evaluation argument* $\mathsf{ROK}_{Eval} = (\mathcal{P}_{\mathtt{Eval}}, \mathcal{V}_{\mathtt{Eval}})$.

Then, Construction 7 is a reduction of knowledge (see Definition 14) ROK *from $\mathfrak{R}$ to $\mathfrak{R}'$ with the following time complexities:*

Setup	*Indexer*	*prover time*	*verifier time*
$T(\mathtt{Setup})$	$T(\mathcal{I}) + n_\Pi(0)T(\mathtt{Commit})$	$T(\mathcal{P}) + nT(\mathtt{Commit}) + T(\mathcal{P}_{\mathtt{Eval}})$	$T(\mathcal{V}) + T(\mathcal{V}_{\mathtt{Eval}})$

and the following size complexities:

Public parameter	*Proving Key*	*Verifying Key*	*proof size*
$\lvert\mathsf{pp}\rvert$	$\lvert\mathsf{ck}_{\mathsf{PCS}}\rvert + n_\Pi(0)d(0)$	$\lvert\mathsf{vk}_{\mathsf{PCS}}\rvert + n_\Pi(0)\lvert\mathsf{cm}\rvert$	$n\lvert\mathsf{cm}\rvert + s\lvert\mathbb{S}\rvert + \lvert\pi_{\mathtt{Eval}}\rvert$