Zinc+: SNARKs for Polynomial Rings

Alexander Abdugafarov[1], Albert Garreta[1], Amit Kumar[1], Michał Osadnik[2], Psi Vesely[3], Ilia Vlasov[1], and Kai Zhe Zheng[1,4]

[1]Nethermind Research
[2]Aalto University, Espoo, Finland
[3]Yale University
[4]Massachusetts Institute of Technology

May 8, 2026

Abstract

Nearly all succinct proof systems express computations as algebraic constraints over a finite field. Operations not native to this field, such as bitwise manipulation, modular arithmetic, and lattice-ring operations, require an arithmetization step that can inflate the witness size by one or more orders of magnitude.

We introduce *Universal Constraint Systems (UCS)* and *Zinc+*. The first is a relation that can express the above constraints with minimal overhead. The second is a framework for building SNARKs for UCS. Concretely, UCS consists of algebraic constraints and ideal membership predicates over multiple polynomial rings simultaneously, such as $\mathbb{F}_q[X], \mathbb{Q}[X], \mathbb{Z}[X]$, etc.

Zinc+ SNARKs are built from 1) a PIOP for UCS, and 2) a hash-based IOPP for multilinear polynomials over $R = \mathbb{Q}[X]$ or $R = \mathbb{F}_q[X]$. For 1), we provide a general compiler that takes standard finite-field PIOPs and turns them into a PIOP for UCS. The IOPP in 2) depends on R: for $R = \mathbb{F}_q[X]$, we construct it via a black-box lift of any existing IOPP for $\mathbb{F}_q$, and for $R = \mathbb{Q}[X]$, we present a novel tensor IOPP design, instantiated with the new code family below.

We introduce *Integer Pseudo-Reed Solomon (IPRS) codes*, a new family of MDS codes over $\mathbb{Q}$ and $\mathbb{Q}[X]$. While not Reed-Solomon codes, these codes have optimal MDS relative minimal distance, support efficient FFT-based encoding, and have bounded norm growth when encoding (unlike a naïve lift of Reed-Solomon codes to the integers).

Our unoptimized, open-source, implementation proves 7 SHA-256 compressions followed by the multi-scalar multiplication (MSM) part of an ECDSA verification (the bulk of the work), with the following performance, benchmarked on a MacBook Air M4:

Prover time: 40.6 ms, Verifier time: 7.0 ms, Proof size: 198 KB.

Zinc+ can be instantiated end-to-end or as a lightweight extension to any existing hash-based SNARK over $\mathbb{F}_q$.

Contents

1 Introduction

A Succinct Non-interactive ARgument of Knowledge (SNARK) for a relation REL is a non-interactive protocol that enables a prover, P, to convince a verifier, V, that they know a witness, $\mathbb{w}$, such that $(\mathbb{x}; \mathbb{w}) \in R$, for a given public input $\mathbb{x}$, and where communication and verifier runtime are sublinear (i.e. the scheme is *succinct*).

In most cases, SNARKs support relations REL defined as a system of polynomial equations over a finite field $\mathbb{F}$ (e.g. R1CS, CCS, AIR, Plonkish, or lookups). In that case we say that REL is an *algebraic relation over* $\mathbb{F}$ and that the SNARK is *native over* $\mathbb{F}$. Such SNARKs have experienced remarkable performance improvements over the past years.

Algebraic relations over $\mathbb{F}$ are expressive enough to capture essentially any computation of interest, even those involving operations across multiple domains not native to $\mathbb{F}$, such as bitwise manipulation, elliptic-curve arithmetic, non-prime-power modular arithmetic, and lattice-ring operations, often within a single computation (e.g., hashing followed by signature verification). The standard approach, called *arithmetization*, encodes the target relation REL' one wishes to prove (not natively expressed over $\mathbb{F}$) into an algebraic relation REL over $\mathbb{F}$: one designs a public encoding Ψ such that knowledge of a witness $\mathbb{w}$ with $(\Psi(\mathbb{x}'), \mathbb{w}) \in \mathsf{REL}$ guarantees knowledge of a witness $\mathbb{w}'$ with $(\mathbb{x}'; \mathbb{w}') \in \mathsf{REL}'$, and uses a SNARK for REL to prove the encoded statement.

The problem is that arithmetization typically inflates the witness by up to one or more orders of magnitude [Gar+25; GWHD25; BFKTWZ24; OKMZ25; DP25]: the encoded witness is much larger than the original. Since SNARK performance depends primarily on witness size, this overhead has a deleterious effect on prover time, proof size, and verifier time. Beyond performance, non-native arithmetization introduces two further challenges:

1) Need for expert arithmetization. Designing efficient and correct arithmetizations over a finite field is challenging both conceptually and in terms of implementation [Pai+23; Wen+24]. Generic tools such as zkVMs [BCTV14; GPR21; AST24] hide this complexity from application developers by introducing a layer of indirection: the computation is first compiled into VM cycles (e.g., RISC-V instructions), which are then arithmetized by the proving backend. However, this incurs a *double* arithmetization overhead: first from compiling the computation into VM cycles, and then from arithmetizing those cycles into a finite-field relation. This double overhead is not inherent to the zkVM paradigm: one could design zkVMs whose underlying proof system avoids arithmetization overhead. We envision the present work as a step in this direction.

2) Correctness and auditability. The gap between the original computation and its arithmetized form makes the final proof system harder to audit and more prone to bugs, e.g., due to underconstrained systems.

In this work, we introduce Zinc+, a framework that largely eliminates this overhead by expressing constraints directly over polynomial rings $R[X]$ with ideal membership predicates, making operations such as bitwise manipulation, rotation, and modular arithmetic native rather than emulated. Zinc+ supports constraints over $\mathbb{Q}[X]$ (and hence $\mathbb{Z}[X]$ and $\mathbb{Z}$) and multiple rings $\mathbb{F}_{q_i}[X]$ simultaneously, compiles them to finite-field satisfiability using any existing finite-field PIOP, and provides hash-based IOPPs for polynomial-ring witnesses. It can be instantiated end-to-end or as a lightweight extension to existing hash-based SNARKs.

1.1 Contributions

The contributions of this work are:

Universal Constraint System. A new arithmetization framework, called *Universal Constraint System* (UCS), that generalizes R1CS, CCS, AIR, and lookup constraints to support constraint do-

mains over $\mathbb{Q}[X]$ (hence $\mathbb{Z}[X]$, $\mathbb{Z}$) and multiple polynomial rings $\mathbb{F}_{q_i}[X]$ (hence $\mathbb{F}_{q_i}$) simultaneously, with ideal membership predicates of the form $Q(\mathtt{x}, \mathtt{w}) \in (g)$ over any of these rings. As shown in Table 1, UCS captures a wide range of practical computations with low arithmetization overhead.

PIOPs for UCS. The *Zinc+ PIOP compiler* presents a framework for building PIOPs for the UCS relation. It takes as input a standard PIOP for a corresponding finite-field relation and extends it to support the UCS relation. We show the resulting PIOP preserves the round-by-round knowledge soundness of the input PIOP.

PCSs/IOPPs for polynomial rings. *Zip+* provides a framework and concrete instantiation for building hash-based polynomial commitment schemes over $\mathbb{Q}[X]$ and $\mathbb{F}_q[X]$. At its core is an IOP of Proximity (IOPP) for linear error-correcting codes over $\mathbb{Q}$, $\mathbb{Q}[X]$, and $\mathbb{F}_q[X]$, which we show is round-by-round knowledge sound. The PCS/IOPP enables proving MLE evaluation claims either over $\mathbb{Q}[X]$ or $\mathbb{F}_q[X]$, or over finite field projections of these.

MDS codes over $\mathbb{Q}$ and $\mathbb{Q}[X]$. We present a novel family of linear error-correcting codes over $\mathbb{Q}$ (and $\mathbb{Q}[X]$) called *Integer Pseudo-Reed Solomon (IPRS) codes*. IPRS codes are, to our knowledge, the first codes over $\mathbb{Q}$ combining MDS, efficient $O(n \log n)$ encoding, and bounded coefficient growth.

Concrete arithmetizations. We arithmetize SHA-256 hashing followed by ECDSA signature verification, as a UCS instance. This is an AIR-like constraint system with a trace of 2^9 rows and 19 witness columns. Further, for the SHA-256 part, all constraints are linear, except for lookup constraints posed over the trace columns.

Instantiation, implementation, and benchmarks. We instantiate and implement our framework (`github.com/NethermindEth/zinc-plus/tree/main-beta`), obtaining a SNARK. We benchmark its performance for proving the SHA-256 + the Multi Scalar Multiplication part of ECDSA. The experiments were run on a MacBook Air M4, targeting 100 bits of security. We provide a highlight of the results below.

Zinc+ on 7x SHA-256 + ECDSA MSM	**Time / Size**
Prover time	**40.6 ms**
Verifier time	**7.0 ms**
Proof size	**198 KB**

The proof is not zero knowledge. For the ECDSA part, the circuit proves the multi-scalar multiplication of ECDSA, which is the bulk of the signature verification[1]. This will be expanded in future repository updates. See Sections 2.5 and 9 for details.

Lightweight instantiation. We propose a lightweight instantiation of our framework, which we call the *Zinc+ add-on*, that, while not as "optimal" as a full Zinc+ instantiation, can be added with minimal overlap to existing hash-based SNARKs over finite fields $\mathbb{F}_q$, allowing such SNARKs to prove UCS statements over $\mathbb{F}_q[X]$, including ideal membership predicates. As demonstrated in Sections 1.1.1 and 2.1, this is sufficient to express computations such as classic hashes, with minimal arithmetization overhead.

Other contributions: PCS or IOPP over $\mathbb{Z}$, and Mutual Correlated Agreement (MCA) for linear codes over $\mathbb{Q}$ up to the 1.5 Johnson bound. Combining our IOPPs (or PCS) for

[1]The rest of the proof would entail proving two multiplications modulo n, i.e. $u_1 \cdot s = e$ and $u_2 \cdot s = r$ modulo n, and $\mathsf{int}(R_x) = r \mod n$, for n the secp256k1 curve prime order, where int denotes integer representative in the range $[0, p-1]$, cgf. Section 8.3.

$\mathbb{Q}$ with any lookup PIOP (by, for example, instantiating our Zinc+ framework) for a range within $\mathbb{Z}$ immediately provides an IOPP (or PCS) over $\mathbb{Z}$. This applies as well to $\mathbb{Q}[X]$ and $\mathbb{Z}[X]$.

We also show that all linear codes over $\mathbb{Q}$ have MCA up to the 1.5 Johnson bound, by adapting the proof of [Zei24] for finite fields to $\mathbb{Q}$.

Below we elaborate on most of the above contributions. Fig. 1 shows how they fit together.

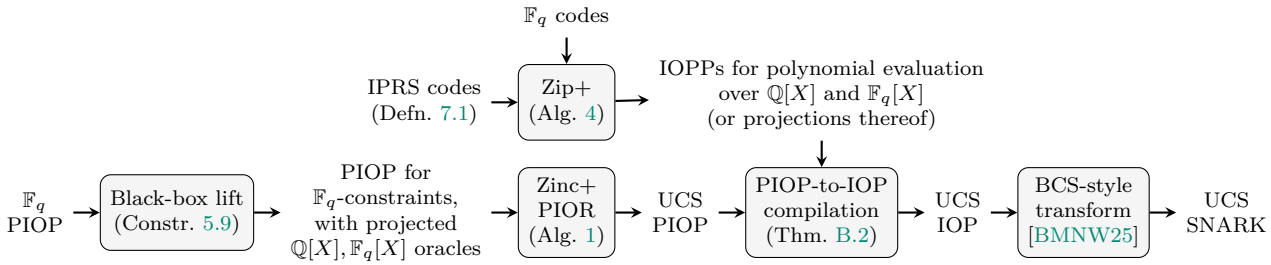

Figure 1: Architecture of the Zinc+ framework. The top row shows the commitment infrastructure from IPRS codes to IOPPs for batched evaluation of multilinear polynomials over $\mathbb{Q}[X]$ and $\mathbb{F}_q[X]$. The bottom row shows the PIOP pipeline from finite-field PIOPs to a PIOP for UCS. Finally, we use our compiler to combine the PIOP and IOPPs into an IOP for UCS, which is then compiled into a SNARK for UCS in the ROM by a BCS-style transformation.

Computation	$\mathbb{Z}/2^w\mathbb{Z}$	**Bit**	$\mathbb{F}_q$	$\mathbb{F}_{2^w}$	$\mathbb{Z}/n\mathbb{Z}$	$\mathbb{Z}$ or $\mathbb{Q}$	R_{latt}
Cryptographic primitives							
Classic hashes (SHA, etc.)	✓	✓					
AES / symmetric encr.	✓	✓		✓			
Elliptic curve crypt.			✓				
Lattice crypt.	✓						✓
RSA crypt.					✓		
Applications							
zkVMs / CPU emulation	✓	✓	✓				
zkID	✓	✓	✓		✓		
Blockchain transactions	✓	✓	✓				
TLS (AEAD)	✓	✓	✓	✓			
Proof recursion (classic hash)	✓	✓	✓		✓		
Hash-to-curve	✓	✓	✓				
FHE	✓					✓	✓
Machine learning						✓	
Finance						✓	

Table 1: Common computations and the types of arithmetic they require. Columns named with a ring R refer to "addition and multiplication in R". The Bit column refers to operations such as XOR, AND, rotations, shifts, etc. For an integer $m \geq 1$, $\mathbb{Z}/m\mathbb{Z}$ denotes integers modulo m (where composite m is relevant in RSA cryptography and lattices, FHE, etc.). Further, $\mathbb{F}_q$ is a finite field (often prime); $\mathbb{F}_{2^w}$ denotes fields of size 2^w; and R_{latt} denotes rings relevant to lattice-based cryptography such as cyclotomic rings $\mathbb{Z}[X]/(X^n+1)$.

zkID refers to performing a classic hash followed by a signature verification with, e.g. ECDSA or RSA. Proof recursion refers to proving acceptance of a SNARK verifier in a context where the original proof system uses classic hashes (due to Fiat-Shamir or Merkle tree commitments); it has $\mathbb{Z}/n\mathbb{Z}$ checkmarked as it sometimes requires using different moduli, e.g. in IVC recursion [KST22]. Hash-to-curve refers to algorithms that iteratively perform hashes and check membership of points on an elliptic curve. zkVMs may use so-called precompiles/builtins to perform computations such as ECDSA verification. In this case they also involve prime field arithmetic.

1.1.1 Universal Constraint Systems (UCS)

We propose a new arithmetization framework that can succinctly express many computations of interest (namely, at least those captured in Table 1), with low arithmetization overhead. Concretely, we design a relation $\mathsf{REL}_{\mathsf{UCS}}$ called *Universal Constraint System* (UCS) that efficiently encapsulates said computations. See Section 2.1 and Definition 4.1.

UCS is a generalization of constraints such as R1CS, CCS, AIR, lookups, etc., along three axes:

Domains and algebraic form. UCS can express algebraic constraints such as R1CS, AIR [Sta21], M3 [Irr26], lookups, etc. simultaneously over different rings such as $\mathbb{Q}$ (rationals), $\mathbb{Z}$ (integers), $\mathbb{F}_q$ (prime or prime power fields), and, importantly, *polynomial rings* such as $\mathbb{Q}[X], \mathbb{Z}[X], \mathbb{F}_q[X]$.

Ideal membership constraints. Typically, an algebraic constraint has the form $Q(\mathbf{b}, \mathbb{x}, \mathbb{w}) = 0 \ \forall \mathbf{b} \in \{0,1\}^{\mu}$, where Q is a polynomial expression on an index $\mathbf{b} \in \{0,1\}^{\mu}$[2], and on the input $\mathbb{x}$ and witness $\mathbb{w}$. When writing constraints over polynomial rings $\mathbb{Q}[X], \mathbb{F}_q[X]$, UCS also allows more general constraints of the form $Q(\mathbf{b}, \mathbb{x}, \mathbb{w}) \in \mathfrak{I}$, where $\mathfrak{I}$ is an ideal of $\mathbb{Q}[X]$ or $\mathbb{F}_q[X]$. We call the latter an *ideal membership constraint*.

As we will see, the usage of ideals allows us to capture operations such as rotation of bitstrings (through membership in the ideal $(X^w - 1)$ generated by $X^w - 1$), relating bitstrings and integers (through membership in $(X - 2)$), constraints over multiple rings and fields (through membership in the kernel of the projection of $\mathbb{Z}[X]$ onto the ring), etc. We provide some examples next. See also Sections 2.1, 4.1 and 8.

Bitstrings and integers. A polynomial ring $R[X]$ with $R \in \{\mathbb{Q}, \mathbb{F}_q\}$ can be used to represent bitstrings: e.g. we can represent strings of, say, 32 bits as polynomials of degree < 32 with binary coefficients. We denote the latter set by $\{0,1\}^{<32}[X]$.

We can now relate bitstrings from $\{0,1\}^{<32}[X]$ and integers from $[0\,..\,2^{32}-1]$ through the ideal membership constraint

$$\hat{a}(X) - c \in (X - 2), \quad \hat{a}(X) \in \{0,1\}^{<32}[X], \ c \in [0\,..\,2^{32}-1], \quad \text{over } R[X].$$

This forces $\hat{a}$ to be the binary representation (in polynomial form) of the integer c.

Rotation. The constraint

$$\hat{b}(X) - X \cdot \hat{a}(X) \in (X^{32} - 1), \quad \hat{a}(X), \hat{b}(X) \in \{0,1\}^{<32}[X], \quad \text{over } R[X],$$

enforces that $\hat{b}$ is the left rotation of $\hat{a}$ by one bit.

Hamming weight. The constraint

$$\hat{a}(X) - c \in (X - 1), \quad \hat{a}(X) \in \{0,1\}^{<32}[X], \ c \in \mathbb{Z}, \quad \text{over } R[X].$$

states that the Hamming weight of $\hat{a}(X)$ is c.

Ring arithmetic. We can express constraints in most rings of interest, e.g. cyclotomic rings, through ideal membership constraints in $\mathbb{Z}[X]$.

Witnesses over $\mathbb{Z}, \mathbb{Z}[X]$ and shareability among different domains. As mentioned, UCS can express constraints over different rings and fields in the same instance. Accordingly, different parts of the witness are typed in different rings, e.g. a witness $\mathbb{w}$ may consist of three vectors $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$ with $\mathbf{f}_0$ having entries in $\mathbb{Z}[X]$ and $\mathbf{f}_1, \mathbf{f}_2$ having entries in finite fields $\mathbb{F}_{q_1}, \mathbb{F}_{q_2}$ (or in polynomial rings $\mathbb{F}_{q_i}[X]$). UCS then allows us to use the vector $\mathbf{f}_0$ in constraints over $\mathbb{Z}[X]$, $\mathbb{F}_{q_1}$, and $\mathbb{F}_{q_2}$

[2] A univariate framework would also be possible, but we do not describe it in this work.

simultaneously. In the latter two, $\mathbf{f}_0$ is projected onto $\mathbb{F}_{q_i}$ via a canonical projection $\psi_i : \mathbb{Z}[X] \to \mathbb{F}_{q_i}$ (the ring $\mathbb{Z}[X]$ homomorphically projects onto any finite field, not necessarily prime).

This way, in the above example, UCS expresses constraints over $\mathbb{Z}[X]$ involving the witness $\mathbf{f}_0$, and constraints over $\mathbb{F}_{q_i}$ involving $\psi_i(\mathbf{f}_0)$ and $\mathbf{f}_i$, for both $i = 1, 2$.

Additionally, one may, for example, express modular arithmetic modulo any integer n over $\mathbb{Z}$ and $\mathbb{Z}[X]$ by adding a term of the form $n \cdot u$ for some extra witness variable u (see the next example)[3].

XOR, rotation, and squaring modulo 2^{32}. As a final example, let $\phi_2 : \mathbb{Z}[X] \to \mathbb{F}_2[X]$ be the projection of $\mathbb{Z}[X]$ onto $\mathbb{F}_2[X]$ consisting of reducing coefficients modulo 2. Then for $\hat{a}(X), \hat{b}(X), \hat{c}(X) \in \{0,1\}^{<32}[X]$, the constraint

$$\hat{c} = \hat{a} \;\; \mathsf{XOR} \;\; (\hat{a}^2 \bmod 2^{32}),$$

can be expressed as

$$\begin{cases} \hat{a}(X), \hat{b}(X), \hat{c}(X) \in \{0,1\}^{<32}[X] \subseteq \mathbb{Z}[X], \quad u \in \mathbb{Z}, \\ \hat{b}(X) - \hat{a}(X)^2 - 2^{32} \cdot u \in (X-2), \quad \text{over } \mathbb{Z}[X], \\ \phi_2(\hat{c}(X)) - \phi_2(\hat{a}(X)) - \phi_2(\hat{b}(X)) = 0 \quad \text{over } \mathbb{F}_2[X], \end{cases}$$

see Sections 2.1 and 8 for a formal treatment.

Instantiations. The UCS constraint is highly flexible. One can instantiate it over multiple rings simultaneously, leveraging integer, polynomial ring, and finite field arithmetic simultaneously. Alternatively, for example, one could only use constraints over $\mathbb{F}_q[X]$, leveraging only the ideal-membership predicate. In this latter case, the scheme does not work with integer constraints whatsoever and is easily incorporable into any hash-based proving stack over $\mathbb{F}_q$. When instantiated this way, we call the resulting scheme the *Zinc+ add-on*. We describe this further in Section 1.1.3.

1.1.2 Zinc+ PIOP framework

Our framework for constructing PIOPs for UCS takes an existing PIOP $\Pi_{\mathbb{F}}$ over a finite field and applies a compilation step to turn $\Pi_{\mathbb{F}}$ into a PIOP for UCS. We illustrate how this works at a high level next.

For the sake of argument, say we have an R1CS-like ideal membership constraint over $R[X]$, where $R \in \{\mathbb{Q}, \mathbb{Z}, \mathbb{F}_q\}$:

$$\begin{aligned} &Q(b_0) \in \mathfrak{I} \qquad \forall\, b_0 \in \{0,1\}, \\ &Q(b_0) = \left(\sum_{\mathbf{b}} \tilde{A}(\mathbf{b}_0, \mathbf{b}) \cdot \tilde{\mathbf{z}}(\mathbf{b})\right) \cdot \left(\sum_{\mathbf{b}} \tilde{B}(\mathbf{b}_0, \mathbf{b}) \cdot \tilde{\mathbf{z}}(\mathbf{b})\right) - \left(\sum_{\mathbf{b}} \tilde{C}(\mathbf{b}_0, \mathbf{b}) \cdot \tilde{\mathbf{z}}(\mathbf{b})\right) \end{aligned} \tag{1}$$

where A, B, C are R1CS matrices with entries in $R[X]$ (typically in R), $\mathbf{z} = (\mathbb{x}, \mathbb{w})$ is a vector with entries in $R[X]$, and $\tilde{\cdot}$ denotes multilinear extension (MLE). Note that $\tilde{\mathbf{z}}$ is a multilinear polynomial whose coefficients are polynomials in $R[X]$.

In simple terms, the Zinc+ PIOP compiler works as follows:

Step 1: Sending oracles over $R[X]$. The prover sends an oracle $[[\tilde{\mathbf{z}}]]$ to $\tilde{\mathbf{z}}$. For simplicity, we assume the verifier can compute evaluation claims on $\tilde{A}, \tilde{B}, \tilde{C}$, instead of only having oracle access to them.

[3] This can also be performed over $\mathbb{F}_q$ and $\mathbb{F}_q[X]$. In Section 4.1 we discuss tradeoffs between working over $\mathbb{Z}[X]$ and $\mathbb{F}_q[X]$.

Step 2: Reducing ideal membership to a strict equality over $\mathbb{F}_{\tilde{q}}[X]$. When $R = \mathbb{Q}$ (or $\mathbb{Z}$), the verifier first samples a random $\Omega(\lambda)$-bit prime q_0 and both parties project all coefficients modulo q_0 via ϕ_{q_0}, bringing the constraint from $\mathbb{Q}[X]$ into $\mathbb{F}_{q_0}[X]$ (we set $\tilde{q} = q_0$). When $R = \mathbb{F}_q$, the constraint is embedded into $\mathbb{F}_{\tilde{q}}[X]$ for a sufficiently large extension $\mathbb{F}_{\tilde{q}}$ of $\mathbb{F}_q$. After this ring projection, all constraints are over $\mathbb{F}_{\tilde{q}}[X]$.

Now, if the constraint were a strict equality $Q(\mathbf{b}_0) = 0 \ \forall \ \mathbf{b}_0 \in \{0,1\}^\mu$, the standard procedure would be to replace it with $\tilde{Q}(\xi) = 0$ for a random point ξ. For ideal membership, Zinc+ proceeds similarly: the prover sends $e \in \mathbb{F}_{\tilde{q}}[X]$, supposedly $\tilde{Q}(\xi)$, the verifier checks $e \in \mathfrak{I}$, and the constraint is replaced by the strict equality

$$\tilde{Q}(\xi) - e = 0 \quad \text{over } \mathbb{F}_{\tilde{q}}[X]. \tag{2}$$

Step 3: From $\mathbb{F}_{\tilde{q}}[X]$ to $\mathbb{F}_{\tilde{q}}$. The verifier samples $a \leftarrow \mathbb{F}_{\tilde{q}}$ and evaluates (2) at $X = a$ via the evaluation map $\psi_a : \mathbb{F}_{\tilde{q}}[X] \to \mathbb{F}_{\tilde{q}}$, yielding a strict equality over $\mathbb{F}_{\tilde{q}}$:

$$\begin{aligned} &\psi_a(\tilde{Q}(\xi)) - \psi_a(e) = 0 \quad \text{over } \mathbb{F}_{\tilde{q}}. \\ &\psi_a(\tilde{Q}(\xi)) = \left(\sum_{\mathbf{b}} \widetilde{\psi_a(A)}(\psi_a(\xi), \mathbf{b}) \cdot \widetilde{\psi_a(\mathbf{z})}(\mathbf{b})\right) \cdot \left(\sum_{\mathbf{b}} \widetilde{\psi_a(B)}(\psi_a(\xi), \mathbf{b}) \cdot \widetilde{\psi_a(\mathbf{z})}(\mathbf{b})\right) \\ &\qquad\qquad - \left(\sum_{\mathbf{b}} \widetilde{\psi_a(C)}(\psi_a(\xi), \mathbf{b}) \cdot \widetilde{\psi_a(\mathbf{z})}(\mathbf{b})\right) \end{aligned} \tag{3}$$

Step 4: Finite-field PIOP. The resulting constraint (3) is a standard algebraic equality over $\mathbb{F}_{\tilde{q}}$, with input-witness vector $\psi_a(\mathbf{z})$, proved using any suitable finite-field PIOP $\Pi_{\mathbb{F}_{\tilde{q}}}$. The prover's oracle $[[\tilde{\mathbf{z}}]]$ supports projected-evaluation queries directly: a query at $\boldsymbol{\beta} \in \mathbb{F}_{\tilde{q}}^\nu$ returns $\widetilde{\psi_a(\mathbf{z})}(\boldsymbol{\beta}) \in \mathbb{F}_{\tilde{q}}$, the value $\Pi_{\mathbb{F}_{\tilde{q}}}$ expects from a finite-field oracle. When $R = \mathbb{Q}$, ψ_a may be undefined on some entry of $\mathbf{z}$ (in which case $\widetilde{\psi_a(\mathbf{z})}$ is shorthand rather than a true polynomial); the oracle returns $\perp$ in that case and the verifier rejects (Definition 5.3 more precisely defines the projected oracles we use).

Bit-size and degree bounds. Our PIOPs support witnesses with entries in $R^{<d,B}[X]$, i.e., polynomials of degree $< d$ with R-coefficients of bit-size $< B$. The PCS must ensure these bounds are approximately satisfied; see Remark 2.17 for details.

1.1.3 Zinc+ add-on

When instantiated with only $\mathbb{F}_q[X]$-constraints (no $\mathbb{Q}[X]$-constraints), the Zinc+ compiler simplifies considerably: the ring projection in Step 2 reduces to the canonical embedding $\iota_{\tilde{q}}$ (or is trivial when q is already large), and no prime projection or well-definedness concerns arise. We call this lightweight instantiation the *Zinc+ add-on*. It extends any existing hash-based SNARK $\Pi_{\mathbb{F}_q}$ over $\mathbb{F}_q$ with the ability to prove algebraic and ideal membership constraints over $(\mathbb{F}_q^{<d}[X])$, requiring only Steps 2 and 3 at the PIOP level and an interleaved commitment at the PCS level, with no modifications to existing components.

The advantage is that arithmetizing over $(\mathbb{F}_q^{<d}[X])$ leads to substantially more compact representations of computations involving bitwise operations, modular arithmetic, and similar primitives. For instance, when q is a prime with over 40 bits, a SHA-256 compression can be arithmetized with a 64×13 witness trace over $(\mathbb{F}_q^{<32}[X])$ using only linear constraints and lookup constraints on the trace columns. The main cost overhead comes from committing to the witness, which effectively

amounts to committing to a vector of size $n \cdot d$ over $\mathbb{F}_q$ (e.g., $n \cdot 32$ for SHA-256); the rest of the PIOP operates on the much smaller trace. See Section 2.2 for further discussion.

1.1.4 Zip+

We construct IOPs of Proximity (IOPPs) for proving evaluation claims such as $\widetilde{\psi(\mathbf{z})}(\beta) = c$, resulting from the last step of the Zinc+ PIOP.

In short, these IOPPs allow one to commit to multilinear polynomials $\tilde{f}$ with coefficients in $R^{<d,B}[X]$, where $R^{<d,B}[X]$ denotes the set of polynomials of degree less than d and coefficients in R of bit-size $< B$. Additionally, the IOPP allows one to prove evaluation claims over $R[X]$ or under a projection map ψ as in the last step of the Zinc+ PIOP.

The IOPP uses at its core an IOPP Π_R^{IOPP} with the same functionality *over* R instead of *over* $R[X]$. Concretely, given $\tilde{f}$ with coefficients in $R^{<d,B}[X]$, we write

$$\tilde{f} = \sum_{i=0}^{d-1} \tilde{f}_i \cdot X^i$$

for some multilinear polynomials $\tilde{f}_i$ with coefficients in $R^{<B}$, i.e. the set of elements from R of bit-size $< B$. We further note that

$$\widetilde{\psi(f)}(\beta) = \sum_{i=0}^{d-1} \widetilde{\psi(f)}_i \cdot a^i$$

Now, at a high level, to commit to $\tilde{f}$, our IOPP batch-commits to $\tilde{f}_0, \ldots, \tilde{f}_{d-1}$ with Π_R^{IOPP}. To prove the evaluation claim $\widetilde{\psi(f)}(\beta) = c$, the prover publishes claims $\widetilde{\psi(f)}_i(\beta) = c_i$, the verifier checks $\sum_{i=0..d-1} c_i \cdot a^i = c$ in $\mathbb{F}_{q'}$, and the prover and verifier use Π_R^{IOPP} to batch-prove the evaluation claims $\widetilde{\psi(f)}_i(\beta) = c_i$, $i = 0..d-1$.

When R is a finite field $\mathbb{F}_q$, Π_R^{IOPP} can be any IOPP over $\mathbb{F}_q$, preferably with good batching capabilities and with capacity to prove evaluation claims over extensions of $\mathbb{F}_q$.

When $R = \mathbb{Q}$, we use our own IOPP over $\mathbb{Q}$, which we call *Zip+* (Section 2.4). We also use the name Zip+ for the IOPP over $\mathbb{Q}[X]$. Zip+ is similar to its predecessor Zip [Gar+25; GWHD25], in that it is a modern version of the Brakedown/Ligero PCS [GLSTW23; AHIV17] over $\mathbb{Q}$. The main improvements are: 1) the so-called test and evaluation phases are merged into one, 2) support for batched commitments and evaluation claim proving, 3) support for distance parameters up to the Mutual Correlated Agreement (MCA) bound of the underlying code (Definition 3.3).

Another crucial difference between Zip+ and Zip is the linear code used to instantiate it, namely *Integer Pseudo Reed-Solomon (IPRS)* codes, as discussed next.

See Section 2.5 for benchmarks of Zip+ (Table 4).

Soft range checks and bit-size checks in Zip and Zip+. Recall that, when $R = \mathbb{Q}$, our PCS and IOPP must enforce that the coefficients of the committed $\tilde{f}$ have bit size at most $\mathsf{poly}(B)$. Thus, we ensure that this is enforced during our batching step, followed by our $\Pi_{\mathbb{Q}}^{\mathsf{IOPP}}$ IOPP.

1.1.5 Integer Pseudo Reed–Solomon (IPRS) codes

We introduce a new linear code over the field of rational numbers $\mathbb{Q}$ and over the ring of integers $\mathbb{Z}$, which we call *Integer Pseudo Reed–Solomon (IPRS) code* (Section 2.3), and which we believe

is of independent interest. This code is not a Reed–Solomon (RS) code, but it presents similar characteristics, namely it is minimum distance separable and admits an efficient FFT encoding process. Additionally, IPRS codes do not blow up the bit-size of encoded messages, as we explain below.

Our IOPP and multilinear PCS Zip+ relies at its core on linear error-correcting codes over $\mathbb{Q}$. As with other code-based IOPPs, the performance of Zip+ is heavily influenced by both the minimum distance of the underlying code and the efficiency of its encoding. Additionally, when working over $\mathbb{Q}$, one also needs to ensure that the bit-size of the codewords does not blow up compared to the bit-size of the messages.

IPRS codes in a nutshell When working over small fields, Reed-Solomon (RS) codes are a natural choice as the underlying code. These codes have optimal minimum distance and support efficient FFT-based encoding. One could try to use RS codes over $\mathbb{Q}$ or $\mathbb{Z}$ as well — for example, by naively defining RS codes over $\mathbb{Q}$ as the set of vectors obtained by evaluating low-degree polynomials on integer points. However, for practical code dimensions, the bit-size of the entries of the resulting codewords would be in the thousands or even millions, which would result in an inefficient IOPP. Alternatively, one could "lift" a standard RS code $\mathsf{RS}[\mathbb{F}_q, n, k]$ from a small field $\mathbb{F}_q$ onto $\mathbb{Q}$ by looking at the Vandermonde generating matrix of $\mathsf{RS}[\mathbb{F}_q, n, k]$ as an integer matrix with entries in $\{0, \ldots, q-1\}$, and defining the lifted code as the set of rational linear combinations of the rows of this matrix. However, this code does not seem to have an efficient FFT encoding.

IPRS codes constitute a middle ground between the two aforementioned RS-based constructions. In short, they are obtained by "lifting" to $\mathbb{Q}$ the FFT encoding algorithm for a standard RS code over a small field $\mathbb{F}_q$. Concretely, we fix such an FFT algorithm over $\mathbb{F}_q$ (by specifying its radix, twiddle factors, depth, etc.), and then we look at its twiddle factors and other constants as integers from $\{-(q-1)/2, \ldots, (q-1)/2\}$, and perform the FFT algorithm over $\mathbb{Q}$, without performing modular reduction. The IPRS code is defined as the set of vectors obtained by applying such a lifted FFT algorithm. In the technical overview (Section 2.3) we further discuss the performance and properties of these codes.

Vector length	2^8	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}
$[0\,..\,2^{32}-1]$	37.7 µs	72.4 µs	213.2 µs	585.7 µs	2.83 ms	15.04 ms
$\{0,1\}^{<32}[X]$	266.9 µs	552.3 µs	2.44 ms	8.42 ms	31.84 ms	176.85 ms

Table 2: Benchmark for encoding vectors of various lengths using IPRS codes on a MacBook Air M4 16GB, multithreaded. Vectors contain either 32-bit integers, or binary polynomials of degree < 32.

RAA and JEA codes. Non-RS-based codes such as RAA and JEA codes [BFKTWZ24; BCFRRZ25] can be defined over $\mathbb{Q}$ and $\mathbb{Z}$ without bit-size blowup issues and with efficient encoding. However, their *proved* minimum distance is significantly smaller than the optimum MDS distance of RS codes and our IPRS codes. Thus, these codes could be a good fit when proof size is not a priority. There is also the possibility that RAA or JEA codes over $\mathbb{Q}$ (and over large fields) have much better minimum distance than what is currently proved . In that case, they could become a good alternative to our IPRS codes in Zip+.

1.2 Related work

SNARKs and PCSs for integer constraints: Zartan[CH24], [BHRRS21; BF22] Zinc[Gar+25; GWHD25], Zinc+, and $\mathbb{Q}\bar{a}ren$ [SSPP26]. Zinc [Gar+25; GWHD25] presents a framework for building proof systems for constraints over $\mathbb{Z}$ or $\mathbb{Q}$ and introduces a hash-based PCS for polynomials with coefficients in $\mathbb{Q}$, built with a tensor IOPP similar to the one in Brakedown [GLSTW23].

Our work improves upon [Gar+25; GWHD25] in the following ways. (1) Our UCS constraints widely generalize Zinc's, which is restricted to constraints over $\mathbb{Z}$ and $\mathbb{Q}$, by additionally supporting multiple finite fields, polynomial rings $R[X]$, and ideal-membership predicates. (2) We design a PIOP framework for these more general constraints. Additionally, we prove round-by-round knowledge soundness, allowing for Fiat-Shamir compilation. (3) Our IOPP Zip+ is an improvement to the PCS from [Gar+25; GWHD25] in that it supports polynomials over a wide range of rings and fields (as opposed to only $\mathbb{Q}$), can handle proximity parameters up to the MCA bound of the underlying code (Definition 3.3) (this greatly helps reduce proof size and verifier time), merges the so-called "test" and "evaluate" phases into one, is proved to be round-by-round knowledge sound, and has batching capabilities. (4) We improve upon the underlying linear code with our IPRS codes. (5) We demonstrate the practicality of our approach with our implementation and evaluation.

Zartan [CH24] builds SNARKs over $\mathbb{Z}$ by compiling so-called mod-AHPs (roughly, a PIOP that has integer polynomials as oracles but makes checks modulo a random prime) with a PCS for integral polynomials based on hidden-order groups [BHRRS21; BF22]. We adopt the idea from [CH24] of working modulo a random prime, but replace the hidden-order-group PCS with a hash-based PCS (Zip+), thereby avoiding the applicability issues and the stronger cryptographic assumptions that hidden-order-group commitments require. Moreover, as in the comparison with Zinc [Gar+25; GWHD25], our scheme works with constraint systems that are substantially more expressive than Zartan's.

The concurrent work $\mathbb{Q}\bar{a}ren$ [SSPP26] provides a compiler that takes an arbitrary PCS for multilinear polynomials over a finite field $\mathbb{F}_p$, and outputs a so-called relaxed mod-PCS for multilinear polynomials over $\mathbb{Q}$. A relaxed mod-PCS is a PCS that allows proving evaluation claims modulo a prime, and whose extractor outputs polynomials with rational coefficients. While of theoretical interest, the construction seems impractical (the paper does not provide time costs) when compared to our Zip+, because, to commit to, say, a vector $\mathbf{v} \in \mathbb{Z}^n$ of integers of B bits each, $\mathbb{Q}\bar{a}ren$ essentially requires committing to at least the vector $\mathbf{v}' \in \{0,1\}^{n \cdot B}$ of the bit decomposition of the entries of $\mathbf{v}$, with the underlying finite field PCS (further, $\mathbb{Q}\bar{a}ren$ requires committing to several vectors of this type). In terms of practicality, we do believe that $\mathbb{Q}\bar{a}ren$ may be a good fit for committing to our vectors with entries in $\{0,1\}^{<32}[X]$. We leave this as future work.

Binary-field SNARKs. Binius [DP25; DP24] is a series of SNARKs for relations over binary fields $\mathbb{F}_{2^k}$. Binary fields are more friendly to bitstring operations than prime fields. For example, bitwise XOR can be modeled as addition in $\mathbb{F}_{2^k}$. However, AND, rotation, and modular-2^w arithmetic remain non-native in binary fields. Despite this, Binius achieves significant improvements on computations involving all said operations, when compared to prime-field SNARKs [Fou]. We believe this is in part due to $\mathbb{F}_{2^k}$ being more amenable to some of these operations, and due to the *pay-per-bit* capacity of Binius' IOPP.

Proof systems over rings. There are a number of works describing SNARKs over different types of specific rings, such as cyclotomic rings [BS23], Galois rings [HMZ25; WZD25; GNS23], and abstract rings with large exceptional sets [BCS21; ACCDE22; Sor22]. [LXY24; BBMRS21; BBMS22] provide approaches for proving statements modulo 2^n based on VOLE techniques, but

yield non-succinct schemes. Unlike our work, none of these approaches support the wide range of rings and fields that we can handle with our framework, including constraints over $\mathbb{Z}$ and $\mathbb{Z}[X]$. The recent works FREPack [SZ25] and Swirl [Con26] treat the problem of allowing different constraint domains within the same proof system.

Applications. In this work we discuss the example of SHA-256 hashing followed by ECDSA signature verification. Two recent approaches to this, which additionally include document parsing, are [Fs24; KS25]. As future work, we plan to include document parsing in our arithmetizations and compare our scheme with these approaches.

Acknowledgments

We thank Remco Bloemen, Benedikt Bünz, Giacomo Fenzi, Ben Fisch, Babis Papamanthou, Katerina Sotiraki, Alin Tomescu, Arantxa Zapico, and our anonymous reviewers for helpful comments and discussions.

The authors used AI-based tools for grammar checking, editing, and assisting with code generation during the preparation of this manuscript. The authors bear full responsibility for the correctness and originality of all content.

2 Technical Overview

We now provide a high-level technical explanation of the main contributions of this work.

2.1 Universal Constraint Systems (UCS)

The goal of this section is to illustrate a constraint system that can express many computations of practical interest with almost no arithmetization overhead. The key question once this system is defined is whether it can be efficiently handled in proof systems. This is answered positively later in the subsequent sections.

2.1.1 Basic notation

We first introduce some notation, see Section 3 for more background. We let R denote either a finite field $\mathbb{F}$, or $\mathbb{Z}$ or $\mathbb{Q}$. Further background can be found in Section 3. Given a ring or field R, we write

$$R^{<d,B}[X] = \{\text{polynomials of degree less than } d \text{ with coefficients in } R \text{ of bit-size at most } B\}$$

When R is a finite field, we often drop B and write $R^{<d}[X]$, understanding there is no bound on the bitsize of coefficients. Note that $R^{<1}[X] = R$.

Given $w \geq 1$ we let $[0\,..\,2^w - 1]$ denote the set of integers $\{0, \ldots, 2^w - 1\}$. We define the set of so-called *bit-polynomials* of degree $< w$ as

$$\{0,1\}^{<w}[X] \;\coloneqq\; \left\{ \sum_{i=0}^{w-1} b_i\, X^i \;\middle|\; b_i \in \{0,1\} \right\} \;\subseteq\; R[X]. \tag{4}$$

When R is $\mathbb{Z}, \mathbb{Q}$ or a finite field of large enough characteristic, the set $\{0,1\}^{<w}[X]$ is in bijective correspondence with $[0\,..\,2^w - 1]$ via the evaluation map

$$\psi_2 \colon R[X] \to R \qquad \text{defined by} \qquad \psi_2(\hat{u}(X)) \coloneqq \hat{u}(2).$$

Looking ahead, we note that ψ_2 is the canonical surjective ring homomorphism $R[X] \twoheadrightarrow R[X]/(X-2) \cong R$ (this will be relevant later when designing our constraint system). Here, $(X-2)$ denotes the ideal generated by $X-2$ in the ring $R[X]$.

We emphasize that, when $R \neq \mathbb{F}_2$, $\{0,1\}^{<w}[X]$ is not the set $\mathbb{F}_2^{<w}[X]$. For example, if $R = \mathbb{Q}$, then adding the elements $(1+X^2), (1+X+X^2) \in \{0,1\}^{<w}[X]$ results in $2+X+2X^2$, while in $\mathbb{F}_2^{<w}[X]$ we have $(1+X^2)+(1+X+X^2) = X$.

2.1.2 Using $R = \mathbb{Q}$ or $R = \mathbb{Z}$, correct typing of witnesses

$\mathbb{Q}$ vs $\mathbb{Z}$ for the typing of witnesses For most applications, we naturally want to arithmetize either over $\mathbb{F}[X]$ or *over* $\mathbb{Z}[X]$, and usually never over $\mathbb{Q}[X]$, i.e. we want to use witnesses with entries in $\mathbb{F}[X]$ or $\mathbb{Z}[X]$ (or $\mathbb{F}$, $\mathbb{Z}$). However, throughout the text, we mostly only consider $R = \mathbb{F}$ or $R = \mathbb{Q}$. The reason is technical, but important: our IOPPs and PCSs work exclusively over fields or over polynomial rings over fields. This excludes $\mathbb{Z}$ and $\mathbb{Z}[X]$. In particular, our IOPP/PCS extractors output witnesses with entries in $\mathbb{F}, \mathbb{F}[X], \mathbb{Q}$, or $\mathbb{Q}[X]$. We then, when desired, enforce integrality by lookup-constraining the relevant witness elements to belong to $\mathbb{Z}$ or $\mathbb{Z}[X]$.

In other words, **the typing $\mathbb{F}, \mathbb{F}[X], \mathbb{Q}, \mathbb{Q}[X]$ for witness elements is enforced by our IOPP/PCS, while the typing $\mathbb{Z}, \mathbb{Z}[X]$ is enforced by lookup constraints, which are proved as part of our PIOP**. We refer to the technical overview of [Gar+25; GWHD25] for a thorough discussion of this design strategy.

We remark that, for all applications we have encountered, most integrality lookup-constraints are subsumed by other constraints placed on the witness. See Section 2.1.3 for an example.

2.1.3 Motivating example: Simultaneous bitstring operations and arithmetic mod 2^w

To describe and motivate our arithmetization framework we focus on two running examples. The first one has the following function f at its core. Let $r \in [w-1]$ be a rotation parameter. Let R be either $\mathbb{Q}$ or a finite field $\mathbb{F}$ of large enough characteristic. Let $n \in [0\,..\,2^w]$ be an integer, possibly non-prime. Now define

$$\begin{aligned} f : [0\,..\,2^w-1] \subseteq R &\to [0\,..\,2^w-1] \subseteq R, \\ x &\mapsto (x^2 \bmod n) \ \ \mathsf{XOR} \ \ \mathsf{ROTR}^r(x), \end{aligned} \tag{5}$$

where XOR is the bitwise exclusive or (XOR) operation and ROTR^r is right rotation by r bits (see Section 3 for formal definitions). We extend these operations to the set $\{0,1\}^{<w}[X] \subseteq R[X]$ by applying them coefficient wise. In the definition of f (Eq. (5)), the expression $x^2 \bmod n$ means that x is understood as an integer in $[0\,..\,2^w-1]$, squared, and then the canonical representative of x^2 modulo 2^w in $[0, n-1]$ is returned.

We think of f as a proxy for some of the subroutines in computations such as classic hashing, CPU and zkVM instructions, zkID authentication systems, TLS, etc. See Table 1. For example, the Σ_0 subroutine in SHA-256 is defined by the function

$$\Sigma_0(x) \coloneqq \mathsf{ROTR}^2(x) \ \mathsf{XOR} \ \mathsf{ROTR}^{13}(x) \ \mathsf{XOR} \ \mathsf{ROTR}^{22}(x),$$

and is called by

$$T_2(a,b,c) \coloneqq \Sigma_0(a) + \mathsf{Maj}(a,b,c) \pmod{2^w},$$

where Maj, like Σ_0, operates bitwise. See Section 8.2 for details on SHA-256.

Given a number of iterations $n \geq 1$, our goal is to design a proof system for the relation REL_f defined as

$$\mathsf{REL}_f = \left\{ (\mathbb{x}; \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{x} = (u_1, u_n) \in [0\,..\,2^w - 1]^2, \\ \mathbb{w} = (u_2, \ldots, u_{n-1}) \in [0\,..\,2^w - 1]^{n-2}, \\ u_{t+1} = f(u_t) \text{ for all } t \in [n-1] \end{array} \right\}. \tag{6}$$

Before diving into arithmetizing REL_f, we make a series of observations about the operations used by the function f, which will motivate our arithmetization framework.

XOR and AND For any $\hat{u}, \hat{v} \in \{0,1\}^{<w}[X] \subseteq R[X]$, the following identity holds in $\mathbb{Z}[X]$:

$$\hat{u} + \hat{v} \;=\; (\hat{u} \;\mathsf{XOR}\; \hat{v}) + 2 \cdot (\hat{u} \;\mathsf{AND}\; \hat{v}) \qquad \text{in } R[X], \tag{7}$$

The converse also holds: if $\hat{u}, \hat{v}, \hat{z}, \hat{w} \in \{0,1\}^{<w}[X]$ satisfy $\hat{u} + \hat{v} = \hat{z} + 2 \cdot \hat{w}$ in $R[X]$, then $\hat{z} = \hat{u} \;\mathsf{XOR}\; \hat{v}$ and $\hat{w} = \hat{u} \;\mathsf{AND}\; \hat{v}$. Both directions follow from the fact that $b + b' = (b \;\mathsf{XOR}\; b') + 2\,(b \;\mathsf{AND}\; b')$ for any two $b, b' \in \{0,1\}$, applied coefficient-wise.

Hence, informally speaking, provided the witness elements belong to $\{0,1\}^{<w}[X]$, the identity (7) arithmetizes both XOR and AND over $R[X]$ simultaneously.

A further consequence of (7) is that, when $R = \mathbb{Q}$, for any $\hat{u}, \hat{v}, \hat{z} \in \{0,1\}^{<w}[X]$,

$$\hat{z} = \hat{u} \;\mathsf{XOR}\; \hat{v} \quad \Longleftrightarrow \quad \phi_2(\hat{z}) = \phi_2(\hat{u}) + \phi_2(\hat{v}), \tag{8}$$

where

$$\phi_2 \colon \mathbb{Z}[X] \to \mathbb{F}_2[X]$$

denotes the map applying coefficient-wise reduction modulo 2 to a given integer polynomial.

In summary, one can expose the XOR of two elements of $\{0,1\}^{<w}[X]$ either via a single equality over $R[X]$ (which additionally yields the AND) or through addition modulo 2 (only when $R = \mathbb{Q}$). We leverage both approaches in our arithmetization of SHA-256.

Looking ahead, our constraint system will allow us to take projections modulo a prime of witness vectors with entries typed in $\mathbb{Q}[X]$.

Rotation through ideal membership For any $\hat{u} \in \{0,1\}^{<w}[X] \subseteq R[X]$, the following ideal-membership relation holds in $R[X]$:

$$\mathsf{ROTR}^r(\hat{u}) - X^{w-r} \cdot \hat{u} \in (X^w - 1). \tag{9}$$

where $(X^w - 1)$ denotes the ideal in $R[X]$ generated by the polynomial $X^w - 1$. Conversely, for any two $\hat{u}, \hat{v} \in \{0,1\}^{<w}[X]$, if $\hat{v} - X^{w-r} \cdot \hat{u} \in (X^w - 1)$ then $\hat{v} = \mathsf{ROTR}^r(\hat{u})$ (Lemma 8.8). Hence rotation *for elements in* $\{0,1\}^{<w}[X]$ can be enforced by the ideal-membership predicate $\hat{v} - X^{w-r}\hat{u} \in (X^w - 1)$.

Mixing XOR and rotation Assume $R = \mathbb{Q}$. Given elements $\hat{u}, \hat{v}, \hat{z} \in \{0,1\}^{<w}[X] \subseteq R[X]$, we want to arithmetize the constraint $\hat{z} = \hat{u} \;\mathsf{XOR}\; \mathsf{ROTR}^r(\hat{v})$. To do so, observe that as long as $\hat{u}, \hat{v}, \hat{z}$ belong to $\{0,1\}^{<w}[X]$, the following equivalence holds:

$$\hat{z} = \hat{u} \;\mathsf{XOR}\; \mathsf{ROTR}^r(\hat{v}) \quad \Longleftrightarrow \quad \phi_2(\hat{u}) + \phi_2(\hat{v}) \cdot X^{w-r} - \phi_2(\hat{z}) \in (X^w - 1) \qquad \text{in } \mathbb{F}_2[X]. \tag{10}$$

We sketch why the converse direction of the equivalence holds; see Lemma 8.9 for a full argument. Assume $\hat{u}, \hat{v}, \hat{z} \in \{0,1\}^{<w}[X]$ satisfy the right-hand side of (10) in $\mathbb{F}_2[X]$. Write

$\phi_2(\hat{v}) \cdot X^{w-r} = \hat{\alpha} + (X^w - 1) \cdot \hat{\beta}$ with $\hat{\alpha} \in \mathbb{F}_2[X]$ of degree less than w, and $\hat{\beta} \in \mathbb{F}_2[X]$. Let $\hat{y} \in \{0,1\}^{<w}[X] \subseteq \mathbb{Z}[X]$ be such that $\phi_2(\hat{y}) = \hat{\alpha}$. Then the right-hand side of Eq. (10) implies that $\phi_2(\hat{u}) + \phi_2(\hat{y}) - \phi_2(\hat{z}) \in (X^w - 1)$ in $\mathbb{F}_2[X]$. Since $\hat{u}, \hat{z} \in \{0,1\}^{<w}[X]$, all elements in the right-hand side of this expression have degree less than w. Hence $\phi_2(\hat{u}) + \phi_2(\hat{y}) - \phi_2(\hat{z}) = 0$ in $\mathbb{F}_2[X]$. Since also $\hat{y} \in \{0,1\}^{<w}[X]$, it follows that $\hat{z} = \hat{u}$ XOR $\hat{y}$ in $\mathbb{Z}[X]$ (cf. Eq. (10)). Finally, we have seen that $\phi_2(\hat{v}) \cdot X^{w-r} - \phi_2(\hat{y}) \in (X^w - 1)$ in $\mathbb{F}_2[X]$. By Eq. (9), $\hat{y} = \mathsf{ROTR}^r(\hat{v})$ in $\mathbb{Z}[X]$. This proves Eq. (10)

The equivalence in (10) can be generalized to more complex expressions involving several XORs and rotations. It is particularly useful for arithmetizing some inner functions of SHA-256 and related hashes and ciphers.

When $R = \mathbb{F}_q$, or if one does not wish to use the projection ϕ_2 one can also perform a similar arithmetization through

$$\begin{aligned} &\hat{z} = \hat{u} \text{ XOR } \mathsf{ROTR}^r(\hat{v}) \\ &\qquad \iff \quad \hat{u} + \hat{v} \cdot X^{w-r} - \hat{z} - 2 \cdot \hat{y} \in (X^w - 1) \qquad \text{in } R[X], \text{ for some } \hat{y} \in \{0,1\}^{<w}[X]. \end{aligned} \tag{11}$$

In this case, $\hat{y}$ is in fact $\hat{u}$ AND $\mathsf{ROTR}^r(\hat{v})$.

Linking bit-polynomials to integers It is often convenient to move between bit-string representations (as a binary polynomial, i.e. an element from $\{0,1\}^{<w}[X]$) to the integer represented by the bitstring. For example, say one has written several constraints as in the previous examples, involving elements from $\{0,1\}^{<w}[X]$, and has encoded several bit-wise operations. One may want to turn the output of some of these operations into integer form, and continue operating with the resulting integers under, say, modular integer arithmetic modulo 2^w.

To do so, we arithmetize the bijection $\psi_2 : \{0,1\}^{<w}[X] \to [0\,..\,2^w - 1]$ by observing that, given $\hat{v} \in \{0,1\}^{<w}[X], u \in [0\,..\,2^w - 1]$, we have

$$u = \psi_2(\hat{v}) \quad \iff \quad \hat{v} - u \in (X - 2) \quad \text{over } R[X].$$

To quickly see why, for the reader familiar with commutative algebra, note ideal membership effectively is equivalent to equality in the quotient $R[X]/(X-2)$. The natural projection onto that quotient is precisely ψ_2, i.e. the map that replaces X by 2. An alternative argument is as follows: if $\hat{v} - u \in (X-2)$, then $\hat{v} = u + (X-2) \cdot \hat{q}$ over $R[X]$, for some polynomial $\hat{q}(X) \in R[X]$. Evaluating ψ_2 on both sides gives $\psi_2(\hat{v}) = \psi_2(u) + 0 \cdot \hat{q}(X) = u$. The converse follows similarly.

Squaring modulo an arbitrary integer n We would now like to, given $\hat{u}, \hat{v} \in \{0,1\}^{<w}[X] \subseteq R[X]$, express the constraint that the integer represented by $\hat{v}$, i.e. $\psi_2(\hat{v})$, is the square of the integer represented by $\hat{u}$, i.e. $\psi_2(\hat{u})$, modulo an arbitrary integer n. We can do so by observing the following:

$$\psi_2(\hat{v}) = \psi_2(\hat{u})^2 \bmod n \quad \iff \quad \hat{v} - \hat{u}^2 + n \cdot \mu \in (X - 2) \qquad \text{in } R[X], \tag{12}$$

for some $\mu \in R$. If R is a field extension of a prime field $\mathbb{F}_{q_0}$, then we must require $|\mu| \leq |\psi_2(\hat{v}) - \psi_2(\hat{u})^2|/n$ as integers.

The justification for (12) is similar to that of the previous example, with the added observation that two integers are congruent modulo n if and only if their difference is a multiple of n, assuming $R = \mathbb{Q}$ or the characteristic q_0 of $R = \mathbb{F}_q$ is large enough in terms of $\psi_2(\hat{v}), \psi_2(\hat{u})$ (and, implicitly, μ). Concretely, in the latter case $R = \mathbb{F}_q$, assume there exists $\mu \in R$ such that $\hat{v}(X) - \hat{u}(X)^2 + n \cdot \mu \in (X-2)$ in $R[X]$. Then $\hat{v}(X) - \hat{u}(X)^2 + n \cdot \mu$ vanishes at $X = 2$, so $\hat{v}(2) - \hat{u}(2)^2 + n \cdot \mu = 0$ in R. Since $\hat{v}(2), \hat{u}(2)^2, n$ all belong to the base prime subfield $\mathbb{F}_{q_0} \subseteq \mathbb{F}_q$, we have q_0 divides $\hat{v}(2) - \hat{u}(2)^2 + n \cdot \mu$

as integers. Because we assumed q_0 is large enough, this means $\hat{v}(2) - \hat{u}(2)^2 + n \cdot \mu = 0$ as integers, which implies $\psi_2(\hat{v}) = \psi_2(\hat{u})^2 \bmod n$. The forward direction is analogous.

A similar approach arithmetizes other algebraic modular operations.

Remark 2.1 (Lookup or typing constraints)**.** In all examples above, the typing of elements is crucial. I.e. most of the ideal membership constraints described above lose their intended effect if one does not pair them with *lookup* or *typing* constraints, e.g. the constraint (12) must be accompanied by the lookup constraints $\hat{v}, \hat{u} \in \{0,1\}^{<w}[X]$ and $\mu \in \mathbb{Z}$. Because of this, our arithmetizations in practice consist mostly of algebraic constraints (strict equalities or ideal membership), plus lookup constraints to enforce correct typing of the witness entries.

In practice, lookup constraints are proved as part of our PIOPs. In particular, these constraints are projected onto a random finite field $\mathbb{F}'$ and proved with standard lookup PIOP over $\mathbb{F}'$. Everything regarding the lookup, including chunk decomposition and chunk commitments, multiplicity commitments, etc., is done entirely over $\mathbb{F}'$.

Arithmetization of REL_f We now put together a full arithmetization for the relation REL_f. Recall that each application of the function f computes

$$u_{t+1} \;=\; \big(u_t^2 \bmod n\big) \;\; \mathsf{XOR} \;\; \mathsf{ROTR}^r\big(u_t\big),$$

and t ranges from 1 to $n-1$. For simplicity we assume $R = \mathbb{Q}$, see Remark 2.2 for comments about $R = \mathbb{F}$.

Witness. For each step $t \in [n-1]$ we introduce three witness elements $\hat{u}_t$, $\hat{v}_t$, and μ_t. For convenience here, we think of the witness as a $n \times 3$ matrix, or *trace*[4]. The following table describes each witness element, its intended meaning, and the set where it is constrained to belong.

Entry	Intended meaning	Lookup constraint
$\hat{u}_t$	bit-polynomial representation of u_t at step t	$\in \{0,1\}^{<w}[X] \subseteq \mathbb{Q}[X]$
$\hat{v}_t$	bit-polynomial representation of $(u_t)^2 \bmod n$	$\in \{0,1\}^{<w}[X] \subseteq \mathbb{Q}[X]$
μ_t	the quotient such that $v_t = u_t^2 + \mu_t \cdot n$	$\in \mathbb{Z}$ (cf. Section 2.1.2)

Constraints. For each step $t \in [n-1]$ we impose two constraints.

1. **Squaring modulo n.** For each $t \in [n-1]$ we impose $\hat{v}_t - \hat{u}_t^2 + n \cdot \mu_t \;\in\; (X-2)$ in $\mathbb{Q}[X]$. By Eq. (12), this forces $v_t \equiv u_t^2 \bmod n$.

2. **XOR and rotation.** For each $t \in [n-1]$ we impose $\phi_2\big(\hat{u}_{t+1}\big) - \phi_2\big(\hat{v}_t\big) - X^{w-r} \cdot \phi_2\big(\hat{u}_t\big) \;\in (X^w - 1)$ in $\mathbb{F}_2[X]$. By (10), this constraint enforces that $\hat{u}_{t+1} = \hat{v}_t \;\mathsf{XOR}\; \mathsf{ROTR}^r(\hat{u}_t)$.

Constraints also include the lookup constraints described in the table above. We omit boundary constraints for brevity.

Remark 2.2 (Arithmetization over $\mathbb{F}[X]$)**.** One can arithmetize REL_f over $\mathbb{F}[X]$, rather than over $\mathbb{Z}[X]$, if the characteristic of $\mathbb{F}$ is large enough. The arithmetization is exactly the same, except that one must place appropriate range constraints on the witness elements μ_t.

[4]This is just for convenience; our definition of UCS can but does not restrict to working with traces and AIR-like systems.

The reader may appreciate the proposed arithmetization as being both conceptually simple and small (in the sense of it requiring a small number of constraints given the relation sought to arithmetize). The question now is whether such arithmetization *is useful in practice*, i.e. whether it can be efficiently handled by proof systems. One of the main contributions of this work is to show that the answer is yes. We outline why and how in the next sections of the present technical overview.

2.1.4 Virtual access to bit-rotation, shift, etc.

It is possible to, given a vector $\mathbf{v} \in \{0,1\}^{<w}[X]^n \subseteq R^{<w}[X]$, gain access to the MLE $\mathsf{mle}[\mathsf{ROTR}^c(\mathbf{v})]$ of the entry-wise rotation $\mathsf{ROTR}^c(\mathbf{v})$ of $\mathbf{v}$, from access to $\mathsf{mle}[\mathbf{v}]$. Concretely, if $\mathbf{v} = (v_i)_{i\in[n]}$, then $\mathsf{ROTR}^c(\mathbf{v}) = (\mathsf{ROTR}^c(v_i))_{i\in[n]}$. This applies as well to other operations such as SHIFT, and in fact to any operation T that can be modeled as a linear R-module map $T : R^{<w}[X] \to R^{<w}[X]$, as shown in the following result.

Lemma 2.3 (R-linear coordinatewise maps commute with MLE)**.** *Let R be a commutative ring, $w \geq 1$, and let $T\colon R^{<w}[X] \to R^{<w}[X]$ be an R-module homomorphism, i.e. a map such that $T(f(X)+g(X)) = T(f(X))+T(g(X))$ and $T(r\cdot f(X)) = r\cdot T(f(X))$ for all $f(X), g(X) \in R^{<w}[X]$ and $r \in R$. For any $\mu \geq 0$, any vector $\mathbf{v} = (v_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^\mu} \in (R^{<w}[X])^{2^\mu}$, and any evaluation point $\mathbf{r} \in R^\mu$,*

$$\widetilde{T(\mathbf{v})}(\mathbf{r}) \;=\; T\big(\widetilde{\mathbf{v}}(\mathbf{r})\big) \qquad \text{in } R^{<w}[X], \tag{13}$$

where $T(\mathbf{v}) \coloneqq (T(\mathbf{v}_{\mathbf{b}}))_{\mathbf{b}\in\{0,1\}^\mu}$.

Proof. By definition, $\widetilde{\mathbf{v}}(\mathbf{r}) = \sum_{\mathbf{b}\in\{0,1\}^\mu} \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{r})\cdot\mathbf{v}_{\mathbf{b}} \in R^{<w}[X]$. Applying T and using R-linearity,

$$T\big(\widetilde{\mathbf{v}}(\mathbf{r})\big) = \sum_{\mathbf{b}\in\{0,1\}^\mu} \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{r})\cdot T(\mathbf{v}_{\mathbf{b}}) = \widetilde{T(\mathbf{v})}(\mathbf{r}).$$

□

Lemma 2.3 allows in practice to enable virtual access to the MLE of entry-wise rotations and shifts of witness vectors. For example, one could modify the arithmetization from the previous Section 2.1.3 as follows. There, the XORrotation constraint reads

$$\phi_2(\hat{u}_{t+1}) - \phi_2(\hat{v}_t) - X^{w-r}\cdot\phi_2(\hat{u}_t) \in (X^w - 1) \qquad \text{in } \mathbb{F}_2[X],$$

where the rotation $\mathsf{ROTR}^r(\hat{u}_t)$ is encoded indirectly as the multiplication $X^{w-r}\cdot\phi_2(\hat{u}_t)$ modulo (X^w-1). Using the lemma, this can be replaced by the equivalent equality

$$\phi_2(\hat{u}_{t+1}) - \phi_2(\hat{v}_t) - \phi_2\big(\mathsf{ROTR}^r(\hat{u}_t)\big) = 0 \qquad \text{in } \mathbb{F}_2[X],$$

where $\mathsf{ROTR}^r(\hat{u}_t)$ is now treated as a *virtual column* obtained from the committed column $\hat{u}_t$.

In some scenarios it can be preferable to keep the ideal-membership constraint approach from Section 2.1.3. For example, in our SHA-256 arithmetization we keep the ideal-based approach, to save on the number of virtual columns used by the prover. On the other hand, we do leverage the above lemma to enable virtual access to SHIFT operations.

2.1.5 Further examples

We highlight other applications of constraints expressed over polynomial rings $R[X]$ and ideal membership in $R[X]$.

- **Lattice-based operations.** Constraints in, for example, cyclotomic rings $R = (\mathbb{Z}/n\mathbb{Z})[X]/(\Phi(X))$ can be enforced through ideal-membership predicates over $\mathbb{Q}[X]$ (or $\mathbb{F}_n[X]$ if n is a prime), where the ideal is $(\Phi(X))$.

  Compared to arithmetizations over the cyclotomic ring itself, this approach has the following advantages: 1) witnesses are typed in $\mathbb{Q}[X]$ (or $\mathbb{F}_n[X]$) rather than in the cyclotomic ring, and this allows, at the PCS level,to avoid the hurdles brought when working over rings with zero divisors (e.g. necessity of large exceptional sets). 2) parts of the witness can be used (perhaps after projecting them modulo some different integer) and used in additional constraints unrelated to lattice-based computations.

- **Hamming weight.** For any $\hat{u} \in \{0,1\}^{<w}[X]$, let $\mathsf{wt}(\hat{u})$ denote the number of nonzero coefficients of $\hat{u}$. Then, given $c \in \mathbb{Q}$ we have that $\mathsf{wt}(\hat{u}) = c \iff \hat{u} - c \in (X-1)$ in $\mathbb{Q}[X]$. This is because evaluation at $X = 1$ sums coefficients.

2.1.6 Universal Constraint Systems (UCS)

We now introduce the relation we use for our arithmetization. We call it the *Universal Constraint System* (UCS) relation $\mathsf{REL}_{\mathsf{UCS}}$. We present it in a wide general form. In practice, we envision practitioners using simpler subsets of it, e.g. the AIR-like version described in Section 8.1, called *Universal AIR with lookups (UAIR+)*.

A UCS instance is specified by an *index* $\mathtt{i}$ that fixes a witness-vector length m, a number of rows n (acting as witness vector lengths), a tuple of prime powers $\mathbf{q} = (q_1, \ldots, q_{|\mathbf{q}|})$, a bit-size bound B, degree bounds $\mathbf{d} = (d_0, \ldots, d_{|\mathbf{q}|})$, and a set of constraints $\mathcal{C}$ described below.

Witness The witness consists of $|\mathbf{q}| + 1$ vectors $\mathbf{f}_0, \ldots, \mathbf{f}_{|\mathbf{q}|}$, each of length m. The vectors are typed[5] as

$$\mathbf{f}_0 \in \left(\mathbb{Q}^{<d_0,B}[X]\right)^m, \qquad \mathbf{f}_i \in \left(\mathbb{F}_{q_i}^{<d_i}[X]\right)^m \text{ for all } i \in [|\mathbf{q}|],$$

where, recall, $R^{<d,B}[X]$ denotes the set of polynomials of degree $< d$ with coefficients in R, of bit-length less than B (when R is a field, we drop the bit bound and allow coefficients in all R). The entries of the witness vectors are thus *polynomials themselves*. In particular, the multilinear extension (MLE) (cf. Section 3.1) of $\mathbf{f}_i$ is a multilinear polynomial whose coefficients lie in $\mathbb{Q}[X]$ or $\mathbb{F}_{q_i}[X]$—that is, a polynomial whose coefficients are polynomials. Regarding why $\mathbf{f}_0$ is typed in $\mathbb{Q}^{<d_0,B}[X]$ rather than $\mathbb{Z}^{<d_0,B}[X]$, see Section 2.1.2 above.

See Remark 2.6 for a discussion of the bit and degree bounds, why they are important, and how they are proved.

Public input The public input is a vector $\mathbf{y} \in (\mathbb{Q}^{<d_0,B}[X])^\ell$.

[5] This typing is proved by the IOPP/PCS. For the PIOP side, we assume malicious provers respect this typing.

Constraints Each constraint in $\mathcal{C}$ is a tuple $(Q, \mathfrak{I})$, where Q is a polynomial with coefficients either in $\mathbb{Q}$ or $\mathbb{F}_{q_i}^{<d_i}[X]$ for some $i \in [|\mathbf{q}|]$, and $\mathfrak{I}$ is an ideal of $\mathbb{Q}[X]$ or $\mathbb{F}_{q_i}[X]$, respectively. In the first case we say $(\mathbb{Q}, \mathfrak{I})$ is a $\mathbb{Q}[X]$-*constraint*. Such constraint is satisfied if and only if

$$Q(k, \mathbf{y}, \mathbf{f}_0) \in \mathfrak{I} \qquad \text{for every } k \in [n], \text{ over } \mathbb{Q}[X]. \tag{14}$$

so, in particular, Q is a polynomial on enough variables to be evaluated $(k, \mathbf{y}, \mathbf{f}_0)$.

Example: For example, if using for R1CS-like type constraints, (14) could take the following form:

$$Q(k, \mathbf{y}, \mathbf{f}_0) = \left(\sum_{\mathbf{b}\in\{0,1\}^\mu} \tilde{A}(k, \mathbf{b}) \cdot \tilde{z}(\mathbf{b})\right) \cdot \left(\sum_{\mathbf{b}\in\{0,1\}^\mu} \tilde{B}(k, \mathbf{b}) \cdot \tilde{z}(\mathbf{b})\right) - \sum_{\mathbf{b}\in\{0,1\}^\mu} \tilde{C}(k, \mathbf{b}) \cdot \tilde{z}(\mathbf{b}) \in \mathfrak{I} \text{ over } \mathbb{Q}[X],$$

for all $k \in [n]$, where $\mathbf{z} = (\mathbf{y}, \mathbf{f}_0)$, A, B, C are R1CS matrices with integer entries, $\tilde{\cdot}$ denotes multilinear extension, and k above is interpreted as an element of a suitable hypercube.

In the later case, i.e. when the constraint domain is $\mathbb{F}_{q_i}[X]$, then we say $(Q, \mathfrak{I})$ is a $\mathbb{F}_{q_i}[X]$-*constraint*. It is satisfied if and only if

$$Q(k, \phi_{q_i}(\mathbf{y}), \phi_{q_i}(\mathbf{f}_0), \mathbf{f}_i) \in \mathfrak{I} \qquad \text{for every } k \in [n], \text{ over } \mathbb{F}_{q_i}[X], \tag{15}$$

where, recall, ϕ_{q_i} denotes (coefficient-wise) reduction modulo q_i (and if q_i is a power of a prime q_i', then $\phi_{q_i} = \phi'_{q_i}$). Hence, in this latter, case, the constraint is exactly the same as in (14) but over $\mathbb{F}_{q_i}[X]$, and with $\mathbf{f}_0$, $\mathbf{y}$ also being part of the constraint, modulo q_i (or q_i'). See Remark 2.5 for a short comment on having $\phi_{q_i}(\mathbf{f}_0)$, $\phi_{q_i}(\mathbf{y})$ be well-defined.

Overall, the relation $\mathsf{REL}_{\mathsf{UCS}}$ consists of all triples $(\mathtt{i}, \mathtt{x}; \mathtt{w})$ such that, for every constraint $(Q, \mathfrak{I}) \in \mathcal{C}$ and every row $k \in [n]$, the following holds:

$Q(k, \mathbf{y}, \mathbf{f}_0) \in \mathfrak{I}$ over $\mathbb{Q}[X]$	if $(Q, \mathfrak{I})$ is a $\mathbb{Q}[X]$-constraint,
$Q(k, \phi_{q_i}(\mathbf{y}), \phi_{q_i}(\mathbf{f}_0), \mathbf{f}_i) \in \mathfrak{I}$ over $\mathbb{F}_{q_i}[X]$	if $(Q, \mathfrak{I})$ is a $\mathbb{F}_{q_i}[X]$-constraint, $i \in [|\mathbf{q}|]$.

Remark 2.4 (Shared witness principle)**.** The witness $\mathbf{f}_0$ acts as a shared witness across different modular arithmetic. This allowed in Eq. (5) to have a single bit-polynomial column $\hat{u} \in \{0,1\}^{<w}[X] \subset \mathbb{Q}[X]$ simultaneously to participate in a squaring of integers modulo some integer, and in XOR operations expressed as additions modulo 2.

The formal definition of UCS is given in Definition 4.1. When restricted to a single finite field and the zero ideal, UCS subsumes standard constraint systems—R1CS, AIR, CCS, Plonkish, and lookup constraints—as special cases (cf. Sections 4.2 and 4.2.2).

After Definition 4.1 we give a detailed discussion of the design choices behind the definition of UCS, and further technical observations about the definition.

2.1.7 Miscellaneous remarks on the definition of UCS

We end the section with a series of comments on the definition of UCS.

Remark 2.5 (Well-defined projections of $\mathbf{f}_0$ and $\mathbf{y}$)**.** In the UCS definition, we also require that the reductions $\phi_{q_i}(\mathbf{f}_0)$, $\phi_{q_i}(\mathbf{y})$ are well-defined. Namely, we ask that $\mathbf{f}_0$ and $\mathbf{y}$ have entries in $\mathbb{Z}_{(q_i)}^{<d_0,B}[X]$, where, in short, $\mathbb{Z}_{(q_i)}$ is the subring of $\mathbb{Q}$ formed by rationals whose denominator is not divisible by q_i (see Section 3.1).

In practice, this requirement is usually subsumed by the lookup or typing constraints on $\mathbf{f}_0$ and $\mathbf{y}$. E.g., in our arithmetization of REL_f, the witness $\mathbf{f}_0$ contains bit-polynomial entries, which are polynomials with integer coefficients, and so belong to $\mathbb{Z}_{(q_i)}$ for any prime or prime power q_i.

Remark 2.6 (Bit bounds and degree bounds)**.** Observe that the entries of $\mathbf{f}_0, \mathbf{f}_i$ are required to have degree less than d_0, d_i and, in the first case, coefficients of bit-length less than B, where B, d_0, d_i are public parameters fixed by the index $\mathtt{i}$. These bounds are crucial to guarantee the soundness of our PIOP, as reflected in the evaluation bounds of Section 5.1 and the field-size hypotheses of Theorem 5.13. The bounds are enforced by our IOPP/PCS, and soundness of the PIOP is proved against attackers respecting these bounds. See Section 2.1.2 for more comments on what is delegated to the IOPP/PCS.

In fact, and this is technical, the bit-bound B is only proved by the IOPP in a soft manner, in the sense that the IOPP guarantees polynomially larger bit-bounds $B' = \mathsf{poly}(B)$, instead of strictly B, with $B' > B$. See Remark 2.17 for a discussion of how this is handled when compiling our PIOPs with our PCSs.

In another direction of comment: having publicly fixed bounds means in particular that our proof system allows to prove knowledge of a witness within publicly specified bit-size B and degree bounds d. In all practical applications we consider, these bounds arise naturally and can be computed by any party (including the verifier), for any witness an honest prover may use: e.g., $B = 64$ and $d = 32$ suffices for SHA-256. Exceptions may be found in computations with groups of unknown order or integer-based math problems. In any case, a prover can always publish bounds B, d satisfied by its witness, and prover and verifier can use these as parameters. When looking to add zero-knowledgeness to our system, this strategy would need to be revisited, since the bounds leak information about the witness. Note that our schemes are not zero-knowledge in any case. We leave zero-knowledgeness as future work.

Remark 2.7 (More generality)**.** The definition of UCS admits several natural straightforward more general variants. E.g. 1) one could split each witness vector $\mathbf{f}_i$ into several vectors, each typed in the same polynomial ring (one would do that to split the whole witness into chunks/columns, and commit to each column separately), 2) we could allow for different witness lengths among $\mathbf{f}_0, \ldots, \mathbf{f}_{|\mathbf{q}|}$, 3) each domain $\mathbb{Q}[X], \mathbb{F}_{q_i}[X]$ could have its own public input, rather than having one single public input $\mathbf{y}$ typed in $\mathbb{Q}[X]$, etc.

Remark 2.8 (Virtual columns)**.** The definition of UCS can be extended to include *virtual witness vectors*. These are vectors $\mathbf{u}$ that are not part of the witness vectors $\mathtt{w}$ but that can be derived from $\mathtt{w}$ through either 1) a linear transformation of the vectors in $\mathtt{w}$; 2) an R-module homomorphism $T : R^{<d}[X] \to R^{<d}[X]$ as in Lemma 2.3 (e.g. entry-wise bit-rotation or entry-wise bit-shift of a vector with entries in $R^{<d}[X]$). In practice, virtual witnesses are not committed, and the verifier gains access to their MLE's in a "virtual" manner through its access to the MLE's of the committed witness vectors.

Them being a standard tool, we do not include virtual witnesses in our formalization for the sake of brevity, but we do use them in our implementation and arithmetizations from Section 8.

2.2 Zinc+: building PIOPs for UCS from PIOPs over finite fields

We now describe how the Zinc+ compiler turns PIOPs over finite fields into a PIOP for UCS. For clarity, we present a simplified version restricted to UCS instances with $R[X]$-constraints for a single $R \in \{\mathbb{Q}, \mathbb{F}_q\}$ (q prime) and a single constraint ideal of degree ≥ 1 (the most interesting case; the zero ideal reduces to a standard equality check). The general case, which handles $\mathbb{Q}[X]$-constraints

and multiple $\mathbb{F}_{q_i}[X]$-constraint rings (for prime-power q_i), each presenting constraints with respect to multiple ideals (zero and proper), is treated in Section 5.

The compiler first has the prover send an oracle to the multilinear extension of the witness. Three reduction steps then transform the $R[X]$-constraint into a finite-field constraint: (1) *ring projection* projects constraints to ideal claims over a prime-power field $\mathbb{F}_{\tilde{q}}[X]$; (2) *ideal batching* collapses the ideal-membership checks across all rows into a single strict polynomial identity over $\mathbb{F}_{\tilde{q}}[X]$; (3) *evaluation projection* evaluates this identity at a random field element $a \in \mathbb{F}_{\tilde{q}}$, yielding a strict equality over $\mathbb{F}_{\tilde{q}}$. Steps 1–3 constitute a *polynomial interactive oracle reduction* (PIOR) from UCS to finite-field satisfiability; a standard finite-field PIOP then proves the resulting constraint, with oracle queries to the $R[X]$-witness handled by the functionality of the oracles (enabled later by our IOPP), cf. Remark 2.9.

The Zinc+ add-on. When $R = \mathbb{F}_q$ (no $\mathbb{Q}[X]$-constraints), the compiler simplifies considerably: Step 1 reduces to the canonical embedding $\iota_{\tilde{q}}$ (or is trivial when q is already large), and no prime projection or well-definedness concerns arise. We call this lightweight instantiation the *Zinc+ add-on*. It extends any existing hash-based SNARK over $\mathbb{F}_q$ with the ability to prove ideal membership constraints over $(\mathbb{F}_q^{<d}[X])$, requiring only Steps 2 and 3 at the PIOP level and an interleaved commitment at the PCS level, with no modifications to existing components. The reader interested primarily in the add-on may follow the $R = \mathbb{F}_q$ case through each step below.

2.2.1 Projections from polynomial rings to finite fields

Since $\mathbb{Q}$ and $\mathbb{F}_q$ are fields, the polynomial rings $\mathbb{Q}[X]$ and $\mathbb{F}_q[X]$ are principal ideal domains: every ideal is principal, and every nonzero ideal is generated by a unique monic polynomial (cf. Section 3.1).

For a prime q_0, the *prime projection*

$$\phi_{q_0} : \mathbb{Z}_{(q_0)}[X] \to \mathbb{F}_{q_0}[X], \qquad f \mapsto f \bmod q_0,$$

reduces each coefficient modulo q_0, where $a/b \in \mathbb{Z}_{(q_0)}$ maps to $a \cdot b^{-1} \bmod q_0$. For any field $\mathbb{F}$ and $a \in \mathbb{F}$, the *evaluation map*

$$\psi_a : \mathbb{F}[X] \to \mathbb{F}, \qquad f \mapsto f(a),$$

evaluates a polynomial at a. Both are surjective ring homomorphisms. When $R = \mathbb{Q}$, their composition $\psi_{q_0,a} := \psi_a \circ \phi_{q_0} : \mathbb{Z}_{(q_0)}[X] \to \mathbb{F}_{q_0}$ maps $f \mapsto \phi_{q_0}(f)(a)$.

2.2.2 UCS constraints and sending oracles

An $R[X]$-constraint in UCS takes the form

$$Q(k, \mathbf{y}, \mathbf{f}) \in (g), \qquad \text{for all rows } k \in [n], \qquad \text{over } R[X], \tag{16}$$

where $g \in R[X]$ is a monic polynomial of degree ≥ 1 generating the constraint ideal, Q is a polynomial with coefficients in R on the row variable k, $\mathbf{y} \in (R^{<d,B}[X])^{\ell}$ is a public input, and $\mathbf{f} \in (R^{<d,B}[X])^{m}$ is a witness. We identify $[n]$ with $\{0,1\}^{\mu}$ where $n = 2^{\mu}$.

The prover sends an oracle $[[\tilde{\mathbf{f}}]]$ to the multilinear extension of $\mathbf{f}$.

Remark 2.9 (Correct typing of $\mathbf{f}$ and evaluation functionality)**.** We assume the entries of $\mathbf{f}$ belong to $(R^{<d,B}[X])$: polynomials of degree $< d$ in X with R-coefficients (and, when $R = \mathbb{Q}$, of bit-size $< B$).

We further assume the oracle $[[\tilde{\mathbf{f}}]]$ exposes the following functionality: let ψ be a projection homomorphism $\psi : R[X] \to \mathbb{F}'$ as in Section 2.1.1, for $R \in \{\mathbb{Q}, \mathbb{F}\}$ and $\mathbb{F}'$ a finite field. Then $[[\tilde{\mathbf{f}}]]$ allows the verifier to query evaluations of $\psi(\tilde{\mathbf{f}})$ on any chosen point $\boldsymbol{\beta}$ with entries in $\mathbb{F}'$. If $R = \mathbb{Q}$ and $\psi(\tilde{\mathbf{f}})$ is not well defined, then the oracle returns $\perp$.

Definition 5.3 formalizes this oracle type. All the above functionality is free in the PIOP soundness model; enforcing it is delegated to the PCS or IOPP. See Remarks 2.6 and 2.17 for details.

2.2.3 Step 1: Projecting to $\mathbb{F}_{\tilde{q}}[X]$

This step projects or embeds the $R[X]$-constraints into a polynomial ring $\mathbb{F}_{\tilde{q}}[X]$ of sufficient size for the soundness of future steps, and also for the soundness of this projection itself in the $R = \mathbb{Q}$ case below.

$\underline{R = \mathbb{Q}.}$ The verifier samples a random $\Omega(\lambda)$-bit prime q_0 and sets $\tilde{q} = q_0$. Both parties replace the constraint (16) by

$$\phi_{q_0}(Q)(\mathbf{b}, \phi_{q_0}(\mathbf{y}), \phi_{q_0}(\mathbf{f})) \in (\phi_{q_0}(g)) \qquad \forall\, \mathbf{b} \in \{0,1\}^{\mu}$$

over $\mathbb{F}_{q_0}[X]$. Since g is monic, $\phi_{q_0}(g)$ is well-defined and has the same degree as g, so the projected ideal $(\phi_{q_0}(g))$ is non-trivial. The projection preserves ideal membership: if $f = g \cdot h$ in $\mathbb{Q}[X]$ and $f, g \in \mathbb{Z}_{(q_0)}[X]$, then $h \in \mathbb{Z}_{(q_0)}[X]$ (since g is monic), so $\phi_{q_0}(f) = \phi_{q_0}(g) \cdot \phi_{q_0}(h) \in (\phi_{q_0}(g))$. The converse can fail, but only with negligible probability over a random $\Omega(\lambda)$-bit prime q_0 (see the soundness sketch below).

The condition $f, g \in \mathbb{Z}_{(q_0)}[X]$ (i.e., q_0 does not divide any denominator) is a completeness concern. If q_0 divides a denominator in the coefficients of $\mathbf{y}$, Q, or g, the verifier detects this and rejects. If q_0 divides a denominator in $\mathbf{f}$, the honest prover cannot compute the projected witness; this occurs with negligible probability since all coefficients have bounded bit-size (see the completeness analysis in Section 5.3).

$\underline{R = \mathbb{F}_q.}$ We set $\tilde{q} = q^{\ell}$ for ℓ chosen so that $|\mathbb{F}_{\tilde{q}}| = \Omega(2^{\lambda})$ (if q is already large enough, $\ell = 1$ and this step is trivial). Both parties apply the canonical embedding $\iota_{\tilde{q}} : \mathbb{F}_q \hookrightarrow \mathbb{F}_{\tilde{q}}$ coefficient-wise to bring the constraints from $\mathbb{F}_q[X]$ into $\mathbb{F}_{\tilde{q}}[X]$. Since $\iota_{\tilde{q}}$ is injective, the embedded ideal remains nonzero and of the same degree. We identify $\mathbb{F}_q[X]$ with its image in $\mathbb{F}_{\tilde{q}}[X]$ and continue to write Q, $\mathbf{y}$, $\mathbf{f}$, g for the same objects viewed in the larger ring.

After this step, all constraints are over $\mathbb{F}_{\tilde{q}}[X]$. When $R = \mathbb{F}_q$ with q small, the extension to $\mathbb{F}_{\tilde{q}}$ provides a field large enough for the Schwartz–Zippel bounds in Steps 2 and 3 to be negligible. When $R = \mathbb{Q}$, the prime projection is applied *before* the ideal batching (Step 2), so that the MLE evaluation in Step 2 operates over $\mathbb{F}_{\tilde{q}}$ rather than $\mathbb{Q}$, avoiding the large bit-sizes that would arise from computing $\widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r})$ over $\mathbb{Q}$.

Soundness sketch ($R = \mathbb{Q}$). If $Q(\mathbf{b}, \mathbf{y}, \mathbf{f}) \notin (g)$ for some $\mathbf{b}$, we claim $\phi_{q_0}(Q(\mathbf{b}, \mathbf{y}, \mathbf{f})) \notin (\phi_{q_0}(g))$ with high probability. Write $Q(\mathbf{b}, \mathbf{y}, \mathbf{f}) = g \cdot h + r$ via Euclidean division in $\mathbb{Q}[X]$, with $\deg r < \deg g$ and $r \neq 0$. Since g is monic, $r \in \mathbb{Z}_{(q_0)}[X]$ for all but negligibly many q_0. After projection, membership in $(\phi_{q_0}(g))$ requires $\phi_{q_0}(r) = 0$; since r has bounded coefficients, only polynomially many primes can cause this (see Lemma A.2).

2.2.4 Step 2: Ideal batching to strict equality constraints

After Step 1, all constraints are over $\mathbb{F}_{\tilde{q}}[X]$ (when $R = \mathbb{Q}$, Q, $\mathbf{y}$, $\mathbf{f}$, g below denote $\phi_{q_0}(Q)$, $\phi_{q_0}(\mathbf{y})$, $\phi_{q_0}(\mathbf{f})$, $\phi_{q_0}(g)$). The verifier samples $\mathbf{r} \leftarrow (\mathbb{F}_{\tilde{q}})^{\mu}$. The prover sends $e \in \mathbb{F}_{\tilde{q}}[X]$, supposedly the

evaluation

$$e = \sum_{\mathbf{b}\in\{0,1\}^\mu} Q(\mathbf{b}, \mathbf{y}, \mathbf{f}) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}; \mathbf{r}) \quad \in \mathbb{F}_{\tilde{q}}[X].$$

The verifier checks $e \in (g)$ over $\mathbb{F}_{\tilde{q}}[X]$. The constraint is then replaced by the strict equality

$$\sum_{\mathbf{b}\in\{0,1\}^\mu} Q(\mathbf{b}, \mathbf{Y}, \mathbf{Z}) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}; \mathbf{r}) - e = 0 \qquad \text{over } \mathbb{F}_{\tilde{q}}[X]. \tag{17}$$

This step generalizes the first step of multilinear sumcheck-based PIOPs. Just as one checks $\widetilde{v}(\mathbf{r}) = 0$ (membership in the zero ideal) to test whether all entries of a vector $v \in (\mathbb{F}_{\tilde{q}}[X])^n$ are zero, here one checks $\widetilde{v}(\mathbf{r}) \in (g)$ to test ideal membership.

Lemma 2.10 (Ideal membership via MLE evaluation; informal version of Lemma 5.1)**.** *Let $\mathbb{K}$ be a field, $g \in \mathbb{K}[X]$ a polynomial with $\deg g \geq 1$, and $v \in (\mathbb{K}[X])^n$ with $n = 2^\mu$. If some entry of v does not belong to (g), then $\Pr_{\mathbf{r}\leftarrow S^\mu}[\widetilde{v}(\mathbf{r}) \in (g)] \leq \mu/|S|$ for any finite $S \subseteq \mathbb{K}$.*

The proof projects onto $\mathbb{K}[X]/(g)$ and applies the Schwartz–Zippel lemma; see Lemma 5.1 for details.

2.2.5 Step 3: Evaluation projection to $\mathbb{F}_q$

The verifier samples $a \leftarrow \mathbb{F}_{\tilde{q}}$ and evaluates the polynomial identity (17) at $X = a$, replacing it with

$$\sum_{\mathbf{b}\in\{0,1\}^\mu} \psi_a(Q)(\mathbf{b},\, \psi_a(\mathbf{y}),\, \psi_a(\mathbf{f})) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}; \mathbf{r}) - \psi_a(e) = 0 \qquad \text{over } \mathbb{F}_{\tilde{q}}. \tag{18}$$

Soundness sketch. Let d_Q denote the total degree of Q in its algebraic variables $(\mathbf{Y}, \mathbf{Z})$. The left-hand side of (17), evaluated on the witness $(\mathbf{y}, \mathbf{f})$, is a polynomial in $\mathbb{F}_{\tilde{q}}[X]$ of degree at most $d \cdot d_Q$ (since each entry of $\mathbf{y}$ and $\mathbf{f}$ has degree $< d$ in X). If this polynomial is nonzero, evaluating at a random $a \in \mathbb{F}_{\tilde{q}}$ gives zero with probability at most $d \cdot d_Q/|\mathbb{F}_{\tilde{q}}|$, which is negligible for $|\mathbb{F}_{\tilde{q}}| = \Omega(2^\lambda)$.

2.2.6 Running a finite-field PIOP with projected oracles

The PIOR (Steps 1-3) outputs $\psi_a(\mathbf{y}) \in (\mathbb{F}_{\tilde{q}})^\ell$ as the new public input (when $R = \mathbb{Q}$, this means $\psi_{q_0,a}(\mathbf{y})$) and constraint (18) as a standard algebraic equality over $\mathbb{F}_{\tilde{q}}$. Any finite-field PIOP $\Pi_{\mathbb{F}_{\tilde{q}}}$ for the corresponding constraint type (e.g., R1CS, CCS, or AIR with lookups) can prove it.

Although $\Pi_{\mathbb{F}_{\tilde{q}}}$ expects $\mathbb{F}_{\tilde{q}}$-polynomial oracles, the prover's projected oracle $[[\tilde{\mathbf{f}}]]$ already exposes the right interface: a query at $\boldsymbol{\beta} \in \mathbb{F}_{\tilde{q}}^\nu$ returns $\widetilde{\psi_a(\mathbf{f})}(\boldsymbol{\beta}) \in \mathbb{F}_{\tilde{q}}$, the multilinear extension of the projected vector $\psi_a(\mathbf{f})$ evaluated at $\boldsymbol{\beta}$. The Zinc+ verifier therefore connects $\Pi_{\mathbb{F}_{\tilde{q}}}$ directly to $[[\tilde{\mathbf{f}}]]$ and forwards each response unchanged.

When $R = \mathbb{Q}$, however, ψ_a is only defined on $\mathbb{Z}_{(p)}[X]$, so $\psi_a(\mathbf{f})$ is undefined when some entry of $\mathbf{f}$ escapes $\mathbb{Z}_{(p)}[X]$, and $\widetilde{\psi_a(\mathbf{f})}$ is then just a formal shorthand rather than a well-defined polynomial. The oracle's contract is therefore: it returns $\widetilde{\psi_a(\mathbf{f})}(\boldsymbol{\beta})$ when the partial projection happens to be defined at the query, and $\perp$ otherwise. The Zinc+ verifier rejects on $\perp$. By Theorem 5.5, this rejection rule detects $\mathbb{Q}[X]$ well-definedness violations with overwhelming probability when queries are uniform.

We compile our PIOP to an IOP, instantiating our polynomial oracles with IOPPs for multilinear polynomial evaluation over projections of $R[X]$. When $R = \mathbb{F}_q$, an IOPP for evaluation of multilinear polynomials over $\mathbb{F}_q[X]$ can be constructed from any IOPP over $\mathbb{F}_q$: writing $\tilde{\mathbf{f}} = \sum_{i=0}^{d-1} \tilde{\mathbf{f}}_i \cdot X^i$

for multilinear $\tilde{\mathbf{f}}_i$ over $\mathbb{F}_q$, the evaluation claim reduces to d claims over $\mathbb{F}_q$ that are batch-verified by the inner IOPP (see Section 6). When $R = \mathbb{Q}$, we use our dedicated IOPP Zip+, built from IPRS codes (Section 2.3).

2.2.7 Protocol summary

The three-step PIOR reduces $R[X]$-constraints to $\mathbb{F}_{\tilde{q}}$-constraints, after which a finite-field PIOP with projected oracles proves the resulting constraint. In the diagram below, the top row shows the constraint ring and the bottom row shows the constraint type at each stage.

$$\underset{\text{ideal}}{R[X]} \xrightarrow[\phi_{q_0} \text{ or } \iota_{\tilde{q}}]{\text{ring proj.}} \underset{\text{ideal}}{\mathbb{F}_{\tilde{q}}[X]} \xrightarrow[\text{MLE at } \mathbf{r}]{\text{ideal batch}} \underset{\text{equality}}{\mathbb{F}_{\tilde{q}}[X]} \xrightarrow[\psi_a]{\text{eval. proj.}} \underset{\text{equality}}{\mathbb{F}_{\tilde{q}}} \xrightarrow[\text{projected oracles}]{\text{PIOP}} \text{Accept/Reject}$$

When $R = \mathbb{F}_q$ (the Zinc+ add-on), the first arrow simplifies to $\iota_{\tilde{q}}$ (or the identity when q is already large), the prime projection and well-definedness machinery are absent, and the IOPP reduces to batch evaluation over $\mathbb{F}_q$. The full version, handling $\mathbb{Q}[X]$-constraints and multiple $\mathbb{F}_{q_i}[X]$-constraint rings (for prime-power q_i, with multiple ideals batched separately), is given in Algorithm 1 (Section 5.3).

2.3 Integer Pseudo Reed Solomon (IPRS) codes

We introduce a new family of integer (or rational) codes, which we call *Integer Pseudo Reed–Solomon* (IPRS) codes, and which have optimal distance, i.e. they are Maximum Distance Separable (MDS). Additionally, IPRS have a practically efficient FFT-like encoding algorithm, and their codewords do not suffer from large bit-size entry growth (something that can be problematic when working with integers or rationals, as we discuss below). We believe that IPRS codes are of independent interest, and may find applications beyond the context of this work.

When witness vectors with entries in $\mathbb{Q}$ or $\mathbb{Q}^{<d,B}[X]$ are present, our schemes use an IOPP for linear codes over $\mathbb{Q}$. In this scenario, as with other code-based SNARKs over finite fields, the efficiency of our final compiled SNARK is highly dependent on the properties of this linear code. Concretely, prover efficiency is heavily affected by encoding time, and verifier speed and proof size are affected by the code's minimum relative distance (or by its Mutual Correlated Agreement error). In the context of SNARKs working over small fields, Reed–Solomon (RS) codes are a popular choice for the IOPP due to their optimal distance and efficient encoding via the FFT algorithm. When working over integers, rationals, or larger finite fields, the options are more limited.

For example, Zinc [Gar+25; GWHD25] used a lifted rational version of Juxtaposed Expand-Accumulate (JEA) codes over a finite field. While enjoying efficient encoding, the proved minimal distance of these codes (both for their finite field and integer/rational versions) is significantly lower than that of RS codes [BFKTWZ24]. RAA codes [BCFRRZ25] are also an intriguing family of codes that can be easily lifted to $\mathbb{Z}$ or $\mathbb{Q}$, and with similar encoding-distance profile as JEA codes. It is possible that both JEA and RAA have better minimum distance than what is currently proved, especially over large prime fields or over $\mathbb{Q}$, but this remains open.

Below, given a vector $x \in \mathbb{Q}^k$, we denote by $\|x\|_\infty$ the *norm* of x, i.e. $\|x\|_\infty = \max_{i\in[k]} |x_i|$. The notation extends to sets as well.

Remark 2.11 (Rational code, integer messages)**.** Since a linear code is a linear subspace of a vector space, we define our codes over $\mathbb{Q}$, and not $\mathbb{Z}$. However, all codes discussed below admit an integer generator matrix, and so when encoding integer messages, the resulting codeword is integral as well. In most of our envisioned applications, honest provers only encode integer messages. In this case, the rational formulation is needed only for the theoretical framework (e.g., our PIOP+IOPP results, cf. Sections 2.2 and 2.4).

2.3.1 Two initial attempts

We start by describing two methods by which one could "port" RS codes from finite fields to $\mathbb{Q}$.

Naive RS codes over $\mathbb{Q}$ and their codeword bit-size blowout. When looking for codes over $\mathbb{Q}$ or $\mathbb{Z}$, one must account for a *fundamental challenge* that does not exist over small finite fields: the code should not create large blowouts in the norm of codewords relative to the norm of the encoded message. I.e. the ratio

$$\|\mathsf{Enc}(x)\|_\infty / \|x\|_\infty$$

should be small for all or almost all messages x, where Enc is an encoding function of the code. Indeed, large ratios $\|\mathsf{Enc}(x)\|_\infty / \|x\|_\infty$ would lead to inefficient encoding, proof sizes, and verification. The reason this is a potential problem is that, of course, $\mathbb{Q}$ and $\mathbb{Z}$ do not enjoy modular reduction, and thus, as we see next, the relative norm of a codeword can grow exponentially in the code dimension if the code is not designed carefully.

For example, consider a naive RS code over $\mathbb{Q}$, where codewords are defined simply as vectors of evaluations of polynomials from $\mathbb{Q}^{<k}[Y]$ over a fixed subset S of $\mathbb{Z}$, i.e. say we define the code as

$$\mathsf{RS}_{\mathsf{naive}}[S,k] = \{(f(\alpha))_{\alpha\in S} \mid f \in \mathbb{Q}^{<k}[Y]\}. \tag{19}$$

Even assuming that encoding can be computed efficiently for $\mathsf{RS}_{\mathsf{naive}}[S,k]$, the norm $\|\mathsf{Enc}(x)\|_\infty$ is in general on the order of $\|x\|_\infty \cdot \|S\|_\infty^{k-1}$. For practical degrees k, this can lead to codewords having entries with hundreds of thousands or even millions of bits.

Naive lifts of RS codes over $\mathbb{Q}$. In search of alternatives, we note the following. Let q be a prime, let $\mathcal{C}_q \subseteq \mathbb{F}_q^n$ be a linear code of dimension k over $\mathbb{F}_q$, and let M_q be a generator matrix of $\mathcal{C}_q$. Consider the linear subspace of $\mathbb{Q}^n$ defined as

$$\mathcal{C}_{\mathbb{Q}}[\hat{M}_q] = \left\{ \hat{M}_q \cdot x \;\middle|\; x \in \mathbb{Q}^k \right\}, \tag{20}$$

where $\hat{M}_q$ is the integer matrix obtained by lifting each entry of M_q to an integer representative (not necessarily in $[0\,..\,q-1]$ or in $\{-(q-1)/2,\ldots,(q-1)/2\}$). We call $\mathcal{C}_{\mathbb{Q}}[\hat{M}_q]$ a *lift* of $\mathcal{C}_q$.

Lemma 2.12 (Lifts of linear codes preserve dimension and distance)**.** *Under the notation above, $\mathcal{C}_{\mathbb{Q}}[\hat{M}_q] \subseteq \mathbb{Q}^n$ is a linear code over $\mathbb{Q}$ (i.e. a linear subspace) of dimension k and minimum distance at least the minimum distance of $\mathcal{C}_q$.*

We defer the proof to Section 7.2. The main idea is that it suffices to study the Hamming weight of nonzero *integer codewords* that are not a multiple of q (by clearing denominators and factoring out powers of q). In that case, the result follows because the Hamming weight can only decrease when reducing modulo q, and the reduction of such a nonzero codeword remains nonzero because it is not a multiple of q and due to the full rank of M_q.

In view of Lemma 2.12, one may try to lift any code over $\mathbb{F}_q$ to $\mathbb{Q}$. In particular, we could lift a standard finite field RS code

$$\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, k] = \left\{ f(\alpha)_{\alpha\in\mathcal{L}} \;\middle|\; f \in \mathbb{F}_q^{<k}[Y] \right\}$$

in this manner, using as generator matrix a Vandermonde matrix V_q with entries reduced modulo q. Such a matrix has entries bounded by q, and thus the norm of the encoding of a message $x \in \mathbb{Q}^k$ is at most $\|x\|_\infty \cdot q \cdot k$.

However, the code $\mathcal{C}_{\mathbb{Q}}[V_q]$ does not have, a priori, efficient encoding. The FFT algorithm one would use in $\mathbb{F}_q$ to multiply V_q by a vector is no longer available when this operation is performed over the integers, where modular reduction cannot be applied. Hence, a priori, encoding time for $\mathcal{C}_{\mathbb{Q}}[V_q]$ reaches a prohibitive $O(nk)$.

2.3.2 A middle ground: IPRS codes

We next describe the IPRS construction. Prior to that, we briefly recall how the FFT encoding algorithm for RS codes over finite fields works. Among others, the algorithm is parameterized by a choice of radix. Throughout this section we fix the radix to be 2 for simplicity. The definitions and ideas apply to general radices as well in a straightforward manner. The full IPRS encoder for arbitrary radices is given in Section 7.

FFT algorithm over $\mathbb{F}_q$, radix 2. Let $\mathbb{F}_q$ be a prime field, let n be a power of two dividing $q-1$ and let $\omega \in \mathbb{F}_q$ be a primitive n-th root of unity. Consider the RS code $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, k]$ with evaluation domain $\mathcal{L} = (1, \omega, \ldots, \omega^{n-1})$. The radix-2 FFT algorithm takes as input a message $x \in \mathbb{F}_q^k$ and outputs the codeword $(f(\omega^j))_{j=0}^{n-1}$, where $f(X) = \sum_{i=0}^{k-1} x_i \cdot X^i$. The algorithm first writes

$$f(\omega^j) = f_0(\omega^{2j}) + \omega^j \cdot f_1(\omega^{2j}), \tag{21}$$

where $f_0(X)$ and $f_1(X)$ are polynomials of degree less than $k/2$, obtained by splitting f into its even- and odd-indexed coefficients, respectively. Since $\mathcal{L}^2 = \{1, \omega^2, \omega^4, \ldots\}$ is a subgroup of $\mathbb{F}_q$ of order $n/2$, the evaluations of f_0 and f_1 on $\mathcal{L}^2$, which are codewords from $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}^2, k/2]$, can be computed recursively with the FFT algorithm. Once this is done, the full codeword $(f(\omega^j))_{j=0}^{n-1}$ is assembled using (21).

The recursion stops at a base case, i.e. when the dimension k' of the current RS code is below certain threshold. At this point, one computes the vector of evaluations of the corresponding polynomial f' by naively multiplying a Vandermonde matrix by the vector of coefficients of f'.

We call the number of recursive levels the *depth* of the algorithm. The elements ω^j are called *twiddle factors*. We denote the resulting RS encoder by $\mathsf{Enc}_{\mathsf{RS}} : \mathbb{F}_q^k \to \mathbb{F}_q^n$.

IPRS codes. We define IPRS codes by "lifting" the FFT algorithm from $\mathbb{F}_q$ to $\mathbb{Z}$ as follows. Let $x \in \mathbb{Q}^k$ be the message to be encoded. We identify elements of $\mathbb{F}_q$ with their *centered* integer representatives via $\iota : \mathbb{F}_q \to \mathbb{Z}$, sending each element to the unique integer in $\{-(q-1)/2, \ldots, (q-1)/2\}$ congruent to it modulo q. We then execute the same radix-2 FFT algorithm as $\mathsf{Enc}_{\mathsf{RS}}$, but with every twiddle factor ω^j replaced by $\iota(\omega^j) \in \mathbb{Z}$, every base-case Vandermonde entry b replaced by $\iota(b)$, and the operations from (21) (as well as the base-case Vandermonde matrix-vector multiplication) are performed over $\mathbb{Q}$ (or $\mathbb{Z}$ when encoding integer messages) with *no modular reduction*.

Denote the resulting encoder by $\mathsf{Enc}_{\mathsf{IPRS}} : \mathbb{Q}^k \to \mathbb{Q}^n$. Then $\mathsf{Enc}_{\mathsf{IPRS}}$ has the same algorithmic structure as $\mathsf{Enc}_{\mathsf{RS}}$ but a completely different algebraic significance: $\mathsf{Enc}_{\mathsf{IPRS}}$ no longer computes evaluations of a polynomial $f(X) = \sum_{i=0}^{k-1} x_i X^i$ over some set. The output of $\mathsf{Enc}_{\mathsf{IPRS}}$ is a vector whose entries are, in general, unrelated to integer or rational polynomial evaluations over $\mathbb{Q}$.

Definition 2.13 (Integer Pseudo Reed–Solomon (IPRS) codes)**.** Given a prime q, a field $\mathbb{F}_q$, a multiplicative subgroup $\mathcal{L} = (1, \omega, \ldots, \omega^{n-1})$ of $\mathbb{F}_q$, a dimension $k < n$, and an FFT encoding algorithm for $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, k]$, the *IPRS code* is the image of the encoder $\mathsf{Enc}_{\mathsf{IPRS}}$ defined above:

$$\mathsf{IPRS}[\mathbb{F}_q, \mathcal{L}, k] \;:=\; \left\{\mathsf{Enc}_{\mathsf{IPRS}}(x) \mid x \in \mathbb{Q}^k\right\} \;\subseteq\; \mathbb{Q}^n. \tag{22}$$

We call $\mathbb{F}_q$ the *base field* of the code.

Theorem 2.14 (Properties of IPRS codes)**.** *The code* $\mathsf{IPRS}[\mathbb{F}, \mathcal{L}, k]$ *is a linear code over* $\mathbb{Q}$ *of dimension* k*, blocklength* n*, and minimum relative distance* $\delta = 1 - k/n + 1/n$*. Moreover, for each* $x \in \mathbb{Q}^k$*, the following norm bound holds*[6]*:*

$$\|\mathsf{Enc}_{\mathsf{IPRS}}(x)\|_\infty \leq \|x\|_\infty \cdot (q/2)^{\mathsf{depth}+1} \cdot k.$$

In our target configurations (cf. Section 2.5), where $k \approx 2^9, 2^{10}$, we use $q \approx 2^{16}$, depth 1–2, and radix 8, we obtain a relative norm increase of roughly 40–60 bits. This is comparable to what field-native schemes achieve over 64-bit fields when encoding messages of small norm, and it is far below the thousands of bits that naive integer RS codes would produce.

We defer the proof of Theorem 2.14 to Section 7.2, where we also state and prove a general-radix version (Theorem 7.3). At a high level, the proof follows by showing that $\mathsf{IPRS}[\mathbb{F}, \mathcal{L}, k]$ is a lift of the RS code $\mathsf{RS}[\mathbb{F}, \mathcal{L}, k]$, in the sense of Lemma 2.12, for some specific generator matrix M_q of $\mathsf{RS}[\mathbb{F}, \mathcal{L}, k]$ (different from the Vandermonde matrix). Then the result follows from Lemma 2.12.

The table below compares IPRS codes with the two naive alternatives described earlier:

Code construction	Relative norm growth	FFT structure
Integer RS code: $\mathsf{RS}_{\mathsf{naive}}[S, k]$, (19)	$\leq k \cdot \|S\|_\infty^k$	?
Lifted Vandermonde matrix: $\mathcal{C}_{\mathbb{Q}}[\hat{V}_q]$, (20)	$\leq k \cdot q$	No
$\mathsf{IPRS}[\mathbb{F}_q, \mathcal{L}, k]$, (22)	$\leq (q/2)^{\mathsf{depth}+1} \cdot k$	Yes

Table 3: Comparison of integer RS-like code constructions. Relative norm growth measures the maximum ratio $\|\mathsf{Enc}(x)\|_\infty / \|x\|_\infty$, where x is a message and $\mathsf{Enc}(x)$ is its encoding. The dimension of the code is k, and depth denotes the number of recursive FFT levels used in the encoder defining $\mathsf{IPRS}[\mathbb{F}_q, \mathcal{L}, k]$. The norm growth in IPRS codes is most of the time smaller than the displayed bound due to using centered representatives. Regarding FFT structure for naive integer RS codes, one could consider additive FFTs over arithmetic integer progressions, but the norm blowout issue persists.

2.3.3 Performance

Table 2 records the performance of IPRS codes in our PoC implementation. Further optimizations may apply. The IPRS codes use a radix-8 FFT algorithm, with rate 1/2 and depths ranging from 1 to 4. The base fields range from the field with $2^{16} + 1$ elements to the field with $5 \cdot 2^{25} + 1$ elements.

We often use IPRS codes to encode messages whose entries are not just rationals or integers, but polynomials of up to a certain degree $< w$, especially polynomials with binary coefficients. As explained in Sections 2.4 and 6, this latter encoding is done in an interleaved manner, i.e. if the message v belongs to, say, $\mathbb{Q}^{<d_0,B}[X]^k$, then we write $v = \sum_{i=0}^{w-1} v_i \cdot X^i$ with $v_i \in \mathbb{Q}^k$, and the encoding of v is the vector from $\mathbb{Q}^{<d_0,B}[X]^n$ obtained as $\mathsf{Enc}_{\mathsf{IPRS}}(v) = \sum_{i=0}^{w-1} \mathsf{Enc}_{\mathsf{IPRS}}(v_i) \cdot X^i$.

In our implementation, for the specific case where all entries of v belong to $\{0, 1\}^{<w}[X]$ for some $w < 64$, we represent each entry of v as a u64 type and use SIMD optimizations to compute $\mathsf{Enc}_{\mathsf{IPRS}}(v)$.

In Table 2 we report results for relatively small message lengths (compared to recent literature trends) because Zinc+ works naturally with small arithmetization sizes, i.e. with small message lengths (as demonstrated in our SHA-256 arithmetization and other examples).

Benchmarks are available at `https://github.com/NethermindEth/zinc-plus`.

[6] In practice, the norm growth is smaller than the above bound, due to the usage of centered integer representatives.

2.3.4 Open questions

In Section 7.3 we list some open questions about IPRS codes. We highlight the first one, which asks whether IPRS codes have the Mutual Correlated Agreement (MCA) property up to the Johnson bound, as Reed-Solomon codes do [Hab25; BCGM25]. A positive answer would directly improve proof sizes in our proof system.

2.4 IOPP's for interleaved codes with Projected Multilinear Evaluation constraints

In our PIOPs for the UCS relation, the prover sends oracles with entries in $\mathbb{K}^{<d}[X]$, where $\mathbb{K}$ is either a finite field $\mathbb{F}$, or the field of rationls $\mathbb{Q}$. As explained in Section 2.2, the PCS must enable proving MLE evaluation claims over a finite field $\mathbb{K}'$ obtained from projecting $\mathbb{K}^{<d}[X]$ onto $\mathbb{K}'$ via one of the maps ψ from Section 2.2.1. Further, following Remarks 2.6 and 2.9 when $\mathbb{K} = \mathbb{Q}$, the PCS must guarantee that the entries of the witness have bounded bit-size B and that they are well-defined under the projection ψ. Concretely, our IOPP instantiates the projected polynomial oracles described in Section 2.2. We formalize this setting next. In what follows, when $\mathbb{K} = \mathbb{Q}$, we understand $\mathbb{K}^{<d}[X]$ to mean $\mathbb{Q}^{<d,B}[X]$.

To this end, we design an IOPP that can test the following claim. Let $W \in (\mathbb{K}^{<d}[X])^t$ be the witness and let k_1 and k_2 be powers of 2 with $k_2 \cdot k_1 = t$. We organize W as a $k_2 \times k_1$ matrix W with entries in $\mathbb{K}^{<d}[X]$. Let $M \in \mathbb{K}^{k_1 \times n}$ be a generator matrix of a linear code over $\mathbb{K}$, and let

$$V \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times n}$$

be a purported interleaved encoding of W under M. THe verifier receives oracle access to V. Let $z \in (\mathbb{K}')^{\mu}$, where $\mu = \log(k_1 k_2)$, and let $\zeta \in \mathbb{K}'$ be an evaluation target. We want an IOPP with the following two properties:

Completeness. If V is the interleaved encoding of some $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$ satisfying $\widetilde{\psi(W)}(z) = \zeta$, then the verifier accepts with probability 1[7].

Soundness. The verifier rejects with high probability if any of the following occur: 1) V is far from every interleaved encoding of a matrix $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$ satisfying $\widetilde{\psi(W)}(z) = \zeta$,; or 2) if $\mathbb{K} = \mathbb{Q}$, V has entries of bit-size above certain bound $\mathsf{poly}(B)$[8], or V has an entry whose projection under ψ is not well-defined.

The rest of this section gives an overview of such IOPP, which we call Zip+. We build up to the construction in three steps, providing at each step IOPPs for similar claims as above, but with different structures and under different types of projections.

We believe each of these intermediate IOPPs are of independent interest.

Non-projected Constraints with Scalar Witness We start with the simplest version of our IOPP: the case where the witness and its encoding satisfy

$$W \in \mathbb{K}^{k_2 \times k_1}, \qquad V \in \mathbb{K}^{k_2 \times n},$$

(i.e. they do not contain entries in the polynomial ring $\mathbb{K}[X]$) and the constraint is simply of the form

$$\widetilde{W}(z) = \zeta \quad \text{in } \mathbb{K}, \quad z \in \mathbb{K}^{\mu}, \zeta \in \mathbb{K}. \tag{23}$$

[7] When $\mathbb{K} = \mathbb{Q}$, we also ask that W has a well-defined projection under ψ.

[8] It is enough for our compilation purposes to guarantee bit-size bound below $\mathsf{poly}(B)$, and not exactly below B. See Remark 2.17.

for some $z \in \mathbb{K}^\mu$ and $\zeta \in \mathbb{K}$. When $\mathbb{K}$ is a finite field, this is a standard setting for IOPPs, and there are many known constructions for testing proximity to interleaved encodings subject to such constraints.

We use a Brakedown-style IOPP. Write the evaluation point as $z = (z_2, z_1)$, where $z_2 \in \mathbb{K}^{\log k_2}$ and $z_1 \in \mathbb{K}^{\log k_1}$. Define the two vectors

$$u_2 = \big(\mathsf{eq}(z_2, b)\big)_{b \in \{0,1\}^{\log k_2}} \in \mathbb{K}^{k_2}, \qquad \text{and} \qquad u_1 = \big(\mathsf{eq}(z_1, b)\big)_{b \in \{0,1\}^{\log k_1}} \in \mathbb{K}^{k_1}.$$

Then the multilinear constraint Eq. (23) can be written as the tensor constraint

$$\widetilde{W}(z) = u_2^T W u_1 = \zeta \qquad \text{over } \mathbb{K}. \tag{24}$$

From here, the prover sends a vector $a \in \mathbb{K}^{k_2}$, which is claimed to equal Wu_1, and the verifier checks whether $u_2^T a = \zeta$. If this check passes, the prover and verifier reduce to checking whether V is close to an encoding of some W satisfying $Wu_1 = a$. This can be done by taking a random linear combination of the rows of V.

The Brakedown-like approach above is standard in the finite-field case. We now discuss a few minor additions needed in the case $\mathbb{K} = \mathbb{Q}$. First, when $\mathbb{K} = \mathbb{Q}$, we extend the Brakedown approach to handle constraints of the form

$$\widetilde{\phi_m(W)}(z) = \zeta \quad \text{in } \mathbb{F}_m,$$

where m is a large prime, and for $z \in \mathbb{F}_m^\mu$ and $\zeta \in \mathbb{F}_m$. Recall that ϕ_m localizes (reduced modulo m) each entry of W to $\mathbb{F}_m$.

This is handled by a minor adaptation of the approach above. The decomposition in Eq. (24) is now carried out over $\mathbb{F}_m$ rather than over $\mathbb{Q}$, and the prover sends $a \in \mathbb{F}_m^{k_2}$, which is claimed to equal $\phi_m(W)u_1$. The check $u_2^T a = \zeta$ is replaced by $u_2^T a = \zeta \mod m$, and the proximity claim is reduced to checking whether V is close to an encoding of some W satisfying $\phi_m(W)u_1 = a$. The final proximity claim is still handled by taking a random linear combination of the rows of V. This combination is taken and analyzed over $\mathbb{K}$, instead of $\mathbb{F}_m$. In the case $\mathbb{K} = \mathbb{Q}$, the analysis of this final step also requires a new MCA result for linear codes over $\mathbb{Q}$.

The second modification over $\mathbb{Q}$ is that we need to ensure that a cheating prover does not use witnesses with large bitsizes. This check is also handled by the final random linear combination: the prover is required to send the message underlying the resulting random linear combination of the rows of V, and this message must have bounded bitsize. Soundness is then ensured by analyzing the bitsizes of rational vectors under random linear combinations.

Non-projected Constraints with Polynomial Witnesses. Now we move to the case where the constraint is of the form

$$\widetilde{W}(z) = a, \qquad a = \sum_{i=0}^{d-1} X^i \zeta^{(i)} \qquad \text{over } \mathbb{K}[X], \qquad z \in \mathbb{K}^\mu.$$

That is, the multilinear evaluation is applied to a matrix whose entries are polynomials of degree less than d, and the target value is itself a polynomial of degree less than d.

These constraints are straightforward to handle. Write

$$W = \sum_{i=0}^{d-1} X^i \cdot W^{(i)} \qquad \text{and} \qquad V = \sum_{i=0}^{d-1} X^i \cdot V^{(i)}.$$

Then we can view the IOPP over $\mathbb{K}^{<d}[X]$ as d parallel IOPPs over $\mathbb{K}$, reducing to the scalar-witness case with non-projected constraints.

Concretely, for each $i \in \{0, \ldots, d-1\}$, one can test whether $V^{(i)}$ is close to an encoding of a message $W^{(i)}$ satisfying

$$\widetilde{W^{(i)}}(z) = \zeta^{(i)} \qquad \text{over } \mathbb{K},$$

together with the rest of the required properties. Using standard random-linear-combination techniques, the verifier can also combine these d checks into a single proximity claim with a multilinear evaluation constraint over $\mathbb{K}$.

The case $\mathbb{K} = \mathbb{F}_q$ and ψ projects $\mathbb{K}[X]$ to $\mathbb{K}$. In this case, ψ is the homomorphism that evaluates at $X = \alpha$ for some fixed $\alpha \in \mathbb{K}$. Our evaluation constraint has the form

$$\widetilde{\psi(W)}(z) = \zeta \qquad \text{over } \mathbb{K}, \quad z \in \mathbb{K}^{\mu}, \zeta \in \mathbb{K}. \tag{25}$$

We first reduce this projected evaluation constraint to a non-projected tensor constraint, that is, to a constraint of the type handled in the previous case. We then use an IOPP for the latter. The details are as follows.

Write the evaluation point as $z = (z_2, z_1)$, where $z_2 \in \mathbb{K}^{\log k_2}$ and $z_1 \in \mathbb{K}^{\log k_1}$. Define the two equality vectors

$$u_2 = \big(\mathsf{eq}(z_2, b)\big)_{b \in \{0,1\}^{\log k_2}} \in \mathbb{K}^{k_2}, \qquad \text{and} \qquad u_1 = \big(\mathsf{eq}(z_1, b)\big)_{b \in \{0,1\}^{\log k_1}} \in \mathbb{K}^{k_1}.$$

Then (25) can be written as the tensor constraint

$$\widetilde{\psi(W)}(z) = u_2^T \psi(W) u_1 = \zeta \qquad \text{over } \mathbb{K}. \tag{26}$$

The prover now provides a value $a \in \mathbb{K}^{<d}[X]$, which is claimed to equal $u_2^T W u_1$. The verifier checks that $\psi(a) = \zeta$. The prover and verifier then reduce (25), together with the proximity claim that $V \in \mathbb{K}^{<d}[X]^{k_2 \times n}$ is close to an encoding of some $W \in \mathbb{K}^{<d}[X]^{k_2 \times k_1}$ satisfying (25) (together with appropriate bit-size bounds and well-definedness properties), to the claim that V is close to an encoding of some W satisfying

$$u_2^T W u_1 = a \qquad \text{in } \mathbb{K}[X],$$

and the corresponding bit-size and well-definedness properties. This is now a non-projected tensor constraint, and it can be handled using the method described above.

Strictly speaking, the constraint $u_2^T W u_1 = a$ is a *tensor* constraint rather than a multilinear evaluation constraint. However, IOPPs for multilinear constraints can be adapted to handle tensor constraints as well; indeed, tensor constraints are a special case of the same general linear-algebraic structure. It is straightforward to see that the Brakedown-style approach described above extends to tensor constraints rather than only to multilinear evaluation constraints.

The case $\mathbb{K} = \mathbb{Q}$ and MLE evaluation constraints are projected onto a finite field. Assume $\mathbb{K} = \mathbb{Q}$ and that our MLE evaluation constraint is of the form

$$\widetilde{\psi(W)}(z) = \zeta \qquad \text{over } \mathbb{F}_q, \quad z \in \mathbb{F}_q^{\mu}, \quad \zeta \in \mathbb{F}_q, \tag{27}$$

where

$$\psi : \mathbb{Z}_{(p)}^{<d}[X] \to \mathbb{F}_q$$

for some prime p and some extension field $\mathbb{F}_q$ of characteristic p, with $q = p^\kappa$. A naive approach would be to lift z to a point

$$z_{\mathsf{lift}} \in \left(\{0, \ldots, p-1\}^{<\kappa}[X]\right)^\mu,$$

then compute the lifted evaluation $\widetilde{W}(z_{\mathsf{lift}})$ over $\mathbb{Q}[X]$ and attempt to use the previous techniques. However, this lifted evaluation is a polynomial over $\mathbb{Q}$ whose X-degree grows with $\kappa\mu$. This is acceptable when κ is 1 or small, and this is what we do in this case.

However, in some applications, however, we expect to use extension fields of high degree, such as $\mathbb{F}_{2^{128}}$; concretely, this arises in scenarios where one arithmetizes constraints over $\mathbb{F}_2[X]$, as in Section 8. In this latter case, the naive approach is prohibitively expensive.

To avoid this blowup, we design a separate approach where we first reduce the projected evaluation constraint (27) to a *tensor constraint*

$$u_2^T \psi(W) u_1 = \zeta \qquad \text{over } \mathbb{F}_q.$$

We then lift to $\mathbb{Q}[X]$ only the tensor vectors u_1 and u_2. In this way, the lifted tensor evaluation

$$u_{2,\mathsf{lift}}^T W u_{1,\mathsf{lift}} \qquad \text{over } \mathbb{Q}[X]$$

has degree less than $d+2\kappa-2$, rather than degree growing with $\kappa\mu$. This is much more manageable. Concretely, the prover and verifier first compute $u_2 \in \mathbb{F}_q^{k_2}$ and $u_1 \in \mathbb{F}_q^{k_1}$ from the evaluation point $z \in \mathbb{F}_q^\mu$, as described above, so that

$$u_2^T \psi(W) u_1 = \widetilde{\psi(W)}(z) \qquad \text{over } \mathbb{F}_q.$$

They then lift these vectors to

$$u_{1,\mathsf{lift}} \in \left(\{0, \ldots, p-1\}^{<\kappa}[X]\right)^{k_1}, \qquad \text{and} \qquad u_{2,\mathsf{lift}} \in \left(\{0, \ldots, p-1\}^{<\kappa}[X]\right)^{k_2}.$$

Specifically, $u_{1,\mathsf{lift}}$ and $u_{2,\mathsf{lift}}$ are obtained by applying a fixed lift, or section, of ψ to u_1 and u_2, respectively. The prover then provides $a \in \mathbb{Q}^{<d+2\kappa-2}[X]$, which is claimed to equal

$$a = u_{2,\mathsf{lift}}^T W u_{1,\mathsf{lift}} \qquad \text{over } \mathbb{Q}[X].$$

The verifier checks that $\psi(a) = \zeta$ over $\mathbb{F}_q$. If this check passes, the prover and verifier reduce to checking whether V is close to an encoding of some W satisfying

$$u_{2,\mathsf{lift}}^T W u_{1,\mathsf{lift}} = a, \tag{28}$$

together with appropriate bit-size bounds and well-definedness properties.

At this point, we are almost in the standard setting of tensor constraints over $\mathbb{Q}$, except for one remaining issue: the constraint vectors $u_{2,\mathsf{lift}}$ and $u_{1,\mathsf{lift}}$ have polynomial entries of degree less than κ, rather than scalar entries. One option would be to reduce this to $O(\kappa)$ scalar tensor constraints. Instead, we avoid this overhead by projecting once more.

The verifier samples a random fresh large prime m and a random $\xi \in \mathbb{F}_m$. Let

$$\psi_{m,\xi} : \mathbb{Z}_{(m)}[X] \to \mathbb{F}_m$$

denote the homomorphism obtained by localizing coefficients to $\mathbb{F}_m$ and then evaluating at $X = \xi$. Also let

$$\phi_m(W) \in \left(\mathbb{F}_m^{<d}[X]\right)^{k_2 \times k_1}$$

denote the matrix obtained by localizing only the coefficients of the entries of W modulo m. The verifier can instead check

$$\psi_{m,\xi}(u_{2,\mathsf{lift}})^T \phi_m(W) \psi_{m,\xi}(u_{1,\mathsf{lift}}) = \psi_{m,\xi}(a). \tag{29}$$

Note that if Eq. (28) does not hold, then Eq. (29) will fail with high probability over the choice of m and ξ.

Thus the prover and verifier run an IOPP to check that V is close to an encoding of some W satisfying the non-projected tensor constraint

$$\psi_{m,\xi}(u_{2,\mathsf{lift}})^T \phi_m(W) \psi_{m,\xi}(u_{1,\mathsf{lift}}) = \psi_{m,\xi}(a).$$

This IOPP is handled by the non-projected constraint case described above, with minor adaptations for witnesses over $\mathbb{Q}$. We remark that m is chosen large enough so that the honest prover's witness W has no entry whose denominator is divisible by m.

Technical observations. We conclude with a series of technical remarks.

Remark 2.15 (List-Decoding Regime and Mutual Correlated Agreement (MCA))**.** In contrast to Zip, which works in the unique decoding regime, Zip+ is designed to work in the list decoding regime (and in particular up to the distance at which MCA holds). As such, the round-by-round knowledge soundness analysis of Zip no longer carries over and we need to rely on a new analysis which shows round-by-round knowledge soundness according to a recent notion from [BCFW25]. By leveraging MCA, we are able to show round-by-round knowledge soundness beyond unique decoding, and as an added bonus, we can merge the test and evaluation phases (which were previously separate in Zip).

Remark 2.16 (MCA for Linear Codes over $\mathbb{Q}$)**.** The analysis of our IOPP's soundness relies on a property called *Mutual Correlated Agreement* (MCA) [ACFY25a]. Over finite fields it is known that any linear code has MCA up to a distance known as the "1.5 Johnson Bound" [Zei24]. We observe that similar techniques also apply over $\mathbb{Q}$ and provide the straightforward adaptation (of [Zei24]) to show MCA up to the "1.5 Johnson Bound" for codes over $\mathbb{Q}$.[9]

Remark 2.17. *Bitlength Bounds for Combinations of Rationals*: When considering encodings over $\mathbb{Q}$, we also want to reject if the prover sends a valid encoding of a witness with entries of large bitsize. This will be handled by our IOPPs as follows. As we follow a Brakedown approach, the prover will eventually conclude by sending a vector $w^\star$ which is supposedly a linear combination of its witness. If the honest prover is using witnesses of large bitsize, then this linear combination is also likely to have large bitsize, and hence the verifier can reject. Incorporating this check into our IOPP requires some careful but straightforward analysis of the bitsizes of sums and linear combinations of rationals. As a result of it, we will have a gap in the bitsizes accepted with perfect completeness and those rejected in the soundness case. That is, the honest prover uses witnesses with entries of bitsize at most B_0, while our soundness only guarantees that the verifier rejects if a prover attempts to use witnesses with entries of bitsize at least $B > B_0$.

[9] We note that in our experiments we use a rate of 1/4 and the Unique Decoding Radius (UDR), since in this parameter regime the UDR leads to better performance than the 1.5 Johnson bound.

2.5 Efficiency and experimental results

We provide benchmark details for Zinc+. Table 4 reports end-to-end Zinc+ costs for 7 consecutive SHA-256 compressions followed by ECDSA verification (except for the three operations modulo the scalar prime n of secp256k1, see below). The implementation and benchmarks are available at `github.com/NethermindEth/zinc-plus/tree/main-beta`. Additional details can be found below and in Section 9.

Zinc+ on 7x SHA-256 + ECDSA MSM	**Prover**	**Verifier**	**Proof size**
Zip+ IOPP			
Commit	8.4 ms	—	—
Open	4.7 ms	4.2 ms	0.06 KB
Subtotal	13.1 ms	4.2 ms	162 KB
Zinc+ PIOR (Steps 1, 2, and 3 of Section 2.2)			
Trace projection $\mathbb{Z}[X] \to \mathbb{F}_q[X]$	0.9 ms	0.0 ms	—
Ideal elements	4.2 ms	0.1 ms	9 KB
Trace projection $\mathbb{F}_q[X] \to \mathbb{F}_q$	0.4 ms	0.1 ms	—
Subtotal	5.5 ms	0.2 ms	9 KB
Finite field PIOP	21.4 ms	2.0 ms	27 KB
Proof compression (zstd-3)	0.6 ms	—	—
Proof uncompression (zstd-3)	—	0.6 ms	—
Total	**40.6 ms**	**7.0 ms**	**198 KB**

Table 4: Zinc+ prover breakdown for 7 SHA-256 compressions followed by ECDSA verification, not including the three operations modulo the scalar prime n of secp256k1. Proof sizes are compressed.

Our implementation is not fully optimized. Additionally, we make the suboptimal choice of arithmetizing entirely over $\mathbb{Z}[X]$ rather than an "optimal" combination of $\mathbb{Z}[X]$, $\mathbb{F}_2[X]$, and $\mathbb{F}_q$ (q being the ECDSA prime), as we have only implemented support for the former.

Full ECDSA verification (cf. Section 8.3) consists of computing a multi-scalar multiplication (MSM) $R = u_1 \cdot G + u_2 \cdot Q$ plus verifying: two constraints $u_1 \cdot s = e, u_2 \cdot s = r$ modulo n; and $\mathsf{int}(R_x) = r$ modulo n, where n is the scalar prime n of secp256k1; the latter three operations have negligible computational cost for a verifier, compared to the MSM. In a context where zero-knowledge is required, one would need to include them. We prove only the MSM, and omit the rest because our implementation does not yet support constraints over multiple domains simultaneously. This will be addressed in future updates.

The large proof size is due in part to our choice of PCS/IOPP, which is based on the Brakedown/Ligero approach [GLSTW23; AHIV22], known for large proof sizes. Ongoing work of ours is to design an IOPP based on WHIR [ACFY25b] and similar schemes, which produce smaller proofs.

A few components of our implementation have been completed with AI (see the end of Section 9 for specific details). These have been reviewed carefully by us and are in the process of being integrated into the `main` branch (full benchmarks are currently available in our `main-beta` branch).

Implementation, parameters, and benchmark details. All benchmarks are run on a MacBook Air M4 (16 GB RAM, 2025), multi-threaded, targeting 100 bits of security. We use IPRS codes of rate 1/8, radix 8, and base field of prime size $2^{16} + 1$. The proximity parameter is set to

the 1.5 Johnson bound. The random prime field has ~ 192 bits, more than enough for our target security.

We arithmetize the computation in the form of a UAIR (a specific type of UCS constraint, see Section 8.1). Our UAIR is very similar to the one described in Section 8, but is not exactly the same due to limitations of our implementation and a design choice that allows for easier handling of boundary constraints, see Section 9 for details. Our trace has 2^9 rows and 19 witness columns (11 from SHA and 8 from ECDSA). Of those, 11 are lookup-constrained to belong to $\{0,1\}^{<32}[X] \subseteq \mathbb{Q}[X]$. SHA contributes 18 sparse public input columns, and ECDSA 15.

Our finite field PIOP is a sumcheck-based protocol similar to Hyperplonk's PIOP (without permutation constraints) [CBBZ23] and the original Binius's PIOP [DP25], from which we use the construction that provides virtual access to shifted trace columns (via a sumcheck), cf. Section 4.3 of [DP25]. See Section 9 for more details of our PIOP.

In a future document, we will provide a full description of our Zinc+ instantiation and arithmetization.

Proof compression. All proof sizes reported above are compressed. Zinc+ proofs compress by about 2x. We refer to [FZ25] for related work on proof compressibility.

3 Preliminaries

As usual, for $k \geq 1$, we let $[k] := \{1, \ldots, k\}$, and let $[0\,..\,k]$ be $\{0\} \cup [k]$. We write $x \leftarrow S$ for uniform sampling from a finite set S, and $x \leftarrow \mathcal{D}$ for sampling from a distribution $\mathcal{D}$.

Given $w \geq 1$, we let $[0\,..\,2^w - 1] = \{0, \ldots, 2^w - 1\}$ and $\{0,1\}^{<w}[X]$ be the set of polynomials on the variable X, of degree less than w with coefficients in $\{0,1\}$.

For two integers $u, v \in [0\,..\,2^w - 1]$ with binary representations $u = \sum_{i=0}^{w-1} u_i 2^i$ and $v = \sum_{i=0}^{w-1} v_i 2^i$, and for $0 \leq r < w$, the *bitwise exclusive or* (XOR), *bitwise conjunction* (AND), *right rotation* by r bits, and *right shift* by r bits are defined as

$$u \;\mathsf{XOR}\; v \;=\; \sum_{i=0}^{w-1} (u_i + v_i \bmod 2)\, 2^i, \qquad u \;\mathsf{AND}\; v \;=\; \sum_{i=0}^{w-1} (u_i \cdot v_i)\, 2^i,$$

$$\mathsf{ROTR}^r(u) \;=\; \sum_{i=0}^{w-1} u_{(i+r) \bmod w}\, 2^i, \qquad \mathsf{SHR}^r(u) \;=\; \sum_{i=0}^{w-1-r} u_{i+r}\, 2^i.$$

These operations are defined on $\{0,1\}^{<w}[X]$ as well in a natural way.

3.1 Algebra

Fields and polynomial rings We use $\mathbb{K}$ to denote (possibly infinite) fields and R to denote rings. We write $\mathbb{F}_q$ for the finite field with q elements (where q is a prime or a prime power), $\mathbb{Q}$ for the field of rational numbers, and $\mathbb{Z}$ for the ring of integers. An ideal I of a ring R generated by a set $S \subseteq R$ is denoted (S), the quotient ring is denoted R/I. A ring is an integral domain if the product of any two nonzero elements is nonzero.

Given a ring R, we let $R[X]$ be the ring of univariate polynomials with coefficients in R, on the variable X. Similarly, for a tuple of variables $\mathbf{Y} = (Y_1, \ldots, Y_\mu)$, $R[\mathbf{Y}]$ denotes multivariate polynomials with coefficients in R. We refer to $R[X]$ and $R[\mathbf{Y}]$ as *polynomial rings*.

Often, we work with multilinear polynomials from $R[\mathbf{Y}]$ where R is itself a polynomial ring, e.g. $R = R'[X]$ with $R' = \mathbb{Q}, \mathbb{Z}$, or $R' = \mathbb{F}_q$. In this case, we view elements of $R[\mathbf{Y}] = (R'[X])[\mathbf{Y}]$

as polynomials in $\mathbf{Y}$ with coefficients in $R'[X]$ (forgetting that the coefficients themselves are polynomials), and not as elements of $R'[X, \mathbf{Y}]$, i.e. as polynomials on the variables $(X, \mathbf{Y})$ with coefficients in R', even though $(R'[X])[\mathbf{Y}]$ and $R'[X, \mathbf{Y}]$ are isomorphic rings.

Following this, we often devoid elements from $R'[X]$ from their polynomiality. In other words, we treat elements from $R'[X]$ as abstract ring elements, and rarely employ their specific properties as polynomials. Accordingly, we usually denote elements of $R'[X]$ by lowercase letters $f, g, h, u, v, \ldots$.

We let $R^{<d,B}[X]$ denote polynomials of degree $< d$ with coefficients in $\mathcal{R}^{<B}$; we may write $\mathcal{R}^{<d}[X]$ when the bit bound is clear. Integers are represented as strings of bits as usual, rationals $a/b \in \mathbb{Q}$ are represented as pairs (a, b) in *lowest terms* (i.e., $\gcd(a, b) = 1$ and $b > 0$), with bit-size the sum of the bit-sizes of a and b.

Ring homomorphisms $f : \mathcal{R}_1 \to \mathcal{R}_2$ extend coefficient-wise to $\mathcal{R}_1[\mathbf{Y}] \to \mathcal{R}_2[\mathbf{Y}]$. Additionally, all maps are extended entrywise to vectors and matrices.

Primitive elements. For a finite extension $\mathbb{F}_{\tilde{q}}/\mathbb{F}_q$ of finite fields with $\tilde{q} = q^\ell$, we denote by $\mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_q)$ the set of primitive elements of $\mathbb{F}_{\tilde{q}}$ over $\mathbb{F}_q$, i.e.,

$$\mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_q) := \{\alpha \in \mathbb{F}_{\tilde{q}} : \mathbb{F}_q(\alpha) = \mathbb{F}_{\tilde{q}}\}.$$

Equivalently, $\alpha \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_q)$ iff α does not lie in any proper subfield of $\mathbb{F}_{\tilde{q}}$ containing $\mathbb{F}_q$.

Sparsity. For a multivariate polynomial $f \in \mathcal{R}[V_1, \ldots, V_k]$, write $\|f\|_0$ for the number of monomials in the support of f—equivalently, the number of nonzero coefficients when f is written in monomial form, after collecting like monomials over the coefficient ring.

For any field $\mathbb{F}_{p^e}$ containing a subfield L , we can map elements of L into $\mathbb{F}_{p^e}$ via the canonical embedding $\iota_{p^e} : L \hookrightarrow \mathbb{F}_{p^e}$. We extend this embedding to the ring $\mathbb{F}_{p^e}[X]$ its subrings $L[X]$ by applying ι_{p^e} coefficient-wise. If $(\mathsf{j}) = \mathcal{I} \subseteq \mathbb{F}_q[X]$ is an ideal, we note $(\iota_{q^\ell}(\mathsf{j})) = \mathcal{I}' \subseteq \mathbb{F}_{q^\ell}[X]$ is also.

Local rings Given a prime p, subset $\mathbb{Z}_{(p)} = \{a/b \in \mathbb{Q} \mid p \nmid b\}$ of $\mathbb{Q}$ is a ring called the *localization of $\mathbb{Z}$ at the prime p*. This ring admits a surjective homomorphism $\phi_p : \mathbb{Z}_{(p)} \to \mathbb{F}_p$, $a/b \mapsto a \cdot b^{-1} \bmod p$, where b^{-1} is the multiplicative inverse of b modulo p. We have $\ker(\phi_p) = p\mathbb{Z}_{(p)}$.

Given a prime power $q = p^e$, we set $\mathbb{Z}_{(q)} := \mathbb{Z}_{(p)}$, and define $\phi_q : \mathbb{Z}_{(q)} \to \mathbb{F}_q$ as $\phi_q = \iota_q \circ \phi_p$. Given prime powers $q_1, \ldots, q_n$ with $q_i = p_i^{e_i}$, we define the *intersection localization ring* on $q_1, \ldots, q_n$ as

$$\mathbb{Z}_{(q_1 \cdots q_n)} \;:=\; \bigcap_{i \in [n]} \mathbb{Z}_{(p_i)} \;=\; \Big\{ a/b \in \mathbb{Q} \;\Big|\; \gcd\big(b, \textstyle\prod_i p_i\big) = 1 \Big\}. \tag{30}$$

Multilinear extensions (MLE). Given a ring R, and μ variables $\mathbf{Y}$, a multilinear polynomial $f(\mathbf{Y}) \in R[\mathbf{Y}]$ is uniquely determined by its values on $\{0,1\}^\mu$. For any map $f : \{0,1\}^\mu \to R$, we write $\widetilde{f}(\mathbf{Y})$ for its *multilinear extension* (MLE), i.e. the unique multilinear polynomial that agrees with f on the Boolean hypercube. For a vector $\mathbf{v} \in R^{2^\mu}$, we write $\widetilde{\mathbf{v}}(\mathbf{Y})$ for the MLE of the map $\mathbf{b} \mapsto v_{\mathbf{b}}$ (identifying $\{0,1\}^\mu$ with $[2^\mu]$ when appropriate). We also use the standard *equality polynomial*

$$\widetilde{\mathsf{eq}}(\mathbf{Z}; \mathbf{Y}) := \prod_{i \in [\mu]} \left(Z_i \cdot Y_i + (1 - Z_i) \cdot (1 - Y_i)\right), \tag{31}$$

where $\mathbf{Z}$ is a tuple of μ variables. We then have

$$\tilde{v}(\mathbf{Y}) = \sum_{\mathbf{b} \in \{0,1\}^\mu} v_{\mathbf{b}} \cdot \widetilde{\mathsf{eq}}(\mathbf{b}; \mathbf{Y}).$$

Schwartz–Zippel lemma for exceptional sets. We will need the following general form of Schwartz-Zippel lemma in order to address certain ideal-membership constraints in our proof system (namely, Step 2 in Section 2.2.4). Given a ring R, a subset $S \subseteq R$ is *exceptional* if $a - b$ has a multiplicative inverse for all distinct $a, b \in S$. The standard Schwartz-Zippel lemma applies whenever R is an integral domain; Lemma 3.1 below is the generalization needed when R has zero divisors (e.g., the ring $\mathbb{K}[X]/\mathfrak{I}$ that arises in the proof of Lemma 5.1).

Lemma 3.1 (Generalized Alon–Füredi Theorem [BCPS18])**.** *Let R be a ring and let $S \subseteq R$ be a finite exceptional subset of R. Let $f(\mathbf{Y}) \in R[\mathbf{Y}]$ be a nonzero multilinear polynomial in μ variables. Then $\Pr_{\mathbf{r} \leftarrow S^\mu}[f(\mathbf{r}) = 0] \leq \mu/|S|$.*

Lemma 3.2 (Bound on large prime divisors of small integers)**.** *Let $B > B_{\mathcal{P}}$. For any $a \in \mathbb{Z}^{<B} \setminus \{0\}$ of bit length less than B, the number of primes of bit length at least $B_{\mathcal{P}}$ dividing it is less than $(B-1)/(B_{\mathcal{P}}-1)$.*

Proof. Say t primes $p_1, \ldots, p_t$ divide a. By assumption $|a| < 2^{B-1}$ and that $p_i \geq 2^{B_{\mathcal{P}}-1}$ for all $i \in [t]$. Then we have $2^{t(B_{\mathcal{P}}-1)} \leq |a| < 2^{B-1}$, which implies $t < (B-1)/(B_{\mathcal{P}}-1)$. □

3.2 Constrained linear codes and Mutual Correlated Agreement (MCA)

Constrained linear codes. A *linear code* over a field $\mathbb{K}$ is a linear subspace $\mathcal{C} \subseteq \mathbb{K}^n$. The *block length* is n, and the *dimension* $k = \mathsf{dim}(\mathcal{C})$ is the dimension of the subspace. A *generator matrix* $M \in \mathbb{K}^{n \times k}$ for $\mathcal{C}$ is a matrix whose columns form a basis for $\mathcal{C}$, so $\mathcal{C} = \{Mw : w \in \mathbb{K}^k\}$. The *relative distance* $\delta \in (0, 1]$ of $\mathcal{C}$ is the minimum, over all distinct $\mathsf{c}, \mathsf{c}' \in \mathcal{C}$, of the fraction of coordinates in which c and c' differ. For a vector $\mathbf{v} \in \mathbb{K}^n$, the *relative Hamming distance* from $\mathbf{v}$ to the code is $\mathsf{dist}(\mathbf{v}, \mathcal{C}) = \min_{\mathsf{c} \in \mathcal{C}} \mathsf{dist}(\mathbf{v}, \mathsf{c})$, where $\mathsf{dist}(\mathbf{v}, \mathsf{c})$ denotes the fraction of coordinates in which $\mathbf{v}$ and c differ.

Given a linear code $\mathcal{C} \subseteq \mathbb{K}^n$ with generator matrix $M \in \mathbb{K}^{n \times k}$, the *$J$-wise interleaving* of $\mathcal{C}$ is the code

$$\mathcal{C}^J \;:=\; \{WM^T : W \in \mathbb{K}^{J \times k}\} \;\subseteq\; \mathbb{K}^{J \times n}.$$

A codeword in $\mathcal{C}^J$ can be viewed as a tuple $(v_1, \ldots, v_J)$ of J codewords $v_i \in \mathcal{C}$.

Mutual Correlated Agreement Let $M \in \mathbb{K}^{n \times k}$ be a generator matrix generating a code $\mathcal{C} \subseteq \mathbb{K}^n$ with relative distance δ. For any $S \subseteq [n]$ and any $v : [n] \to \mathbb{K}$ we let $v|_S : S \to \mathbb{K}$ denote the restriction of v to its coordinates in S and let $\mathcal{C}|_S$ denote the projection of $\mathcal{C}$ to the coordinates in S. Hence, we write $v|_S \in \mathcal{C}|_S$ to mean that there exists some codeword $c \in \mathcal{C}$ such that $c|_S = v|_S$. We also let $\nu(S) = |S|/n$ denote the fractional size of S. Now for a distance parameter $\beta \in [0, 1]$, we define the *agreement domains* up to distance β of v with respect to $\mathcal{C}$ as

$$\mathsf{AD}_{\mathcal{C},\beta}(v) = \{S \subseteq [n] \mid \mu(S) > 1 - \beta, v|_S \in \mathcal{C}|_S\}.$$

In words, these are all the subsets of coordinates of fractional size greater than $1 - \beta$ such that the restriction of v to those coordinates is in the code.

The mutual correlated agreement property, which we define next, states that for any two vectors $v_1, v_2 \in \mathbb{K}^n$, for most choices of coefficients r_1, r_2 from some set $R \subseteq \mathbb{K}$, the linear combination $r_1 v_1 + r_2 v_2$ does not have any new agreement domains compared to v_1 and v_2.

Definition 3.3. Given a code $\mathcal{C} \subseteq \mathbb{K}^n$ and a set of coefficients $R \subseteq \mathbb{K}$ we say that $\mathcal{C}$ satisfies *mutual correlated agreement (MCA)* with respect to R up to distance β with error $\mathsf{err}_{\mathsf{pg}}$ if for every $v_1, v_2 \in \mathbb{K}^n$

$$\Pr_{r_1,r_2 \in R}\left[\mathsf{AD}_{\mathcal{C},\beta}(r_1 \cdot v_1 + r_2 \cdot v_2) \not\subseteq \bigcap_{i=1}^{2} \mathsf{AD}_{\mathcal{C},\beta}(v_i)\right] \leq \mathsf{err}_{\mathsf{pg}}.$$

In words, the event of the probability above is that there exists S of fractional size at least $1-\beta$ such that $(r_1 \cdot v_1 + r_2 \cdot v_2)|_S$ is in the code, but there is no large subset $S' \subseteq S$ of measure greater than $\beta - \eta$ such that both $v_1|_{S'}$ and $v_2|_{S'}$ are in the code.

The next theorem states that every linear code (even those over infinite fields such as $\mathbb{Q}$) satisfies MCA up to a distance known as the 1.5 Johnson bound.

Theorem 3.4. *Let $\mathcal{C} \subseteq \mathbb{K}^n$ be a linear code with relative distance δ. Then, for any set of coefficients $R \subseteq \mathbb{K}$, any $\varepsilon, \beta \in [0,1)$ such that $\beta = a/n$ for some $a \in \mathbb{N}$ and*

$$\beta > (1 - \delta + \varepsilon)^{1/3}, \quad \varepsilon < 0.18,$$

the code $\mathcal{C}$ satisfies MCA with respect to R up to distance β with error $\frac{n}{\varepsilon|R|}$.

Proof. The proof is essentially the same as in [Zei24] and is included in Appendix A.2.3 for completeness. □

Any code satisfying MCA also satisfies the following two properties which can be thought of as a generalization of MCA to linear combinations of J functions for any $J \geq 2$ and a list-decoding preservation statement respectively.

Theorem 3.5. *Let $\mathcal{C} \subseteq \mathbb{K}^n$ be a linear code with relative distance δ and let $R \subseteq \mathbb{K}$ be a set of coefficients and such that $\mathcal{C}$ satisfies MCA up to distance $\beta < \delta$ and error $\mathsf{err}_{\mathsf{pg}}$ with respect to R. Then for any $v_1, \ldots, v_J \in \mathbb{K}^n$, we have,*

$$\Pr_{r_j \in R}\left[\mathsf{AD}_{\mathcal{C},\beta}\left(\sum_{i=1}^{J} r_i \cdot v_i\right) \not\subseteq \bigcap_{i=1}^{J} \mathsf{AD}_{\mathcal{C},\beta}(v_i)\right] \leq (J-1)\mathsf{err}_{\mathsf{pg}}.$$

Proof. The proof is deferred to Appendix A.2.3. □

Finally we show that MCA implies a property called list preservation. Given a code $\mathcal{C} \subseteq \mathbb{K}^n$ and a vector $v \in \mathbb{K}^n$, we define

$$\mathsf{List}_{\mathcal{C},\beta}(v) = \{u \in \mathcal{C} \mid \mathsf{dist}(u, v) \leq \beta\},$$

and given a list of vectors $(v_1, \ldots, v_J) \in \mathbb{K}^n$ we define

$$\mathsf{List}_{\mathcal{C}^J,\beta}(v_1, \ldots, v_J) = \{(u_1, \ldots, u_J) \in \mathcal{C}^J \mid \exists S \subseteq [n], |S| > (1-\beta)n, u_j|_S = v_j|_S, \forall j \in [J]\}.$$

Then Theorem 3.5 implies the following list preservation property.

Theorem 3.6. *Let $\mathcal{C} \subseteq \mathbb{K}^n$ be a linear code with relative distance δ and let $R \subseteq \mathbb{K}$ be a set of coefficients and such that $\mathcal{C}$ satisfies MCA up to distance $\beta < \delta$ and error $\mathsf{err}_{\mathsf{pg}}$ with respect to R. Then for any $v_1, \ldots, v_J \in \mathbb{K}^n$ we have,*

$$\Pr_{r_j \in R}\left[\begin{array}{c}\mathsf{List}_{\mathcal{C},\beta}\left(\sum_{j=1}^{J} r_j v_j\right) \subseteq \left\{\sum_{j=1}^{J} r_j u_j \mid (u_1,\ldots,u_J) \in \mathsf{List}_{\mathcal{C}^J,\beta}(v_1,\ldots,v_J)\right\} \\ \bigvee \mathsf{AD}_{\mathcal{C},\beta}(v^*) \not\subseteq \cap_{j=1}^{J} \mathsf{AD}_{\mathcal{C},\beta}(v_j)\end{array}\right] \leq (J-1)\mathsf{err}_{\mathsf{pg}}.$$

Proof. The proof is deferred to Appendix A.2.3. □

3.3 Interactive Oracle Reductions

Indexed relations and oracles. An *indexed relation* $\mathcal{R}$ is a set of index-instance-witness triples $(\mathbb{i}, \mathbb{x}; \mathbb{w})$. The *language* associated with $\mathcal{R}$ is $\mathsf{LANG}(\mathcal{R}) := \{(\mathbb{i}, \mathbb{x}) \mid \exists\, \mathbb{w} \text{ such that } (\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathcal{R}\}$. Each relation declares a partition of the entries of $(\mathbb{i}, \mathbb{x})$ into *explicit parts* and *oracle parts*. The verifier reads explicit parts directly and accesses oracle parts via oracle queries.

A relation declares the type of each oracle entry, which falls into one of three kinds:

- *String oracles.* Vectors over an alphabet Σ, queried by index. We write $[[\pi]]$ for an oracle to a string π, returning π_j on query j.
- *Polynomial oracles.* Polynomials in some ring $R[Y_1, \ldots, Y_\mu]$, queried by evaluation. We write $[[f]]$ for an oracle to a polynomial f, returning $f(\mathbf{r})$ on query $\mathbf{r}$. A polynomial-oracle type may constrain the polynomial along any combination of: number of variables, total or per-variable degree bound, coefficient ring, and (for $\mathbb{Q}$-coefficients) coefficient bit-size bound.
- **Projected polynomial oracles.** Polynomial oracles whose response is computed by a lift-evaluate-project sequence: each query is lifted into the polynomial's coefficient ring via a section, the polynomial is evaluated at the lifted point, and the result is projected into a target ring via a (possibly partial) map. Responses are in the target ring, or $\perp$ when the projection is undefined.

We define the trivial relation $\mathcal{R}_{\mathsf{triv}} := \{(\perp, \perp; \perp)\}$ with $\mathsf{LANG}(\mathcal{R}_{\mathsf{triv}}) = \{(\perp, \perp)\}$. For an IOR with target $\mathcal{R}_{\mathsf{triv}}$, we identify the verifier's output $(\mathbb{i}', \mathbb{x}') = (\perp, \perp)$ with acceptance (or outputting 1) and any other output with rejection (or outputting 0). For nontrivial target relations, rejection means the verifier outputs an instance outside the target language; the protocol specifies how this is encoded.

IOR protocols. A public-coin *Interactive Oracle Reduction (IOR)* $\Pi = (\mathsf{P}, \mathsf{V})$ from a relation $\mathcal{R}$ to a relation $\mathcal{R}'$ is specified as follows. The prover P receives $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathcal{R}$. The verifier V receives $(\mathbb{i}, \mathbb{x})$: explicit parts directly, oracle parts via oracle access. They interact for k rounds. In round $j \in [k]$, the prover sends an oracle π_j and the verifier sends a uniformly random message $\rho_j \in \{0,1\}^{r_j}$. After the k rounds, the verifier makes oracle queries to the oracle parts of $(\mathbb{i}, \mathbb{x})$ and to $\pi_1, \ldots, \pi_k$. Then:

- V outputs a new index-instance pair $(\mathbb{i}', \mathbb{x}')$. The oracle parts of $\mathbb{x}'$ are determined implicitly as functions of the oracle parts of $(\mathbb{i}, \mathbb{x})$ and of $\pi_1, \ldots, \pi_k$.
- P outputs a witness $\mathbb{w}'$.

Without loss of generality, V's output is a deterministic function of its inputs and the transcript (including oracle responses).

Types of IORs used in this paper. All variants below are IORs; we distinguish them informally by oracle type and target relation. The categories overlap.

- *IOP:* string oracles, trivial target relation.
- *IOPP (IOP of proximity):* an IOP whose source relation enforces proximity to a code.
- *PIOR (polynomial IOR):* polynomial oracles, non-trivial target relation.
- *PIOP (polynomial IOP):* polynomial oracles, trivial target relation.

3.3.1 Completeness

Definition 3.7 (Completeness)**.** Π has *completeness* with error $\epsilon(\cdot)$ if for every $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathcal{R}$, in honest execution producing transcript $(\rho_1, \ldots, \rho_k)$, V's output $(\mathbb{i}', \mathbb{x}')$, and P's output $\mathbb{w}'$,

$$\Pr_{\rho_1, \ldots, \rho_k}\left[\, (\mathbb{i}', \mathbb{x}'; \mathbb{w}') \notin \mathcal{R}' \,\right] \;\leq\; \epsilon(\mathbb{i}, \mathbb{x}).$$

Perfect completeness corresponds to $\epsilon \equiv 0$.

3.3.2 Round-by-round knowledge soundness

Two formulations of round-by-round knowledge soundness (RBR-KS) appear in the literature; we use both. The *state function* formulation [CMS19; BMNW25] uses a witness-free predicate $\mathsf{State}(\mathbb{i}, \mathbb{x}, \mathsf{tr})$ paired with a global extractor $\mathcal{E}_{\mathsf{rbr}}(\mathbb{i}, \mathbb{x}, \mathsf{tr})$. The *knowledge state function* formulation [BCFW25] threads a candidate *knowledge state witness* w through both the predicate $\mathsf{KState}(\mathbb{i}, \mathbb{x}, \mathsf{tr}, \mathsf{w})$ and the extractor. State-function-form RBR-KS implies knowledge-state-form RBR-KS with the same per-round errors by the construction of [BCFW25, Theorem C.1]: the new extractor returns w when $(\mathbb{i}, \mathbb{x}; \mathsf{w}) \in \mathcal{R}$, and otherwise returns the state-form extractor on the pre-randomness transcript.

Knowledge state function version.

Definition 3.8 (Knowledge state function)**.** Let $\Pi = (\mathsf{P}, \mathsf{V})$ be an IOR from a relation $\mathcal{R}$ to $\mathcal{R}'$. A *knowledge state function* for Π is a (possibly inefficient) function KState that, on input a statement $(\mathbb{i}, \mathbb{x})$, interaction transcript tr, and knowledge state witness w, outputs a bit and has the following properties.

- **Empty transcript.** If $\mathsf{tr} = \emptyset$ is the empty transcript, then $\mathsf{KState}(\mathbb{i}, \mathbb{x}, \mathsf{tr}, \mathsf{w}) = 1$ if and only if $(\mathbb{i}, \mathbb{x}; \mathsf{w}) \in \mathcal{R}$.
- **Prover moves.** If tr is a transcript where the prover is about to move and $\mathsf{KState}(\mathbb{i}, \mathbb{x}, \mathsf{tr}, \mathsf{w}) = 0$, then $\mathsf{KState}(\mathbb{i}, \mathbb{x}, \mathsf{tr} \| \pi, \mathsf{w}) = 0$ for every prover message π.
- **Full transcript.** If $\mathsf{tr} = (\pi_1, \rho_1, \ldots, \pi_k, \rho_k)$ is a full transcript, then $\mathsf{KState}(\mathbb{i}, \mathbb{x}, \mathsf{tr}, \mathsf{w}) = 1$ if and only if V on tr outputs $(\mathbb{i}', \mathbb{x}')$ such that $(\mathbb{i}', \mathbb{x}'; \mathsf{w}) \in \mathcal{R}'$.

Definition 3.9 (Round-by-round knowledge soundness)**.** Π has *round-by-round knowledge soundness* with errors $(\kappa_1, \ldots, \kappa_k)$ if there exists a knowledge state function KState as in Definition 3.8 and a deterministic extractor $\mathcal{E}_{\mathsf{rbr}}$ such that for every $(\mathbb{i}, \mathbb{x})$, every $j \in [k]$, and every transcript tr where the verifier is about to send ρ_j,

$$\Pr_{\rho_j}\left[\begin{array}{l} \exists\, \mathsf{w} : \\ \wedge\ \mathsf{KState}\big(\mathbb{i}, \mathbb{x}, \mathsf{tr}, \mathcal{E}_{\mathsf{rbr}}(\mathbb{i}, \mathbb{x}, \mathsf{tr} \| \rho_j, \mathsf{w})\big) = 0 \\ \wedge\ \mathsf{KState}(\mathbb{i}, \mathbb{x}, \mathsf{tr} \| \rho_j, \mathsf{w}) = 1 \end{array}\right] \;\leq\; \kappa_j(\mathbb{i}, \mathbb{x}).$$

Remark 3.10 (Auxiliary witness slots in KState arguments)**.** When an outer IOR is built using an inner IOR as a black box, the outer KState invokes the inner KState as a sub-formula and the outer extractor invokes the inner extractor as a subroutine, so the outer's w must include an inner-protocol witness alongside the outer relation's witness. To accommodate this without modifying the source or target relations, we work with augmentations $\mathcal{R}_{\mathsf{aug}}$ and $\mathcal{R}'_{\mathsf{aug}}$ that admit an extra inert component $\mathsf{w}_{\mathsf{aux}}$:

$$(\mathbb{i}, \mathbb{x}; (\mathbb{w}, \mathsf{w}_{\mathsf{aux}})) \in \mathcal{R}_{\mathsf{aug}} \iff (\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathcal{R},$$

and analogously for $\mathcal{R}'_{\mathsf{aug}}$. Since $\mathsf{LANG}(\mathcal{R}_{\mathsf{aug}}) = \mathsf{LANG}(\mathcal{R})$ (and likewise for $\mathcal{R}'$), the resulting RBR knowledge soundness guarantee is for the original relations.

State function version.

Definition 3.11 (State function)**.** Let $\Pi = (\mathsf{P}, \mathsf{V})$ be an IOR from a relation $\mathcal{R}$ to $\mathcal{R}'$. A *state function* for Π is a (possibly inefficient) function State that, on input a statement $(\mathbb{i}, \mathbb{x})$ and interaction transcript tr, outputs a bit and has the following properties.

- **Empty transcript.** $\mathsf{State}(\mathbb{i}, \mathbb{x}, \emptyset) = 0$.
- **Prover moves.** If tr is a transcript where the prover is about to move and $\mathsf{State}(\mathbb{i}, \mathbb{x}, \mathsf{tr}) = 0$, then $\mathsf{State}(\mathbb{i}, \mathbb{x}, \mathsf{tr} \| \pi) = 0$ for every prover message π.
- **Full transcript.** If $\mathsf{tr} = (\pi_1, \rho_1, \ldots, \pi_k, \rho_k)$ is a full transcript, then $\mathsf{State}(\mathbb{i}, \mathbb{x}, \mathsf{tr}) = 1$ if and only if V on tr outputs $(\mathbb{i}', \mathbb{x}')$ such that there exists $\mathbb{w}'$ with $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}'$.

Definition 3.12 (Strong round-by-round knowledge soundness)**.** Π has *strong round-by-round knowledge soundness* with errors $(\kappa_1, \ldots, \kappa_k)$ if there exist a state function State as in Definition 3.11 and a deterministic extractor $\mathcal{E}_{\mathsf{rbr}}$ such that for every $(\mathbb{i}, \mathbb{x})$, every $j \in [k]$, and every transcript tr where the verifier is about to send ρ_j,

$$\Pr_{\rho_j}\left[\begin{array}{l} \mathsf{State}(\mathbb{i}, \mathbb{x}, \mathsf{tr}) = 0 \\ \wedge\ \mathsf{State}(\mathbb{i}, \mathbb{x}, \mathsf{tr} \| \rho_j) = 1 \\ \wedge\ (\mathbb{i}, \mathbb{x}; \mathcal{E}_{\mathsf{rbr}}(\mathbb{i}, \mathbb{x}, \mathsf{tr})) \notin \mathcal{R} \end{array}\right] \leq \kappa_j(\mathbb{i}, \mathbb{x}).$$

Compiling to non-interactive arguments of knowledge. The compiled Zinc+ IOP (Theorem 5.17) satisfies knowledge-state-form RBR knowledge soundness. By [BCFW25, Theorem B.4], this implies the straightline state-restoration knowledge soundness variant of [BCFW25, Definition B.2]. The BCS-style transformation of [BMNW25, Theorem 5.9], which adapts [BCS16] to this setting, then yields a SNARK in the random oracle model (ROM).

Quantum-random-oracle-model security is conjectural: [BCFW25] conjectures the QROM result of [CMS19] extends to the BCS-style transformation in this setting, which in its current form requires the IOP has strong RBR knowledge soundness.

4 Universal Constraint Systems (UCS)

We start this section by presenting the main relation we are interested in creating proofs for, which we call the *universal constraint system* (UCS) relation. After that, we discuss some structural semantics of UCS, namely how one can write lookup, permutation, R1CS, CCS, and AIR constraints within a UCS, over different rings, and exploiting ideal membership predicates.

Our presentation is purely technical. We refer to Sections 2.1 and 8 for intuition and explanations of how UCS can be used to arithmetize many computations of practical interest.

4.1 General UCS definition

In this section we define the *universal constraint system* (UCS) relation $\mathsf{REL}_{\mathsf{UCS}}$.

Definition 4.1 (Universal constraint system (UCS))**.** The relation $\mathsf{REL}_{\mathsf{UCS}}$ is the set of all triples $(\mathbb{i}, \mathbb{x}; \mathbb{w})$ of the following form.

Index. An *index* $\mathbb{i}$ is a tuple

$$\mathbb{i} = (m, \ell, n, \mathbf{q}, B, \mathbf{d}, \mathcal{C}),$$

where $m \geq 0$ is the *witness length*, $\ell \geq 0$ is the *input length*, $n \geq 0$ is the *number of rows*, $\mathbf{q} = (q_1, \ldots, q_{|\mathbf{q}|})$ is a tuple of prime powers, $B \geq 0$ is a *bit-size bound*, $\mathbf{d} = (d_0, \ldots, d_{|\mathbf{q}|})$ is a tuple of *degree bounds*, and $\mathcal{C}$ is a family of *constraints* (defined below). Let K be a formal variable (interpreted as ranging over rows, where rows are indexed by $[n]$), let $\mathbf{Y}$ be a tuple of ℓ formal variables (interpreted as placeholders for public input entries), and let $\mathbf{Z}_0, \mathbf{Z}_1$ be tuples of m formal variables each (interpreted as placeholders for witness vectors). As we will see, $\mathbf{Z}_0$ serves as a placeholder for a witness with entries in $\mathbb{Q}^{<d_0, B}[X]$ and $\mathbf{Z}_1$ for witnesses with entries in $\mathbb{F}_{q_i}^{<d_i}[X]$;

the $\mathbb{Q}[X]$-constraint polynomials have coefficients in the intersection localization ring $\mathbb{Z}^{<d_0,B}_{(q_1\cdots q_{|\mathbf{q}|})}[X]$. Additionally, in the finite-field constraints, $\mathbf{Z}_0$ also serves as a placeholder for the projected integer witness $\phi_{q_i}(\mathbf{f}_0)$.

The constraint family $\mathcal{C}$ is a tuple

$$\mathcal{C} = \Big((Q_{0t}, \mathfrak{I}_{0t})_{t\in[|\mathcal{C}|]},\ (Q_{it}, \mathfrak{I}_{it})_{i\in[|\mathbf{q}|],t\in[|\mathcal{C}|]}\Big),$$

where, for every $t \in [|\mathcal{C}|]$ and every $i \in [|\mathbf{q}|]$,

$$Q_{0t} \in \big(\mathbb{Z}^{<d_0,B}_{(q_1\cdots q_{|\mathbf{q}|})}[X]\big)[K, \mathbf{Y}, \mathbf{Z}_0], \qquad Q_{it} \in \big(\mathbb{F}^{<d_i}_{q_i}[X]\big)[K, \mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1],$$

and $\mathfrak{I}_{0t} = (\mathsf{j}_{0t})$ and $\mathfrak{I}_{it} = (\mathsf{j}_{it})$ are ideals given by generators

$$\mathsf{j}_{0t} \in \mathbb{Z}^{<d_0,B}_{(q_1\cdots q_{|\mathbf{q}|})}[X], \qquad \mathsf{j}_{it} \in \mathbb{F}_{q_i}[X], \qquad t \in |\mathcal{C}|,\ i \in |\mathbf{q}|.$$

We call $(Q_{it}, \mathfrak{I}_{it})$ a $\mathbb{Q}[X]$- (or $\mathbb{F}_{q_i}[X]$-) *constraint*.

This restriction on the X-degree of explicit constraint coefficients is harmless for the instances considered here. More general explicit X-coefficients can be accommodated by increasing the relevant d_i, or by the standard coefficient-vector encoding used for polynomial-ring entries.

Scope of bounds. Bounds on the public input and witness are needed to constrain the admissible circuit I/O. The UCS index surfaces some of the bounds that appear in the completeness and soundness analyses of our compiler. Other bounds that factor into those analyses are omitted: the degrees of Q_{it} in subsets of the variables $(\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1)$, and the sparsity of Q_{it} over those subsets, would add unnecessary complexity if surfaced here. Still other bounds—relevant to the downstream finite-field PIOPs that instantiate our compiler, such as the algebraic degree and sparsity of the projected constraint polynomials produced by the Zinc+ PIOR—are not tracked at the UCS level for the same reason. Some bounds are omitted because they do not factor into our analyses at all; for example, $\deg_X(\mathsf{j}_{it})$ for $i \in [|\mathbf{q}|]$ never appears.

Input and witness. The public input is $\mathbb{x} = \mathbf{y}$ with

$$\mathbf{y} \in (\mathbb{Z}^{<d_0,B}_{(q_1\cdots q_{|\mathbf{q}|})}[X])^{\ell},$$

(recall that $\mathbb{Z}_{(q_1\cdots q_{|\mathbf{q}|})}$ denotes the intersection localization ring from Eq. (30)). The witness is $\mathbb{w} = (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|})$ with

$$\mathbf{f}_0 \in \big(\mathbb{Q}^{<d_0,B}[X]\big)^m, \qquad \mathbf{f}_i \in \big(\mathbb{F}^{<d_i}_{q_i}[X]\big)^m \ \text{ for all } \ i \in [|\mathbf{q}|]. \tag{32}$$

Constraint satisfaction. We let $\mathsf{REL}_{\mathsf{UCS}}$ consist of all triples $(\mathbb{i}, \mathbb{x}; \mathbb{w})$ satisfying all the following conditions.

(i) **$\mathbb{Q}[X]$-constraints.** For every row index $k \in [n]$ and every constraint index $t \in [|\mathcal{C}|]$,

$$Q_{0t}(k, \mathbf{y}, \mathbf{f}_0) \in \mathfrak{I}_{0t} \subseteq \mathbb{Q}[X].$$

(ii) **$\mathbb{F}_{q_i}[X]$-constraints.** For every prime power index $i \in [|\mathbf{q}|]$, every row index $k \in [n]$, and every constraint index $t \in [|\mathcal{C}|]$,

$$Q_{it}\big(k, \phi_{q_i}(\mathbf{y}), \phi_{q_i}(\mathbf{f}_0), \mathbf{f}_i\big) \in \mathfrak{I}_{it} \subseteq \mathbb{F}_{q_i}[X].$$

(iii) **Well-definedness of ϕ_{q_i}.** The homomorphisms ϕ_{q_i} from Section 3.1 are defined only on the intersection localization ring $\mathbb{Z}_{(q_1 \cdots q_{|\mathbf{q}|})}$. Since in the previous condition we applied ϕ_{q_i} to $\mathbf{f}_0$, we require the following well-definedness condition:

$$\mathbf{f}_0 \in \left(\mathbb{Z}^{<d_0,B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X]\right)^m. \tag{33}$$

When $|\mathbf{q}| = 0$, this condition is vacuous.

We show in Section 4.2 that the well-definedness condition can be expressed as a $\mathbb{Q}[X]$-constraint, i.e. as a constraint as in Item 1. Hence, (33) does not require its own separate syntax, as long as one of the $\mathbb{Q}[X]$-constraints in the UCS instance enforces it.

Moreover, the Zinc+ PIOR (Section 5.3) enforces well-definedness through constraint (i) of the output $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ instances (Definition 5.8), so including an explicit well-definedness lookup in the UCS index is not required.

In all our applications, condition (33) is subsumed by other lookup constraints placed on $\mathbf{f}_0$, which are written as $\mathbb{Q}[X]$-constraints, cf. Section 4.2. See Remarks 2.1 and 2.5 and Section 2.1.2 for further comments on this.

When an ideal $\mathfrak{I}_{it}$ is zero, the corresponding constraint $Q_{it}(\cdots) \in \mathfrak{I}_{it}$ reduces to the equality $Q_{it}(\cdots) = 0$. When $|\mathbf{q}| = 0$, the tuple $\mathbf{q}$ is empty and UCS reduces to a purely $\mathbb{Q}[X]$-constraint system with no $\mathbb{F}_{q_i}[X]$-constraints. Notice that the elements $Q_{0t}(k, \mathbf{y}, \mathbf{f}_0)$ and $Q_{it}\big(k, \phi_{q_i}(\mathbf{y}), \phi_{q_i}(\mathbf{f}_0), \mathbf{f}_i\big)$ are, in general, elements from $\mathbb{Q}[X]$ and $\mathbb{F}_{q_i}[X]$, respectively, i.e. they are polynomials as opposed to constant elements from $\mathbb{Q}$ or $\mathbb{F}_{q_i}$.

For ease of reference, below we restate the definition of $\mathsf{REL}_{\mathsf{UCS}}$ in compact form.

$$\mathsf{REL}_{\mathsf{UCS}} = \left\{ \begin{pmatrix} \mathbb{i}, \\ \mathbb{x}; \\ \mathbb{w} \end{pmatrix} \;\middle|\; \begin{array}{l} \textbf{Index-input-witness} \\ \quad \textbf{Index} \\ \quad\quad \mathbb{i} = (m, \ell, n, \mathbf{q}, B, \mathbf{d}, \mathcal{C}), \\ \quad\quad \mathbf{q} = (q_1, \ldots, q_{|\mathbf{q}|}), \quad \mathbf{d} = (d_0, \ldots, d_{|\mathbf{q}|}), \\ \quad\quad \mathcal{C} = ((Q_{0t}, \mathfrak{I}_{0t} = (\mathsf{j}_{0t})), (Q_{it}, \mathfrak{I}_{it} = (\mathsf{j}_{it})))_{i \in [|\mathbf{q}|], t \in [|\mathcal{C}|]}, \\ \quad\quad \mathsf{j}_{0t} \in \mathbb{Z}^{<d_0, B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X],\ \mathsf{j}_{it} \in \mathbb{F}_{q_i}[X], \\ \quad\quad Q_{0t} \in (\mathbb{Z}^{<d_0, B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X])[K, \mathbf{Y}, \mathbf{Z}_0],\ Q_{it} \in (\mathbb{F}^{<d_i}_{q_i}[X])[K, \mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1], \\ \quad\quad |\mathbf{Z}_0| = |\mathbf{Z}_1| = m \\ \quad \textbf{Input} \\ \quad\quad \mathbb{x} = (\mathbf{y}) \in (\mathbb{Z}^{<d_0, B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X])^{\ell} \\ \quad \textbf{Witness} \\ \quad\quad \mathbb{w} = (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|}),\ \mathbf{f}_i \in (\mathbb{F}^{<d_i}_{q_i}[X])^m \ \text{ for all } i \in [|\mathbf{q}|], \\ \quad\quad \mathbf{f}_0 \in (\mathbb{Q}^{<d_0, B}[X])^m \\ \\ \textbf{Constraints} \\ \quad \textbf{Algebraic constraints over } \mathbb{Q}[X] \textbf{ mod ideals} \\ \quad\quad Q_{0t}\,(k, \mathbf{y}, \mathbf{f}_0) \in \mathfrak{I}_{0t} \quad \text{for all } k \in [n],\ t \in [|\mathcal{C}|] \\ \quad \textbf{Algebraic constraints over } \mathbb{F}_{q_i}[X] \\ \quad\quad Q_{it}\,(k, \phi_{q_i}(\mathbf{y}), \phi_{q_i}(\mathbf{f}_0), \mathbf{f}_i) \in \mathfrak{I}_{it}, \\ \quad\quad\quad\quad \text{for all } k \in [n],\ i \in [|\mathbf{q}|],\ t \in [|\mathcal{C}|] \\ \quad \textbf{Well-definedness} \\ \quad\quad \mathbf{f}_0 \in (\mathbb{Z}^{<d_0, B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X])^m \end{array} \right\}$$

Design choices. We discuss some of the design choices behind the definition of UCS. We format the discussion as a series of reasonable questions and answers.

Why are witnesses and inputs typed in polynomial rings $R^{<w}[X]$ rather than in R? Why not try to reinterpret $R^{<w}[X]$ as w copies of R and treat each witness entry as an element of R^w, or as w elements of R?

We see at least three advantages in preserving the ring structure of $R[X]$. First, $R[X]$ makes ideal membership constraints natural and efficient to reason about. It is not clear how to translate ideal membership predicates into R^w without resorting to lookups, and even when lookups are available, the translation loses the algebraic structure of the ideals. Our PIOP framework leverages this algebraic structure by resolving ideal membership predicates with a single MLE randomized check, analogously to how standard proof systems check strict equalities.

Second, $R[X]$ enables multiple ideal membership predicates in the same relation. If there were only one ideal $\mathfrak{I}$, one could work over $R[X]/\mathfrak{I}$ directly (in which case we would be working directly over a non-polynomial ring R'). In the presence of membership predicates involving more than one ideal, staying in $R[X]/\mathfrak{I}$ would complicate the arithmetization of all but one of the predicates. Working in $R[X]$ allows us to glue together constraints that would otherwise be separated into

different quotient rings. E.g., in the example from Section 2.1 we have ideal membership constraints simultaneously in $(X - 2)$ and $(X^w - 1)$.

Third, our PIOP projects constraints over $R[X]$ onto a finite field $\mathbb{F}$ via a randomized projection. In practice $\mathbb{F}$ has size comfortably less than 2^{192}. When the original elements from $R[X]$ are large, this projection provides equally large compression benefits, in that elements from $R[X]$, which may be thousands of bits long (for example, in lattice-based settings), become < 192 bits long.

Why allow integers? Why not work entirely over $\mathbb{F}_q$ or $\mathbb{F}_q[X]$ for a sufficiently large prime q?

A plausible alternative to having constraints over $\mathbb{Z}$ or $\mathbb{Z}[X]$ is to work over $\mathbb{F}_q$ or $\mathbb{F}_q[X]$ for q larger than any value that the integer computation ever produces (let's call such a value the *computation norm*), and avoid modular overflow via lookups or range checks (this presence of lookups would not necessarily suppose an overhead, since our example arithmetization also relies on having lookup constraints on the witness).

We see several advantages of staying over $\mathbb{Z}/\mathbb{Z}[X]$, though some use-cases may indeed benefit from working on $\mathbb{F}_q$ or $\mathbb{F}_q[X]$, particularly those where the computation norm is small (e.g. classic hashing). See Section 1.1.3 and the end of Section 2.2 for comments on an instantiation of our framework over $\mathbb{F}_q[X]$, which we call the Zinc+ add-on.

First, by the same reasoning as the second point above, staying in $\mathbb{Z}$ or $\mathbb{Z}[X]$ allows projecting onto fields and rings $\mathbb{F}_q$ or $\mathbb{F}_q[X]$ for different values of q within the same constraint system. For example, our SHA-256 (+ECDSA verification) arithmetization showcases simultaneous constraints modulo 2^{32}, 2, and p for a 256-bit prime p.

Second, the third point above also applies: in applications such as verifiable FHE (e.g. the CKKS scheme) or RSA cryptography, one would require working with a very large prime q (i.e. the computation norm in such applications is large). The random-projection technique may provide significant compression benefits in these cases.

Third, when q is large, one faces the familiar overhead of field arithmetic in $\mathbb{F}_q$ or $\mathbb{F}_q[X]$ throughout the proof. For instance, Reed–Solomon encoding becomes expensive over large fields. Our Integer Pseudo Reed–Solomon (IPRS) codes operate over $\mathbb{Z}$ or $\mathbb{Q}$ and show good practical performance (Section 2.3.3). We think that specifically designed IPRS codes can be set up with good distance over $\mathbb{F}_q$ for a fixed large q, but we lack theoretical guarantees for this; we pose this as an open question in Section 7.3.

We note that combining an IOPP (or PCS) for $\mathbb{Q}^{<d_0,B}[X]$ with a lookup PIOP for $\mathbb{Z}^{<d}[X]$ provides an IOPP (or PCS) for $\mathbb{Z}^{<d_0,B}[X]$, and hence our work provides that.

Definitional extensions and specializations The definition of UCS can be both extended and specialized in various convenient ways, as required by the application or framework where it is used. For example, in Section 8.1 we provide an AIR-like specialization of UCS. In Remarks 2.7, 2.8 and 8.2 to 8.4 we discuss extensions to the definition. Extensions and specializations may be applied simultaneously.

The definition of UCS used in this paper targets the widest amount possible of generality, while deliberately simplifying some syntax (Remarks 2.7 and 8.4), and leaving some functionality out (Remarks 2.8, 8.2 and 8.3).

4.2 Lookup, permutation, R1CS, CCS, and AIR constraints as UCS constraints

In this section and in Section 4.2.2, we show how several common relations and constraints—namely lookups, permutations, R1CS, CCS, and AIR—can be written as UCS constraints over different rings and, perhaps, with ideal membership constraints. We present these constructions in isolation;

however, in practice one can combine multiple constraint types over different domains and with different ideals within a single UCS instance.

4.2.1 Lookup constraints

The construction follows Section 4.4 of [Gar+25; GWHD25], which proves the result for arbitrary integral domains; we include it here for completeness but omit a detailed proof. We work over rings $R[X]$ with $R = \mathbb{Q}$ or $R = \mathbb{F}_q$ for some prime power q.

We define the *lookup relation* over $R[X]$ as:

$$\mathsf{REL}_{\mathsf{Look}} = \left\{ (\mathbb{i}, \mathbb{x}; \mathbb{w}) \;\middle|\; \begin{array}{l} \mathbb{i} = (m, \tau, \mathbf{t}, R[X]), \\ m, \tau \geq 0,\ \mathbf{t} \in R[X]^{\tau}, \\ \mathbb{x} = \emptyset,\ \mathbb{w} = \mathbf{w} \in R[X]^{m}, \\ \mathbf{w}_i \in \{\mathbf{t}_j \mid j \in [\tau]\} \text{ for all } i \in [m]. \end{array} \right\}.$$

Let $\mathbb{i}_{\mathsf{Look}} := (m, \tau, \mathbf{t}, R)$ denote an index for $\mathsf{REL}_{\mathsf{Look}}$. We now construct a UCS index $\mathbb{i}_{\mathsf{UCS}}$ such that $(\mathbb{i}_{\mathsf{Look}}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{Look}}$ if and only if $(\mathbb{i}_{\mathsf{UCS}}, \mathbb{x}'; \mathbb{w}') \in \mathsf{REL}_{\mathsf{UCS}}$ for an explicit transformation $\mathbb{x}', \mathbb{w}'$ of $\mathbb{x}, \mathbb{w}$.

Given $N \geq 1$, the *Lagrange basis polynomial* for $i \in [N]$ is

$$\mathsf{Lag}_{[N],i}(T) := \prod_{j \in [N] \setminus \{i\}} \frac{T - j}{i - j}. \tag{34}$$

This polynomial is well-defined over $\mathbb{Q}$ or any $\mathbb{F}_q$ with $q > N$. Define, given a tuple $\mathbf{Z}$ of m variables and a variable K,

$$L(K, \mathbf{Z}) := \sum_{k \in [m]} \mathbf{Z}_k \cdot \mathsf{Lag}_{[m],k}(K) \tag{35}$$

so that $L(k, \mathbf{z}) = z_k$ for all $k \in [m]$ and any vector $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$. Define

$$Q_{\mathsf{Look}}(K, \mathbf{Z}) = \prod_{j \in [\tau]} \big(L(K, \mathbf{Z}) - \mathbf{t}_j\big). \tag{36}$$

If $R = \mathbb{Q}$, define $\mathbb{i}_{\mathsf{UCS}} = (m, \ell{=}0, n = m, \mathbf{q}{=}\emptyset, B, \mathbf{d}{=}d, \mathcal{C})$ with $\mathcal{C} = (Q_{0,1}, \mathfrak{I}_{0,1}) := (Q_{\mathsf{Look}}, (0))$, where d and B are degree and coefficient bit-size bounds such that $\mathbf{t} \subseteq \mathbb{Q}^{<d,B}[X]$ and $Q_{\mathsf{Look}} \in (\mathbb{Q}^{<d,B}[X])[K, \mathbf{Z}_0]$.

If $R = \mathbb{F}_q$, define $\mathbb{i}_{\mathsf{UCS}} = (m, \ell{=}0, n = m, \mathbf{q}{=}q, B, \mathbf{d}{=}(0, d), \mathcal{C})$ with $\mathcal{C} = ((Q_{0,1}, \mathfrak{I}_{0,1}), (Q_{1,1}, \mathfrak{I}_{1,1}))$ where $Q_{0,1} \equiv 0$, $\mathfrak{I}_{0,1} = (0)$ (i.e. the $\mathbb{Q}[X]$-constraints are void), and $Q_{1,1}(K, \mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1) := Q_{\mathsf{Look}}(K, \mathbf{Z}_1)$, $\mathfrak{I}_{1,1} = (0)$, and d is such that $Q_{\mathsf{Look}} \in (\mathbb{F}_q^{<d}[X])[K, \mathbf{Z}_1]$.

Lemma 4.2 (Lookup as UCS)**.** *Let $\mathbb{i}_{\mathsf{Look}}$ and $\mathbb{i}_{\mathsf{UCS}}$ be defined as above. For every $\mathbf{w} \in R[X]^m$, letting $\mathbb{x} = \emptyset$, we have:*

- *If $R = \mathbb{Q}$ then $(\mathbb{i}_{\mathsf{Look}}, \emptyset; \mathbf{w}) \in \mathsf{REL}_{\mathsf{Look}}$ if and only if $(\mathbb{i}_{\mathsf{UCS}}, \emptyset; \mathbf{w}) \in \mathsf{REL}_{\mathsf{UCS}}$.*
- *If $R = \mathbb{F}_q$ then, writing $\mathbf{0}$ for the all-zero $\mathbb{Q}$-witness component of length m, $(\mathbb{i}_{\mathsf{Look}}, \emptyset; \mathbf{w}) \in \mathsf{REL}_{\mathsf{Look}}$ if and only if $(\mathbb{i}_{\mathsf{UCS}}, \emptyset; (\mathbf{0}, \mathbf{w})) \in \mathsf{REL}_{\mathsf{UCS}}$.*

The proof is analogous to the one in Section 4.4 of [Gar+25; GWHD25] and we omit it. The key observation is that R is an integral domain, and so

$$Q_{\mathsf{Look}}(i, \mathbf{w}) = \prod_{j \in [\tau]} (\mathbf{w}_i - \mathbf{t}_j) = 0$$

if and only if $\mathbf{w}_i \in \{\mathbf{t}_j\}$, for all $i \in [m]$.

Remark 4.3 (Degree blowup and practical lookup arguments)**.** The purpose of the above construction is to exhibit lookups as UCS instances, so that our generic UCS compilers and soundness statements apply to lookup constraints over $R[X]$. In practice, we will not prove a lookup via generic proving methods for polynomial identity (as the lookup constraint has too large a total degree); instead, we will use lookup arguments that exploit the specific semantics of the lookup constraints, following the approach of [Gar+25; GWHD25].

Concretely, we prove lookup constraints by using our Zinc+ compiler steps (see, e.g. Section 2.2), and then using a lookup PIOP. For our applications, a simple sumcheck-based protocol for proving booleanity of elements suffices, cf. Section 9. In general, one can use generic lookup PIOPs such as LogUp [Hab22], Twist&Shout [ST25], etc. over the field $\mathbb{F}'$ used in the projection step of the Zinc+ compiler.

Remark 4.4 (Lookup constraints on polynomial expressions of witness entries)**.** The lookup relation $\mathsf{REL}_{\mathsf{Look}}$ defined above requires each entry of a witness vector $\mathbf{w}$ to belong to a table $\mathbf{t}$. In applications, one sometimes needs to constrain *polynomial expressions* of witness entries to belong to a lookup table. Concretely, given a witness vector $\mathbf{w} \in R^m$ and a polynomial $P \in R[\mathbf{Z}]$ with $|\mathbf{Z}| = m$, one may require that

$$P(\mathbf{w}_i) \ \in \ \{\mathbf{t}_j \mid j \in [\tau]\} \quad \text{for all } i \in [m].$$

The extension to this setting follows by replacing the selector polynomial $L(K, \mathbf{Z})$ in (35) with

$$L'(K, \mathbf{Z}) \ := \ P\Big(\sum_{i \in [m]} \mathbf{Z}_i \cdot \mathsf{Lag}_{[m],i}(K)\Big),$$

and then defining $Q'_{\mathsf{Look}} = \prod_{j \in [\tau]} \big(L'(K, \mathbf{Z}) - \mathbf{t}_j\big)$. analogously. This generalization is needed in our definition of the relation UAIR$^+$ (Definition 8.1) and in some of our applications in Section 8, notably in the SHA-256 arithmetization (Section 8.2); see Remark 8.3 for further discussion.

Remark 4.5 (Well-definedness as a lookup constraint)**.** Recall that Definition 4.1 includes a well-definedness condition (33) requiring $\mathbf{f}_0 \in (\mathbb{Z}^{<d_0,B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X])^m$. This is simply a lookup constraint: the table is $\mathbf{t} := \mathbb{Z}^{<d_0,B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X]$, and each entry of $\mathbf{f}_0$ must belong to this finite set. Thus, the well-definedness condition can be enforced via a $\mathbb{Q}[X]$-constraint in UCS and does not require separate syntax in the definition of UCS.

In the Zinc+ PIOR (Section 5.3), the well-definedness condition is enforced by constraint (i) of the output $\mathsf{PESAT}_{\mathbb{Q}[X]}$ instances (Definition 5.8), so including this lookup in the UCS index is not required.

In many applications, other lookup constraints placed on $\mathbf{f}_0$ already subsume the well-definedness condition (see Section 8), so it does not need to be explicitly included.

4.2.2 Permutation constraints

Define the *permutation relation* over $R[X]$ as

$$\mathsf{REL}_{\mathsf{perm}} \ = \ \left\{ (\mathbb{i}, \mathbb{x}; \mathbb{w}) \ \middle| \ \begin{array}{l} \mathbb{i} = (n, \sigma, R), \ n \geq 1, \ \sigma : [n] \to [n] \text{ is a permutation}, \\ \mathbb{x} = \emptyset, \quad \mathbb{w} = (\mathbf{u}, \mathbf{v}) \in R^n \times R^n, \\ \mathbf{u}_i = \mathbf{v}_{\sigma(i)} \text{ for all } i \in [n]. \end{array} \right\}.$$

To write this relation as a subset of the UCS relation, we view the witness as a length-$2n$ vector $\mathbf{w} = (\mathbf{u}, \mathbf{v})$ and define the selector polynomial

$$Q_{\mathsf{Perm}}(K, \mathbf{Z}) \;:=\; \sum_{i \in [n]} \big(\mathbf{Z}_i - \mathbf{Z}_{n+\sigma(i)}\big) \cdot \mathsf{Lag}_{[n],i}(K).$$

Then evaluating at $K = i \in [n]$ yields $Q_{\mathsf{Perm}}(i, \mathbf{w}) = \mathbf{u}_i - \mathbf{v}_{\sigma(i)}$. Thus, $Q_{\mathsf{Perm}}(i, \mathbf{w}) = 0$ for all $i \in [n]$ if and only if $(\mathbf{u}, \mathbf{v}) \in \mathsf{REL}_{\mathsf{perm}}$. The encoding of $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{Perm}}$ as a UCS index-input-witness triple then proceeds analogously to that in Lemma 4.2: for $R = \mathbb{Q}$, one sets $\mathcal{C} = (Q_{01}, \mathfrak{I}_{01}) = (Q_{\mathsf{Perm}}, (0))$; for $R = \mathbb{F}_q$, one sets $(Q_{01}, \mathfrak{I}_{01}) = (0, (0))$ and $Q_{1,1}(K, \mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1) := Q_{\mathsf{Perm}}(K, \mathbf{Z}_1)$.

4.2.3 R1CS, CCS, and AIR

We refer to Section 4.4 of [Gar+25; GWHD25] for a detailed treatment. Recall that the R1CS relation over a ring R_0 constrains $(M_1 \cdot \mathbf{z}) \circ (M_2 \cdot \mathbf{z}) - (M_3 \cdot \mathbf{z}) = \mathbf{0}$, where $M_1, M_2, M_3 \in R_0^{n \times (\ell+m)}$ are publicly known constraint matrices and $\mathbf{z} = (\mathbf{y}, \mathbf{w}) \in R_0^{\ell} \times R_0^{m}$ is the concatenation of a public input and witness. We capture such a constraint in UCS form as follows: Given a matrix $M \in R_0^{n \times (\ell+m)}$, define the polynomial $L_M(K, \mathbf{Z}) = \sum_{j \in [\ell+m]} \tilde{M}(K, j) \cdot \mathbf{Z}_j$, where $\mathbf{Z}$ is a tuple of $\ell + m$ variables, $\tilde{M}$ is the multilinear extension of M, and we identify $[\ell + m]$ with $\{0,1\}^{\log(\ell+m)}$ (assume $\ell + m$ is a power of 2). Now, define the following polynomial:

$$Q_{\mathsf{R1CS}}(K, \mathbf{Z}) \;:=\; L_{M_1}(K, \mathbf{Z}) \cdot L_{M_2}(K, \mathbf{Z}) - L_{M_3}(K, \mathbf{Z}).$$

Given $k \in [n]$, we have $Q_{\mathsf{R1CS}}(k, \mathbf{z}) = [(M_1 \cdot \mathbf{z}) \circ (M_2 \cdot \mathbf{z}) - (M_3 \cdot \mathbf{z})]_k$. Hence $Q_{\mathsf{R1CS}}(k, \mathbf{z}) = 0$ for all $k \in [n]$ if and only if $\mathbf{z}$ satisfies the initial R1CS constraint (again, we assume k is a power of two and identify $[n]$ with $\{0,1\}^{\log(n)}$). For R1CS constraints over $\mathbb{Q}[X]$, we let the UCS index have $\mathbf{q} = \emptyset$ and set the constraint tuple as $\mathcal{C} = (Q_{\mathsf{R1CS}}, (0))$. For R1CS constraints over $\mathbb{F}_q[X]$, we instead leave the $\mathbb{Q}[X]$-part empty, and use the $\mathbb{F}_q[X]$-constraint as $(Q_{\mathsf{R1CS}}, (0))$.

In many scenarios, it may be useful to use R1CS-ideal membership constraints, rather than strict R1CS equalities. In this case, the constraint may look like

$$(M_1 \cdot \mathbf{z}) \circ (M_2 \cdot \mathbf{z}) - (M_3 \cdot \mathbf{z}) \in \mathfrak{I} \subseteq R[X], \qquad \text{for all } k \in [n],$$

where $R \in \{\mathbb{Q}, \mathbb{F}_q\}$, and $\mathfrak{I}$ is an ideal of $R[X]$. This constraint can be captured in UCS form by using the constraint $(Q_{\mathsf{R1CS}}, \mathfrak{I})$, instead of $(Q_{\mathsf{R1CS}}, (0))$.

CCS and AIR. CCS relations [STW23] admit an analogous transformation to R1CS relations. Further, since AIR relations [Sta21] can be written as CCS relations [STW23], they too can be instantiated as UCS instances. For an AIR-like instantiation tailored to the UCS framework, see Section 8.1.

5 Zinc+: from finite-field PIOPs to a SNARK for UCS

In this section we construct a round-by-round knowledge sound PIOP for UCS from finite-field PIOPs, where the resulting PIOP uses multilinear oracles over $\mathbb{Q}[X]$ and $\mathbb{F}_q[X]$. We then obtain a round-by-round knowledge sound IOP for UCS using Theorem B.2 and IOPPs for multilinear polynomial evaluation over these domains (such as Zip+ in Section 6). Finally, we apply the BCS transformation [BCS16] to obtain a SNARK for UCS.

The construction proceeds in two steps. First, the Zinc+ PIOR reduces UCS to a conjunction of instances of the *polynomial equation satisfiability* relation $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$—one per finite field in the UCS constraints, plus one for the $\mathbb{Q}[X]$-constraints—where $\tilde{q}$ is some power of q. We refer to each index $i \in [0, |\mathbf{q}|]$ as a *field branch*: branch 0 corresponds to the $\mathbb{Q}[X]$-constraints, and branches $i \in [|\mathbf{q}|]$ correspond to the $\mathbb{F}_{q_i}[X]$-constraints. $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ has $\mathbb{F}_{\tilde{q}}$-satisfiability constraints (capturing R1CS, CCS, AIR, and more as special cases), but uses *projected* $\mathbb{F}_q[X]$ and $\mathbb{Q}[X]$ oracles (Definition 5.3). Second, a black-box construction lifts any PIOP for $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$ (the standard finite-field version with only $\mathbb{F}_{\tilde{q}}$ oracles) into a PIOP for $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$. The lift also detects well-definedness violations of the $\mathbb{Q}[X]$ oracle probabilistically at query time, removing the need for explicit $\mathbb{Q}[X]$-constraints enforcing the well-definedness condition. Section 5.1 introduces our techniques for batching ideal membership constraints and projecting them onto finite fields. Section 5.2 defines PESAT and $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ and presents the black-box PIOP lift. Section 5.3 presents the Zinc+ PIOR, together with its completeness and round-by-round knowledge soundness analysis. Section 5.4 states how to set parameters to obtain a SNARK for UCS.

5.1 Batched ideal checks, projected oracles, and UCS well-definedness

In this section, we provide results and definitions which will later be used to reduce ideal membership constraints in $\mathsf{REL}_{\mathsf{UCS}}$ to strict polynomial equalities in $\mathbb{Q}[X]$ and $\mathbb{F}_{q_i}[X]$, and these to polynomial equalities over finite fields. We also formalize the projected polynomial oracle, the oracle type used for every $\mathbb{Q}[X]$- and $\mathbb{F}_q[X]$-oracle in our schemes, and show how the well-definedness condition of UCS is enforced for free at projected-oracle query time. We first introduce the Extend map.

Extend map. Let $\mathbb{P}$ be the set of all prime powers. Throughout the rest of the section, we fix the extend map

$$\mathsf{Extend} : \mathbb{P} \to \mathbb{P}, \quad q \mapsto \tilde{q},$$

that maps each prime power q to a sufficiently large power $\tilde{q} = q^{\ell}$; see Theorem 5.17 for the precise requirements. We write $\tilde{q} = \mathsf{Extend}(q)$ throughout. Throughout, given a prime power $q = p^e$, we write $p_i = \mathrm{char}(\mathbb{F}_{q_i}) = \mathrm{char}(\mathbb{F}_{\tilde{q}_i})$ for the characteristic of the corresponding finite-field branch; equivalently, $q_i = p_i^{e_i}$ and $\tilde{q}_i = p_i^{e_i \ell_i}$.

5.1.1 Ideal batching

We first give our ideal batching lemma, which captures the fact that with high probability all entries in a vector $\mathbf{v}$ belong to an ideal $\mathfrak{I}$ if the evaluation of the multilinear extension of $\mathbf{v}$ at a random point belongs to $\mathfrak{I}$. This is a consequence of the Schwartz-Zippel lemma over rings (Lemma 3.1), which follows by looking at membership in $\mathfrak{I}$ as equality to zero over the quotient ring by $\mathfrak{I}$.

Lemma 5.1 (Testing ideal membership in $\mathbb{K}[X]$ via one random MLE evaluation)**.** *Let $\mathbb{K}$ be a field, and let $\mathfrak{I} \subseteq \mathbb{K}[X]$ be an ideal such that $\mathfrak{I} \cap \mathbb{K} = (0)$. Let $\mu \geq 1$, let $\mathbf{v} : \{0,1\}^{\mu} \to \mathbb{K}[X]$ be a vector indexed by $\{0,1\}^{\mu}$, and let $\widetilde{\mathbf{v}}(\mathbf{Y}) \in (\mathbb{K}[X])[\mathbf{Y}]$ be its multilinear extension (as in (31)). The following hold:*

1. *If $\mathbf{v}(\mathbf{x}) \in \mathfrak{I}$ for all $\mathbf{x} \in \{0,1\}^{\mu}$, then $\widetilde{\mathbf{v}}(\mathbf{r}) \in \mathfrak{I}$ for every $\mathbf{r} \in \mathbb{K}^{\mu}$.*
2. *If there exists $\mathbf{x}_0 \in \{0,1\}^{\mu}$ such that $\mathbf{v}(\mathbf{x}_0) \notin \mathfrak{I}$, then, for every nonempty finite set $S \subseteq \mathbb{K}$,*

$$\Pr_{\mathbf{r} \leftarrow S^{\mu}}\left[\widetilde{\mathbf{v}}(\mathbf{r}) \in \mathfrak{I}\right] \;\leq\; \frac{\mu}{|S|}.$$

The proof is deferred to Appendix A.1.1.

5.1.2 Projecting from rings to finite fields

We now define the evaluation projections that map polynomials over $\mathbb{F}_q[X]$ and $\mathbb{Z}_{(p)}[X]$ onto finite fields, together with their sections (right-inverses). Recall that $\iota_{\tilde{q}} : \mathbb{F}_q \to \mathbb{F}_{\tilde{q}}$ denotes the canonical embedding of $\mathbb{F}_q$ as a subfield of $\mathbb{F}_{\tilde{q}}$, extended coefficient-wise to $\iota_{\tilde{q}} : \mathbb{F}_q[X] \to \mathbb{F}_{\tilde{q}}[X]$. Recall $\alpha \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_q)$ iff α does not lie in any proper subfield of $\mathbb{F}_{\tilde{q}}$ containing $\mathbb{F}_q$.

Definition 5.2 (Evaluation projections and sections)**.** Let $q = p^e$ be a prime power, write $\tilde{q} = q^\ell$, and let $a \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_p)$. Write $\kappa := [\mathbb{F}_{\tilde{q}} : \mathbb{F}_p] = e\ell$.

Projection over $\mathbb{F}_q[X]$. We define the surjective ring homomorphism

$$\psi_{q,a} : \mathbb{F}_q[X] \to \mathbb{F}_{\tilde{q}}, \qquad f(X) \mapsto \iota_{\tilde{q}}(f)(a).$$

A *section* of $\psi_{q,a}$ is any map $\sigma : \mathbb{F}_{\tilde{q}} \to \mathbb{F}_q[X]$ satisfying $\psi_{q,a} \circ \sigma = \mathsf{id}_{\mathbb{F}_{\tilde{q}}}$. We fix the section

$$\psi_{q,a}^{-1} : \mathbb{F}_{\tilde{q}} \to \mathbb{F}_q^{<\ell}[X]$$

defined as follows. Since $a \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_q)$, the set $\{1, a, \ldots, a^{\ell-1}\}$ is an $\mathbb{F}_q$-basis for $\mathbb{F}_{\tilde{q}}$, and every $\alpha \in \mathbb{F}_{\tilde{q}}$ has a unique representation $\alpha = \sum_{k=0}^{\ell-1} c_k \cdot a^k$ with $c_k \in \mathbb{F}_q$. We set $\psi_{q,a}^{-1}(\alpha) := \sum_{k=0}^{\ell-1} c_k X^k$. This is the unique section with image in $\mathbb{F}_q^{<\ell}[X]$. We have $\psi_{q,a} \circ \psi_{q,a}^{-1} = \mathsf{id}_{\mathbb{F}_{\tilde{q}}}$, while $\psi_{q,a}^{-1} \circ \psi_{q,a} = \mathsf{id}$ only on the restricted domain $\mathbb{F}_q^{<\ell}[X] \subseteq \mathbb{F}_q[X]$.

Projection over $\mathbb{Z}_{(p)}[X]$. We extend the projection to polynomials with coefficients in the localization ring $\mathbb{Z}_{(p)} \subset \mathbb{Q}$. Define the surjective ring homomorphism

$$\psi_{p,a} : \mathbb{Z}_{(p)}[X] \to \mathbb{F}_{\tilde{q}}, \qquad f(X) \mapsto \iota_{\tilde{q}}(\phi_p(f))(a),$$

where $\phi_p : \mathbb{Z}_{(p)}[X] \to \mathbb{F}_p[X]$ reduces coefficients modulo p (mapping $a/b \in \mathbb{Z}_{(p)}$ to $a \cdot b^{-1} \bmod p \in \mathbb{F}_p$). This map depends only on the characteristic p and the generator a, not on the prime power q: for any $q_j = p^{e_j}$, the maps $\psi_{q_j,a}$ and $\psi_{p,a}$ coincide on $\mathbb{Z}_{(p)}[X]$, since $\phi_{q_j}|_{\mathbb{Z}_{(p)}} = \iota_{q_j} \circ \phi_p$ and $\iota_{\tilde{q}} \circ \iota_{q_j} = \iota_{\tilde{q}}$.

We use the same notation ψ^{-1} for a section of $\psi_{p,a}$. We fix the section

$$\psi_{p,a}^{-1} : \mathbb{F}_{\tilde{q}} \to \mathbb{Z}_{(p)}^{<\kappa}[X]$$

defined as follows. If $a \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_p)$, the set $\{1, a, \ldots, a^{\kappa-1}\}$ is an $\mathbb{F}_p$-basis for $\mathbb{F}_{\tilde{q}}$. Every $\alpha \in \mathbb{F}_{\tilde{q}}$ has a unique representation $\alpha = \sum_{k=0}^{\kappa-1} \alpha_k \, a^k$ with $\alpha_k \in \mathbb{F}_p$. For each k, choose a representative $c_k \in \mathbb{Z}$ satisfying $\phi_p(c_k) = \alpha_k$. Set $\psi_{p,a}^{-1}(\alpha) := \sum_{k=0}^{\kappa-1} c_k \, X^k$. Any choice of representatives satisfying $\phi_p(c_k) = \alpha_k$ yields a valid section; for concreteness, we use the centered representative $c_k = \mathsf{ctr}_p(\alpha_k) \in \{-(p-1)/2, \ldots, (p-1)/2\}$. We have $\psi_{p,a} \circ \psi_{p,a}^{-1} = \mathsf{id}_{\mathbb{F}_{\tilde{q}}}$. Neither $\psi_{q,a}^{-1}$ nor $\psi_{p,a}^{-1}$ is a ring homomorphism.

Note that primitivity over $\mathbb{F}_p$ is strictly stronger than primitivity over $\mathbb{F}_q$ when $e > 1$; for prime q they coincide. Our protocol requires both sections $\psi_{q,a}^{-1}$ and $\psi_{p,a}^{-1}$ to be well-defined for the same a, which is captured by $a \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_p)$. While for definitional clarity we define $\psi_{q,a}^{-1}$ and $\psi_{p,a}^{-1}$ separately above, we sometimes write $\psi_{q,a}^{-1}$ for both; the specific case should always be clear from context.

Definition 5.3 (Projected polynomial oracle)**.** For a ring $K \in \{\mathbb{Q}, \mathbb{F}_{q'}\}$ (with q' a prime power), an X-degree bound $d \in \mathbb{N}$, and (when $K = \mathbb{Q}$) a bit-length bound $B \in \mathbb{N}$, a *projected polynomial oracle* $[[\tilde{f}]]$ is an oracle to a multilinear polynomial $\tilde{f}$ in variables W with $|W| = \nu$. The oracle accepts queries (u, q, a) where q is a prime power (and $q = q'$ in the $K = \mathbb{F}_{q'}$ case), $\tilde{q} = \mathsf{Extend}(q)$, $u \in \mathbb{F}_{\tilde{q}}^\nu$, and $a \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_p)$ with $p = \mathrm{char}(q)$. Following Definition 5.2, the projection ψ has target $\mathbb{F}_{\tilde{q}}$ in both cases, with ψ^{-1} a corresponding section. The polynomial type of $\tilde{f}$, the source ring and admissible parameterizations of ψ, and the response semantics depend on K:

- $K = \mathbb{F}_{q'}$: We have $\tilde{f} \in (\mathbb{F}_{q'}^{<d}[X])[W]$ and $\psi_{q',a} : \mathbb{F}_{q'}[X] \to \mathbb{F}_{\tilde{q}'}$. The response to (u, q', a) is $\psi_{q',a}(\tilde{f}(\psi_{q',a}^{-1}(u))) \in \mathbb{F}_{\tilde{q}'}$.
- $K = \mathbb{Q}$: We have $\tilde{f} \in (\mathbb{Q}^{<d,B}[X])[W]$ and $\psi_{p,a} : \mathbb{Z}_{(p)}[X] \to \mathbb{F}_{\tilde{q}}$. The response to (u, q, a) is $\psi_{p,a}(\tilde{f}(\psi_{p,a}^{-1}(u))) \in \mathbb{F}_{\tilde{q}}$ when $\tilde{f}(\psi_{p,a}^{-1}(u)) \in \mathbb{Z}_{(p)}[X]$, and $\perp$ otherwise.

Lemma 5.4. *Let $\psi : \mathcal{R} \to \mathcal{R}'$ be a surjective ring homomorphism. Let $n = 2^\mu$ and let $\mathbf{v} \in \mathcal{R}^n$ be a vector and $\tilde{\mathbf{v}}$ its multilinear extension. Let $\widetilde{\psi(\mathbf{v})}$ be the multilinear extension of $\psi(\mathbf{v}) \in \mathcal{R}'^n$. Then, for any section $\psi^{-1} : \mathcal{R}' \to \mathcal{R}$ (i.e. any map such that $\psi \circ \psi^{-1} = \mathsf{id}_{\mathcal{R}'}$),*

$$\psi(\tilde{\mathbf{v}}(\psi^{-1}(\mathbf{u}))) = \widetilde{\psi(\mathbf{v})}(\mathbf{u}) \qquad \text{for all } \mathbf{u} \in \mathcal{R}'^\mu.$$

The proof is deferred to Appendix A.1.1.

5.1.3 Enforcing UCS well-definedness

The projection $\psi_{p,a} : \mathbb{Z}_{(p)}[X] \to \mathbb{F}_{\tilde{q}}$ is defined only on the localization ring $\mathbb{Z}_{(p)}[X] \subset \mathbb{Q}[X]$. A projected $\mathbb{Q}[X]$-oracle (Definition 5.3) for a multilinear $\tilde{f}$ over $\mathbb{Q}[X]$ may therefore return $\perp$ on some queries. The following theorem shows that for a uniformly random query, $\perp$ is returned with overwhelming probability whenever some entry of $\tilde{f}$'s underlying vector escapes $\mathbb{Z}_{(p)}[X]$. The Zinc+ PIOP (Section 5.2) uses this to enforce the UCS well-definedness condition without an explicit constraint, by translating $\perp$ responses into verifier rejection.

Theorem 5.5 (Well-definedness detection via random evaluation)**.** *Let p be prime and $\kappa, \nu \geq 1$. Let $S \subseteq \mathbb{Z}_{(p)}$ with $|S| = p$ and $\phi_p(S) = \mathbb{F}_p$. Let $\tilde{\mathbf{f}} \in (\mathbb{Q}[X])[\mathbf{W}]$, $|\mathbf{W}| = \nu$, be multilinear with $\tilde{\mathbf{f}} \notin (\mathbb{Z}_{(p)}[X])[\mathbf{W}]$. Sample $z_{i,k} \leftarrow S$ independently for $i \in [\nu]$, $k \in \{0, \ldots, \kappa - 1\}$, and set $z_i := \sum_{k=0}^{\kappa-1} z_{i,k} X^k \in \mathbb{Z}_{(p)}^{<\kappa}[X]$. Then*

$$\Pr\big[\tilde{\mathbf{f}}(z_1, \ldots, z_\nu) \in \mathbb{Z}_{(p)}[X]\big] \;\leq\; \nu/p^\kappa.$$

The proof is deferred to Appendix A.1.1.

Remark 5.6. In our schemes, projected $\mathbb{Q}[X]$-oracles are queried at points u uniformly sampled from $\mathbb{F}_{\tilde{q}}^\nu$. Setting $z_i := \psi_{p,a}^{-1}(u_i)$ produces $\mathbb{F}_p$-basis coefficients independent and uniform over $\mathbb{F}_p$, so Theorem 5.5 bounds the probability that a query whose underlying witness escapes $\mathbb{Z}_{(p)}[X]$ returns a non-$\perp$ response by ν/p^κ.

5.2 The polynomial satisfiability relation with projected oracles

We begin this section by introducing two satisfiability relations. The *polynomial equation satisfiability relation* $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$ has both constraints and oracles over the finite field $\mathbb{F}_{\tilde{q}}$. The relation $\mathsf{PESAT}_{\mathbb{Q}[X]}(\mathbb{F}_{\tilde{q}})$ has constraints over $\mathbb{F}_{\tilde{q}}$, but *projected* oracles over $\mathbb{Q}[X]$ and $\mathbb{F}_q[X]$, for some q such that $\tilde{q} = \mathsf{Extend}(q)$. We present a black-box PIOP construction of the latter from any PIOP for the former. In Section 5.4 we combine this with our Zinc+ PIOR, which reduces UCS to a conjunction of $\mathsf{PESAT}_{\mathbb{Q}[X]}$ instances over different $\mathbb{F}_{\tilde{q}_i}$, to obtain a PIOP for UCS from PIOPs for $\mathsf{PESAT}_{\mathbb{Q}[X]}$.

We note our formulation of PESAT is essentially equivalent to the one given in [BCFW25] with a split witness oracle and no degree bound on the constraint polynomials. The PESAT relation naturally captures AIR, Plonkish, R1CS, CCS, and more.

Definition 5.7 ($\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$)**.** Let $\tilde{q}$ be a prime power. $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$ consists of all triples $(\mathbb{i}, \mathbb{x}; \mathbb{w})$ of the following form.

$$\mathsf{PESAT}(\mathbb{F}_{\tilde{q}}) = \left\{ \begin{pmatrix} \mathbb{i}, \\ \mathbb{x}; \\ \mathbb{w} \end{pmatrix} \;\middle|\; \begin{array}{l} \textbf{Index} \\ \quad \mathbb{i} = \big(m = 2^{\nu},\, \ell,\, \mathcal{C}\big),\;\; \mathcal{C} = [Q_t]_{t \in [|\mathcal{C}|]}, \\ \quad Q_t \in \mathbb{F}_{\tilde{q}}[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}1],\;\; |\mathbf{Y}| = \ell,\;\; |\mathbf{Z}_0| = |\mathbf{Z}_1| = m, \\ \textbf{Input} \\ \quad \mathbb{x} = (\mathbf{y},\, [[\widetilde{\mathbf{f}}_0]],\, [[\widetilde{\mathbf{f}}_1]]),\;\; \mathbf{y} \in \mathbb{F}_{\tilde{q}}^{\ell},\;\; \widetilde{\mathbf{f}}_0,\, \widetilde{\mathbf{f}}_1 \in \mathbb{F}_{\tilde{q}}[\mathbf{W}],\;\; |\mathbf{W}| = \nu \\ \textbf{Witness} \\ \quad \mathbb{w} = (\mathbf{f}_0,\, \mathbf{f}_1) \in \mathbb{F}_{\tilde{q}}^{m} \times \mathbb{F}_{\tilde{q}}^{m} \\ \textbf{Constraints} \\ \quad Q_t(\mathbf{y},\, \mathbf{f}_0,\, \mathbf{f}_1) = 0 \;\text{ for all } t \in [|\mathcal{C}|] \end{array} \right\}.$$

Definition 5.8 ($\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$)**.** Let q be a power of a prime p such that $\tilde{q} = \mathsf{Extend}(q)$, and let $a \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}}/\mathbb{F}_p)$. Then:

$$\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}}) = \left\{ \begin{pmatrix} \mathbb{i}, \\ \mathbb{x}; \\ \mathbb{w} \end{pmatrix} \;\middle|\; \begin{array}{l} \textbf{Index} \\ \quad \mathbb{i} = \big(m = 2^{\nu},\, \ell,\, a,\, d_0,\, B,\, q,\, d_1,\, \mathcal{C}\big), \\ \quad \mathcal{C} = [Q_t]_{t \in [|\mathcal{C}|]},\;\; Q_t \in \mathbb{F}_{\tilde{q}}[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1], \\ \textbf{Input} \\ \quad \mathbb{x} = (\mathbf{y},\, [[\widetilde{\mathbf{f}}_0]],\, [[\widetilde{\mathbf{f}}_1]]),\;\; \mathbf{y} \in \mathbb{F}_{\tilde{q}}^{\ell}, \\ \quad \widetilde{\mathbf{f}}_0 \in (\mathbb{Q}^{<d_0,B}[X])[\mathbf{W}],\;\; \widetilde{\mathbf{f}}_1 \in (\mathbb{F}_q^{<d_1}[X])[\mathbf{W}],\;\; |\mathbf{W}| = \nu \\ \textbf{Witness} \\ \quad \mathbb{w} = (\mathbf{f}_0,\, \mathbf{f}_1),\;\; \mathbf{f}_0 \in (\mathbb{Q}^{<d_0,B}[X])^m,\;\; \mathbf{f}_1 \in (\mathbb{F}_q^{<d_1}[X])^m \\ \textbf{Constraints} \\ \quad \text{(i)}\;\; \mathbf{f}_0 \in (\mathbb{Z}_{(p)}[X])^m \\ \quad \text{(ii)}\;\; Q_t(\mathbf{y},\, \psi_{p,a}(\mathbf{f}_0),\, \psi_{q,a}(\mathbf{f}_1)) = 0 \;\text{ for all } t \in [|\mathcal{C}|] \end{array} \right\}.$$

Constraint polynomial degree and sparsity. While for simplicity we don't bound the degree of the constraint polynomials or their sparsity above, many PIOP constructions of interest for PESAT have prover, verifier, and proof-size complexities that scale with $D'_t := \deg(Q'_t)$ or $s'_t := \|Q'_t\|_0$. To that end, we later show the constraint polynomials Q'_t output by the Zinc+ PIOR inherit sparsity and algebraic-degree bounds from the originating UCS constraint polynomials Q_{it}. Specifically, we prove $\deg(Q'_t) \le \deg_{\mathbf{Y},\mathbf{Z}_0,\mathbf{Z}_1}(Q_{it})$ (Lemma 5.15), and $\|Q'_t\|_0$ is at most one plus the number of distinct $(\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1)$-monomials appearing in the support of Q_{it} (Lemma 5.16). Both inequalities are upper bounds; coefficient cancellations after the projection or row summation may make actual values smaller.

5.2.1 Lifting standard finite-field PIOPs to support projected oracles

In the Zinc+ PIOR (Section 5.3), one instance of $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ is produced per field branch. The construction below turns any PIOP for $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$ into a PIOP for $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$ with no internal modification: the wrapping verifier forwards each query and response between the inner verifier and the projected oracles unchanged, except that a $\perp$ response triggers immediate rejection.

Construction 5.9 (PIOP Π' for $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$ from PIOP Π for $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$)**.** Let $\Pi = (\mathsf{P}_\Pi, \mathsf{V}_\Pi)$ be a PIOP for $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$. Write $\psi_0 := \psi_{p,a}$ and $\psi_1 := \psi_{q,a}$.

Derived instance. Given $(\mathbb{i}, \mathbb{x})$ for $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$ with $\mathbb{x} = (\mathbf{y}, [[\widetilde{\mathbf{f}_0}]], [[\widetilde{\mathbf{f}_1}]])$, let $(\bar{\mathbb{i}}, \bar{\mathbb{x}})$ be the $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$ instance with $\bar{\mathbb{i}} := (m, \ell, \mathcal{C})$ and $\bar{\mathbb{x}} := (\mathbf{y}, [[\widetilde{\mathbf{f}_0}]], [[\widetilde{\mathbf{f}_1}]])$, reusing the same oracle handles, with the constraint list $\mathcal{C}$ unchanged. To V_Π the oracles in $\bar{\mathbb{x}}$ are presented as $\mathbb{F}_{\tilde{q}}$-polynomial oracles, with V' mediating their responses as below.

Prover. P' runs P_Π on $(\bar{\mathbb{i}}, \bar{\mathbb{x}}; (\psi_0(\mathbf{f}_0), \psi_1(\mathbf{f}_1)))$ and forwards its messages.

Verifier. V' runs V_Π, acting as a thin wrapper on its oracle queries: when V_Π queries $[[\widetilde{\mathbf{f}_j}]]$ at $\mathbf{u} \in \mathbb{F}_{\tilde{q}}^\nu$, V' forwards the query $(\mathbf{u}, q, a)$ to the projected oracle and receives a response $\alpha \in \mathbb{F}_{\tilde{q}} \cup \{\perp\}$. If $\alpha = \perp$, V' rejects; otherwise V' returns α to V_Π. (Since $\psi_{q,a}$ is total on $\mathbb{F}_q[X]$, $\perp$ can occur only for $j = 0$.) V' outputs whatever V_Π outputs.

When constraint (i) of $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ holds, Lemma 5.4 gives that on every query $\mathbf{u}$ the responses V' forwards to V_Π equal $\widetilde{\psi_0(\mathbf{f}_0)}(\mathbf{u})$ and $\widetilde{\psi_1(\mathbf{f}_1)}(\mathbf{u})$ respectively, i.e., V' presents V_Π with exactly the $\mathbb{F}_{\tilde{q}}$-polynomial oracles for $\widetilde{\psi_0(\mathbf{f}_0)}$ and $\widetilde{\psi_1(\mathbf{f}_1)}$.

Remark 5.10 (Single uniform query assumption)**.** For simplicity, in the theorem below we assume Π makes a single uniformly random query to $[[\widetilde{\mathbf{f}_0}]]$ in some round $i^\star$. This is true of all sumcheck-based finite-field PIOPs we are aware of. If Π makes multiple queries including at least one uniformly random query, standard batching reduces them to a single uniformly random query.

Theorem 5.11. *Let Π be a k-round PIOP for $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$ with completeness error ϵ and round-by-round knowledge soundness errors $(\kappa_1, \ldots, \kappa_k)$ (Definition 3.9). Suppose Π makes a single uniformly random query to $[[\widetilde{\mathbf{f}_0}]]$ in round $i^\star$. Then Π' (Construction 5.9) is a PIOP for $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$ with the same round and query complexity, completeness error ϵ, and RBR knowledge soundness errors*

$$\kappa'_i = \begin{cases} \max(\kappa_i,\ \nu/|\mathbb{F}_{\tilde{q}}|) & \text{if } i = i^\star, \\ \kappa_i & \text{otherwise.} \end{cases}$$

The proof is deferred to Appendix A.1.2.

5.3 The Zinc+ PIOR

The Zinc+ PIOR reduces $\mathsf{REL}_{\mathsf{UCS}}$ (Definition 4.1) to a conjunction of $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ instances $\mathcal{R}'$, one per field branch, by projecting ideal membership constraints over rings to strict equalities over finite fields. We index field branches by $i \in [0, |\mathbf{q}|]$. Branch 0 corresponds to the random prime branch F_{q_0}, where q_0 is sampled randomly from prime set $\mathcal{P}$, and $\tilde{q}_0 := q_0$. This is where our $Q[X]$-constraints are projected. Branches $i \in [|\mathbf{q}|]$ correspond to the fixed finite-field branches $\mathbb{F}_{\tilde{q}_i}$, where $\tilde{q}_i := \mathsf{Extend}(q_i)$ (see Section 5.1), and are where our $\mathbb{F}_q[X]$-constraints are projected. The construction proceeds in three steps per branch (following the pipeline of Section 2.2):

1. **Projection to rings $\mathbb{F}_{\tilde{q}_i}[X]$.** Rational coefficients are projected via $\phi_{\tilde{q}_i} := \iota_{\tilde{q}_i} \circ \phi_{q_i}$; finite-field coefficients are embedded via $\iota_{\tilde{q}_i}$. For branch 0, if q_0 divides any denominator, the verifier rejects; for other branches this is not a concern as all explicit elements of the index-instance are type-constrained to $\mathbb{Z}_{(\mathbf{q})}$.

2. **Batching ideal constraints to strict equalities.** For each constraint, the prover replaces the 2^μ row-wise ideal-membership checks $Q_{it}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0, \mathbf{f}_i) \in \langle \mathsf{j}_{it} \rangle$ for $\mathbf{b} \in \{0,1\}^\mu$ with the strict equality $e_{it} = \sum_{\mathbf{b} \in \{0,1\}^\mu} Q_{it}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0, \mathbf{f}_i) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i)$ at a random $\mathbf{r}_i$, where e_{it} is a claimed prover evaluation. Ideal membership of e_{it} then implies, with high probability over $\mathbf{r}_i$, ideal membership of every row.
3. **Evaluation projection to $\mathbb{F}_{\tilde{q}_i}$.** The verifier samples a primitive element a_i and applies ψ_{q_i, a_i} to project the constraint polynomials and explicit instance $\mathbf{y}$ onto $\mathbb{F}_{\tilde{q}_i}$. The strict equality claims over $\mathbb{F}_{\tilde{q}_i}[X]$ from the previous step reduce to equalities in $\mathbb{F}_{\tilde{q}_i}$ that hold at a_i only if (with high probability over a_i) they hold over $\mathbb{F}_{\tilde{q}_i}[X]$.

Definition 5.12 (Zinc+ PIOR target relation)**.** $\mathcal{R}'$ consists of all triples $(\mathbb{i}', \mathbb{x}'; \mathbb{w}')$ of the following form.

$$\mathcal{R}' = \left\{ \begin{pmatrix} \mathbb{i}', \\ \mathbb{x}'; \\ \mathbb{w}' \end{pmatrix} \;\middle|\; \begin{array}{l} \textbf{Index} \\ \mathbb{i}' = (m, \ell, n, q_0, \mathbf{q}, \mathbf{a}, B, \mathbf{d}, \mathbf{Q}'), \\ m, \ell, n, B, \nu \in \mathbb{N},\ q_0 \in \mathcal{P}, \\ \text{For } i \in [0, |\mathbf{q}|]: \\ \quad q_i \text{ a prime-power},\ a_i \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i}),\ d_i \in \mathbb{N}, \\ \quad Q'_{it} \in \mathbb{F}_{\tilde{q}_i}[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1] \text{ for } t \in [|\mathcal{C}|],\ |\mathbf{Y}| = \ell,\ |\mathbf{Z}_0| = |\mathbf{Z}_1| = m \\ \textbf{Input} \\ \mathbb{x}' = ((y'_i)_{i \in [0, |\mathbf{q}|]}, [[\widetilde{\mathbf{f}}_0]], ([[\widetilde{\mathbf{f}}_i]])_{i \in [|\mathbf{q}|]}), \\ y'_i \in \mathbb{F}^\ell_{\tilde{q}_i},\ \widetilde{\mathbf{f}}_0 \in (\mathbb{Q}^{<d_0, B}[X])[\mathbf{W}],\ \widetilde{\mathbf{f}}_i \in (\mathbb{F}^{<d_i}_{q_i}[X])[\mathbf{W}],\ |\mathbf{W}| = \nu \\ \textbf{Witness} \\ \mathbb{w}' = (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|}), \\ \mathbf{f}_0 \in (\mathbb{Q}^{<d_0, B}[X])^m,\ \mathbf{f}_i \in (\mathbb{F}^{<d_i}_{q_i}[X])^m \text{ for } i \in [|\mathbf{q}|] \\ \textbf{Constraints} \\ \text{For } i \in [0, |\mathbf{q}|]: \\ \quad ((m, \ell, a_i, d_0, B, q_i, d_i, [Q'_{it}]_{t \in [|\mathcal{C}|]}),\ (y'_i, [[\widetilde{\mathbf{f}}_0]], [[\widetilde{\mathbf{f}}_i]]),\ (\mathbf{f}_0, \mathbf{f}_i)) \in \mathsf{PESAT}_{\mathbb{Q}[X]}(\mathbb{F}_{\tilde{q}_i}) \end{array} \right\}$$

Conjuncting constraint (i) of $\mathsf{PESAT}_{\mathbb{Q}[X]}(\mathbb{F}_{\tilde{q}_i})$ (Definition 5.8) across branches recovers the well-definedness condition of $\mathsf{REL}_{\mathsf{UCS}}$ (Definition 4.1, constraint (iii)). The reject symbol $\perp_{\mathsf{UCS}} := \perp_{\mathsf{UCS}}$ is distinct from $\mathcal{R}_{\mathsf{triv}}$'s accept symbol $\perp$; $\mathsf{V}_{\mathsf{UCS}}$ outputs the pair $(\perp_{\mathsf{UCS}}, \perp_{\mathsf{UCS}}) \notin \mathsf{LANG}(\mathcal{R}')$ on rejection.

Algorithm 1 Protocol $\Pi^{\mathsf{UCS}} = (\mathsf{P}_{\mathsf{UCS}}, \mathsf{V}_{\mathsf{UCS}})$: Zinc+ PIOR from $\mathsf{REL}_{\mathsf{UCS}}$ to $\mathcal{R}'$

Input: An index and input $(\mathbb{i}, \mathbb{x})$ for $\mathsf{REL}_{\mathsf{UCS}}$ (Definition 4.1). $\mathsf{P}_{\mathsf{UCS}}$ additionally receives a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{UCS}}$. Write

$$\begin{aligned} \mathbb{i} &= (m = 2^\nu,\ \ell,\ n = 2^\mu,\ \mathbf{q},\ B,\ \mathbf{d},\ \mathcal{C} = [(Q_{it}, \langle \mathsf{j}_{it} \rangle)]_{i \in [0, |\mathbf{q}|],\, t \in [|\mathcal{C}|]}), \\ \mathbb{x} &= (\mathbf{y}) \in (\mathbb{Z}^{<d_0, B}_{(\mathbf{q})}[X])^\ell, \\ \mathbb{w} &= (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|}), \quad \mathbf{f}_0 \in (\mathbb{Q}^{<d_0, B}[X])^m, \quad \mathbf{f}_i \in (\mathbb{F}^{<d_i}_{q_i}[X])^m \text{ for } i \in [|\mathbf{q}|]. \end{aligned}$$

1: $\mathsf{P}_{\mathsf{UCS}}$ sends polynomial oracles $[[\widetilde{\mathbf{f}}_0]], [[\widetilde{\mathbf{f}}_1]], \ldots, [[\widetilde{\mathbf{f}}_{|\mathbf{q}|}]]$, where $\widetilde{\mathbf{f}}_0 \in (\mathbb{Q}^{<d_0, B}[X])[\mathbf{W}]$ is the MLE of $\mathbf{f}_0$ and $\widetilde{\mathbf{f}}_i \in (\mathbb{F}^{<d_i}_{q_i}[X])[\mathbf{W}]$ is the MLE of $\mathbf{f}_i$, with $|\mathbf{W}| = \nu$.

2: $\mathsf{V}_{\mathsf{UCS}}$ samples $q_0 \leftarrow \mathcal{P}$ and $\mathbf{r}_i \leftarrow \mathbb{F}^\mu_{\tilde{q}_i}$ for $i \in [0, |\mathbf{q}|]$. If Q_{0t}, j_{0t}, or $\mathbf{y}$ have entries outside $\mathbb{Z}_{(q_0)}[X]$ for any $t \in [|\mathcal{C}|]$, $\mathsf{V}_{\mathsf{UCS}}$ rejects. % When $\mathbf{f}_0 \notin (\mathbb{Z}_{(q_0)}[X])^m$, the honest prover cannot compute e_{0t} at step 3 in a way that lies in $\langle \phi_{q_0}(\mathsf{j}_{0t}) \rangle$, leading to verifier rejection at step 5.

3: $\mathsf{P_{UCS}}$ sends, for each $i \in [0, |\mathbf{q}|]$ and $t \in [|\mathcal{C}|]$,

$$\begin{aligned} e_{0t} &:= \sum_{\mathbf{b}\in\{0,1\}^\mu} \phi_{q_0}(Q_{0t}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_0) && \in \mathbb{F}_{q_0}^{<\delta_{0t}}[X], \\ e_{it} &:= \sum_{\mathbf{b}\in\{0,1\}^\mu} \iota_{\tilde{q}_i}(Q_{it})(\mathbf{b},\, \phi_{\tilde{q}_i}(\mathbf{y}),\, \phi_{\tilde{q}_i}(\mathbf{f}_0),\, \iota_{\tilde{q}_i}(\mathbf{f}_i)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i) && \in \mathbb{F}_{\tilde{q}_i}^{<\delta_{it}}[X], \end{aligned}$$

where the degree bounds δ_{it} are as in Eqs. (37) and (38).

4: **for** $i \in [0, |\mathbf{q}|]$ and $t \in [|\mathcal{C}|]$ **do**

5: $\mathsf{V_{UCS}}$ rejects if $e_{0t} \notin \langle \phi_{q_0}(\mathsf{j}_{0t}) \rangle$ ($i = 0$ case), or $e_{it} \notin \langle \iota_{\tilde{q}_i}(\mathsf{j}_{it}) \rangle$ ($i \in [|\mathbf{q}|]$ case).

6: $\mathsf{V_{UCS}}$ samples $a_i \leftarrow \mathbb{F}_{\tilde{q}_i}$, rejecting if $a_i \notin \mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i})$.

7: $\mathsf{V_{UCS}}$ computes

$$Q'_{it} = \sum_{\mathbf{b}\in\{0,1\}^\mu} \psi_{q_i,a_i}(Q_{it})(\mathbf{b}, \mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i) - e_{it}(a_i) \in \mathbb{F}_{\tilde{q}_i}[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1].$$

8: $\mathsf{V_{UCS}}$ outputs $\mathbb{i}'_i = (m,\, \ell,\, a_i,\, d_0,\, B,\, q_i,\, d_i,\, [Q'_{it}]_{t\in[|\mathcal{C}|]})$ and $\mathbb{x}'_i = (\psi_{q_i,a_i}(\mathbf{y}),\, [[\widetilde{\mathbf{f}_0}]],\, [[\widetilde{\mathbf{f}_i}]])$.

9: **end for**

10: $\mathsf{P_{UCS}}$ outputs $\mathbb{w}' = \mathbb{w}$.

5.3.1 Completeness, round-by-round knowledge soundness, and efficiency

We first present our round-by-round knowledge theorem for Zinc+. We then present our completeness bound as a corollary in terms of the security parameter λ, i.e. we show the requirements imposed on the prime set $\mathcal{P}$ and choice of $\tilde{q}_i$ to achieve per-round knowledge error $2^{-\lambda}$ imply a certain completeness error dependent on λ. For most instantiations, simply setting λ as desired for security will provide a sufficiently small completeness error on the order of $2^{-\lambda/2}$. For certain relations the completeness error vanishes (e.g., integral with constraints over fields of large characteristic); for others rejection sampling can bound the error closer to $2^{-\lambda}$ (see discussion below for tradeoffs).

We additionally present two lemmas related to the efficiency of our reduction, showing the degree and sparsity of the projected constraint polynomials are upper bounded by the original constraint polynomials. This is important because many PIOP constructions of interest for PESAT have prover, verifier, and proof-size complexities that scale with $D'_t := \deg(Q'_t)$ or $s'_t := \|Q'_t\|_0$.

For soundness, we must enforce strict upper bounds on the X-degrees of the evaluated constraint polynomials. For any $\mathbf{b} \in \{0,1\}^\mu$, $\mathbf{y} \in (\mathbb{F}^{<d_0}[X])^\ell$, $\mathbf{f}_0 \in (\mathbb{F}^{<d_0}[X])^m$, and $\mathbf{f}_i \in (\mathbb{F}^{<d_i}[X])^m$, we define

$$\deg_X Q_{0t}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0) \;<\; \delta_{0t} := d_0 + (d_0 - 1)\deg_{\mathbf{Y},\mathbf{Z}_0}(Q_0) \tag{37}$$

$$\begin{aligned} \deg_X Q_{it}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0, \mathbf{f}_i) \;<\; \delta_{it} &:= d_i + \max_{M\in\mathrm{supp}(Q_i)} \big[(d_0 - 1)\,\deg_{\mathbf{Y},\mathbf{Z}_0}(M) + (d_i - 1)\,\deg_{\mathbf{Z}_1}(M)\big] \\ &\le d_i\big(\deg_{\mathbf{Z}_1}(Q_i) + 1\big) + d_0 \deg_{\mathbf{Y},\mathbf{Z}_0}(Q_i). \end{aligned} \tag{38}$$

While not enforced in Algorithm 1, our soundness analysis also requires a strict upper bound B_{0t}^{eval} on the bitlength of the rational coefficients of an evaluation of $Q_{0t}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0)$. In Lemma A.1 we show

$$B_{0t}^{\mathrm{eval}} \le B + \deg_{\mathbf{Y},\mathbf{Z}_0}(Q_{0t}) \cdot (B + \log d_0) + \log \|Q_{0t}\|_0 + 1,$$

noting this bound is enforced by the UCS definition and polynomial oracle typing.

Theorem 5.13 (Round-by-round knowledge soundness of the Zinc+ PIOR)**.** *Assume* $\lambda \geq 80$. *Suppose for all* $q_0 \in \mathcal{P}$ *and all* $i \in [0, |\mathbf{q}|]$*:*

$$|\mathcal{P}| \geq 2^{\lambda} \cdot \max_{t\in[|\mathcal{C}|]}\left(\delta_{0t}^2 \cdot B_{0t}^{\text{eval}}\right),$$
$$\tilde{q}_i \geq 2^{\lambda} \cdot \max\left\{2\log n,\ \max_{t\in[|\mathcal{C}|]} \delta_{it}\right\}.$$

Then Π_{UCS} *(Algorithm 1) has strong round-by-round knowledge soundness (Definition 3.12) with errors* $\kappa_1 = \kappa_2 = 2^{-\lambda}$.

We defer the proof to Appendix A.1.3.

Corollary 5.14 (Completeness of the Zinc+ PIOR)**.** *Under the hypotheses of Theorem 5.13,* Π_{UCS} *is complete (Definition 3.7) with error*

$$\epsilon \ \leq\ \frac{N_{\text{rat}} \cdot B}{2^{\lambda} \cdot \lambda \cdot \max_t(\delta_{0t}^2 \cdot B_{0t}^{\text{eval}})} \ +\ \sum_{\substack{i\in[|\mathbf{q}|]\\ \kappa_i>1}} \sum_{\substack{d|\kappa_i\\ d<\kappa_i}} p_i^{d-\kappa_i},$$

where $N_{\text{rat}} := d_0\Big(\ell + m + |\mathcal{C}| + \sum_{t\in[|\mathcal{C}|]} \|Q_{0t}\|_0\Big)$ *counts the rational coefficients of* $\mathbf{y}$, $\mathbf{f}_0$, *the ideal generators, and the constraint polynomials, each scaled by the degree bound* d_0*, with* $\|Q_{0t}\|_0$ *counting the monomials of* Q_{0t} *in* $(K, \mathbf{Y}, \mathbf{Z}_0)$ *that have nonzero* $\mathbb{Q}[X]$*-coefficient, and* $\kappa_i := [\mathbb{F}_{\tilde{q}_i} : \mathbb{F}_{p_i}]$.

We defer the proof to Appendix A.1.3.

On rejection sampling and perfect completeness. In the completeness bound, the second term generally dominates. This term bounds the probability that any sampled a_i fails to be a primitive element of the extension $\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i}$. When all fields have large characteristic it disappears. Otherwise, assuming 2 divides $[\mathbb{F}_{\tilde{q}_i} : \mathbb{F}_{p_i}]$ whenever $\tilde{q}_i > p_i$, it's $\approx 2^{\lambda/2}$.

The protocol samples from $\mathbb{F}_{\tilde{q}}$ rather than using rejection sampling to sample from Prim to prioritize recursion-friendliness. Even sampling uniformly from $\mathbb{F}_{\tilde{q}}$ in constant time is not possible in practice, and practitioners usually accept a small bias in the sampling to achieve constant time (e.g., following RFC 9380). Our system as written supports constant-time sampling from Prim using this established method and accepting the associated completeness error, with the caveat that, of course, our proofs assume the a_i are sampled uniformly. Using rejection sampling, an expected polynomial-time verifier could ensure each $a_i \in \mathsf{Prim}$ at the cost of recursion-friendliness.

The smaller first term bounds the probability that the random prime q_0 divides a denominator in the instance, witness, or constraint coefficients. This term vanishes when the denominators of (admissible) $\mathbf{y}$, $\mathbf{f}_0$, j_{0t}, or Q_{0t} have absolute value strictly less than the smallest prime in $\mathcal{P}$ (e.g., when integral).

Projection preserves constraint polynomial sparsity and degree. We present the aforementioned results bounding the degree and sparsity of the projected constraint polynomials.

Lemma 5.15 (Algebraic-degree preservation)**.** *For each* (i, t) *with* $i \in [0, |\mathbf{q}|]$ *and* $t \in [|\mathcal{C}|]$*, the projected constraint polynomial* Q'_{it} *produced by* Π_{UCS} *satisfies*

$$\deg(Q'_{it}) \ \leq\ \deg_{\mathbf{Y},\mathbf{Z}_0,\mathbf{Z}_1}(Q_{it}).$$

Proof. The projection ψ_{q_i,a_i} acts on X-coefficients only and preserves algebraic support. Boolean substitution $K = \mathbf{b}$ projects onto $(\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1)$, the weight $\widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i)$ is scalar, row summation aggregates monomials, and subtracting $e_{it}(a_i)$ adds a constant—none increase $\deg_{\mathbf{Y},\mathbf{Z}_0,\mathbf{Z}_1}$. □

Lemma 5.16 (Sparsity preservation)**.** *For each (i,t) with $i \in [0,|\mathbf{q}|]$ and $t \in [|\mathcal{C}|]$, the projected constraint polynomial Q'_{it} produced by Π_{UCS} satisfies*

$$\|Q'_{it}\|_0 \;\le\; 1 + \big|\{(\mathbf{Y},\mathbf{Z}_0,\mathbf{Z}_1)\text{-monomials in } \mathrm{supp}(Q_{it})\}\big|.$$

Proof. Boolean substitution $K = \mathbf{b}$ collapses K-monomials with the same $(\mathbf{Y},\mathbf{Z}_0,\mathbf{Z}_1)$-projection, bounding post-substitution sparsity by the number of distinct $(\mathbf{Y},\mathbf{Z}_0,\mathbf{Z}_1)$-projections in $\mathrm{supp}(Q_{it})$. The projection ψ_{q_i,a_i} and row summation over $\mathbf{b} \in \{0,1\}^\mu$ can only further reduce sparsity. Subtracting the scalar $e_{it}(a_i)$ adds at most one constant monomial. □

5.4 A SNARK for UCS

To obtain a SNARK for UCS, two additional ingredients are needed: a finite-field PIOP for each $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}_i})$ instance (e.g., any sumcheck-based PIOP for CCS, AIR, or R1CS), and IOPPs for projected multilinear evaluation over both $\mathbb{Q}[X]$ and finite fields (such as those provided by Zip+ for $\mathbb{Q}[X]$, Section 6). The PIOP-to-IOP compilation of Theorem B.2 assembles these into an IOP, which then yields a SNARK in the random oracle model via the BCS-style pipeline described in Section 3.3.2.

The following theorem states the parameter constraints that arise from the Zinc+ PIOR and its interaction with the input primitives. These input primitives may impose additional constraints on Zinc+ PIOR parameters (such as structural requirements on the moduli, e.g. multiplicative subgroups of specific order, when the IOPPs are instantiated with Reed–Solomon-like codes), and their per-round knowledge errors may depend on $\mathbb{F}_{\tilde{q}}$ and parameters in the UCS index $\mathtt{i}$ or projected PESAT indices $\bar{\mathtt{i}}'_i$.

We restrict the $\mathsf{PESAT}_{\mathbb{Q}[X]}$ PIOPs constructed via Construction 5.9 to *commit-and-prove* (CP) form: each Π'_i proves a statement about polynomial oracles handed to it by Algorithm 1, with all prover messages explicit rather than oracles. This satisfies Theorem B.2's structural interface, and is the natural form anyway.

Theorem 5.17 (Compiled IOP for UCS)**.** *Let $\lambda' := \lambda + \lceil \log_2(|\mathbf{q}|+1) \rceil$. For each $i \in [0,|\mathbf{q}|]$, let Π_i be a CP-PIOP for $\mathsf{PESAT}(\mathbb{F}_{\tilde{q}_i})$ (Definition 5.7), let Π'_i be its lift via Construction 5.9, and let Π_Σ be the PIOP that runs the protocol of Algorithm 1 and then runs $\Pi'_0,\dots,\Pi'_{|\mathbf{q}|}$ in parallel. Suppose:*

1. *Each Π_i has per-round RBR knowledge soundness error at most $2^{-\lambda'}$ and satisfies the single-uniform-query hypothesis of Theorem 5.11.*
2. *For every final IOPP round j in the invocation of the compiler of Theorem B.2 on Π_Σ, the sum of the round-j RBR knowledge errors over all final IOPP instances generated by the compiler is at most $2^{-\lambda}$.*

Assume that the prime set $\mathcal{P}$ and prime powers $\tilde{q}_i$ satisfy the hypotheses of Theorem 5.13 and additionally

$$|\mathbb{F}_{\tilde{q}_i}| \ge 2^{\lambda'}\nu \qquad \text{for every } i \in [0,|\mathbf{q}|],$$

and that the out-of-domain error introduced by Theorem B.2 satisfies $\varepsilon_{\mathsf{ood}} \le 2^{-\lambda}$.

If k_{PIOR} is the round count of Algorithm 1 and k_i is the round count of Π'_i, then

$$k_\Sigma = k_{\mathsf{PIOR}} + \max_{i\in[0,|\mathbf{q}|]} k_i$$

is the round count of Π_Σ. With the auxiliary-slot witness convention of Remark 3.10, Π_Σ has knowledge-state-form RBR knowledge soundness with per-round error at most $2^{-\lambda}$.

Applying Theorem B.2 to Π_Σ yields an IOP for $\mathsf{REL}_{\mathsf{UCS}}$ with per-round RBR knowledge soundness error at most $2^{-\lambda}$.

We defer the proof to Appendix A.1.4.

6 Zip+: an IOPP for constrained codes over polynomial rings

In this section, we present our IOPP, Zip+. To be compatible with the PIOPs in Zinc+, our IOPP must test proximity to a *constrained code* over $\mathbb{K}[X]$, where the constraints are given by what we call *projected multilinear evaluations*. To be more precise, given a witness $w \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$, which we view as a matrix, a projected multilinear evaluation of w is obtained by first projecting w from $\mathbb{K}^{<d}[X]$ to some finite field $\mathbb{K}'$ via a suitable map, ψ, and then evaluating the multilinear evaluation of the resulting matrix, $\widetilde{\psi(w)}$, at some point $z \in \mathbb{K}'^{\mu}$, where μ is such that $2^\mu = k_1 k_2$ (here we are assuming the dimensions k_2, k_1 are powers of 2).

When $\mathbb{K} = \mathbb{Q}$, then we may project $\mathbb{Q}[X]$ to any finite field $\mathbb{F}_m$, provided $w \in \mathbb{Z}_{(p)}$, where p is the characteristic of $\mathbb{F}_m$. When $\mathbb{K}$ is a finite field, we only allow for projections from $\mathbb{K}[X]$ to $\mathbb{K}$, and in this case we also only deal with prime fields, $\mathbb{K}$.

In this section, we build up to the full IOPP for projected multilinear evaluations in three steps.

In the first part, we give an IOPP for inner product constrained interleaved codes. This part deals with witnesses $W \in \mathbb{K}^{k_2 \times k_1}$ and the constraints are of the form $Wv = a$ for $a \in \mathbb{K}^{k_1}$ and $v \in \mathbb{K}^{k_2}$. When $\mathbb{K} = \mathbb{Q}$, we allow for constraints of the form $\phi_m(W)v = a$ where $v \in \mathbb{F}_m^{k_1}, a \in \mathbb{F}_m^{k_2}$ and m is a prime.

In the second part, we handle tensor evaluation constraints over witnesses $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$. These constraints are of the form $u_2^T W u_1 = \alpha$ for $u_1 \in \mathbb{K}^{k_1}, u_2 \in \mathbb{K}^{k_2}$ and $\alpha \in \mathbb{K}$. Similarly, when $\mathbb{K} = \mathbb{Q}$, we allow for constraints of the form $u_2^T \phi_m(W) u_1 = \alpha$ where $u_1 \in \mathbb{F}_m^{k_1}, u_2 \in \mathbb{F}_m^{k_2}$ and $\alpha \in \mathbb{F}_m$. The main difference compared to the first part is the fact that the witness contains polynomial, rather than only scalar, entries. In the $d = 1$ case, the two parts are essentially the same.

Finally, in the last part, we handle projected multilinear evaluation constraints. Here the witness is $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$, but now the constraints are of the form $\widetilde{\psi(W)}(z) = \alpha$ for $z \in \mathbb{K}'^{\mu}$ and $\alpha \in \mathbb{K}'$, and ψ is a projection from $\mathbb{K}[X]$ to some finite field $\mathbb{K}'$. When $\mathbb{K}$ is a finite field, we only allow for projections from $\mathbb{K}[X]$ to $\mathbb{K}$ and when $\mathbb{K} = \mathbb{Q}$ we can project to any finite field $\mathbb{F}_m$ as long as $W \in \mathbb{Z}_{(p)}$ where p is the characteristic of $\mathbb{F}_m$.

6.1 An IOPP for codes over $\mathbb{Q}$ with Inner Product Constraints over a finite field or $\mathbb{Q}$

We start by giving a simple Brakedown-like IOPP for inner-product constrained interleaved codes, that is, interleaved codes whose underlying message satisfies an inner-product constraint. For our purposes, we will also allow the constraint to be taken modulo m for some (large) prime m.

More formally, our IOPP is an IOP for the following proximity relation, which we call the *constrained interleaved code proximity relation.*

Definition 6.1 (Constrained interleaved code proximity relation)**.** The *constrained interleaved code proximity relation* $\mathsf{REL}_{\mathsf{CIC}}$ is the set of all triples $(\mathbb{i}, \mathbb{x}; \mathbb{w})$ where the index is

$$\mathbb{i} = (M, B_{\mathsf{code}}, n, k, J, B, \beta, (u, a, m))$$

consisting of a linear code generator matrix $M \in \left(\mathbb{Q}^{<B_{\mathsf{code}}}\right)^{n \times k}$, a number of rows parameter J, a bitsize bound $B \in \mathbb{N}$, a proximity parameter $\beta \in [0, 1]$, and a *constraint* given by (u, a, m). Here,

$m \in \mathbb{N}$ is a prime modulus for the t-th constraint (where $m = 0$ means no modular reduction), $u \in \mathbb{F}^k$ is the constraint vector, and $a \in \mathbb{F}_m^J$ is the target value for the constraint. When $m = 0$, we abuse notation and write $\mathbb{F}_m = \mathbb{Q}$.

The input is $\mathbb{x} = ([[v_1]], \ldots, [[v_J]])$, where each $v_j \in \mathbb{Q}^n$ and $[[v_i]]$ denotes oracle access to v_i. Let V be the $J \times n$ matrix with rows $v_1, \ldots, v_J$. The witness is $\mathbb{w} = (w_1, \ldots, w_J) \in \left((\mathbb{Q})^k\right)^J$ and likewise let W be the $J \times k$ matrix with rows $w_1, \ldots, w_J$. We have $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{CIC}}$ if and only if the following hold

(i) $\phi_m(W)u = a$ where the equality is in $\mathbb{F}_m^J$,

(ii) $\mathsf{dist}(WM^T, V) \leq \beta$ where we view WM^T and V as length-n vectors over the alphabet $\mathbb{Q}^J$.

(iii) $W \in \left(\mathbb{Q}^{<B}\right)^{J \times k}$, i.e., each entry of W has bitsize less than B.

Recall in condition (i) above that $\phi_m(W)$ is obtained by reducing the entries of W modulo m. Hence, if W contains an entry not in $\mathbb{Z}_{(m)}$, then $\phi_m(W)$ is undefined and condition (i) is not satisfied.

In words, the relation $\mathsf{REL}_{\mathsf{CIC}}$ consists of all inputs which are close to interleaved-encodings of bounded bitsize witnesses that satisfy the given inner product constraints specified by the index. We remark that our IOP for $\mathsf{REL}_{\mathsf{CIC}}$ will not have perfect completeness for *all* witness-index pairs in the relation, but only for pairs where 1) the witness satisfies a stricter bitsize bound, and 2) the oracles are the actual encodings for the witness (rather than only close to the actual encodings). Typical IOPPs are similar in that they are also IOPs for a proximity relation where perfect completeness only holds for index-witness pairs satisfying condition 2). Having condition 1) for perfect completeness is unique to our setting working over the rationals however, and it is due to bitsize considerations. In order to reject witnesses that violate some bitsize bound, we can only guarantee perfect completeness for witnesses that satisfy a stricter bitsize bound, $B_0 < B$. This bound is derived from our protocol parameters.

Our IOP for $\mathsf{REL}_{\mathsf{CIC}}$ (Definition 6.1) is described below. The protocol takes in two protocol parameters $C, K \in \mathbb{N}$ which determine the bitsize for the verifier's random coefficients and the number of spot-check queries, respectively.

Algorithm 2 Protocol $\Pi_{\mathsf{CIC}}(K, C)$: IOPP for the relation $\mathsf{REL}_{\mathsf{CIC}}$ (one constraint case).

Input: P and V receive an index $\mathbb{i} = (M, B_{\mathsf{code}}, n, k, J, B, \beta, (u, a, m))$ as in Definition 6.1, and $\mathbb{x} = ([[v_1]], \ldots, [[v_J]])$.

Additionally, $C, K \in \mathbb{N}$ are protocol parameters where C is a number of spot-checks parameter and K is a bitsize parameter for the verifier's random coefficients. We assume that the honest prover holds a witness $\mathbb{w} = (w_1, \ldots, w_J) \in \left(\left(\mathbb{Q}^{<B_0}\right)^k\right)^J$, where $B_0 \in \mathbb{N}$ is the smaller bitsize bound for the completeness case satisfying

$$6JB_0 + (6J + 1)K + 3\log(J) + 1 = B. \tag{39}$$

Throughout the protocol, the verifier truncates all of the oracle outputs so that they are in $\mathbb{Q}^{<2k(B_0 + B_{\mathsf{code}}) + \log k}$, and hence we may assume $v_j \in \left(\mathbb{Q}^{<2k(B_0 + B_{\mathsf{code}}) + \log k}\right)^n$ for each $j \in [J]$. We remark that this bitsize is chosen so that the oracles for the honest prover are not truncated and perfect completeness is unaffected for the honest prover.

1: V chooses $r_1, \ldots, r_J \in [2^K]$ uniformly and sends them to the prover. Set

$$v^* = \sum_{j=1}^{J} r_j \cdot v_j.$$

2: P responds with $w \in \left(\mathbb{Q}^{<2J(B_0+K)+\log(m)}\right)^k$. The honest prover sends $w^* = \sum_{j=1}^J r_j \cdot w_j$ so that $Mw^* = v^*$ and $\phi_m(w^*) \cdot u = \alpha^*$, where $\alpha^* = \sum_{j=1}^J \phi_m(r_j) \cdot a_j$.

3: V first checks if

$$\phi_m(w) \cdot u = \alpha^*,$$

and if not rejects, sends $\perp$ to the prover, and terminates the protocol. If the equality holds, then V chooses C indices $\ell_1, \ldots, \ell_C \in [n]$ and checks

$$(v^*)_{\ell_i} = (Mw)_{\ell_i} \quad \forall i \in [C].$$

To obtain the ℓ_i-th entry of v^*, V queries the ℓ_i-th entry of v_j for each $j \in [J]$. If any of the C equalities does not hold, V rejects; otherwise, V accepts.

Theorem 6.2. *Let $\mathtt{i} = (M, B_{\mathsf{code}}, n, k, J, B, \beta, (u, a, m))$ be an index for $\mathsf{REL}_{\mathsf{CIC}}$ (Definition 6.1), and let $K, C \in \mathbb{N}$ be the chosen parameters for Algorithm 2. Suppose M generates a code over $\mathbb{Q}$ which satisfies MCA up to distance β with respect to $[2^K]$ with error $\mathsf{err}_{\mathsf{pg}}(M, \beta, K)$, and suppose $\mathcal{C}^J$ is (L, β)-list decoding. Let $B_0 \in \mathbb{N}$ be a bound on the bitsize of the witness entries in the completeness case given by Eq. (39), and assume either $m = 0$ or $m > \max(2^{B_0}, 2^K)$. Then $\Pi_{\mathsf{CIC}}(K, C)$ in Algorithm 2 is a two-round IOPP for the relation $\mathsf{REL}_{\mathsf{CIC}}$ with the following guarantees.*

- ***Completeness.*** *If $(\mathtt{i}, \mathtt{x}; \mathtt{w}) \in \mathsf{REL}_{\mathsf{CIC}}$ with $\mathtt{x} = (v_1, \ldots, v_J)$ and $\mathtt{w} = (w_1, \ldots, w_J) \in \left(\left(\mathbb{Q}^{<B_0}\right)^k\right)^J$ satisfying $Mw_j = v_j$ and $\phi_m(w_j) \cdot u = a_j$ for all $j \in [J]$, then the verifier accepts with probability 1.*
- ***Round-by-Round Knowledge Soundness*** *(Definition 3.9) with errors $(\mathsf{err}_1, \mathsf{err}_2)$, where*

$$\mathsf{err}_1 = (J-1) \cdot \mathsf{err}_{\mathsf{pg}}(M, \beta, K) + \frac{L}{2^{K-1}} \quad \text{and} \quad \mathsf{err}_2 = (1-\beta)^C.$$

Proof. We verify the properties of Algorithm 2. We show all of the properties except for round-by-round knowledge soundness, which is shown in Appendix A.2.1. The round complexity and query complexity are easy to see. For completeness, suppose $(\mathtt{i}, \mathtt{x}; \mathtt{w}) \in \mathsf{REL}_{\mathsf{CIC}}$. Then, it is straightforward to check that:

- for each $j \in [J]$, $v_j = Mw_j \in \left(\mathbb{Q}^{<2k(B_0+B_{\mathsf{code}})+\log k}\right)^n$,
- with probability 1 over $r_1, \ldots, r_J$ that the message from the honest prover, $w = \sum_{j=1}^J r_j w_j$, in step 2 satisfies the inner product constraints, $\phi_m(w) \cdot u = \alpha^*$ and $w = \sum_{j=1}^J r_j w_j \in \left(\mathbb{Q}^{<2J(B_0+K)+\log(J)}\right)^k$. Note that by Lemma 6.9 (stated and proved later) with probability 1, the honest prover's w^* has entries with bitsize $2J(B_0+k)+\log(J)+1$ as required in step 2.

If both items above hold, it is clear that the spot checks in Step 3 pass with probability 1. The proof of round-by-round knowledge soundness is deferred to Appendix A.2.1. □

6.2 An IOPP for codes over $\mathbb{K}^{<d}[X]$ with Tensor Constraints over prime fields or $\mathbb{Q}$

In this section, we extend from inner-product constraints to tensor constraints. Given a witness matrix $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$, a tensor constraint is of the form $u_2^T W u_1 = \alpha$ for some constraint

vectors $u_1 \in \mathbb{K}^{k_1}, u_2 \in \mathbb{K}^{k_2}$ and some target value $\alpha \in \mathbb{K}$. We remark that in this section, we also allow the witness to have polynomial entries, rather than only scalar entries as in the prior section. Reducing from tensor constraints to inner product constraints is straightforward, and the main difference between the protocol we present here versus Algorithm 2 is the presence of polynomial entries in the witness matrix. In the $d = 1$ case, the IOPP of this section is almost exactly the same as Algorithm 2.

The relation which we construct an IOP for is the following.

Definition 6.3 (Tensor Constrained Interleaved Code Proximity Relation)**.** The *tensor constrained interleaved code proximity relation* over field $\mathbb{K}$ (which is either a prime finite field or $\mathbb{Q}$), $\mathsf{REL}_{\mathsf{TE},\mathbb{K}}$ is the set of all triples $(\mathtt{i}, \mathtt{x}; \mathtt{w})$ where the index is

$$\mathtt{i} = (M, B_{\mathsf{code}}, n, k_1, k_2, d, B, m, \beta, (u_1, u_2, \alpha, m))$$

where $M \in \left(\mathbb{K}^{<B_{\mathsf{code}}}\right)^{n \times k_1}$ is a generator matrix; $d \in \mathbb{N}$ is a degree bound on witness entries over $\mathbb{K}[X]$; $B \in \mathbb{N}$ is the bit-size bound for coefficients of witness entries; $\beta \in [0, 1]$ is the proximity parameter; and (u_1, u_2, α, m) defines a tensor constraint. In the tensor constraint, $m \in \{0\} \cup \mathbb{N}$ is a prime modulus (with $m = 0$ meaning no modular reduction), $u_1 \in \mathbb{F}_m^{k_1}$ and $u_2 \in \mathbb{F}_m^{k_2}$ are vectors defining the bilinear form for the constraint (where we abuse notation to let $\mathbb{F}_0 = \mathbb{Q}$) and $\alpha \in \mathbb{F}_m^{<d}[X]$ is the target value for the constraint.

When $\mathbb{K}$ is a prime field, we ignore all bit-size bounds and furthermore the relation is invalid unless all the moduli are $m = |\mathbb{K}|$ (i.e., we only handle constraints over $\mathbb{K}$ when the witness is over $\mathbb{K}$).

The input is $\mathtt{x} = ([[v_1]], \ldots, [[v_{k_2}]])$ where each $v_i \in \left(\mathbb{K}^{<d}[X]\right)^n$ and the witness is $\mathtt{w} = (w_1, \ldots, w_{k_2}) \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$. We let V be the $k_2 \times k_1$ matrix with rows $v_1, \ldots, v_{k_2}$ and we let W be the $k_2 \times k_1$ matrix with rows $w_1, \ldots, w_{k_2}$. Then, $(\mathtt{i}, \mathtt{x}; \mathtt{w}) \in \mathsf{REL}_{\mathsf{TE},\mathbb{K}}$ if and only if:

(i) $u_2^T \phi_m(W) u_1 \equiv \alpha \pmod{m}$,

(ii) $\mathsf{dist}(WM^T, V) \leq \beta$ where we view WM^T and V as length n vectors over $\left(\mathbb{K}^{<d}[X]\right)^{k_2}$,

(iii) $W \in \left(\mathbb{K}^{<d,B}[X]\right)^{k_2 \times k_1}$.

If the witness is over $\mathbb{Q}$ and contains any entry with denominator divisible by m, then the instance is not in the relation, as the modular reduction in condition (i) is not well defined.

Remark 6.4. While Definition 6.3 is described over both finite fields and $\mathbb{Q}$, in this section, we focus on building IOPPs for $\mathsf{REL}_{\mathsf{TE},\mathbb{Q}}$. Over finite fields, IOPPs for $\mathsf{REL}_{\mathsf{TE},\mathbb{K}}$ can be obtained either by a straightforward adaptation of the IOPP we describe here or by adapting other known code-based IOPPs over finite fields.

We now describe the IOP for the tensor constrained relation $\mathsf{REL}_{\mathsf{TE}}$ (Definition 6.3).

Algorithm 3 Protocol $\Pi_{\mathsf{TE}}(K, C)$: IOPP for batched tensor constrained interleaved codes (one-constraint case).

Input: P and V receive an index $\mathtt{i} = (M, B_{\mathsf{code}}, n, k_1, k_2, d, B, m, \beta, (u_1, u_2, \alpha, m))$ as in Definition 6.3. Assume the honest prover holds a valid witness $\mathtt{w} = (w_1, \ldots, w_{k_2}) \in \left(\mathbb{Q}^{<d,B_0}[X]\right)^{k_2 \times k_1}$, where $B_0 < B$ is a stricter bitlength bound defined by

$$B_{\mathsf{agg}} := 2d(B_0 + K) + \log(d) + 1 \text{ satisfies } B = 3B_{\mathsf{agg}} + K + 1. \tag{40}$$

Additionally, $K, C \in \mathbb{N}$ are parameters determining the number of bits in the verifier's random coefficients and the number of spot-checks, respectively.
Throughout the protocol, the verifier truncates all of the oracle outputs so that they are in $\mathbb{Q}^{<d,B_1}[X]$ where $B_1 = 2k_1(B_0 + B_{\mathsf{code}}) + \log(k_1) + 1$. We remark that for the honest prover, all of the oracle outputs are indeed in $\mathbb{Q}^{<d,B_1}[X]$.

1: V chooses coefficients $\gamma_0, \ldots, \gamma_{d-1} \in [2^K]$ uniformly at random and sends them to the prover.

2: Let us decompose $v_j = \sum_{i<d} v_j^{(i)} \cdot X^i$ for $v_j^{(i)} \in \left(\mathbb{Q}^{<d}\right)^n$, $w_j = \sum_{i<d} w_j^{(i)} \cdot X^i$ for $w_j^{(i)} \in \left(\mathbb{Q}^{<d}\right)^{k_1}$, and $\alpha = \sum_{i<d} \alpha^{(i)} \cdot X^i$ for $\alpha^{(i)} \in \mathbb{F}_m$. In words, $v_j^{(i)}$ is the i-th coefficient vector of v_j, $w_j^{(i)}$ is the i-th coefficient vector of w_j, and $\alpha^{(i)}$ is the i-th coefficient of α.
Set $\alpha^* = \sum_{i<d} \phi_m(\gamma_i)\alpha^{(i)} \in \mathbb{F}_m$. Let $v_j^* = \sum_{i<d} \gamma_i v_j^{(i)} \in \mathbb{Q}^n$, $w_j^* = \sum_{i<d} \gamma_i w_j^{(i)} \in \mathbb{Q}^{k_2}$ for each $j \in [k_2]$, and let $W^* \in \left(\mathbb{Q}^{<d(B_0+K)+\log(d)}\right)^{k_2 \times k_1}$ be the matrix with rows $w_1^*, \ldots, w_{k_2}^*$.

3: P responds with $a \in \mathbb{F}_m^{k_2}$. The honest prover responds with an a such that $\phi_m(W^*) \cdot u_1 = a$.

4: V checks that $u_2 \cdot a = \alpha^*$, where this equality is over $\mathbb{F}_m$. If not, V rejects and the protocol terminates. If the check passes, P and V run Algorithm 2 (Π_{CIC}) on the derived instance for $\mathsf{REL}_{\mathsf{CIC}}$ given by:

$$\mathbb{i}_{\mathsf{CIC}} = (M, B_{\mathsf{code}}, n, k_1, k_2, B', \beta, \{(u_1, a, m)\}), \quad \mathbb{x}_{\mathsf{CIC}} = [[v_1^*]], \ldots, [[v_{k_2}^*]] \text{ and } \mathbb{w}_{\mathsf{CIC}} = (w_1^*, \ldots, w_{k_2}^*).$$

where $B' := 6k_2 B_{\mathsf{agg}} + (6k_2 + 1)K + 3\log(k_2) + 1$ and B_{agg} is as in Eq. (40). In the IOPP, V simulates a query to $[[v_j^*]]$ by querying $[[v_j]]$ and taking the suitable linear combination of the output's coefficients.

5: V accepts if and only if the Π_{CIC} verifier accepts.

Theorem 6.5. *Let $\mathbb{i} = (M, B_{\mathsf{code}}, n, k_1, k_2, d, B, \beta, (u_1, u_2, \alpha, m))$ be an index for $\mathsf{REL}_{\mathsf{TE},\mathbb{Q}}$ (Definition 6.3), and let $K, C \in \mathbb{N}$ be the chosen parameters for Algorithm 3. Suppose M generates a code $\mathcal{C}$ which satisfies MCA up to distance β with respect to $[2^K]$ with error $\mathsf{err}_{\mathsf{pg}}(M, \beta, K)$, and suppose $\mathcal{C}^{k_2}$ is (L, β)-list decodable. Let $B_0 \in \mathbb{N}$ be a bound on the bitsize of the witness entries in the completeness case given by Eq. (40), and assume either $m = 0$ or $m > \max(2^{B_0}, 2^K)$. Finally, suppose Algorithm 2 ($\Pi_{\mathsf{CIC}}(K, C)$) has round-by-round knowledge soundness errors*

$$(\mathsf{err}_1^{\mathsf{CIC}}(K, C, \mathbb{i}_{\mathsf{CIC}}, \mathbb{x}_{\mathsf{CIC}}), \mathsf{err}_2^{\mathsf{CIC}}(K, C, \mathbb{i}_{\mathsf{CIC}}, \mathbb{x}_{\mathsf{CIC}})),$$

where $\mathbb{i}_{\mathsf{CIC}}, \mathbb{x}_{\mathsf{CIC}}$ are the index and input as derived in step 5. Then Algorithm 3 is an IOP for the relation $\mathsf{REL}_{\mathsf{TE},\mathbb{Q}}$ with the following guarantees.

- ***Round Complexity.*** *$1 + \mathsf{rd}_{\mathsf{CIC}}$, where $\mathsf{rd}_{\mathsf{CIC}}$ is the round complexity of $\Pi_{\mathsf{CIC}}(K, C)$.*
- ***Completeness.*** *If $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{TE},\mathbb{Q}}$ with $\mathbb{x} = ([[v_1]], \ldots, [[v_{k_2}]])$ and $\mathbb{w} = (w_1, \ldots, w_{k_2}) \in \left(\mathbb{Q}^{<d,B_0}[X]\right)^{k_2 \times k_1}$ such that $WM^T = V$, where $V \in \left(\mathbb{Q}^{<d}[X]\right)^{k_2 \times n}$ is the matrix with rows $v_1, \ldots, v_{k_2}$ and $W \in \left(\mathbb{Q}^{<d,B_0}[X]\right)^{k_2 \times k_1}$ is the matrix with rows $w_1, \ldots, w_{k_2}$ then the verifier accepts with probability 1.*
- ***Round-by-Round Knowledge Soundness*** *(Definition 3.9).*

$$\left((d-1)\mathsf{err}_{\mathsf{pg}}(M, \beta, K) + \frac{L}{2^{K-1}}, \mathsf{err}_1^{\mathsf{CIC}}(K, C, \mathbb{i}_{\mathsf{CIC}}, \mathbb{x}_{\mathsf{CIC}}), \mathsf{err}_2^{\mathsf{CIC}}(K, C, \mathbb{i}_{\mathsf{CIC}}, \mathbb{x}_{\mathsf{CIC}})\right).$$

Proof. The round complexity is straightforward to see. For the completeness, suppose the honest prover has a witness $\mathbb{w} = (w_1, \ldots, w_{k_2})$ as in the completeness case described above and let W be

the matrix with rows $w_1, \ldots, w_{k_2}$. Let W^* and a be as in step 4. By the assumption from the completeness case, the honest prover can send $a = \phi_m(W^*)u_1$ and this a passes the verifier's check in step 5. Furthermore, one can check that $W^*M^T = V^*$ and by Lemma 6.9 and the assumption on W's bitsizes from the completeness case, W^* satisfies, with probability 1, the bitlength requirement of the completeness case in Theorem 6.2 with index $\mathbb{i}$ for $\mathsf{REL}_{\mathsf{CIC}}$. This concludes the analysis of completeness. We defer round-by-round knowledge soundness to Appendix A.2.2. □

6.3 An IOPP for codes over $\mathbb{Q}^{<d}[X]$ with Multilinear Evaluation Constraints over arbitrary finite fields

In this section we consider IOPPs for constraints given by multilinear evaluations over an extension field. We consider witnesses $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_2}$ similar to the previous section, except now the constraints are on the projection of the witness W to a finite field $\mathbb{K}'$. It will be convenient to split our discussion into the case where $\mathbb{K} = \mathbb{Q}$ and the case where $\mathbb{K}$ is a finite field. We start with the latter since it is simpler.

Over Finite $\mathbb{K}$. In this case the type of projections we allow are only from $\mathbb{K}[X] \to \mathbb{K}$. Here, we use the ring homomorphism $\psi_{q,\zeta} : \mathbb{K}[X] \to \mathbb{K}$ from Definition 5.2 where $q = |\mathbb{K}|$ and $\zeta \in \mathbb{K}$. Recall that this projection evaluates a $\mathbb{K}[X]$ element at $X = \zeta$. For $W \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_2}$, we consider constraints of the form $\widetilde{\psi_{q,\zeta}(W)}(z) = \gamma$ for $z \in \mathbb{K}^\mu$ and $\gamma \in \mathbb{K}$.

Over $\mathbb{Q}$. In this case we allow for projections of the witness to any finite field $\mathbb{F}_q$. Recall from Definition 5.2 that for any $\mathbb{F}_q$ with characteristic p, there is a ring homomorphism $\mathbb{Z}_{(p)}[X] \to \mathbb{F}_q$, which we denote $\psi_{q,\zeta}$, obtained by localizing coefficients into $\mathbb{F}_p$ and then evaluating at $\zeta \in \mathbb{F}_q$. Hence, for $W \in \left(\mathbb{Q}^{<d}[X]\right)^{k_2 \times k_2}$, we consider constraints of the form $\widetilde{\psi_{q,\zeta}(W)}(z) = \gamma$ for $z \in \mathbb{F}_q^\mu$ and $\gamma \in \mathbb{F}_q$.

For simplicity, we will assume that one suitable ζ has been chosen for each field size q and hence we will drop the ζ from the subscript and only write ψ_q. Note that this implies that if $q = p^\kappa$, where p is the characteristic of $\mathbb{F}_q$, then $1, \zeta, \ldots, \zeta^{\kappa-1}$ form an $\mathbb{F}_p$ basis of $\mathbb{F}_q$.

Furthermore, we will abuse notation and use ψ_q to refer to both the homomorphism from $\mathbb{Z}_{(p)}[X] \to \mathbb{F}_q$, and from $\mathbb{F}_q[X] \to \mathbb{F}_q$, where it will be clear from context which we are referring to.

Before proceeding, it will be helpful to also define the inverse map $\psi_q^{-1} : \mathbb{F}_q \to \mathbb{Z}_{(p)}^{<\kappa}[X]$ which maps $\alpha \in \mathbb{F}_q$ to the unique polynomial $\psi_q^{-1}(\alpha) \in \mathbb{Z}_{(p)}^{<\kappa}[X]$ such that $\psi_q(\psi_q^{-1}(\alpha)) = \alpha$. We naturally extend both ψ_q and ψ_q^{-1} to a map from vectors over $\mathbb{F}_q$ to vectors over $\mathbb{Z}_{(p)}^{<\kappa}[X]$ by applying ψ_q^{-1} coordinate-wise.

Definition 6.6 (Projected multilinear evaluation constrained interleaved code proximity relation)**.** The *Projected MLE constrained proximity relation* $\mathsf{REL}_{\mathsf{PMLE},\mathbb{K}}$ is the set of all triples $(\mathbb{i}, \mathbb{x}; \mathbb{w})$ where the index is

$$\mathbb{i} = (M, B_{\mathsf{code}}, n, k_1, k_2, d, B, m, \beta, (z, \alpha, q)),$$

where $M \in \mathbb{K}^{n \times k_1}$ is a generator matrix for a code $\mathcal{C} \subseteq \mathbb{K}^n$; $n \in \mathbb{N}$ is a code length; $k_1, k_2 \in \mathbb{N}$ are dimensions such that $k_1 k_2 = 2^\mu$ for some $\mu \in \mathbb{N}$; $d \in \mathbb{N}$ a degree bound on witness coefficients over $\mathbb{K}[X]$; $B \in \mathbb{N}$ the bit-size bound for witness coefficients; $\beta \in [0, 1]$ the proximity parameter; and constraints given by (z, α, q). For each constraint, q is the size of an extension field $\mathbb{F}_q$ with characteristic that we denote by p; $z \in \mathbb{F}_q^\mu$ is the evaluation, and $\alpha \in \mathbb{F}_q$ is the target value.

The input is $\mathtt{x} = [[V]] = ([[v_1]], \ldots, [[v_{k_2}]])$, where $V \in \left(\left(\mathbb{K}^{<d}[X]\right)^{k_2}\right)^n$ and we think of $v_j \in \mathbb{K}^n$ as the j-th row of V. The witness is $\mathtt{w} = (w_1, \ldots, w_{k_2}) \in \left(\mathbb{K}^{<d}[X]\right)^{k_2 \times k_1}$. We have $(\mathtt{i}, \mathtt{x}; \mathtt{w}) \in \mathsf{REL}_{\mathsf{PMLE},\mathbb{K}}$ if and only if:

(i) $\widetilde{\psi_q(W)}(z) = \alpha$, where W is the matrix with rows given by $w_1, \ldots, w_{k_2}$,

(ii) $\mathsf{dist}(WM^T, V) \leq \beta$.

(iii) $W \in \left(\mathbb{K}^{<d,B}[X]\right)^{k_2 \times k_1}$.

We remark that in the case $\mathbb{K}$ is a finite field, we only allow for $q = |\mathbb{K}|$ in item (i) above and we ignore the bitsize bounds and in particular item (iii) above. When $\mathbb{K} = \mathbb{Q}$, we assume that condition (i) is not satisfied if $W \notin \mathbb{Z}_{(p)}[X]^{k_2 \times k_1}$ where p is the characteristic of $\mathbb{F}_q$ (similar to in Definition 6.1 and Definition 6.3).

Remark 6.7. The $\mathsf{REL}_{\mathsf{PMLE}}$ relation captures the projected polynomial oracle abstraction (Definition 5.3): an IOPP for $\mathsf{REL}_{\mathsf{PMLE}}$ realizes such an oracle, with the witness bounds d, B matching the oracle's type bounds and the constraint $\widetilde{\psi_q(W)}(z) = \alpha$ encoding a single query. An IOPP rejection covers invalid claims, including the undefined-projection case represented by $\perp$.

We fix an extension field $\mathbb{F}_q$ throughout this section and let p be its characteristic. It will be helpful to state a basic but key fact which allows us to express multilinear evaluation constraints over an extension field as a quadratic form with vectors from $\mathbb{Q}^{<\kappa}[X]$. Fix $z \in \mathbb{F}_q^{\mu}$, and let $u_1 \in \mathbb{F}_q^{k_1}$ be the vector indexed by $\mathsf{eq}(b; z_1)$ for $b \in \{0,1\}_1^k$ and let $u_2 \in \mathbb{F}_q^{k_2}$ be the vector indexed by $\mathsf{eq}(a; z_2)$ for $a \in \{0,1\}^{k_2}$, where z_1 and z_2 are the first k_1 and last k_2 coordinates of z, respectively. Then, for any $W \in \left(\mathbb{Q}^{<d,B}[X]\right)^{k_2 \times k_1}$, we have

$$\widetilde{\psi_q(W)}(z) = u_2^T \cdot (\psi_q(W) \cdot u_1) = \psi_q(\psi_q^{-1}(u_2)^T \cdot W \cdot \psi_q^{-1}(u_1)). \tag{41}$$

In the case where $\mathbb{K} = \mathbb{Q}$, the above only holds when $W \in \mathbb{Z}_{(p)}[X]^{k_2 \times k_1}$ as otherwise the projection $\psi_q(W)$ is not well-defined.

Algorithm 4 Protocol $\Pi_{\mathsf{PMLE}}(K, C, \mathcal{P})$: IOP for $\mathsf{REL}_{\mathsf{PMLE},\mathbb{F}_q}$

Input: P and V receive an index

$$\mathtt{i} = (M, B_{\mathsf{code}}, n, k_1, k_2, \varphi, B, \beta, (z, \alpha, q))$$

and input

$$\mathtt{x} = ([[v_1]], \ldots, [[v_{k_2}]])$$

as in Definition 6.6. Suppose q is a degree-κ extension over $\mathbb{F}_p$ where p is the characteristic of $\mathbb{F}_q$. The honest prover holds

$$W \in (\mathbb{Q}^{<d,B_0}[X])^{k_2 \times k_1}$$

such that $\widetilde{\psi_q(W)}(z) = \alpha$ and $WM^T = V$ where W and V are matrices with rows $w_1, \ldots, w_{k_2}$ and $v_1, \ldots, v_{k_2}$ respectively. Here B_0 is a bit-length bound for the completeness case defined by

$$B_{\mathsf{agg}} := 2d(B_0 + K) + \log(d) + 1 \text{ satisfies } B = 3B_{\mathsf{agg}} + K + 1. \tag{42}$$

the same as in Algorithm 3. Additionally $K, C \in \mathbb{N}$ are protocol parameters where C is a number of spot-checks parameter, K is a bitsize for the verifier's random prime. Finally, the parameter $\mathcal{P}$ is a set of prime numbers each of bitsize at least K.

Throughout the protocol, the verifier truncates all of the oracle outputs so that they are in $\mathbb{Q}^{<d,B_1}[X]$ where $B_1 = 2k_1(B_0 + B_{\mathsf{code}}) + \log(k_1) + 1$. We remark that for the honest prover, all of the oracle outputs are indeed in $\mathbb{Q}^{<d,B_1}[X]$.

1: Let $u_1 \in \mathbb{F}_q^{k_1}$ and $u_2 \in \mathbb{F}_q^{k_2}$ be defined as in (41) for the point $z \in \mathbb{F}_q^{\mu}$ and let $u_1' = \psi_q^{-1}(u_1) \in ([0 \dots p-1]^{<\kappa}[X])^{k_1} \subseteq (\mathbb{Q}^{<\kappa}[X])^{k_1}$ and $u_2' = \psi_q^{-1}(u_2) \in ([0 \dots p-1]^{<\kappa}[X])^{k_2} \subseteq (\mathbb{Q}^{<\kappa}[X])^{k_2}$.

2: P responds with $a \in \mathbb{Q}^{<d+2\kappa-1, 2(B_0+\log p)+\log(dk_1k_2)}[X]$. The honest prover sends $a = u_2'^T W u_1'$, where this equality is over $\mathbb{Q}[X]$.

3: V checks if $\psi_q(a) = \alpha$, where this equality is over $\mathbb{F}_q$. If not, V rejects. Note, in particular, that if some entry in a has a coefficient not in $\mathbb{Z}_{(p)}$ then $\psi_q(a)$ is undefined and the verifier also rejects.

If the check passes, the verifier samples a random prime $m \in \mathcal{P}$ and $\xi \in \mathbb{F}_m$, and sends (m, ξ) to the prover.

4: P responds with $a' \in \mathbb{F}_m^{<d}[X]$. The honest prover sends $a' = \psi_{m,\xi}(u_2')^T \phi_m(W) \psi_{m,\xi}(u_1')$. Here, the equation is in the prime field $\mathbb{F}_m$, and recall $\phi_m(W) \in \mathbb{F}_m^{<d}[X]$ is obtained by reducing the coefficients of entries in W modulo m.

5: V checks if $\psi_{\xi,m}(a') = \psi_{\xi,m}(a)$ where this equality is in the prime field $\mathbb{F}_m$. If the equality does not hold, the verifier rejects. Otherwise, P and V run $\Pi_{\mathsf{TE}}(K, C)$ on the derived instance

$$\mathbb{i}_{\mathsf{TE}} = \Big(M, B_{\mathsf{code}}, n, k_1, k_2, B', \beta, \{(\psi_{m,\xi}(u_1'), \psi_{m,\xi}(u_2'), a', m)\}\Big), \qquad \mathbb{x}_{\mathsf{TE}} = ([[v_1]], \dots, [[v_{k_2}]]),$$

6: V accepts if and only if the derived $\Pi_{\mathsf{TE},\mathbb{K}}(K, C)$ verifier accepts.

We show that Algorithm 4 is an IOPP for $\mathsf{REL}_{\mathsf{PMLE},\mathbb{Q}}$, i.e. the $\mathbb{K} = \mathbb{Q}$ case of Definition 6.6. The case where the witness is over a finite field $\mathbb{K} \neq \mathbb{Q}$ is similar.

Theorem 6.8. *Fix an extension field $\mathbb{F}_q$ of degree κ over $\mathbb{F}_p$ where p is the characteristic of q and let*

$$\mathbb{i} = (M, B_{\mathsf{code}}, n, k_1, k_2, \varphi, B, \beta, (z, \alpha, q)),$$

and suppose the code $\mathcal{C}$ generated by M is such that $\mathcal{C}^{dk_2}$ is (L, β) list-decodable. Suppose $\Pi_{\mathsf{TE}}(K, C)$ is an IOP for $\mathsf{REL}_{\mathsf{TE}}$ with round complexity $\mathsf{rd}_{\mathsf{TE}}$ and round-by-round knowledge soundness

$$(\mathsf{err}_1^{\mathsf{TE}}(K, C), \dots, \mathsf{err}_{\mathsf{rd}_{\mathsf{TE}}}^{\mathsf{TE}}(K, C))$$

. Then $\Pi_{\mathsf{PMLE}}(K, C, \mathcal{P})$ is an IOP for $\mathsf{REL}_{\mathsf{PMLE},\mathbb{Q}}$ with the following guarantees.

- ***Round complexity.*** $1 + \mathsf{rd}_{\mathsf{TE}}$,
- ***Completeness.*** *Suppose $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{PMLE},\mathbb{K}}$ with $\mathbb{w} = W \in \left(\mathbb{Q}^{<d,B_0}[X]\right)^{k_2 \times k_1}$ and $WM^T = V$. Then the verifier accepts with probability 1.*
- ***Round-by-round knowledge soundness.*** $\big(L\varepsilon, \mathsf{err}_1^{\mathsf{TE}}, \dots, \mathsf{err}_{\mathsf{rd}_{\mathsf{TE}}}^{\mathsf{TE}}\big)$, *where* $\varepsilon = \frac{d+2\kappa}{2^K} + \frac{dB+\kappa\log(p)}{K2^K}$.

Proof. The round complexity is immediate from the protocol.

For completeness, suppose $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{PMLE},\mathbb{F}_q}$ with witness W satisfying as described in the completeness case. Then one can check that the honest prover's messages indeed satisfy the equality checks in steps 3 and 5. It remains to check that W satisfies the completeness case of

Theorem 6.5 with index and input as in step 5. First, by the completeness case here, we have that $V = (v_1, \ldots, v_{k_2})$ satisfies $V = WM^T$. Next, the bitsize bounds for W are the same here as in Theorem 6.5, and finally the tensor constraint $a' \equiv \psi_{m,\xi}(u_2')^T \phi_m(W) \psi_{m,\xi}(u_1') \pmod m$ due to the honest prover's response in step 4.

We conclude by checking round-by-round knowledge soundness. After the verifier's message in step 3, the knowledge state, KState_1 and extractor are as follows. Given transcript $\tau = (b, m, \xi)$, we have $\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau, \mathbb{w}) = 1$ if and only if $\mathbb{w} = W \in \left(\mathbb{Q}^{<d}[X]\right)^{k_2 \times k_1}$ for W such that the equality check in Step 3 passes, W is a valid witness for the derived $\mathsf{REL}_{\mathsf{TE}}$-instance in Step 4. Namely, this means $V = WM^T$, $W \in \left(\mathbb{Q}^{<d,B_0}[X]\right)^{k_2 \times k_1}$, $a' = \psi_{m,\xi}(u_2')^T \phi_m(W) \psi_{m,\xi}(u_1')$, where this equality is over $\mathbb{F}_m$. The extractor is the trivial one which simply outputs the knowledge witness, $\mathsf{E}_1(\mathbb{i}, \mathbb{x}, \tau, \mathbb{w}) = \mathbb{w}$. Fix a W such that $\mathsf{dist}(WM^T, V) \leq \beta$ and note that by the list-decodability of $\mathcal{C}^{dk_2}$, there are at most L such W's. Here, we are viewing each coefficient vector of V as an interleaving, so that V is viewed as a (purported) dk_2 wise interleaved encoding.

The following, combined with a union bound over all L possible witnesses, is sufficient for showing round-by-round knowledge soundness,

$$\Pr_{\xi \in [2^K], m \in \mathcal{P}}[(\mathbb{i}, \mathbb{x}, W) \notin \mathsf{REL}_{\mathsf{PMLE}} \wedge (\mathbb{i}_{\mathsf{TE}}, \mathbb{x}_{\mathsf{TE}}) \in \mathsf{REL}_{\mathsf{TE}} \wedge \psi_q(a) = \alpha] \leq \frac{d + 2\kappa}{2^K} + \frac{dB + \kappa \log(p)}{K|\mathcal{P}|}.$$

It remains to prove the above for a fixed W. First note that if $(\mathbb{i}, \mathbb{x}, W) \notin \mathsf{REL}_{\mathsf{PMLE}}$ then the probability above is clearly 0, so we assume this is not the case. Then the event above can occur only if the following holds. There exists $W' \in \left(\mathbb{Q}^{<d,B_0}[X]\right)^{k_2 \times k_1}$ such that $\mathsf{dist}(W'M^T, V) \leq \beta$, $u_2'^T W' u_1' \neq a'$, and $\phi_{m,\xi}(u_2'^T W' u_1') = \phi_{m,\xi}(a')$. We bound the probability over m, ξ of this occurring. First, note that both $u_2'^T W' u_1'$ and a' are in $\mathbb{Q}^{<d+2\kappa, 2(B_0 + \log p) + \log(dk_1k_2)}[X]$. Then, the probability that they are not equal over $\mathbb{Q}[X]$ but are equal after reducing modulo a prime m of at least K bits is at most

$$\frac{2(B_0 + \log p) + \log(dk_1k_2)}{K|\mathcal{P}|},$$

and conditioned on this, the reduced polynomials are equal at $\xi \in \mathbb{F}_m$ with probability at most $(d+2\kappa)/2^K$ by the Schwartz-Zippel lemma. Overall, this shows that the round-by-round knowledge soundness error is at most $\frac{d+2\kappa}{2^K} + \frac{2(B_0 + \log p) + \log(dk_1k_2)}{K|\mathcal{P}|}$ for this step.

Finally, the round-by-round knowledge soundness of the remaining rounds follows from the observation that for any $\mathbb{w}$ such that $\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau, \mathbb{w}) = 0$, we have $(\mathbb{i}_{(TE)}, \mathbb{x}_{\mathsf{TE}}, \mathbb{w})$ is not in the derived relation $\mathsf{REL}_{\mathsf{TE}}$ for any prover message sent in step 4. Hence, in step 5 we may define the knowledge state to be membership in $\mathsf{REL}_{\mathsf{TE}}$ with the derived index and input, and have the extractor be the trivial extractor. The remainder of the round-by-round knowledge soundness follows from that of $\Pi_{\mathsf{TE}}(K, C)$ in Theorem 6.5. □

6.4 Random Linear Combinations of Rational Numbers

In this section we show necessary lemmas regarding random linear combinations of rational numbers. Similar results are shown in [Gar+25; GWHD25], but there the results are tailored for IOPPs where the honest prover only uses integral witnesses. For our purposes, we will also want to allow for provers who use bounded degree rational witnesses and hence more results on the bitlengths of random linear combinations of rational numbers are needed.

Our first lemma bounds the bitlength of sums of rationals.

Lemma 6.9. *For any $a_1, \ldots, a_m \in \mathbb{Q}$, we have*

$$\mathsf{bitlen}\left(\sum_{i=1}^{m} a_i\right) \leq \log(m) + \left(2\sum_{i=1}^{m} \mathsf{bitlen}(a_i)\right) - \min_{i\in[m]} \mathsf{bitlen}(a_i).$$

In particular, when $m = 2$ and $\mathsf{bitlen}(a_1), \mathsf{bitlen}(a_2) \leq B$, we have

$$\mathsf{bitlen}\,(a_1 + a_2) \leq 3B + 1.$$

Also, as an immediate consequence, if $u_1, \ldots, u_m \in \left(\mathbb{Q}^{<B}\right) k$, then

$$u_1 + \cdots + u_m \in \left(\mathbb{Q}^{<2mB+\log(m)}\right)^k.$$

Proof. Write $a_i = p_i/q_i$ for coprime integers p_i, q_i and let $j_i = \log|p_i|$, $j_i' = \log|q_i|$, so that $\mathsf{bitlen}(a_i) = j_i + j_i'$. Then letting $J = \sum_{i=1}^m j_i$ and $J' = \sum_{i=1}^m j_i'$,

$$\sum_{i=1}^{m} a_i = \frac{\sum_{i=1}^m p_i \prod_{k\neq i} q_k}{\prod_{k=1}^m q_k}.$$

Each numerator term satisfies $|p_i \prod_{k\neq i} q_k| \leq 2^{J'+j_i-j_i'}$ and $|\prod_k q_k| \leq 2^{J'}$, so $\mathsf{bitlen}\,(\sum_{i=1}^m a_i) \leq \log\left(\sum_{i=1}^m 2^{J'+j_i-j_i'}\right) + J'$. Now let $i \in [m]$ be an index that maximizes $j_i - j_i'$. Then

$$\begin{aligned}
\mathsf{bitlen}\left(\sum_{i=1}^{m} a_i\right) &\leq J' + \log(m \cdot 2^{J'+j_i-j_i'}) \\
&= 2J' + j_i - j_i' + \log(m) \\
&\leq \log(m) + \left(2\sum_{i=1}^{m} \mathsf{bitlen}(a_i)\right) - \min_{i\in[m]} \mathsf{bitlen}(a_i).
\end{aligned}$$

The two consequences are straightforward to see. □

Lemma 6.10. *For $u \in \mathbb{Q}$ and $r \in \mathbb{Z}$,*

$$\mathsf{bitlen}(r \cdot u) \geq \mathsf{bitlen}(u) - \mathsf{bitlen}(r).$$

Proof. Write $u = a/b$ for coprime integers a and b and let $c = \gcd(b, r)$. Then, $r \cdot u = \frac{a \cdot r/c}{b/c}$ and

$$\mathsf{bitlen}(r \cdot u) = \log(a) + \log(r) + \log(b) - 2\log(c) \geq \log(a) + \log(b) - \log(r) = \mathsf{bitlen}(u) - \mathsf{bitlen}(r),$$

where we use the fact that $|c| \leq |r|$. □

The next lemma shows that a random linear combination of two rationals, one of which has large bitlength, is unlikely to have small bitlength.

Lemma 6.11. *For any $u, v \in \mathbb{Q}$ and $B, K \in \mathbb{N}$ such that $\mathsf{bitlen}(u) \geq 3B + K + 1$,*

$$\Pr_{r\in[2^K]}[\mathsf{bitlen}(ru + v) \leq B] \leq \frac{1}{2^K}.$$

Proof. Suppose for the sake of contradiction there are two distinct $r_1, r_2 \in [2^K]$ such that $\mathsf{bitlen}(r_1 u + v), \mathsf{bitlen}(r_2 u + v) \leq B$. Then by Lemma 6.9 $\mathsf{bitlen}((r_1 - r_2)u) \leq 3B + 1$ and by Lemma 6.10 $\mathsf{bitlen}((r_1 - r_2)u) \geq \mathsf{bitlen}(u) - \mathsf{bitlen}(r_1 - r_2)$. Together with the fact that $\mathsf{bitlen}(u) \geq 3B + K + 1$, we get

$$3B + 1 + \mathsf{bitlen}(r_2 - r_1) \geq \mathsf{bitlen}(u) \geq 3B + K + 1,$$

and

$$\mathsf{bitlen}(r_2 - r_1) \geq K.$$

However, this implies that $|r_2 - r_1| \geq 2^K$, which is a contradiction. Hence, there can only be at most one coefficient $r \in [2^K]$ satisfying $\mathsf{bitlen}(ru + v) \leq B$ and the lemma follows. □

The following two lemmas are immediate consequences of Lemma 6.11.

Lemma 6.12. *For any positive integers D, B, K and vectors $w_1, \ldots, w_D \in \mathbb{Q}^k$ such that for some $i \in [D]$, $w_i \notin \left(\mathbb{Q}^{<3B+K+1}\right)^k$, the following holds:*

$$\Pr_{r_1,\ldots,r_D \in [2^K]} \left[\sum_{i=1}^{D} r_i \cdot w_i \in \left(\mathbb{Q}^{<B}\right)^k \right] \leq \frac{1}{2^K}.$$

Proof. Without loss of generality, let w_1 be a vector not in $\left(\mathbb{Q}^{<3B+K+1}\right)^k$. Choose $r_2, \ldots, r_D \in [2^K]$ first and set $w' = r_2 \cdot w_2 + \cdots + r_D \cdot w_D$. By Lemma 6.11, for any choice of $r_2, \ldots, r_D$, we have

$$\Pr_{r_1 \in [2^K]} [r_1 w_1 + w' \in \left(\mathbb{Q}^{<B}\right)^k] \leq \frac{1}{2^K},$$

and the lemma follows. □

Finally, we have the following lemma which states that if a rational number has denominator divisible by a large prime, then any random linear combination involving it is also likely to have denominator divisible by that prime. This lemma will be useful when we want to constrain a rational code using constraints modulo some large prime.

Lemma 6.13. *Let $u \in \mathbb{Q}$ be a rational number such that $u = a/b$ for coprime integers a and b, and suppose $p \mid b$. Then for any $v \in \mathbb{Q}$ and $M \in \mathbb{N}$ such that $M \leq p$, we have*

$$\Pr_{r \in [M]} [v + ru \text{ has denominator not divisible by } p] \leq \frac{1}{M},$$

where in the probability above we mean the denominator of $v + ru$ when written in simplest form.

Proof. Write $v = c/d$ for coprime integers c and d. Also let us write $b = p^i b'$ where i is the largest integer such that $p^i | b$ and $p \nmid b'$, and similarly let us write $c = p^j c'$ and $d = p^\ell d'$ so that p^j and p^ℓ are the largest powers of p dividing c and d respectively. Note that $p \nmid a$ because a and b are coprime.

Then, for any $r \in [M]$ we have

$$v + ru = \frac{bc + r \cdot ad}{bd} = \frac{p^{i+j} b'c' + r \cdot p^\ell ad'}{p^{i+\ell} b'd'}.$$

Now let $r_1, r_2 \in [M]$ be two distinct coefficients such that both $v + r_1 u$ and $v + r_2 u$ have denominator not divisible by p. Then $p^{i+\ell} \mid p^{i+j} b'c' + r_s \cdot p^\ell ad'$ for each $s \in \{1, 2\}$, and hence

$$p^\ell ad'(r_1 - r_2) \equiv 0 \pmod{p^{i+\ell}}.$$

Since a, d' are not divisible by p and $i > 0$ by assumption, it follows that $p \mid (r_1 - r_2)$. Since $|r_1 - r_2| \leq M - 1 < p$, it follows that $r_1 = r_2$, and there can only be one coefficient $r \in [M]$ such that $v + ru$ has denominator not divisible by p. Hence, the probability above is at most $1/M$ and the lemma follows. □

7 IPRS codes: general radix algorithms and proofs

In this section we provide the general radix-r IPRS encoding algorithm, and the corresponding formal definition of IPRS codes, complementing the radix-2 description in Section 2.3. We also state and prove Theorem 7.3, a general-radix version of Theorem 2.14, and provide the full proof of Lemma 2.12.

7.1 General IPRS encoder

The IPRS encoder follows the recursive, divide-and-conquer structure of a radix-r FFT, but executes all arithmetic over the rationals without any modular reduction.

We identify elements of $\mathbb{F} = \mathbb{F}_q$ with their *centered* integer representatives: define $\iota : \mathbb{F} \to \mathbb{Z}$ by sending each field element to the unique integer in $\{-(q-1)/2, \ldots, (q-1)/2\}$ congruent to it modulo q. Recall that $\phi_q : \mathbb{Z} \to \mathbb{F}$ denotes the reduction map modulo q, extended entrywise to vectors; we also extend ϕ_q to $\mathbb{Q}$ by mapping $a/b \mapsto \phi_q(a)/\phi_q(b)$ when $\gcd(b, q) = 1$.

Algorithm 5 Rational FFT (IPRS encoder)

Parameters. A prime q and a field $\mathbb{F}$ with q elements; a code rate ρ; integers $\gamma \geq 1$, $\beta \geq 1$, $k \geq 0$; a code dimension $m := 2^{\gamma+\beta k}$ and code length $n := m/\rho$; a primitive n-th root of unity $\omega \in \mathbb{F}$; an evaluation domain $\mathcal{L} = (1, \omega, \omega^2, \ldots, \omega^{n-1})$; a radix $r := 2^\beta$; a base-case size $m_0 := 2^\gamma$; the lifting map $\iota : \mathbb{F} \to \mathbb{Z}$ and the reduction map $\phi_q : \mathbb{Q} \to \mathbb{F}$. We assume n is a power of two and n divides $q - 1$.

Pre-computed constants. The following are fixed before encoding:

- **Twiddle factors:** For each power $\omega^j \in \mathbb{F}$, the lifted integer $\iota(\omega^j) \in \mathbb{Z}$, $j \in [n]$.
- **Base-case matrix entries:** The entries $\iota(\omega^{ij})$ of the lifted base-case Vandermonde matrix, $j \in [0\,..\,m_0 - 1]$, $i \in [0\,..\,n - 1]$.

Input. A message $\mathbf{x} = (x_0, \ldots, x_{m-1}) \in \mathbb{Q}^m$.
Output. The IPRS encoding $\mathsf{Enc}_{\mathsf{IPRS}}(\mathbf{x}) \in \mathbb{Q}^n$.

Algorithm

1: **function** $\mathsf{RatFFT}(\mathbf{x}, \omega, n)$
2: **if** $|\mathbf{x}| \leq m_0$ **then**
3: For each $i \in [0\,..\,n-1]$, compute $y_i \leftarrow \sum_{j \in [0\,..\,m_0-1]} x_j \cdot \iota(\omega^{ij})$ over $\mathbb{Q}$.
4: **return** $(y_0, \ldots, y_{n-1})$
5: **else**
6: Decompose $\mathbf{x}$ into $r = 2^\beta$ messages $\mathbf{x}_0, \ldots, \mathbf{x}_{r-1}$ of length m/r each, corresponding to the decomposition of the polynomial $f(X) = \sum_{i=0}^{m-1} x_i X^i$ with coefficients $\mathbf{x}$ into r sub-polynomials of degree $< m/r$:

$$f(X) \;=\; \sum_{s=0}^{r-1} X^s \cdot f_s(X^r), \quad \text{where } f_s(Y) := \sum_{t=0}^{m/r-1} x_{tr+s}\, Y^t.$$

7: Recursively compute $\mathsf{RatFFT}(\mathbf{x}_s, \omega^r, n/r)$ for each $s \in \{0, \ldots, r-1\}$. The evaluation domain for each recursive call is the set of (n/r)-th powers $\{1, \omega^r, \omega^{2r}, \ldots\}$.

8: **Butterfly.** Combine the r recursive results. For each $i \in [0\,..\,n-1]$, compute

$$y_i \;\leftarrow\; \sum_{s=0}^{r-1} \iota(\omega^{is}) \cdot f_s(\omega^{ir}) \quad \text{over } \mathbb{Q}.$$

9: **return** $(y_0, \ldots, y_{n-1})$

10: **end if**

Crucially, *no modular reduction is performed at any step of the encoding*, even though the twiddle factors and base-case matrix entries are initially lifted from a finite field. The encoding itself consists purely of rational additions and multiplications (which becomes integer arithmetic when the input is integral).

7.2 Formal definition and properties of IPRS codes

We are ready to define general IPRS codes and to state their properties. Throughout this subsection, $\|\cdot\|_\infty$ denotes the ℓ_∞ norm.

Definition 7.1 (Integer Pseudo Reed–Solomon (IPRS) code, general definition)**.** Assume the parameters and notation of Algorithm 5. The *Integer Pseudo Reed–Solomon encoder* $\mathsf{Enc}_{\mathsf{IPRS}} : \mathbb{Q}^m \to \mathbb{Q}^n$ is the $\mathbb{Q}$-linear map obtained by executing Algorithm 5. The *Integer Pseudo Reed–Solomon code* associated to these parameters is

$$\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}, \mathcal{L}, m] \;:=\; \left\{\mathsf{Enc}_{\mathsf{IPRS}}(\mathbf{x}) \mid \mathbf{x} \in \mathbb{Q}^m\right\} \;\subseteq\; \mathbb{Q}^n.$$

Note that IPRS codes are not Reed–Solomon codes: indeed, their codewords have a priori little relation to polynomial evaluations over $\mathbb{Z}$ (albeit they *are* polynomial evaluations when reduced modulo q).

We now state the lifting lemma underlying our construction and the general properties of IPRS codes.

Lemma 7.2 (Restatement of Lemma 2.12: lifts of linear codes preserve dimension and distance)**.** *Let q be a prime, let $\mathcal{C}_q \subseteq \mathbb{F}_q^n$ be a linear code of dimension k over $\mathbb{F}_q$, and let $M_q \in \mathbb{F}_q^{n\times k}$ be a generator matrix of $\mathcal{C}_q$. Let $\hat{M}_q \in \mathbb{Z}^{n\times k}$ be any integer matrix obtained by lifting each entry of M_q to an integer representative. Then*

$$\mathcal{C}_{\mathbb{Q}}[\hat{M}_q] := \left\{\hat{M}_q \cdot x \;\middle|\; x \in \mathbb{Q}^k\right\} \subseteq \mathbb{Q}^n$$

is a linear code over $\mathbb{Q}$ of dimension k and minimum distance at least the minimum distance of $\mathcal{C}_q$.

Proof. Assume towards contradiction that the maximal minors of $\hat{M}_q$ are zero in $\mathbb{Q}$. Then all maximal minors of M_q are zero in $\mathbb{F}_q$, hence M_q does not have full rank k in $\mathbb{F}_q$, contradicting the fact that M_q is a generator matrix of $\mathcal{C}_q$. Thus $\hat{M}_q$ has full rank k over $\mathbb{Q}$, and $\mathcal{C}_{\mathbb{Q}}[\hat{M}_q]$ has dimension k.

To prove the claim about minimum distance, consider first $x \in \mathbb{Z}^k \setminus \{0\}$ (as opposed to $x \in \mathbb{Q}^k \setminus \{0\}$) and write $x = q^t x'$ with $t \geq 0$ maximal, so that $x' \in \mathbb{Z}^k$ and x' is not the zero vector modulo q. Then $\hat{M}_q \cdot x = q^t \hat{M}_q \cdot x'$ has the same Hamming weight as $\hat{M}_q \cdot x'$. Let $\bar{x}' \in \mathbb{F}_q^k$ denote x' reduced modulo q. The reduction of $\hat{M}_q \cdot x'$ modulo q equals $M_q \cdot \bar{x}'$ modulo q. The latter is

a nonzero codeword of $\mathcal{C}_q$ (since M_q has full rank k over $\mathbb{F}_q$ and $\bar{x}' \neq 0$ in $\mathbb{F}_q$). Since reduction modulo q can only decrease Hamming weight, $\mathsf{wt}(\hat{M}_q \cdot x) = \mathsf{wt}(\hat{M}_q \cdot x') \geq \mathsf{wt}(M_q \cdot \bar{x}')$, and since $\bar{x}' \neq 0$, the latter is at least the minimum distance $d(\mathcal{C}_q)$ of $\mathcal{C}_q$. Here we use wt to denote Hamming weight, and $d(\mathcal{C}_q)$ to denote the minimum distance of $\mathcal{C}_q$.

For a general $x \in \mathbb{Q}^k \setminus \{0\}$, let $D \in \mathbb{Z}$ be the least common multiple of the denominators of the entries of x (in lowest form), so that $Dx \in \mathbb{Z}^k \setminus \{0\}$. Then $\hat{M}_q \cdot x = \frac{1}{D}\hat{M}_q \cdot Dx$, and since $Dx \neq 0$, the previous case gives $\mathsf{wt}(\hat{M}_q \cdot Dx) \geq d(\mathcal{C}_q)$. Scaling by $1/D$ does not change the support, so $\mathsf{wt}(\hat{M}_q \cdot x) = \mathsf{wt}(\hat{M}_q \cdot Dx) \geq d(\mathcal{C}_q)$. □

Theorem 7.3 (Properties of IPRS codes, general radix; generalization of Theorem 2.14)**.** *Assume the parameters and notation of Algorithm 5, and let d denote the recursion depth of the encoder, so that $m = m_0 \cdot r^d$. The code $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}_q, \mathcal{L}, m]$ is a linear code over $\mathbb{Q}$ of dimension m, blocklength n, and minimum relative distance $\delta = 1 - m/n + 1/n$. Moreover, for each $x \in \mathbb{Q}^m$,*

$$\|\mathsf{Enc}_{\mathsf{IPRS}}(x)\|_\infty \leq \|x\|_\infty \cdot (q/2)^{d+1} \cdot m.$$

Proof. We establish two claims: (1) $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}_q, \mathcal{L}, m]$ is a lift of $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, m]$, in the sense of Lemma 2.12, and hence inherits its dimension and distance by that lemma; (2) the norm bound holds.

Dimension and minimum distance. Since $\mathsf{Enc}_{\mathsf{IPRS}}$ performs recursively only additions and multiplications by fixed integer constants (the lifted twiddle factors $\iota(\omega^j)$ and base-case Vandermonde entries), it is a $\mathbb{Q}$-linear map. Hence there exists a matrix $\hat{M}_q \in \mathbb{Z}^{n \times m}$ such that $\mathsf{Enc}_{\mathsf{IPRS}}(x) = \hat{M}_q\, x$ for all $x \in \mathbb{Q}^m$.

We claim that

$$\phi_q(\mathsf{Enc}_{\mathsf{IPRS}}(x)) \;=\; \mathsf{Enc}_{\mathsf{RS}}(\phi_q(x)) \qquad \text{for all } x \in \mathbb{Z}^m, \tag{43}$$

where $\mathsf{Enc}_{\mathsf{RS}} : \mathbb{F}_q^m \to \mathbb{F}_q^n$ denotes the standard radix-r FFT encoder for $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, m]$ (using the same parameters as in the statement of the theorem) and $\phi_q : \mathbb{Z} \to \mathbb{F}_q$ is reduction modulo q. To see this, observe that ϕ_q is a ring homomorphism, so it commutes with the additions and multiplications performed during the FFT recursion. Moreover, every lifted twiddle factor satisfies $\phi_q(\iota(\omega^j)) = \omega^j$, and similarly for the base-case Vandermonde entries. Therefore, reducing each intermediate value modulo q at every step of $\mathsf{Enc}_{\mathsf{IPRS}}$ recovers exactly the computation of $\mathsf{Enc}_{\mathsf{RS}}$, which yields (43).

Now, $\mathsf{Enc}_{\mathsf{RS}}(y) = V_q\, y$ where V_q is the $n \times m$ Vandermonde matrix with (j,i)-entry ω^{ji}, a generator matrix of $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, m]$. Applying (43) to the standard basis vectors $e_i \in \mathbb{Z}^m$ (for which $\phi_q(e_i) = e_i$ since $q \geq 2$) gives $\phi_q(\hat{M}_q\, e_i) = V_q\, e_i$ for each i, and so $\phi_q(\hat{M}_q) = V_q$. Hence $\hat{M}_q$ is an integer lift of V_q in the sense of Lemma 7.2, and so $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}_q, \mathcal{L}, m] = \mathcal{C}_{\mathbb{Q}}[\hat{M}_q]$ in the notation of Lemma 7.2. By Lemma 2.12, $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}_q, \mathcal{L}, m]$ has dimension m and minimum relative distance at least $1 - m/n + 1/n$, the same as $\mathsf{RS}[\mathbb{F}_q, \mathcal{L}, m]$.

Norm bound. We prove by induction on the recursion depth d that, when the encoder is run with base-case dimension m_0 and radix r at depth d on a message $x \in \mathbb{Q}^{m_d}$ of dimension $m_d := m_0 \cdot r^d$,

$$\|\mathsf{Enc}_{\mathsf{IPRS}}(x)\|_\infty \;\leq\; \|x\|_\infty \cdot (q/2)^{d+1} \cdot m_d.$$

Setting d to the actual recursion depth gives $m_d = m$ and recovers the bound in the statement.

Base case ($d = 0$). The encoder performs a direct matrix–vector multiplication with the $n_0 \times m_0$ lifted base-case Vandermonde matrix, where $n_0 = m_0/\rho$. Each entry of this matrix has magnitude at most $(q-1)/2 < q/2$, and each row has m_0 nonzero entries. Hence $\|\mathsf{Enc}_{\mathsf{IPRS}}(x)\|_\infty \leq m_0 \cdot (q/2) \cdot \|x\|_\infty$, matching the claim for $d = 0$.

Inductive step. At depth $d \geq 1$, the message $x \in \mathbb{Q}^{m_d}$ is split into r messages $x_0, \ldots, x_{r-1} \in \mathbb{Q}^{m_{d-1}}$, each satisfying $\|x_s\|_\infty \leq \|x\|_\infty$. Each message is encoded recursively at depth $d-1$. By the induction hypothesis, the recursive outputs $x'_0, \ldots, x'_{r-1} \in \mathbb{Q}^{n/r}$ satisfy

$$\left\|x'_s\right\|_\infty \;\leq\; \|x\|_\infty \cdot (q/2)^d \cdot m_{d-1} \qquad \text{for all } s \in \{0, \ldots, r-1\}.$$

The butterfly stage combines them as

$$y_i \;=\; \sum_{s=0}^{r-1} \iota(\omega^{is}) \cdot (x'_s)_{i \bmod n/r}.$$

Since $\iota(\omega^0) = 1$ and $|\iota(\omega^{is})| \leq (q-1)/2 < q/2$ for $s \geq 1$,

$$\|y\|_\infty \;\leq\; \left\|x'_0\right\|_\infty + (r-1)(q/2) \cdot \max_{s \geq 1} \left\|x'_s\right\|_\infty \;\leq\; \big(1 + (r-1)(q/2)\big) \cdot \|x\|_\infty \cdot (q/2)^d \cdot m_{d-1}.$$

Since $1 + (r-1)(q/2) \leq r \cdot (q/2)$ for every $r \geq 2$ and $q \geq 2$, we obtain

$$\|y\|_\infty \;\leq\; r \cdot (q/2) \cdot \|x\|_\infty \cdot (q/2)^d \cdot m_{d-1} \;=\; \|x\|_\infty \cdot (q/2)^{d+1} \cdot m_d,$$

completing the induction. □

7.3 Open questions

Reed–Solomon codes over finite fields are known to enjoy remarkable properties in the context of SNARK systems: among others, they achieve Mutual Correlated Agreement (MCA) up to the Johnson bound [Hab25; BCGM25]. On the other hand, general linear codes over finite fields are only known to achieve MCA up to the so-called 1.5 Johnson bound [Zei24]. In Section 6 we generalize the latter result to $\mathbb{Q}$.

We pose the following question:

Question 7.4 (MCA up to the Johnson bound for IPRS codes (informal))**.** *Let* $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}, \mathcal{L}, m]$ *be an IPRS code as in Definition 7.1. Then* $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}, \mathcal{L}, m]$ *achieves MCA up to the Johnson bound* $1 - \sqrt{m/n}$*.*

A related question is whether IPRS codes are algorithmically (efficiently) list-decodable up to the Johnson bound, as RS codes are.

Question 7.5. *Let* $\mathsf{IPRS}_{\beta,\gamma}[\mathbb{F}, \mathcal{L}, m]$ *be an IPRS code as in Definition 7.1. Let* $q' > q$ *be a prime different from* q*. Consider a new code over* $\mathbb{F}_{q'}$ *defined as the image of the IPRS encoder* $\mathsf{Enc}_{\mathsf{IPRS}}$ *modulo* q'*. What is the minimum distance of the resulting code?*

The above is motivated by the following scenario: assume one wishes to or must work on a large prime field $\mathbb{F}_{q'}$, but on the other hand the witness elements are significantly smaller than q'. A problem with using RS codes in this setting is that encoded witnesses suffer a large norm blowout, similarly to naïvely using RS codes over $\mathbb{Z}$, as discussed in Section 2.3. One may instead use an IPRS code, as the variation defined above, with a small base field and small depth, and retain small norm increases while still having an FFT encoder. However, we do not know whether the resulting code has good minimum distance.

8 Arithmetization case study: SHA-256 + ECDSA verification

In this section we arithmetize the computation consisting of hashing a message with the SHA-256 hash function, and verifying an ECDSA signature over the message digest, over the secp256k1 curve.

We begin by describing a specific subclass of the UCS relation, which we call *Universal AIRs with lookups* (UAIR$^+$), and then show how to express the aforementioned computation as a UAIR$^+$ instance.

8.1 Universal AIR with lookups (UAIR$^+$), a specialization of UCS

The general definition of UCS (cf. Sections 2.1 and 4.1) is designed to be as general and all-encompassing as possible. In practice, we expect practitioners to use more restricted subclasses of UCS, e.g. AIR-like or R1CS/CCS-like subsets of UCS (expressed over multiple rings, with ideal membership predicates, and with lookups). In this section we describe one such subclass, which we call *Universal AIRs with lookups* (UAIR$^+$).

UAIR+ is an AIR-like [Sta21] relation that: 1) allows having multiple traces typed in different rings of the form $\mathbb{Q}[X], \mathbb{F}_{q_1}[X], \ldots, \mathbb{F}_{q_{|\mathbf{q}|}}[X]$, 2) allows the use of columns from the trace over $\mathbb{Q}[X]$ as columns in the traces over $\mathbb{F}_{q_i}[X]$ (after projecting them with the morphism $\phi_{q_i} : \mathbb{Z}_{(q_i)}[X] \to \mathbb{F}_{q_i}[X]$), and 3) expresses transition constraints as ideal membership constraints, instead of strict equalities.

UAIR$^+$ does not introduce new expressive power relative to UCS: any UAIR$^+$ instance can be written as a UCS instance (cf. Lemma 8.6). We focus on UAIR$^+$ in this paper because it aligns well with the use-case we consider, i.e. SHA-256 and ECDSA. However, working with an R1CS-like or CCS-like subclass of UCS would also be perfectly reasonable.

Definition 8.1 (UAIR$^+$ relation)**.** The relation $\mathsf{REL}_{\mathsf{UAIR+}}$ consists of all triples $(\mathtt{i}, \mathtt{x}; \mathtt{w})$ of the following form.

Index. An index $\mathtt{i}$

$$\mathtt{i} = (n, c, \mathbf{q}, B, \mathbf{d}, \mathcal{C}, \mathcal{S}),$$

where $n \geq 1$ is the number of rows, $c \geq 1$ is the number of columns, $\mathbf{q} = (q_1, \ldots, q_{|\mathbf{q}|})$ is a tuple of prime powers, B is a bit-size bound, $\mathbf{d} = (d_0, \ldots, d_{|\mathbf{q}|})$, $\mathcal{C}$ is a tuple of constraints (called *transition constraints*, described below), and $\mathcal{S}$ is a tuple of lookup sets (also described below).

Input and witness. The public input is a trace (i.e., a matrix) $\mathtt{x} = \mathbf{y}$ with

$$\mathbf{y} \in \left(\mathbb{Z}^{<d_0, B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X]\right)^{n \times c}.$$

In practice, the public-input matrix $\mathbf{y}$ is sparse (e.g., most entries are zero or a fixed constant), and the verifier can evaluate the multilinear extensions of each column of $\mathbf{y}$ directly without prover assistance.

The witness is a tuple of traces $\mathtt{w} = (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|})$ with

$$\mathbf{f}_0 \in \left(\mathbb{Q}^{<d_0, B}[X]\right)^{n \times c}, \qquad \mathbf{f}_i \in \left(\mathbb{F}^{<d_i}_{q_i}[X]\right)^{n \times c} \quad \text{for} \quad i \in [|\mathbf{q}|].$$

Constraints and constraint satisfaction. For a matrix $\mathbf{v} \in \{\mathbf{y}, \mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|}\}$, we write $\mathbf{v}_j[k]$ for its entry at row $k \in [n]$ and column $j \in [c]$. We define the *row vector*

$$\mathbf{v}[k] := (\mathbf{v}_1[k], \ldots, \mathbf{v}_c[k]).$$

For a witness matrix $\mathbf{f}_i$ and a row index $k \in [n-1]$, the *shifted row vector* is defined as

$$\mathbf{f}_i^{\downarrow}[k] := \mathbf{f}_i[k+1] = (\mathbf{f}_{i,1}[k+1], \ldots, \mathbf{f}_{i,c}[k+1]),$$

and we set $\mathbf{f}_i^{\downarrow}[n] = \mathbf{0}$. Other approaches are possible, see e.g. [Tho24].

Let $\mathbf{Y}$ be a tuple of c formal variables representing a row of the public-input matrix (see below). Let $\mathbf{Z}_0, \mathbf{Z}_1$ be tuples of c formal variables each. We interpret $\mathbf{Z}_0$ as a placeholder for a row of the $\mathbb{Q}[X]$-trace $\mathbf{f}_0$ (and its projections under ϕ_{q_i}), and similarly for $\mathbf{Z}_1$ and the $\mathbb{F}_{q_i}[X]$-trace. We further let $\mathbf{Z}_0^{\downarrow}, \mathbf{Z}_1^{\downarrow}$ be two tuples of c formal variables each, which we interpret as placeholders for the shifted row vectors $\mathbf{f}_0^{\downarrow}[k]$ (possibly under a projection ϕ_{q_i}) and $\mathbf{f}_i^{\downarrow}[k]$, respectively.

The transition constraints $\mathcal{C}$ are formed by a tuple

$$\mathcal{C} = \Big((Q_{0t}, \mathfrak{I}_{0t}), ((Q_{it}, \mathfrak{I}_{it}))_{i \in [|\mathbf{q}|]} \Big)_{t \in [|\mathcal{C}|]},$$

where for every $t \in [|\mathcal{C}|]$ and every $i \in [|\mathbf{q}|]$,

$$Q_{0t} \in \big(\mathbb{Q}^{<d_0, B}[X]\big)[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_0^{\downarrow}] \qquad \text{and} \qquad \mathfrak{I}_{0t} \subseteq \mathbb{Q}[X] \text{ is an ideal,}$$

$$Q_{it} \in \big(\mathbb{F}_{q_i}^{<d_i}[X]\big)[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_0^{\downarrow}, \mathbf{Z}_1, \mathbf{Z}_1^{\downarrow}] \qquad \text{and} \qquad \mathfrak{I}_{it} \subseteq \mathbb{F}_{q_i}[X] \text{ is an ideal.}$$

Finally, $\mathcal{S} = (S_{ij})_{i \in \{0\} \cup [|\mathbf{q}|],\, j \in [c]}$ is a collection of *column-wise* lookup sets satisfying the following validity conditions for every $j \in [c]$ (recall that $\mathbb{Z}_{(q_1 \cdots q_{|\mathbf{q}|})}$ denotes the intersection localization ring from Eq. (30)):

$$S_{0j} \subseteq \mathbb{Z}_{(q_1 \cdots q_{|\mathbf{q}|})}^{<d_0, B}[X], \qquad S_{ij} \subseteq \mathbb{F}_{q_i}^{<d_i}[X] \quad \text{for} \quad i \in [|\mathbf{q}|].$$

Constraint satisfaction We say that $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{UAIR+}}$ if all of the following conditions hold.

(i) **Transition constraints.** For every row index $k \in [n]$ and every constraint index $t \in [|\mathcal{C}|]$,

$$Q_{0t}\big(\mathbf{y}[k], \mathbf{f}_0[k], \mathbf{f}_0^{\downarrow}[k]\big) \in \mathfrak{I}_{0t},$$

and for every $k \in [n]$ and every prime power index $i \in [|\mathbf{q}|]$,

$$Q_{it}\big(\phi_{q_i}(\mathbf{y})[k], \phi_{q_i}(\mathbf{f}_0)[k], \phi_{q_i}(\mathbf{f}_0)^{\downarrow}[k], \mathbf{f}_i[k], \mathbf{f}_i^{\downarrow}[k]\big) \in \mathfrak{I}_{it},$$

where, recall, $\phi_{q_i} : \mathbb{Z}_{(q_i)}[X] \to \mathbb{F}_{q_i}[X]$ is the natural modular reduction from $\mathbb{Z}_{(q_i)}[X]$ to $\mathbb{F}_{q_i}[X]$.

(ii) **Column-wise lookups.** For every index $i \in \{0\} \cup [|\mathbf{q}|]$ and every column index $j \in [c]$,

$$\mathsf{set}(\mathbf{f}_{i,j}) \subseteq S_{ij},$$

where $\mathsf{set}(\mathbf{f}_{i,j})$ denotes the set of all entries in the vector $\mathbf{f}_{i,j}$.

For ease of reference, we restate the definition of $\mathsf{REL}_{\mathsf{UAIR}+}$ in Eq. (44).

$$
\mathsf{REL}_{\mathsf{UAIR}+} = \left\{ \begin{pmatrix} \mathbb{i}, \\ \mathbb{x}; \\ \mathbb{w} \end{pmatrix} \;\middle|\; \begin{array}{l}
\textbf{Index-input-witness} \\
\quad \textbf{Index} \\
\qquad \mathbb{i} = (n, c, \mathbf{q}, B, \mathbf{d}, \mathcal{C}, \mathcal{S}),\ \mathbf{q} = (q_1, \ldots, q_{|\mathbf{q}|}), \\
\qquad \mathbf{d} = (d_0, \ldots, d_{|\mathbf{q}|}), \\
\qquad \mathcal{C} = \Big((Q_{0t}, \mathfrak{I}_{0t}), ((Q_{it}, \mathfrak{I}_{it}))_{i \in [|\mathbf{q}|]} \Big)_{t \in [|\mathcal{C}|]}, \\
\qquad Q_{0t} \in \big(\mathbb{Q}^{<d_0, B}[X] \big)[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_0^{\downarrow}],\ \mathfrak{I}_{0t} \subseteq \mathbb{Q}[X], \\
\qquad Q_{it} \in \big(\mathbb{F}_{q_i}^{<d_i}[X] \big)[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_0^{\downarrow}, \mathbf{Z}_1, \mathbf{Z}_1^{\downarrow}],\ \mathfrak{I}_{it} \subseteq \mathbb{F}_{q_i}[X], \\
\qquad |\mathbf{Y}| = |\mathbf{Z}_0| = |\mathbf{Z}_0^{\downarrow}| = |\mathbf{Z}_1| = |\mathbf{Z}_1^{\downarrow}| = c, \\
\qquad \mathcal{S} = (S_{ij})_{i \in \{0\} \cup [|\mathbf{q}|],\ j \in [c]}, \\
\qquad S_{0j} \subseteq \mathbb{Z}_{(q_1 \cdots q_{|\mathbf{q}|})}^{<d_0, B}[X], \quad S_{ij} \subseteq \mathbb{F}_{q_i}^{<d_i}[X] \ \text{ for } \ i \in [|\mathbf{q}|],\ j \in [c], \\
\quad \textbf{Input} \\
\qquad \mathbb{x} = \mathbf{y} \in \big(\mathbb{Z}_{(q_1 \cdots q_{|\mathbf{q}|})}^{<d_0, B}[X] \big)^{n \times c}, \\
\quad \textbf{Witness} \\
\qquad \mathbb{w} = (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|}), \\
\qquad \mathbf{f}_0 \in \big(\mathbb{Q}^{<d_0, B}[X] \big)^{n \times c},\ \mathbf{f}_i \in \big(\mathbb{F}_{q_i}^{<d_i}[X] \big)^{n \times c} \ \text{ for } \ i \in [|\mathbf{q}|], \\
\\
\textbf{Constraints} \\
\quad \textbf{Transition constraints} \\
\qquad Q_{0t}\big(\mathbf{y}[k], \mathbf{f}_0[k], \mathbf{f}_0^{\downarrow}[k] \big) \in \mathfrak{I}_{0t}, \\
\qquad\quad \text{for all } k \in [n],\ t \in [|\mathcal{C}|], \\
\qquad Q_{it}\big(\phi_{q_i}(\mathbf{y})[k], \phi_{q_i}(\mathbf{f}_0)[k], \phi_{q_i}(\mathbf{f}_0)^{\downarrow}[k], \mathbf{f}_i[k], \mathbf{f}_i^{\downarrow}[k] \big) \in \mathfrak{I}_{it}, \\
\qquad\quad \text{for all } k \in [n],\ i \in [|\mathbf{q}|],\ t \in [|\mathcal{C}|], \\
\\
\quad \textbf{Column-wise lookup constraints} \\
\qquad \mathsf{set}(\mathbf{f}_{i,j}) \subseteq S_{ij} \quad \text{for all } i \in \{0\} \cup [|\mathbf{q}|],\ j \in [c].
\end{array} \right\}. \tag{44}
$$

We now discuss some extensions to UAIR$^+$ that are conceptually straightforward but notationally tedious. We use both extensions in our arithmetizations but omit them from Definition 8.1 to keep the notation lighter.

Remark 8.2 (Extension: Shift access to rows)**.** In some applications (e.g., our SHA-256 arithmetization) transition constraint polynomials from $\mathcal{C}$ need to access not only the immediate successor rows $\mathbf{f}_0^{\downarrow}[k], \mathbf{f}_i^{\downarrow}[k], \phi_{q_i}(\mathbf{f}_0)^{\downarrow}[k]$, but also rows at a fixed constant offset. Concretely, for different shift parameters $\Delta_1, \ldots, \Delta_k \geq 1$, one may allow constraints to depend on elements $\mathbf{f}_0[k], \mathbf{f}_0^{\downarrow\Delta_1}[k], \ldots, \mathbf{f}_0^{\downarrow\Delta_k}[k]$ (and their projections under ϕ_{q_i}), and $\mathbf{f}_i[k], \mathbf{f}_i^{\downarrow\Delta_1}[k], \ldots, \mathbf{f}_i^{\downarrow\Delta_k}[k]$, where $\mathbf{f}^{\downarrow\Delta}[k]$ is the $n \times c$ shifted trace defined as

$$
\mathbf{f}^{\downarrow\Delta}[k] \ := \ \begin{cases} \mathbf{f}[k + \Delta] & \text{if } k + \Delta \leq n, \\ \mathbf{0} & \text{if } k + \Delta > n. \end{cases}
$$

To obtain MLE evaluations of vectors of the form $\mathbf{v}^{\downarrow\Delta}$ in our PIOP instantiation, we use known sumcheck techniques to reduce MLE evaluation claims of $\tilde{W}^{\downarrow\Delta}$ to MLE evaluation claims of $\tilde{W}$, cf. Section 2.5 and [DP25].

Remark 8.3 (Extension: Lookup constraints on algebraic combinations of columns)**.** The lookup constraints in Definition 8.1 constrain every column $\mathbf{f}_{i,j}$ of every witness trace $\mathbf{f}_i$ to be in a fixed set S_{ij}. In practice, it can be convenient to constrain *algebraic combinations* of different columns of $\mathbf{f}_i$ to belong to a set S. For example, one may require that the affine expression $\alpha_1\mathbf{f}_{i,j_1} + \alpha_2\mathbf{f}_{i,j_2} + \cdots + \alpha_k\mathbf{f}_{i,j_k} + \beta$ involving the columns $j_1, \ldots, j_k$ of $\mathbf{f}_i$ is contained in S, for some public constants $\alpha_1, \ldots, \alpha_k, \beta$. See Section 8.2 for a concrete example.

Remark 8.4 (Extenstion: Different trace dimensions)**.** For notational simplicity, we assume the public-input matrix $\mathbf{y}$ has the same number of columns c as the witness matrices. If the public input has fewer columns, one pads it with zeros without affecting costs. One could instead introduce a separate parameter c_{pub} specifying the number of public-input columns. Similarly, one could use different trace sizes per index $i \in \{0\} \cup [|\mathbf{q}|]$.

Remark 8.5 (Extension: Virtual access to column entry-wise bit-operations)**.** Following Section 2.1.4, Lemma 2.3, and Remark 2.8, one could have the UAIR constraint accept columns of the form $T(\mathbf{u})$ where $\mathbf{u}$ is a column in the witness trace (or a vector of the form $\mathbf{u} = \mathbf{f}^{\downarrow\Delta}$ for a witness trace column $\mathbf{f}$, cf. Remark 8.3). As per Lemma 2.3, one can grant virtual MLE access to these columns from MLE access to $\mathbf{u}$, even after reducing coefficient modulo a prime, in case $\mathbf{u}$ is a vector with entries in $\mathbb{Q}[X]$.

We now argue that UAIR$^+$ is a specialization of UCS, and thus that all our protocols for UCS apply to UAIR$^+$ as well.

Lemma 8.6 (UAIR$^+$ as a specialization of UCS)**.** *There is a polynomial-time map sending each UAIR$^+$ index $\mathbb{i}_{\mathsf{UAIR}}$ to a UCS index $\mathbb{i}_{\mathsf{UCS}}$, and polynomial-time mutually inverse maps Ψ, Ψ^{-1} on instance–witness pairs (depending on $\mathbb{i}_{\mathsf{UAIR}}$), such that for every $\mathbb{x}, \mathbb{w}$,*

$$(\mathbb{i}_{\mathsf{UAIR}}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{UAIR}+} \quad \Longleftrightarrow \quad (\mathbb{i}_{\mathsf{UCS}}, \Psi(\mathbb{x}, \mathbb{w})) \in \mathsf{REL}_{\mathsf{UCS}}.$$

Proof. Write $\mathbb{i}_{\mathsf{UAIR}} = (n, c, \mathbf{q}, B, \mathbf{d}, \mathcal{C}, \mathcal{S})$, $\mathbb{x} = \mathbf{y}$, $\mathbb{w} = (\mathbf{f}_0, \ldots, \mathbf{f}_{|\mathbf{q}|})$, and set $m := \ell := nc$.

Flattening and row-extraction selectors. For any ring R, let $\mathsf{flat}\colon R^{n\times c} \to R^m$ be the row-major flattening $\mathsf{flat}(\mathbf{v})_{(k-1)c+j} := \mathbf{v}_j[k]$, with inverse unflat; both are computable in time polynomial in nc and commute entry-wise with ϕ_{q_i}. Define $\Psi(\mathbb{x}, \mathbb{w}) := (\mathsf{flat}(\mathbf{y}); \mathsf{flat}(\mathbf{f}_0), \ldots, \mathsf{flat}(\mathbf{f}_{|\mathbf{q}|}))$ and Ψ^{-1} by unflat on each component.

Using the Lagrange basis polynomials $\mathsf{Lag}_{[n],k}$ from Eq. (34), define for each $j \in [c]$

$$\mathsf{Col}_j(K, \mathbf{Z}) := \sum_{k\in[n]} \mathbf{Z}_{(k-1)c+j}\, \mathsf{Lag}_{[n],k}(K), \qquad \mathsf{Col}_j^{\downarrow}(K, \mathbf{Z}) := \sum_{k\in[n-1]} \mathbf{Z}_{kc+j}\, \mathsf{Lag}_{[n],k}(K),$$

on m variables $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_m)$. By the defining property of the Lagrange basis, for $\mathbf{z} = \mathsf{flat}(\mathbf{v})$ and $k \in [n]$,

$$\mathsf{Col}_j(k, \mathbf{z}) = \mathbf{v}_j[k], \qquad \mathsf{Col}_j^{\downarrow}(k, \mathbf{z}) = \begin{cases} \mathbf{v}_j[k+1] & k < n, \\ 0 & k = n, \end{cases} \tag{45}$$

matching the UAIR$^+$ convention $\mathbf{v}^{\downarrow}[n] = \mathbf{0}$. Write $\mathsf{Col}(K, \mathbf{Z}) := (\mathsf{Col}_j(K, \mathbf{Z}))_{j\in[c]}$ and $\mathsf{Col}^{\downarrow}(K, \mathbf{Z})$ similarly, so that $\mathsf{Col}(k, \mathbf{z}) = \mathbf{v}[k]$ and $\mathsf{Col}^{\downarrow}(k, \mathbf{z}) = \mathbf{v}^{\downarrow}[k]$.

For each $t \in [|\mathcal{C}|]$ and $i \in [|\mathbf{q}|]$, define UCS constraint polynomials Q'_{0t}, Q'_{it} by (45):

$$Q'_{0t}(K, \mathbf{Y}', \mathbf{Z}'_0) := Q_{0t}\big(\mathsf{Col}(K, \mathbf{Y}'),\, \mathsf{Col}(K, \mathbf{Z}'_0),\, \mathsf{Col}^{\downarrow}(K, \mathbf{Z}'_0)\big),$$
$$Q'_{it}(K, \mathbf{Y}', \mathbf{Z}'_0, \mathbf{Z}'_1) := Q_{it}\big(\mathsf{Col}(K, \mathbf{Y}'),\, \mathsf{Col}(K, \mathbf{Z}'_0),\, \mathsf{Col}^{\downarrow}(K, \mathbf{Z}'_0),\, \mathsf{Col}(K, \mathbf{Z}'_1),\, \mathsf{Col}^{\downarrow}(K, \mathbf{Z}'_1)\big),$$

paired with the same ideals $\mathfrak{I}_{0t}, \mathfrak{I}_{it}$. By the row-extraction identities and entry-wise commutation of ϕ_{q_i} with flat, evaluating at $K = k$ yields

$$Q'_{0t}\big(k, \mathsf{flat}(\mathbf{y}), \mathsf{flat}(\mathbf{f}_0)\big) = Q_{0t}\big(\mathbf{y}[k], \mathbf{f}_0[k], \mathbf{f}_0^{\downarrow}[k]\big),$$

and analogously for Q'_{it}. Hence each UAIR$^+$ transition constraint at row k is equivalent to the corresponding UCS constraint at the same row.

For each column-wise lookup $\mathsf{set}(\mathbf{f}_{i,j}) \subseteq S_{ij}$, apply the algebraic-combination lookup gadget of Remark 4.4 with selector Col_j (acting on $\mathsf{flat}(\mathbf{f}_i)$) and table S_{ij}. By Eq. (45), the resulting UCS constraint at row $k \in [n]$ holds iff $\mathbf{f}_{i,j}[k] \in S_{ij}$. Since $S_{0j} \subseteq \mathbb{Z}^{<d_0,B}_{(q_1 \cdots q_{|\mathbf{q}|})}[X]$, the lookups on $\mathbf{f}_0$ subsume the UCS well-definedness condition (33) (cf. Remark 4.5).

To conclude, set $\mathbb{i}_{\mathsf{UCS}} := (m, \ell, n, \mathbf{q}, B, \mathbf{d}, \mathcal{C}')$, where $\mathcal{C}'$ collects all $(Q'_{0t}, \mathfrak{I}_{0t}), (Q'_{it}, \mathfrak{I}_{it})$ together with the lookup constraints above; this index depends only on $\mathbb{i}_{\mathsf{UAIR}}$ and is computable from it in polynomial time. The arguments above show that the UCS constraints in $\mathbb{i}_{\mathsf{UCS}}$ are satisfied by $\Psi(\mathbb{x}, \mathbb{w})$ iff the UAIR$^+$ constraints in $\mathbb{i}_{\mathsf{UAIR}}$ are satisfied by $(\mathbb{x}, \mathbb{w})$, proving both implications. □

8.2 SHA-256 compression

In this section we propose an arithmetization of the SHA-256 compression function as a UAIR$^+$ instance.

We emphasize that this arithmetization is not exactly the same as the one we used in our experiments. Rather, this arithmetization is more "optimal" than the one in our experiments, see Section 9 for details.

8.2.1 SHA-256 overview and target relation definition

We begin with a brief overview of SHA-256 to establish notation, then formalize the relation we seek to arithmetize.

For the purposes of SHA-256, a *word* is a ≤ 32 bit integer. The SHA-256 compression function receives as input a 512-bit string, represented as sixteen *message words* $(M_t)_{t\in[16]} \in [0\,..\,2^{32}-1]^{16}$. The algorithm also uses 64 fixed *round constants* $(K_t)_{t\in[64]} \in [0\,..\,2^{32}-1]^{64}$. Throughout the rest of the section, the word size is 32.

For a word $x = \sum_{i=0}^{32-1} x_i 2^i \in [0\,..\,2^{32}-1]$ with bits $(x_0, \ldots, x_{32-1}) \in \{0,1\}^{32}$, bitwise and , right rotation ROTR^r, and right shift SHR^r are defined as in Section 3. Bitwise negation is defined by

$$\neg x \;:=\; \sum_{i=0}^{32-1} (1 - x_i)\, 2^i \;=\; (2^{32} - 1) - x.$$

We next describe how the SHA-256 compression function operates.

SHA-256 auxiliary functions. SHA-256 employs six auxiliary functions built from rotations, shifts, and bitwise operations. For $x, y, z \in [0\,..\,2^{32}-1]$, define:

$$\begin{aligned}
\Sigma_0(x) &\coloneqq \mathsf{ROTR}^2(x)\ \mathsf{XOR}\ \mathsf{ROTR}^{13}(x)\ \mathsf{XOR}\ \mathsf{ROTR}^{22}(x),\\
\Sigma_1(x) &\coloneqq \mathsf{ROTR}^6(x)\ \mathsf{XOR}\ \mathsf{ROTR}^{11}(x)\ \mathsf{XOR}\ \mathsf{ROTR}^{25}(x),\\
\mathsf{Ch}(x,y,z) &\coloneqq (x\ \mathsf{AND}\ y)\ \mathsf{XOR}\ (\neg x\ \mathsf{AND}\ z),\\
\mathsf{Maj}(x,y,z) &\coloneqq (x\ \mathsf{AND}\ y)\ \mathsf{XOR}\ (x\ \mathsf{AND}\ z)\ \mathsf{XOR}\ (y\ \mathsf{AND}\ z),\\
\sigma_0(x) &\coloneqq \mathsf{ROTR}^7(x)\ \mathsf{XOR}\ \mathsf{ROTR}^{18}(x)\ \mathsf{XOR}\ \mathsf{SHR}^3(x),\\
\sigma_1(x) &\coloneqq \mathsf{ROTR}^{17}(x)\ \mathsf{XOR}\ \mathsf{ROTR}^{19}(x)\ \mathsf{XOR}\ \mathsf{SHR}^{10}(x).
\end{aligned} \tag{46}$$

Message schedule. At the beginning of the SHA-256 compression function, the sixteen input message words $(M_t)_{t\in[16]}$ are expanded into a so-called *message schedule*, which consists of 64 words $(W_t)_{t\in[64]} \in [0\,..\,2^{32}-1]^{64}$ defined as follows:

$$W_t \coloneqq \begin{cases} M_t & t \in [16],\\ W_{t-16} + \sigma_0(W_{t-15}) + W_{t-7} + \sigma_1(W_{t-2}) \pmod{2^{32}} & t \in \{17,\dots,64\}. \end{cases} \tag{47}$$

Compression loop. After the message schedule is computed, SHA-256 performs the so-called *compression loop*. Throughout it, the function maintains an 8-word internal state, conventionally labeled as $(a,b,c,d,e,f,g,h) \in [0\,..\,2^{32}-1]^8$. The compression loop iterates 64 rounds, each of which updates this state using the current schedule word W_t and round constant K_t. At each round $t \in [64]$, the state entering that round is denoted $(a_t,b_t,c_t,d_t,e_t,f_t,g_t,h_t) \in [0\,..\,2^{32}-1]^8$. Writing $(a_{65},\dots,h_{65})$ for the final state (after round 64), the per-round update is given by

$$T_{1,t} \coloneqq h_t + \Sigma_1(e_t) + \mathsf{Ch}(e_t,f_t,g_t) + K_t + W_t \pmod{2^{32}}, \tag{48}$$

$$T_{2,t} \coloneqq \Sigma_0(a_t) + \mathsf{Maj}(a_t,b_t,c_t) \pmod{2^{32}}, \tag{49}$$

$$a_{t+1} \coloneqq T_{1,t} + T_{2,t} \pmod{2^{32}}, \tag{50}$$

$$e_{t+1} \coloneqq d_t + T_{1,t} \pmod{2^{32}}, \tag{51}$$

$$(b_{t+1},c_{t+1},d_{t+1},f_{t+1},g_{t+1},h_{t+1}) \coloneqq (a_t,b_t,c_t,e_t,f_t,g_t).$$

The full SHA-256 compression function adds the initial state to the terminal state word-wise modulo 2^{32} (the so-called *feed-forward step*); we omit this step for brevity. For our purposes, the output of the compression function is the final state $(a_{65},\dots,h_{65})$.

Throughout, we consider UAIR$^+$ instances with extended semantics as discussed in Remarks 8.2 to 8.4.

Remark 8.7 (Shift-register structure)**.** The registers b,c,d,f,g,h are shifted copies of earlier values of a and e: specifically, $b_{t+1} = a_t$, $c_{t+1} = b_t = a_{t-1}$, $d_{t+1} = c_t = a_{t-2}$, and similarly $f_{t+1} = e_t$, $g_{t+1} = f_t = e_{t-1}$, $h_{t+1} = g_t = e_{t-2}$. This shift-register structure is important to our arithmetization: we store only the a and e registers in the trace, recovering all other state registers via shifted row accesses (cf. Section 8.2.3).

The relation we want to arithmetize. We now formally define the relation that we want to express as a UAIR$^+$ instance. We denote this relation by $\mathsf{REL}_{\mathsf{SHA256\text{-}64}}$ and define it as follows:

$$\mathsf{REL}_{\mathsf{SHA256\text{-}64}} \coloneqq \left\{ \begin{pmatrix} \mathbb{i}, \\ \mathbb{x}; \\ \mathbb{w} \end{pmatrix} \;\middle|\; \begin{array}{l} \mathbb{i} = (K_t)_{t\in[64]}, \\ \mathbb{x} = \big((M_t)_{t\in[16]},\ (a_1,\dots,h_1),\ (a_{65},\dots,h_{65})\big), \\ \mathbb{w} = \big((W_t)_{t\in[64]},\ (a_t,\dots,h_t)_{t=1}^{65}\big), \\ (W_t)_{t\in[64]} \text{ satisfies (47), and for every } t\in[64]: \\ \quad a_{t+1} \equiv h_t + \Sigma_1(e_t) + \mathsf{Ch}(e_t,f_t,g_t) \\ \qquad\qquad +K_t + W_t + \Sigma_0(a_t) + \mathsf{Maj}(a_t,b_t,c_t) \pmod{2^{32}}, \\ \quad e_{t+1} \equiv d_t + h_t + \Sigma_1(e_t) + \mathsf{Ch}(e_t,f_t,g_t) + K_t + W_t \pmod{2^{32}}, \\ \quad (b_{t+1},c_{t+1},d_{t+1},f_{t+1},g_{t+1},h_{t+1}) = (a_t,b_t,c_t,e_t,f_t,g_t). \end{array} \right\}. \tag{52}$$

The remainder of this section describes how to express the above relation as a UAIR$^+$ instance.

Background notation We recall some notation introduced in Section 2.1.1. We let $\psi_{32} : [0\,..\,2^{32}-1] \to \{0,1\}^{<32}[X]$ denote the bijection that sends a word to its *bit-polynomial representative* $\hat{x}$. Concretely, given a word $x \in [0\,..\,2^{32}-1]$ with bit decomposition $x = \sum_{i=0}^{32-1} x_i 2^i$, we define

$$\hat{x} \coloneqq \psi_{32}(x) = \sum_{i=0}^{32-1} x_i X^i \in \{0,1\}^{<32}[X]$$

and call $\hat{x}$ the *bit-polynomial representative* of x. Since $\hat{x}(2) = x$ for every $x \in [0\,..\,2^{32}-1]$, we can recover words from their bit-polynomial representatives via membership constraints to the ideal $(X-2)$, as explained in Section 2.1.3.

Finally, we let $\phi_2 : \mathbb{Z}[X] \to \mathbb{F}_2[X]$ denote the natural projection ring homomorphism that reduces coefficients of integer polynomials modulo 2.

8.2.2 Arithmetizing the compression and message schedule operations

In this section we express the functions in Eqs. (46) and (47), i.e. the core SHA-256 compression and message schedule functions, as constraints compatible with our relations $\mathsf{REL}_{\mathsf{UCS}}$ and $\mathsf{REL}_{\mathsf{UAIR+}}$. The characterization is self-contained and does not depend on the register structure of the compression loop; those details are addressed in Section 8.2.3, where we assemble the full 64-round loop using the results established here.

We begin with a general lemma characterizing rotation as ideal membership.

Lemma 8.8 (Rotation as ideal membership)**.** *Let R be a commutative ring, and let $0 \le r < 32$. Then for any $u, v \in R^{<32}[X]$,*

$$v - X^{32-r}\cdot u \;\in\; (X^{32}-1) \qquad\Longleftrightarrow\qquad v = \mathsf{ROTR}^r(u).$$

Proof. Assume first $v = \mathsf{ROTR}^r(u)$. Write $u = \sum_{i=0}^{32-1} u_i X^i$. Then

$$X^{32-r}\cdot u \;=\; \sum_{i=0}^{32-1} u_i\, X^{i+32-r} \;=\; \sum_{i=0}^{r-1} u_i\, X^{i+32-r} \;+\; \sum_{i=r}^{32-1} u_i\, X^{i+32-r}.$$

In the quotient ring $R[X]/(X^{32}-1)$, we have $X^{32}=1$, so the second sum becomes $\sum_{i=r}^{32-1} u_i\, X^{i-r}$ in $R[X]/(X^{32}-1)$. For the first sum, each exponent $i+32-r$ satisfies $32-r \le i+32-r \le 32-1 < 32$,

so these terms already have degree less than 32 and remain unchanged. Thus, modulo $(X^{32}-1)$,

$$X^{32-r}\cdot u \;\equiv\; \sum_{i=r}^{32-1} u_i\, X^{i-r} + \sum_{i=0}^{r-1} u_i\, X^{i+32-r} \;=\; \sum_{j=0}^{32-1} u_{(j+r) \bmod 32}\, X^j \;=\; \mathsf{ROTR}^r(u),$$

which gives $\mathsf{ROTR}^r(u) - X^{32-r}\cdot u \in (X^{32}-1)$.

Now assume $v - X^{32-r}\cdot u \in (X^{32}-1)$ for $v, u \in \{0,1\}^{<32}[X]$. We have $v \equiv X^{32-r}\cdot u \pmod{X^{32}-1}$. Since $\deg(v) < 32$, the polynomial v is the unique representative of degree less than 32 of the residue class of $X^{32-r}\cdot u$ modulo $(X^{32}-1)$. By the previous argument, $\mathsf{ROTR}^r(u) \in R^{<32}[X]$ is also a representative of the same class. Since the representative of degree less than 32 is unique, $v = \mathsf{ROTR}^r(u)$. □

Lemma 8.9 (Arithmetization of Σ_0 and Σ_1)**.** *Let*

$$\rho_0(X) \coloneqq X^{32-2} + X^{32-13} + X^{32-22}, \qquad \rho_1(X) \coloneqq X^{32-6} + X^{32-11} + X^{32-25} \qquad \text{in } \mathbb{F}_2[X]. \tag{53}$$

Let $a, e \in [0\,..\,2^{32}-1]$ with bit-polynomial representatives $\hat{a}, \hat{e} \in \{0,1\}^{<32}[X]$. Let $\hat{\mathbf{y}}_{\Sigma_0}, \hat{\mathbf{y}}_{\Sigma_1} \in \{0,1\}^{<32}[X]$. Then $\hat{\mathbf{y}}_{\Sigma_0}$ is the bit-polynomial representative of $\Sigma_0(a)$, i.e., $\hat{\mathbf{y}}_{\Sigma_0} = \widehat{\Sigma_0(a)}$, if and only if

$$\phi_2(\hat{a})\cdot\rho_0(X) \;-\; \phi_2(\hat{\mathbf{y}}_{\Sigma_0}) \;\in\; (X^{32}-1) \qquad \text{in } \mathbb{F}_2[X], \tag{54}$$

and $\hat{\mathbf{y}}_{\Sigma_1}$ is the bit-polynomial representative of $\Sigma_1(e)$, i.e., $\hat{\mathbf{y}}_{\Sigma_1} = \widehat{\Sigma_1(e)}$, if and only if

$$\phi_2(\hat{e})\cdot\rho_1(X) \;-\; \phi_2(\hat{\mathbf{y}}_{\Sigma_1}) \;\in\; (X^{32}-1) \qquad \text{in } \mathbb{F}_2[X]. \tag{55}$$

Proof. We prove the claim for Σ_0; the argument for Σ_1 is identical.

Forward direction. Since the coefficients of $\hat{a}$ lie in $\{0,1\}$, reduction modulo 2 commutes with right-rotation, so $\phi_2(\widehat{\mathsf{ROTR}^r(a)}) = \mathsf{ROTR}^r(\phi_2(\hat{a}))$ in $\mathbb{F}_2[X]$. Applying Lemma 8.8 over $R = \mathbb{F}_2$ then yields, for each $r \in \{2, 13, 22\}$,

$$\phi_2(\widehat{\mathsf{ROTR}^r(a)}) \;-\; X^{32-r}\cdot\phi_2(\hat{a}) \;\in\; (X^{32}-1).$$

Expanding ρ_0 by distributivity in $\mathbb{F}_2[X]$,

$$\phi_2(\hat{a})\cdot\rho_0(X) \;=\; X^{32-2}\cdot\phi_2(\hat{a}) + X^{32-13}\cdot\phi_2(\hat{a}) + X^{32-22}\cdot\phi_2(\hat{a}),$$

and reducing modulo $(X^{32}-1)$ we obtain, in $\mathbb{F}_2[X]/(X^{32}-1)$,

$$\phi_2(\hat{a})\cdot\rho_0(X) \;\equiv\; \phi_2(\widehat{\mathsf{ROTR}^2(a)}) + \phi_2(\widehat{\mathsf{ROTR}^{13}(a)}) + \phi_2(\widehat{\mathsf{ROTR}^{22}(a)}).$$

Since XOR on $\{0,1\}$ coincides with addition in $\mathbb{F}_2$, we have $\phi_2(x \;\widehat{\mathsf{XOR}}\; y \;\widehat{\mathsf{XOR}}\; z) = \phi_2(\hat{x}) + \phi_2(\hat{y}) + \phi_2(\hat{z})$ in $\mathbb{F}_2[X]$ for any $x, y, z \in [0\,..\,2^{32}-1]$. Therefore, in $\mathbb{F}_2[X]/(X^{32}-1)$,

$$\phi_2(\hat{a})\cdot\rho_0(X) \;=\; \phi_2(\widehat{\Sigma_0(a)}),$$

as needed.

Converse direction. Assume (54) holds. Euclidean division by the monic polynomial $X^{32}-1$ shows that every residue class in $\mathbb{F}_2[X]/(X^{32}-1)$ has a unique representative of degree less than 32. Since $\phi_2(\hat{\mathbf{y}}_{\Sigma_0}) \in \mathbb{F}_2^{<32}[X]$, it is the unique such representative of the class of $\phi_2(\hat{a})\cdot\rho_0(X)$. By the forward direction, $\phi_2(\widehat{\Sigma_0(a)}) \in \mathbb{F}_2^{<32}[X]$ is also a representative of the same class, so $\phi_2(\hat{\mathbf{y}}_{\Sigma_0}) = \phi_2(\widehat{\Sigma_0(a)})$ in $\mathbb{F}_2[X]$. Finally, the restriction $\phi_2\colon \{0,1\}^{<32}[X] \to \mathbb{F}_2[X]$ is injective (reduction modulo 2 is a bijection on $\{0,1\}$), so we conclude that $\hat{\mathbf{y}}_{\Sigma_0} = \widehat{\Sigma_0(a)}$ in $\mathbb{Z}[X]$. □

Lemma 8.10 (Arithmetization of Ch)**.** *Let $e, f, g \in [0\,..\,2^{32}-1]$ with bit-polynomial representatives $\hat{e}, \hat{f}, \hat{g} \in \{0,1\}^{<32}[X]$. Let $\hat{t}_{ef}, \hat{t}_{\neg e,g} \in \{0,1\}^{<32}[X]$. Define $\mathbf{1}_{32} \coloneqq \sum_{i=0}^{32-1} X^i \in \{0,1\}^{<32}[X]$, so that $\mathbf{1}_{32} - \hat{e} = \widehat{\neg e}$.*

Then $\hat{t}_{ef}$ and $\hat{t}_{\neg e,g}$ are the bit-polynomial representatives of e AND f and $(\neg e)$ AND g, respectively, if and only if

$$\hat{e} + \hat{f} - 2\hat{t}_{ef} \;\in\; \{0,1\}^{<32}[X] \qquad \text{in } \mathbb{Q}[X], \tag{56}$$

and

$$(\mathbf{1}_{32} - \hat{e}) + \hat{g} - 2\hat{t}_{\neg e,g} \;\in\; \{0,1\}^{<32}[X] \qquad \text{in } \mathbb{Q}[X]. \tag{57}$$

Moreover, if (56) and (57) hold, then

$$\hat{\mathsf{Ch}} \;\coloneqq\; \hat{t}_{ef} + \hat{t}_{\neg e,g} \qquad \text{in } \mathbb{Q}[X] \tag{58}$$

lies in $\{0,1\}^{<32}[X]$ and equals the bit-polynomial representative of $\mathsf{Ch}(e,f,g)$.

Proof. The first two claims follow from (7): for any $x, y \in [0\,..\,2^{32}-1]$, a bit-polynomial $\hat{t} \in [0\,..\,2^{32}-1]$ satisfies $\hat{x} + \hat{y} - 2\hat{t} \in \{0,1\}^{<32}[X]$ if and only if $\hat{t} = x \,\widehat{\mathsf{AND}}\, y$.

For the final claim, recall that $\mathsf{Ch}(e,f,g) = (e \;\mathsf{AND}\; f) \;\mathsf{XOR}\; ((\neg e) \;\mathsf{AND}\; g)$. Since e and $\neg e$ have disjoint bit-support (no bit position can be 1 in both), the terms $e \;\mathsf{AND}\; f$ and $(\neg e) \;\mathsf{AND}\; g$ also have disjoint bit-support. Therefore, their bitwise XOR equals their integer sum:

$$(e \;\mathsf{AND}\; f) \;\mathsf{XOR}\; ((\neg e) \;\mathsf{AND}\; g) \;=\; (e \;\mathsf{AND}\; f) + ((\neg e) \;\mathsf{AND}\; g) \qquad \text{in } \mathbb{Q}.$$

Hence $\widehat{\mathsf{Ch}}(\hat{e}, \hat{f}, \hat{g}) = e \,\widehat{\mathsf{AND}}\, f + (\neg e) \,\widehat{\mathsf{AND}}\, g \in \{0,1\}^{<32}[X]$ in $\mathbb{Q}[X]$. By the first two claims, (56) and (57) imply $\hat{t}_{ef} = e \,\widehat{\mathsf{AND}}\, f$ and $\hat{t}_{\neg e,g} = (\neg e) \,\widehat{\mathsf{AND}}\, g$; summing gives $\hat{\mathsf{Ch}} = \widehat{\mathsf{Ch}(e,f,g)} \in \{0,1\}^{<32}[X]$. □

Lemma 8.11 (Arithmetization of Maj for bit inputs)**.** *For all $a, b, c, u, m \in \{0,1\}$, we have*

$$a + b + c \;=\; u + 2m \qquad \text{in } \mathbb{Z}$$

if and only if $u = a \;\mathsf{XOR}\; b \;\mathsf{XOR}\; c$ and $m = \mathsf{Maj}(a,b,c)$.

Proof. Given $(a,b,c) \in \{0,1\}^3$, the integer sum $s \coloneqq a + b + c$ lies in $\{0,1,2,3\}$. There is a unique decomposition $s = u + 2m$ with $(u,m) \in \{0,1\}^2$, namely $u \coloneqq s \bmod 2$ and $m \coloneqq \lfloor s/2 \rfloor$. The value u equals the parity $a \;\mathsf{XOR}\; b \;\mathsf{XOR}\; c$, while $m = 1$ if and only if at least two of the three bits are 1, that is, $m = \mathsf{Maj}(a,b,c)$. □

Lemma 8.12 (Arithmetization of Maj for bit-polynomial inputs)**.** *Let $a, b, c \in [0\,..\,2^{32}-1]$ with bit-polynomial representatives $\hat{a}, \hat{b}, \hat{c} \in \{0,1\}^{<32}[X]$. Let $\hat{m} \in \{0,1\}^{<32}[X]$. Then $\hat{m}$ is the bit-polynomial representative of $\mathsf{Maj}(a,b,c)$ if and only if*

$$\hat{a} + \hat{b} + \hat{c} - 2\hat{m} \;\in\; \{0,1\}^{<32}[X] \qquad \text{in } \mathbb{Q}[X]. \tag{59}$$

Proof. For each $i \in \{0, \ldots, 32-1\}$, let $a_i, b_i, c_i, m_i \in \{0,1\}$ denote the i-th coefficients of $\hat{a}, \hat{b}, \hat{c}, \hat{m}$, respectively. Since addition and scalar multiplication in $\mathbb{Q}[X]$ act coefficient-wise, the i-th coefficient of $\hat{a} + \hat{b} + \hat{c} - 2\hat{m}$ equals $a_i + b_i + c_i - 2m_i \in \mathbb{Z}$. Hence $\hat{a} + \hat{b} + \hat{c} - 2\hat{m} \in \{0,1\}^{<32}[X]$ if and only if, for every i, there exists $u_i \in \{0,1\}$ with $a_i + b_i + c_i = u_i + 2m_i$. By Lemma 8.11, this holds if and only if $m_i = \mathsf{Maj}(a_i, b_i, c_i)$ for every i, which is exactly the condition $\hat{m} = \widehat{\mathsf{Maj}(a,b,c)}$. □

Remark 8.13 (Lookup-based enforcement of Ch and Maj)**.** The conditions (56), (57), and (59) are membership constraints of the form "a polynomial lies in $\{0,1\}^{<32}[X]$." In our UAIR$^+$ setting, we enforce these via lookup constraints into the table $\{0,1\}^{<32}[X]$ (cf. Section 4.2 and Remark 4.4): since each constraint involves a linear combination of witness polynomials, we use the linear-combination lookup mechanism described in Remark 8.3.

Register updates modulo 2^{32} In SHA-256, each round updates certain registers via modular addition of several terms, cf. Eqs. (48) to (51), and Eq. (47). The following lemma will be used later to arithmetize such updates. It allows us to express additions of multiple terms in $[0\,..\,2^{32}-1]$ and the integer represented by terms in $\{0,1\}^{<32}[X]$, modulo 2^{32}.

Lemma 8.14 (Modular addition via the $(X-2)$ bridge)**.** *Let $m_1, m_2 \geq 1$. Let $y, x_1, \ldots, x_{m_1} \in [0\,..\,2^{32}-1]$, with $\hat{y} \in \{0,1\}^{<32}[X]$ the bit-polynomial representative of y, and let $\hat{z}_1, \ldots, \hat{z}_{m_2} \in \{0,1\}^{<32}[X]$. Assume there exists $\mu \in [0\,..\,m_1+m_2-1]$ such that*

$$\hat{y} - \sum_{j=1}^{m_1} x_j - \sum_{j=1}^{m_2} \hat{z}_j + \mu \cdot X^{32} \;\in\; (X-2). \tag{60}$$

Then $y \equiv \sum_{j=1}^{m_1} x_j + \sum_{j=1}^{m_2} \hat{z}_j(2) \pmod{2^{32}}$.

Conversely, if $y \equiv \sum_{j=1}^{m_1} x_j + \sum_{j=1}^{m_2} \hat{z}_j(2) \pmod{2^{32}}$, then there exists $\mu \in [0\,..\,m_1+m_2-1]$ such that (60) holds.

Proof. A polynomial $p \in \mathbb{Q}[X]$ lies in $(X-2)$ if and only if $p(2)=0$. Evaluating (60) at $X=2$ yields

$$\hat{y}(2) - \sum_{j=1}^{m_1} x_j - \sum_{j=1}^{m_2} \hat{z}_j(2) + \mu \cdot 2^{32} \;=\; 0 \qquad \text{in } \mathbb{Q}.$$

Since $\hat{y}(2) = y$, this implies the desired congruence modulo 2^{32}.

For the converse, let $S := \sum_{j=1}^{m_1} x_j + \sum_{j=1}^{m_2} \hat{z}_j(2) \in \mathbb{Z}$. Since each summand lies in $[0\,..\,2^{32}-1]$, we have $0 \leq S < (m_1+m_2)\cdot 2^{32}$. Let $\mu := \lfloor S/2^{32} \rfloor \in [0\,..\,m_1+m_2-1]$. If $y \equiv S \pmod{2^{32}}$ with $y \in [0\,..\,2^{32}-1]$, then $S = y + \mu \cdot 2^{32}$. Therefore $\hat{y}(2) - S + \mu \cdot 2^{32} = 0$, so the left-hand side of (60) vanishes at $X=2$ and hence belongs to $(X-2)$. □

The functions σ_0 and σ_1. We now arithmetize the functions σ_0 and σ_1. Recall that

$$\sigma_0(x) = \mathsf{ROTR}^7(x) \;\mathsf{XOR}\; \mathsf{ROTR}^{18}(x) \;\mathsf{XOR}\; \mathsf{SHR}^3(x),$$
$$\sigma_1(x) = \mathsf{ROTR}^{17}(x) \;\mathsf{XOR}\; \mathsf{ROTR}^{19}(x) \;\mathsf{XOR}\; \mathsf{SHR}^{10}(x).$$

Similarly as in Lemma 8.9, we define

$$\rho_{\sigma_0}(X) := X^{32-7} + X^{32-18}, \qquad \rho_{\sigma_1}(X) := X^{32-17} + X^{32-19} \qquad \text{in } \mathbb{F}_2[X]. \tag{61}$$

Lemma 8.15 (Arithmetization of σ_0 and σ_1)**.** *Let $x \in [0\,..\,2^{32}-1]$ be a word with bit-polynomial representative $\hat{x} \in \{0,1\}^{<32}[X]$, and let $\hat{y}_{\sigma_0}, \hat{y}_{\sigma_1} \in \{0,1\}^{<32}[X]$. Then $\hat{y}_{\sigma_0} = \widehat{\sigma_0(x)}$ if and only if*

$$\phi_2(\hat{x}) \cdot \rho_{\sigma_0}(X) + \phi_2(\widehat{\mathsf{SHR}^3(x)}) - \phi_2(\hat{y}_{\sigma_0}) \;\in\; (X^{32}-1) \qquad \text{in } \mathbb{F}_2[X]. \tag{62}$$

Similarly, $\hat{y}_{\sigma_1} = \widehat{\sigma_1(x)}$ if and only if

$$\phi_2(\hat{x}) \cdot \rho_{\sigma_1}(X) + \phi_2(\widehat{\mathsf{SHR}^{10}(x)}) - \phi_2(\hat{y}_{\sigma_1}) \;\in\; (X^{32}-1) \qquad \text{in } \mathbb{F}_2[X]. \tag{63}$$

Proof. We prove the claim for σ_0; the argument for σ_1 is identical. Similarly as in the proof of Lemma 8.9, in $\mathbb{F}_2[X]$,

$$\phi_2(\hat{x}) \cdot \rho_{\sigma_0}(X) = \phi_2(\hat{x}) \cdot (X^{32-7} + X^{32-18}) \equiv \phi_2(\widehat{\mathsf{ROTR}^7(x)}) + \phi_2(\widehat{\mathsf{ROTR}^{18}(x)}) \pmod{X^{32}-1}.$$

Adding $\phi_2(\widehat{\mathsf{SHR}^3(x)})$ to both sides gives

$$\begin{aligned}&\phi_2(\hat{x}) \cdot \rho_{\sigma_0}(X) + \phi_2(\widehat{\mathsf{SHR}^3(x)}) \\ &\quad \equiv \phi_2(\widehat{\mathsf{ROTR}^7(x)}) + \phi_2(\widehat{\mathsf{ROTR}^{18}(x)}) + \phi_2(\widehat{\mathsf{SHR}^3(x)}) = \phi_2(\widehat{\sigma_0(x)}) \pmod{X^{32}-1}\end{aligned}$$

in $\mathbb{F}_2[X]$, since addition in $\mathbb{F}_2$ corresponds to XOR. The remainder of the argument follows exactly as in Lemma 8.9: the ideal membership (62) holds if and only if $\phi_2(\hat{y}_{\sigma_0})$ and $\phi_2(\hat{x}) \cdot \rho_{\sigma_0}(X) + \phi_2(\widehat{\mathsf{SHR}^3(x)})$ agree modulo $(X^{32}-1)$, which occurs if and only if $\phi_2(\hat{y}_{\sigma_0}) = \phi_2(\widehat{\sigma_0(x)})$ in $\mathbb{F}_2^{<32}[X]$. By injectivity of ϕ_2 on $\{0,1\}^{<32}[X]$, this is equivalent to $\hat{y}_{\sigma_0} = \widehat{\sigma_0(x)}$ in $\mathbb{Z}[X]$. □

Remark 8.16 (Virtual right-shift columns)**.** In our SHA-256 arithmetization (Section 8.2.3), the term $\phi_2(\widehat{\mathsf{SHR}^r(x)})$ in (62) and (63) is realized as a virtual column derived from the committed bit-polynomial column $\hat{x}$, via Lemma 2.3. No separate witness is committed for $\widehat{\mathsf{SHR}^r(x)}$.

Once the σ_0 and σ_1 functions are arithmetized using Lemma 8.15, the message schedule recurrence (47) can be enforced using Lemma 8.14. We provide the details in Section 8.2.3.

8.2.3 Full UAIR$^+$ arithmetization of SHA-256

This section presents the complete UAIR$^+$ arithmetization of the SHA-256 compression function. We construct a UAIR$^+$ index–input–witness triple that belongs to $\mathsf{REL}_{\mathsf{UAIR}+}$ if and only if a corresponding triple belongs to our initial target relation $\mathsf{REL}_{\mathsf{SHA256\text{-}64}}$ (Eq. (52)).

Throughout the section we use bracket notation $v[t]$ to denote the entry of a column vector v at row t.

Our goal is, given $\mathbb{i}_{\mathsf{SHA}}$, to construct a UAIR$^+$ index $\mathbb{i}_{\mathsf{UAIR}}$ such that for every input-witness $(\mathbb{x}_{\mathsf{SHA}}, \mathbb{w}_{\mathsf{SHA}})$ such that $(\mathbb{i}_{\mathsf{SHA}}, \mathbb{x}_{\mathsf{SHA}}; \mathbb{w}_{\mathsf{SHA}}) \in \mathsf{REL}_{\mathsf{SHA256\text{-}64}}$ (Eq. (52)), there exists $\mathbb{x}_{\mathsf{UAIR}}, \mathbb{w}_{\mathsf{UAIR}}$ such that $(\mathbb{i}_{\mathsf{UAIR}}, \mathbb{x}_{\mathsf{UAIR}}; \mathbb{w}_{\mathsf{UAIR}}) \in \mathsf{REL}_{\mathsf{UAIR}+}$, and vice versa. Moreover, the triple $(\mathbb{i}_{\mathsf{UAIR}}, \mathbb{x}_{\mathsf{UAIR}}; \mathbb{w}_{\mathsf{UAIR}})$ can be computed from $(\mathbb{i}_{\mathsf{SHA}}, \mathbb{x}_{\mathsf{SHA}}; \mathbb{w}_{\mathsf{SHA}})$, and vice versa, in polynomial time.

UAIR$^+$ index. Fix a triple $(\mathbb{i}_{\mathsf{SHA}}, \mathbb{x}_{\mathsf{SHA}}; \mathbb{w}_{\mathsf{SHA}}) \in \mathsf{REL}_{\mathsf{SHA256\text{-}64}}$ (Eq. (52)) with

$$\begin{aligned}\mathbb{i}_{\mathsf{SHA}} &= (K_t)_{t\in[64]}, \\ \mathbb{x}_{\mathsf{SHA}} &= \big((M_t)_{t\in[16]},\ (a_1,\ldots,h_1),\ (a_{65},\ldots,h_{65})\big), \\ \mathbb{w}_{\mathsf{SHA}} &= \big((W_t)_{t\in[64]},\ (a_t,\ldots,h_t)_{t=1}^{65}\big).\end{aligned}$$

Here $(K_t)_{t\in[64]}$ are the SHA-256 round constants, $(M_t)_{t\in[16]}$ are the message words, $(a_1,\ldots,h_1)$ and $(a_{65},\ldots,h_{65})$ are the initial and final states, $(W_t)_{t\in[64]}$ is the message schedule, and $(a_t,\ldots,h_t)_{t=1}^{65}$ is the sequence of internal states. Recall that the word size is 32, so each value lies in $[0\,..\,2^{32}-1]$.

Towards our goals, let $\mathbb{i}_{\mathsf{UAIR}}$ be a UAIR$^+$ index as follows:

$$\mathbb{i}_{\mathsf{UAIR}} = (n, c, \mathbf{q}, B, \mathbf{d}, \mathcal{C}, \mathcal{S}),$$

where:

Prime power tuple $\mathbf{q}$. We set $\mathbf{q} = (2)$, i.e. the prime power tuple $\mathbf{q}$ consists solely of the prime 2. Accordingly, the UAIR$^+$ will have two trace components: a $\mathbb{Q}[X]$-trace (component 0) and an $\mathbb{F}_2[X]$-trace (component 1).

Number of rows n. We set $n = 65$. Among others, row 1 holds the initial state $(a_1, \dots, h_1)$, rows $t = 2, \dots, 64$ hold the intermediate states $(a_t, \dots, h_t)$, and row 65 holds the final state $(a_{65}, \dots, h_{65})$.

Number of columns c. We set $c = 13$ for the $\mathbb{Q}[X]$-trace and $c = 0$ for the $\mathbb{F}_2[X]$-trace (see column descriptions below). The $\mathbb{F}_2[X]$-trace carries no committed columns; it is retained only to register that the relation has constraints over $\mathbb{F}_2[X]$. See Remark 8.4 for a discussion of allowing a different number of columns per trace component.

Bit-size B and degree bounds $\mathbf{d} = (d_0, d_1)$. We set $B = 32$ and $\mathbf{d} = (d_0, d_1) = (32, 32)$.

Witness The witness $\mathbb{w}_{\mathsf{UAIR}} = (\mathbf{f}_0, \mathbf{f}_1)$ contains $\mathbf{f}_0 \in (\mathbb{Q}^{<B, d_0}[X])^{n \times 13}$, with $\mathbf{f}_1$ being the empty $n \times 0$ matrix; equivalently, all of the witness is committed in the $\mathbb{Q}[X]$-trace. The 13 columns of $\mathbf{f}_0$ have the following intended semantics:

Bit-polynomial columns (10 columns). $\hat{a}$, $\hat{e}$, $\hat{W}$, $\hat{\Sigma}_0$, $\hat{\Sigma}_1$, $\hat{\mathsf{Maj}}$, $\hat{u}_{ef}$, $\hat{u}_{\neg e,g}$, $\hat{\sigma}_0$, $\hat{\sigma}_1$. Each entry of these columns is constrained to lie in $\{0,1\}^{<32}[X] \subset \mathbb{Q}[X]$ via a lookup constraint. These columns store the bit-polynomial representatives of the following elements, respectively: the state registers a_t and e_t, i.e., $\hat{a}[t] = \widehat{a_t}$ and $\hat{e}[t] = \widehat{e_t}$; the schedule word W_t, i.e., $\hat{W}[t] = \widehat{W_t}$ for $t \leq 64$ (the entry $\hat{W}[65]$ is unconstrained); the bit-polynomial representatives of the auxiliary functions $\Sigma_0(a_t)$, $\Sigma_1(e_t)$, $\mathsf{Maj}(a_t, b_t, c_t)$; the columns $\hat{u}_{ef}$, $\hat{u}_{\neg e,g}$ store the bit-polynomial witnesses used in the arithmetization of $\mathsf{Ch}(e_t, f_t, g_t)$ (cf. Lemma 8.10); and, finally, the columns $\hat{\sigma}_0$, $\hat{\sigma}_1$ store the bit-polynomial representatives of the values taken by the message-schedule functions σ_0, σ_1.

Integer columns (3 columns). μ_a, μ_e, μ_W. The columns μ_a, μ_e, μ_W store the carry values for the modular additions in the a-update (50), e-update (51), and message-schedule recurrence (47), respectively; these take values in $[0\,..\,6]$, $[0\,..\,5]$, $[0\,..\,3]$. They are lookup-constrained to belong to some integer range. Any integer range containing $[0\,..\,6]$ works, but they must be lookup-constrained to avoid malicious provers using rational values for μ_a, μ_e, μ_W, as opposed to integers.
The $\mathbb{F}_2[X]$-trace $\mathbf{f}_1$ has no columns. In an earlier version of this arithmetization, four columns S_0, S_1, T_0, T_1 stored the right-shift quotients of $\hat{W}$ used to arithmetize σ_0 and σ_1. By Lemma 2.3, SHR^r extends to an $\mathbb{F}_2$-linear coordinatewise map on $\mathbb{F}_2^{<32}[X]$ that commutes with multilinear extension; we can therefore treat $\mathsf{SHR}^r(\hat{W})$ as a *virtual column* in our constraints (cf. the discussion preceding Lemma 2.3), removing the need to commit S_0, S_1, T_0, T_1 and the consistency constraints linking them to $\hat{W}$.

Remark 8.17. The state registers d_t and h_t do not require dedicated trace columns: by the shift-register identities (Remark 8.7), $d_t = a_{t-3}$ and $h_t = e_{t-3}$ for $t \geq 4$, and for $t \in \{1, 2, 3\}$ the values of d_t and h_t are determined by the initial state $(a_1, \dots, h_1)$ from the public input via $d_1 = d_1$, $d_2 = c_1$, $d_3 = b_1$ and $h_1 = h_1$, $h_2 = g_1$, $h_3 = f_1$ (cf. Remark 8.7). We inline the identities $d_t = a_{t-3}$ and $h_t = e_{t-3}$ into our constraints for $t \geq 4$, and use these base-case substitutions for $t \in \{1, 2, 3\}$.

Public input The public input is a matrix $\mathbb{x}_{\mathsf{UAIR}} = \mathbf{y} \in (\mathbb{Q}^{<d_0, B}[X])^{n \times c_{\mathsf{pub}}}$ with $c_{\mathsf{pub}} = 10$ columns. It is organized as follows:

State columns (8 columns). For each register $x \in \{a, b, c, d, e, f, g, h\}$, $\mathbf{y}$ contains a column $\mathbf{y}_x$ that is zero except at rows $t \in \{1, 65\}$, where it takes the following values:

$$\mathbf{y}_x[1] = \begin{cases} \widehat{x_1} & \text{if } x \in \{a, b, c, e, f, g\}, \\ x_1 & \text{if } x \in \{d, h\}, \end{cases} \qquad \mathbf{y}_x[65] = \begin{cases} \widehat{x_{65}} & \text{if } x \in \{a, b, c, e, f, g\}, \\ x_{65} & \text{if } x \in \{d, h\}. \end{cases}$$

Message-word column. A column $\mathbf{y}_M \in (\mathbb{Q}^{<d_0,B}[X])^{65}$ with $\mathbf{y}_M[t] = \widehat{M_t}$ for $t \in [16]$ and $\mathbf{y}_M[t] = 0$ for $t \in \{17, \ldots, 65\}$.

Round-constant column. A column $\mathbf{y}_K \in (\mathbb{Q}^{<d_0,B}[X])^{65}$ with $\mathbf{y}_K[t] = K_t$ for $t \in [64]$ and $\mathbf{y}_K[65] = 0$.

Constraints We proceed to describe the constraints $\mathcal{C}$. We let $\mathbf{1}_{32} \coloneqq \sum_{i=0}^{32-1} X^i \in \{0,1\}^{<32}[X]$ and let $\rho_0, \rho_1, \rho_{\sigma_0}, \rho_{\sigma_1} \in \mathbb{F}_2[X]$ be the polynomials

$$\rho_0(X) \coloneqq X^{32-2} + X^{32-13} + X^{32-22}, \qquad \rho_1(X) \coloneqq X^{32-6} + X^{32-11} + X^{32-25},$$
$$\rho_{\sigma_0}(X) \coloneqq X^{32-7} + X^{32-18}, \qquad \rho_{\sigma_1}(X) \coloneqq X^{32-17} + X^{32-19},$$

as defined in (53) and (61). Recall that $\phi_2 : \mathbb{Z}[X] \to \mathbb{F}_2[X]$ is the natural reduction modulo 2.

Σ-functions. The following constraints enforce that the columns $\hat{\Sigma}_0$ and $\hat{\Sigma}_1$ of $\mathbf{f}_0$ contain the values $\Sigma_0(a_t)$ and $\Sigma_1(e_t)$ for $t \in [65]$, by Lemma 8.9:

$$\phi_2(\hat{a}[t]) \cdot \rho_0(X) - \phi_2(\hat{\Sigma}_0[t]) \in (X^{32} - 1), \quad \text{for all } t \in [65], \quad \text{in } \mathbb{F}_2[X] \tag{64}$$
$$\phi_2(\hat{e}[t]) \cdot \rho_1(X) - \phi_2(\hat{\Sigma}_1[t]) \in (X^{32} - 1), \quad \text{for all } t \in [65], \quad \text{in } \mathbb{F}_2[X] \tag{65}$$

Ch *function.* We use Lemma 8.10 to enforce that the columns $\hat{u}_{ef}$ and $\hat{u}_{\neg e,g}$ contain values such that

$$\widehat{\mathsf{Ch}(e_t, f_t, g_t)} = \hat{u}_{ef} + \hat{u}_{\neg e,g} \quad \text{in } \mathbb{Q}[X].$$

For $t \geq 2$, we use that $f_t = e_{t-1}$ and $g_t = e_{t-2}$ (cf. Remark 8.7) and $\mathsf{Ch}(e_t, f_t, g_t) = \mathsf{Ch}(e_t, e_{t-1}, e_{t-2})$. The corresponding constraints, in the form of lookup constraints, are:

$$\hat{e}[t] + \hat{e}[t-1] - 2\hat{u}_{ef}[t] \in \{0,1\}^{<32}[X], \quad \text{for all } t \in [65] \setminus \{1,2\}, \quad \text{in } \mathbb{Q}[X] \tag{66}$$
$$(\mathbf{1}_{32} - \hat{e}[t]) + \hat{e}[t-2] - 2\hat{u}_{\neg e,g}[t] \in \{0,1\}^{<32}[X], \quad \text{for all } t \in [65] \setminus \{1,2\}, \quad \text{in } \mathbb{Q}[X] \tag{67}$$

When $t = 1, 2$ we use the same constraint, except that f_t and g_t are substituted with their initial values from the public input $(a_1, \ldots, h_1)$, see the boundary constraints below, concretely Eqs. (82), (83), (85) and (86).

Maj *function.* We use Lemma 8.12 to enforce that the column $\hat{\mathsf{Maj}}$ contains values such that

$$\hat{\mathsf{Maj}}[t] = \widehat{\mathsf{Maj}(a_t, b_t, c_t)} \quad \text{in } \mathbb{Q}[X].$$

For $t \geq 3$, we use that $b_t = a_{t-1}$ and $c_t = a_{t-2}$ (cf. Remark 8.7) and $\mathsf{Maj}(a_t, b_t, c_t) = \mathsf{Maj}(a_t, a_{t-1}, a_{t-2})$. The corresponding constraint, in the form of a lookup constraint, is:

$$\hat{a}[t] + \hat{a}[t-1] + \hat{a}[t-2] - 2\hat{\mathsf{Maj}}[t] \in \{0,1\}^{<32}[X], \quad \text{for all } t \in [65] \setminus \{1,2\}, \quad \text{in } \mathbb{Q}[X]. \tag{68}$$

When $t = 1, 2$ we use the same constraint, except that b_t and c_t are substituted with their initial values from the public input $(a_1, \ldots, h_1)$, see the boundary constraints below, concretely Eqs. (87) and (88).

Update of the registers a and e. We use Lemma 8.14 to enforce correct updating of the registers a and e, as per equations Eqs. (50) and (51). For $t \geq 4$, we use that $d_t = a_{t-3}$ and $h_t = e_{t-3}$ (cf. Remark 8.7) and inline these identities directly into the update. The corresponding constraints are:

$$\hat{a}[t+1] - \big(\hat{e}[t-3] + \hat{\Sigma}_1[t] + \hat{\mathsf{Ch}}[t] + \mathbf{y}_K[t] + \hat{W}[t] + \hat{\Sigma}_0[t] + \hat{\mathsf{Maj}}[t]\big)$$

$$+ \mu_a[t] \cdot X^{32} \in (X-2), \quad \text{for all } t \in [64] \setminus \{1,2,3\}, \quad \text{in } \mathbb{Q}[X] \tag{69}$$

$$\hat{e}[t+1] - \big(\hat{a}[t-3] + \hat{e}[t-3] + \hat{\Sigma}_1[t] + \hat{\mathsf{Ch}}[t] + \mathbf{y}_K[t] + \hat{W}[t]\big) + \mu_e[t] \cdot X^{32} \in (X-2), \quad \text{for all } t \in [64] \setminus \{1,2,3\}, \quad \text{in } \mathbb{Q}[X] \tag{70}$$

When $t = 1, 2, 3$ we use the same constraints, except that d_t and h_t are substituted with their initial values from the public input $(a_1, \ldots, h_1)$ via the shift identities $d_1 = d_1$, $d_2 = c_1$, $d_3 = b_1$ and $h_1 = h_1$, $h_2 = g_1$, $h_3 = f_1$, see boundary constraints below.

Update of the remaining registers. The round update

$$(b_{t+1}, c_{t+1}, d_{t+1}, f_{t+1}, g_{t+1}, h_{t+1}) = (a_t, b_t, c_t, e_t, f_t, g_t) \tag{71}$$

requires no dedicated constraints. For $t \geq 3$, we use that $b_t = a_{t-1}$, $c_t = a_{t-2}$, $f_t = e_{t-1}$, $g_t = e_{t-2}$, and for $t \geq 4$ that $d_t = a_{t-3}$, $h_t = e_{t-3}$ (cf. Remark 8.7), and in fact we do not even have witness columns storing values for the b, c, d, f, g, h registers: the corresponding values are obtained from the appropriate entries of the columns $\hat{a}$ and $\hat{e}$, and from the public input when $t \in \{1, 2, 3\}$ (see below). Using such values instead of values from dedicated witness columns for b, c, d, f, g, h implicitly enforces the round update (71).

For $t \leq 3$, some of the back-references above fall before row 1 of the trace. The registers that cannot be recovered are precisely $c_2, d_2, g_2, h_2, d_3, h_3$. For these, we obtain the appropriate values from the public input $(a_1, \ldots, h_1)$, using the back-references from Remark 8.7 as follows:

$$\begin{aligned} &t = 2: \quad c_2 = b_1,\ d_2 = c_1,\ g_2 = f_1,\ h_2 = g_1, \\ &t = 3: \quad d_3 = b_1,\ h_3 = f_1. \end{aligned} \tag{72}$$

Message schedule. We enforce the recurrence

$$W_t \equiv W_{t-16} + \sigma_0(W_{t-15}) + W_{t-7} + \sigma_1(W_{t-2}) \pmod{2^{32}} \quad \text{for } t \in \{17, \ldots, 64\}. \tag{73}$$

The arithmetization consists in:

(i) The σ_0 and σ_1 functions. Following Lemma 8.15, we arithmetize σ_0 and σ_1 as

$$\phi_2(\hat{W}[t]) \cdot \rho_{\sigma_0}(X) + \phi_2(\widehat{\mathsf{SHR}^3(W)}[t]) - \phi_2(\hat{\sigma}_0[t+15]) \in (X^{32} - 1), \quad \text{for all } t \in [49] \setminus \{1\}, \quad \text{in } \mathbb{F}_2[X], \tag{74}$$

$$\phi_2(\hat{W}[t]) \cdot \rho_{\sigma_1}(X) + \phi_2(\widehat{\mathsf{SHR}^{10}(W)}[t]) - \phi_2(\hat{\sigma}_1[t+2]) \in (X^{32} - 1), \quad \text{for all } t \in [62] \setminus [14], \quad \text{in } \mathbb{F}_2[X]. \tag{75}$$

where $\widehat{\mathsf{SHR}^3(W)}$ and $\widehat{\mathsf{SHR}^{10}(W)}$ are treated as virtual vectors, with access to their MLE's granted virtually from access to the MLE of $\hat{W}$, following Lemma 2.3 and Remark 8.5.

In Eqs. (74) and (75) we have indexed rows so that $\widehat{\mathsf{SHR}^i(W)}$ is always accessed without shifts in the row indices, i.e. as $\widehat{\mathsf{SHR}^i(W)}[t]$ instead of $\widehat{\mathsf{SHR}^i(W)}[t-15]$ or $\widehat{\mathsf{SHR}^i(W)}[t-2]$. This is mainly for presentation simplicity, to avoid arguing how one can obtain MLE evaluations of vectors of the form $\widehat{\mathsf{SHR}^i(W)}^{\downarrow\Delta}$, cf. Remark 8.2 for the notation $\downarrow \Delta$. We briefly remark that it is perfectly possible to do so: one can show that $\mathsf{MLE}[(\widehat{\mathsf{SHR}^i(W)})^{\downarrow\Delta}] = \mathsf{SHR}^i(\mathsf{MLE}[\hat{W}^{\downarrow\Delta}])$. In our PIOP instantiation, to obtain MLE evaluations of vectors of the form $\mathbf{v}^{\downarrow\Delta}$, we use known sumcheck techniques to reduce MLE evaluation claims of $\tilde{W}^{\downarrow\Delta}$ to MLE evaluation claims of $\tilde{W}$, cf. Section 2.5 and [DP25]).

(ii) Modular sum (Lemma 8.14), with carry $\mu_W[t]$.

$$\hat{W}[t] - \hat{W}[t-16] - \hat{\sigma}_0[t] - \hat{W}[t-7] - \hat{\sigma}_1[t] + \mu_W[t] \cdot X^{32} \in (X-2), \quad \text{for all } t \in [64] \setminus [16], \quad \text{in } \mathbb{Q}[X] \tag{76}$$

Boundary constraints. We also add the following constraints, which are to be applied on a small number of row indices $t \in [65]$.

Initial and final states.

$$\hat{a}[1] = \mathbf{y}_a[1], \qquad \hat{e}[1] = \mathbf{y}_e[1], \tag{77}$$

$$\hat{a}[65] = \mathbf{y}_a[65], \qquad \hat{e}[65] = \mathbf{y}_e[65], \tag{78}$$

$$\hat{a}[62] - \mathbf{y}_d[65] \in (X-2), \qquad \hat{e}[62] - \mathbf{y}_h[65] \in (X-2), \tag{79}$$

$$\hat{a}[64] = \mathbf{y}_b[65], \qquad \hat{a}[63] = \mathbf{y}_c[65], \qquad \hat{e}[64] = \mathbf{y}_f[65], \qquad \hat{e}[63] = \mathbf{y}_g[65]. \tag{80}$$

Eqs. (77) and (78) enforce that the columns $\hat{a}$ and $\hat{e}$ contain the initial and final values of the a and e registers, respectively. The remaining constraints in Eqs. (79) and (80) enforce that the remaining registers b, c, d, f, g, h also take their correct final values at row 65, using the shift-register identities from Remark 8.7.

Message-schedule initialization.

$$\hat{W}[t] = \mathbf{y}_M[t] \quad \text{for all } t \in [16]. \tag{81}$$

Base-case lookup constraints for Ch. We take the constraints from Eqs. (66) and (67) and substitute the back-references to f_t and g_t with the corresponding initial values from the public input, concretely $f_1 \to \mathbf{y}_f[1]$ and $g_1 \to \mathbf{y}_g[1]$ for $t = 1$, and $g_2 \to \mathbf{y}_f[1]$ (cf. Eq. (72)) for $t = 2$. The resulting constraints are for arithmetizing $\mathsf{Ch}(e_1, f_1, g_1)$:

$$\hat{e}[1] + \mathbf{y}_f[1] - 2\hat{u}_{ef}[1] \in \{0,1\}^{<32}[X], \quad \text{in } \mathbb{Q}[X], \tag{82}$$

$$(\mathbf{1}_{32} - \hat{e}[1]) + \mathbf{y}_g[1] - 2\hat{u}_{\neg e,g}[1] \in \{0,1\}^{<32}[X], \quad \text{in } \mathbb{Q}[X], \tag{83}$$

$$\tag{84}$$

and for arithmetizing $\mathsf{Ch}(e_2, f_2, g_2)$:

$$\hat{e}[2] + \hat{e}[1] - 2\hat{u}_{ef}[2] \in \{0,1\}^{<32}[X], \quad \text{in } \mathbb{Q}[X], \tag{85}$$

$$(\mathbf{1}_{32} - \hat{e}[2]) + \mathbf{y}_f[1] - 2\hat{u}_{\neg e,g}[2] \in \{0,1\}^{<32}[X], \quad \text{in } \mathbb{Q}[X]. \tag{86}$$

Base-case lookups for Maj. Similarly, the constraints from Eq. (68) are modified for $t = 1, 2$ by substituting the back-references to b_t and c_t with the corresponding initial values from the public input, concretely $b_1 \to \mathbf{y}_b[1]$ and $c_1 \to \mathbf{y}_c[1]$ for $t = 1$, and $c_2 \to \mathbf{y}_b[1]$ (cf. Eq. (72)) for $t = 2$. The resulting constraints are, respectively for $t = 1$ and $t = 2$:

$$\hat{a}[1] + \mathbf{y}_b[1] + \mathbf{y}_c[1] - 2\hat{\mathsf{Maj}}[1] \in \{0,1\}^{<32}[X], \quad \text{in } \mathbb{Q}[X], \tag{87}$$

$$\hat{a}[2] + \hat{a}[1] + \mathbf{y}_b[1] - 2\hat{\mathsf{Maj}}[2] \in \{0,1\}^{<32}[X], \quad \text{in } \mathbb{Q}[X]. \tag{88}$$

Base-case updates. Similarly as above, for $t \in \{1,2,3\}$, the update constraints (69)–(70) are modified by replacing the back-references $\hat{e}[t-3]$ (standing for h_t) and $\hat{a}[t-3]$ (standing for d_t) with the corresponding public-input values: at $t = 1$, $h_1 \to \mathbf{y}_h[1]$ and $d_1 \to \mathbf{y}_d[1]$; at $t = 2$, $h_2 = g_1 \to \mathbf{y}_g[1]$ and $d_2 = c_1 \to \mathbf{y}_c[1]$; at $t = 3$, $h_3 = f_1 \to \mathbf{y}_f[1]$ and $d_3 = b_1 \to \mathbf{y}_b[1]$. See Table 8 for the explicit resulting constraints.

Remark 8.18 (Selector polynomials and shift polynomials.)**.** Most of the constraints we described apply only on a subset of row indices $t \in [65]$. However, in a UAIR$^+$ instance all constraints must be defined over all rows. To address this, we first reindex rows through the change of variable $t \to t{+}17$. This way, all shifted column accesses have the form $\mathbf{x}[t + \Delta]$ for some $t \in [65]$ and non-negative constant Δ. Then, we use shift columns $\mathbf{x}^{\downarrow\Delta}$ as in Remark 8.2 in place of $\mathbf{x}[t+\Delta]$. This way, $\mathbf{x}^{\downarrow\Delta}[t]$ is defined for all $t \in [65]$, and further, for the relevant values of t we have $\mathbf{x}^{\downarrow\Delta}[t] = \mathbf{x}[t + \Delta]$, and $\mathbf{x}^{\downarrow\Delta}$ is zero otherwise. Then, whenever necessary, we multiply constraint polynomials by selector polynomials, i.e. polynomials whose evaluation at t is 1 if the constraint should apply at row t, and 0 otherwise.

We omit the details for brevity and because these are standard techniques. We also remark that there are other reasonable approaches, see e.g. [Tho24].

8.2.4 Arithmetization summary

Tables Tables 5 to 9, summarize the public input and witness $\mathbb{x}_{\mathsf{UAIR}}, \mathbb{w}_{\mathsf{UAIR}}$, and the constraints in $\mathbb{i}_{\mathsf{UAIR}}$. We separate the constraint list into three lists. In Table 6 we describe non-boundary constraints, i.e. constraints that apply to almost all row indices $t \in [65]$; in Table 7 we describe affine lookup constraints. Note that Table 5 already includes the lookup constraints on the witness columns — these are not included in Table 7. Finally, in Tables 8 and 9 we describe boundary constraints, i.e. constraints that apply only to a small number of rows (e.g., $t = 1$ or $t = 65$).

Selector polynomials and shift columns are not explicitly described here, see Remark 8.18.

Again we emphasize that this arithmetization is similar but not exactly the same as the one we used in our experiments, see Section 9 for details.

Lookup set	$\subseteq$ Domain	#Rows	#Cols	Columns
$\mathbb{Q}[X]$-*trace columns*				
$\{0,1\}^{<32}[X]$	$\subseteq \mathbb{Q}^{<32}[X]$	65	10	$\hat{a}, \hat{e}, \hat{W}, \hat{\Sigma}_0, \hat{\Sigma}_1, \hat{\mathsf{Maj}}, \hat{u}_{ef}, \hat{u}_{\neg e,g}, \hat{\sigma}_0, \hat{\sigma}_1$
$[0\,..\,6], [0\,..\,5], [0\,..\,3]$	$\subseteq \mathbb{Q}$	65	3	μ_a, μ_e, μ_W
$\mathbb{F}_2[X]$-*trace columns*: *none*				
Public input columns				
$\{0,1\}^{<32}[X]$	$\subseteq \mathbb{Q}^{<32}[X]$	65	6	$\mathbf{y}_a, \mathbf{y}_b, \mathbf{y}_c, \mathbf{y}_e, \mathbf{y}_f, \mathbf{y}_g$ (zero except at $t \in \{1, 65\}$)
$[0\,..\,2^{32}-1]$	$\subseteq \mathbb{Q}$	65	2	$\mathbf{y}_d, \mathbf{y}_h$ (zero except at $t \in \{1, 65\}$)
$\{0,1\}^{<32}[X]$	$\subseteq \mathbb{Q}^{<32}[X]$	65	1	$\mathbf{y}_M$ (zero except at $t \in [16]$)
$[0\,..\,2^{32}-1]$	$\subseteq \mathbb{Q}$	65	1	$\mathbf{y}_K$ (zero at $t = 65$)

Table 5: Traces: The columns of the witnesses $\mathbf{f}_0$ and $\mathbf{f}_1$ in our SHA-256 UAIR$^+$ arithmetization. The "Lookup set" column indicates lookup constraints enforced on the entries of each column (to be proved at the PIOP level); the "Domain" column specifies the type of elements each column contains (enforced via the IOPP/PCS), cf. Section 2.1.2.

Domain	Ideal	Constraint polynomial	Rows t	Reference
$\mathbb{F}_2[X]$	$(X^{32}-1)$	$\phi_2(\hat{a}[t]) \cdot \rho_0(X) - \phi_2(\hat{\Sigma}_0[t])$	[65]	Σ_0 rotation, (64)
$\mathbb{F}_2[X]$	$(X^{32}-1)$	$\phi_2(\hat{e}[t]) \cdot \rho_1(X) - \phi_2(\hat{\Sigma}_1[t])$	[65]	Σ_1 rotation, (65)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{a}[t+1] - \hat{e}[t-3] - \hat{\Sigma}_1[t] - \hat{\mathsf{Ch}}[t] - \mathbf{y}_K[t] - \hat{W}[t] - \hat{\Sigma}_0[t] - \hat{\mathsf{Maj}}[t] + \mu_a[t]X^{32}$	$[64] \setminus \{1,2,3\}$	a-update, (69)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{e}[t+1] - \hat{a}[t-3] - \hat{e}[t-3] - \hat{\Sigma}_1[t] - \hat{\mathsf{Ch}}[t] - \mathbf{y}_K[t] - \hat{W}[t] + \mu_e[t]X^{32}$	$[64] \setminus \{1,2,3\}$	e-update, (70)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{W}[t] - \hat{W}[t-16] - \hat{\sigma}_0[t] - \hat{W}[t-7] - \hat{\sigma}_1[t] + \mu_W[t]X^{32}$	$[64] \setminus [16]$	schedule recurrence, (76)
$\mathbb{F}_2[X]$	$(X^{32}-1)$	$\phi_2(\hat{W}[t]) \cdot \widehat{\rho_{\sigma_0}}(X) + \phi_2(\widehat{\mathsf{SHR}^3}(W_t)) - \phi_2(\hat{\sigma}_0[t+15])$	$[49] \setminus [1]$	σ_0 rotation + virtual shift, (74)
$\mathbb{F}_2[X]$	$(X^{32}-1)$	$\phi_2(\hat{W}[t]) \cdot \widehat{\rho_{\sigma_1}}(X) + \phi_2(\widehat{\mathsf{SHR}^{10}}(W_t)) - \phi_2(\hat{\sigma}_1[t+2])$	$[62] \setminus [14]$	σ_1 rotation + virtual shift, (75)

Table 6: Non-boundary polynomial constraints of our SHA-256 arithmetization. The "Ideal" column specifies the ideal in which the constraint polynomial must lie. The "Rows t" column lists the row indices $t \in [65]$ at which the constraint is enforced. Note that these are all linear polynomial constraints over $\mathbb{Q}[X]$ or $\mathbb{F}_2[X]$.

Domain	Lookup set	Constraint polynomial	Rows t	Reference
$\mathbb{Q}[X]$	$\in \{0,1\}^{<32}[X]$	$\hat{e}[t] + \hat{e}[t-1] - 2\hat{u}_{ef}[t]$	$[65] \setminus \{1,2\}$	Ch lookup, (66)
$\mathbb{Q}[X]$	$\in \{0,1\}^{<32}[X]$	$(\mathbf{1}_{32} - \hat{e}[t]) + \hat{e}[t-2] - 2\hat{u}_{\neg e,g}[t]$	$[65] \setminus \{1,2\}$	Ch lookup, (67)
$\mathbb{Q}[X]$	$\in \{0,1\}^{<32}[X]$	$\hat{a}[t] + \hat{a}[t-1] + \hat{a}[t-2] - 2\hat{\mathsf{Maj}}[t]$	$[65] \setminus \{1,2\}$	Maj lookup, (68)

Table 7: Affine-combination lookup constraints of our SHA-256 arithmetization. Each constraint requires the given affine combination of witness/input entries to lie in $\{0,1\}^{<32}[X]$, to be enforced with a lookup PIOP. The "Rows t" column lists the row indices $t \in [65]$ at which the constraint is enforced.

Domain	Ideal	Constraint polynomial	Rows t	Reference
$\mathbb{Q}[X]$	(0)	$\hat{a}[1]-\mathbf{y}_a[1],\ \hat{e}[1]-\mathbf{y}_e[1]$	$\{1\}$	initial state, (77)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{a}[2]-\mathbf{y}_h[1]-\hat{\Sigma}_1[1]-\hat{\mathsf{Ch}}[1]-\mathbf{y}_K[1]$ $-\hat{W}[1]-\hat{\Sigma}_0[1]-\hat{\mathsf{Maj}}[1]+\mu_a[1]X^{32}$	$\{1\}$	a-update (t=1)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{e}[2]-\mathbf{y}_d[1]-\mathbf{y}_h[1]-\hat{\Sigma}_1[1]-\hat{\mathsf{Ch}}[1]$ $-\mathbf{y}_K[1]-\hat{W}[1]+\mu_e[1]X^{32}$	$\{1\}$	e-update (t=1)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{a}[3]-\mathbf{y}_g[1]-\hat{\Sigma}_1[2]-\hat{\mathsf{Ch}}[2]-\mathbf{y}_K[2]$ $-\hat{W}[2]-\hat{\Sigma}_0[2]-\hat{\mathsf{Maj}}[2]+\mu_a[2]X^{32}$	$\{2\}$	a-update (t=2)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{e}[3]-\mathbf{y}_c[1]-\mathbf{y}_g[1]-\hat{\Sigma}_1[2]-\hat{\mathsf{Ch}}[2]$ $-\mathbf{y}_K[2]-\hat{W}[2]+\mu_e[2]X^{32}$	$\{2\}$	e-update (t=2)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{a}[4]-\mathbf{y}_f[1]-\hat{\Sigma}_1[3]-\hat{\mathsf{Ch}}[3]-\mathbf{y}_K[3]$ $-\hat{W}[3]-\hat{\Sigma}_0[3]-\hat{\mathsf{Maj}}[3]+\mu_a[3]X^{32}$	$\{3\}$	a-update (t=3)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{e}[4]-\mathbf{y}_b[1]-\mathbf{y}_f[1]-\hat{\Sigma}_1[3]-\hat{\mathsf{Ch}}[3]$ $-\mathbf{y}_K[3]-\hat{W}[3]+\mu_e[3]X^{32}$	$\{3\}$	e-update (t=3)
$\mathbb{Q}[X]$	(0)	$\hat{a}[65]-\mathbf{y}_a[65],\ \hat{e}[65]-\mathbf{y}_e[65]$	$\{65\}$	final state, (78)
$\mathbb{Q}[X]$	$(X-2)$	$\hat{a}[62]-\mathbf{y}_d[65],\ \hat{e}[62]-\mathbf{y}_h[65]$	$\{65\}$	final state, (79)
$\mathbb{Q}[X]$	(0)	$\hat{a}[64]-\mathbf{y}_b[65],\ \hat{a}[63]-\mathbf{y}_c[65]$	$\{65\}$	final state, (80)
$\mathbb{Q}[X]$	(0)	$\hat{e}[64]-\mathbf{y}_f[65],\ \hat{e}[63]-\mathbf{y}_g[65]$	$\{65\}$	final state, (80)
$\mathbb{Q}[X]$	(0)	$\hat{W}[t]-\mathbf{y}_M[t]$ for $t\in[16]$	$[16]$	message init. (81)

Table 8: Boundary polynomial constraints of our SHA-256 arithmetization. Boundary constraints apply only at specific rows. The "Ideal" column specifies the ideal in which the constraint polynomial must lie ((0) denotes exact equality). The "Rows t" column lists the row indices $t\in[65]$ at which the constraint is enforced. Affine-combination boundary lookup constraints are listed separately in Table 9.

Domain	Lookup set	Constraint polynomial	Rows t	Reference
$\mathbb{Q}[X]$	$\in\{0,1\}^{<32}[X]$	$\hat{e}[1]+\mathbf{y}_f[1]-2\hat{u}_{ef}[1]$	$\{1\}$	Ch (t=1), (82)
$\mathbb{Q}[X]$	$\in\{0,1\}^{<32}[X]$	$(\mathbf{1}_{32}-\hat{e}[1])+\mathbf{y}_g[1]-2\hat{u}_{\neg e,g}[1]$	$\{1\}$	Ch (t=1), (83)
$\mathbb{Q}[X]$	$\in\{0,1\}^{<32}[X]$	$\hat{e}[2]+\hat{e}[1]-2\hat{u}_{ef}[2]$	$\{2\}$	Ch (t=2), (85)
$\mathbb{Q}[X]$	$\in\{0,1\}^{<32}[X]$	$(\mathbf{1}_{32}-\hat{e}[2])+\mathbf{y}_f[1]-2\hat{u}_{\neg e,g}[2]$	$\{2\}$	Ch (t=2), (86)
$\mathbb{Q}[X]$	$\in\{0,1\}^{<32}[X]$	$\hat{a}[1]+\mathbf{y}_b[1]+\mathbf{y}_c[1]-2\hat{\mathsf{Maj}}[1]$	$\{1\}$	Maj (t=1), (87)
$\mathbb{Q}[X]$	$\in\{0,1\}^{<32}[X]$	$\hat{a}[2]+\hat{a}[1]+\mathbf{y}_b[1]-2\hat{\mathsf{Maj}}[2]$	$\{2\}$	Maj (t=2), (88)

Table 9: Boundary affine-combination lookup constraints of our SHA-256 arithmetization, extracted from the Ch and Maj base cases.

8.3 ECDSA signature verification

In this section we arithmetize ECDSA signature verification on the secp256k1 curve as a UAIR$^+$ instance. We also sketch how to compose this arithmetization with the SHA-256 arithmetization of Section 8.2 into a single UAIR$^+$ that verifies a signature on the SHA-256 digest of a message.

8.3.1 ECDSA overview and target relation definition

The ECDSA signature scheme, instantiated on secp256k1, operates on the elliptic curve $E/\mathbb{F}_p\colon y^2 = x^3+7$, where p is a 256-bit prime. A fixed base-point $G\in E(\mathbb{F}_p)$ generates a prime-order subgroup $\langle G\rangle\subseteq E(\mathbb{F}_p)$ of order n (also 256-bit, with $p<2n$). The public key is a point $Q\in\langle G\rangle$ (i.e. $Q=dG$ for some secret scalar $d\in\mathbb{F}_n$), messages are elements $e\in\mathbb{F}_n$, and signatures are pairs

$(r, s) \in \mathbb{F}_n \times \mathbb{F}_n^*$. Verification accepts (r, s) as a signature on e under Q if the point $R \coloneqq u_1 G + u_2 Q$ is not the point at infinity and its affine x-coordinate R_x satisfies $R_x \equiv r \pmod{n}$, where $u_1 = e \cdot s^{-1}$ and $u_2 = r \cdot s^{-1}$ in $\mathbb{F}_n$.

The relation we want to arithmetize. The target relation is

$$\mathsf{REL}_{\mathsf{ECDSA}} \coloneqq \left\{ (\mathbb{i}, \mathbb{x}; \mathbb{w}) \;\middle|\; \begin{array}{l} \textbf{Index-input-witness definition:} \\ \quad \mathbb{i} = (G),\ \mathbb{x} = (e, r, s, Q, u_1, u_2),\ \mathbb{w} = (R), \\ \quad u_1, u_2, e, r, s \in \mathbb{F}_n,\ Q, R \in E(\mathbb{F}_p), \\ \quad u_1 \cdot s = e \text{ and } u_2 \cdot s = r \text{ in } \mathbb{F}_n \\ \textbf{Constraints:} \\ \quad R = u_1 G + u_2 Q \neq \infty,\ \mathsf{int}(R_x) \equiv r \pmod{n} \end{array} \right\}, \tag{89}$$

where $\mathsf{int}(R_x)$ denotes the unique integer representative of the affine x-coordinate $R_x \in \mathbb{F}_p$ of R in $[0\,..\,p-1]$.

Remark 8.19 (u_1, u_2 in the public input)**.** We place u_1, u_2 in the public input rather than the witness; the verifier trusts or is responsible for checking that u_1, u_2 are correctly derived from (e, r, s) as $u_1 = e \cdot s^{-1}$ and $u_2 = r \cdot s^{-1}$ in $\mathbb{F}_n$. This shifts these two $\mathbb{F}_n$ multiplications out of the arithmetization, removing the need to include s^{-1} in the witness and to enforce the corresponding well-formedness constraints. In use cases where any of e, r, s must remain private, u_1, u_2 would have to be moved back into the witness and the corresponding well-formedness constraints reinstated; the same applies to other components of the public input below, e.g. $(Q,\, G+Q)$ when their privacy is required.

Note that the computation/verification of u_1, u_2 constitutes a negligible fraction of the overall cost of computing R.

Remark 8.20 (Bit decompositions of u_1, u_2 as public input)**.** Once u_1, u_2 are public, so are their 256-bit canonical decompositions b_1, b_2. We therefore lift the bit columns b_1, b_2 from the witness directly into the public input: the verifier checks externally that each $b_i[t] \in \{0, 1\}$ and that $\sum_{t=1}^{256} b_i[t] \cdot 2^{256-t}$ equals the canonical integer representative of $u_i \in \mathbb{F}_n$ in $[0\,..\,n-1]$. This eliminates the in-circuit bit-decomposition accumulator columns U_1, U_2 and all of their associated constraints (initialization, transition, and final binding to u_1, u_2); since these were the only constraints and trace columns over $\mathbb{F}_n$, the $\mathbb{F}_n$-trace disappears altogether and the prime tuple shrinks to $\mathbf{q} = (p)$. The remaining double-and-add loop reads the bits $b_1[t], b_2[t]$ as integers in $\{0, 1\} \subset \mathbb{Z}$ via the $\mathbb{F}_p$-side embedding ϕ_p (cf. (93)), and computes $u_1 G + u_2 Q$ correctly because G has order n, so the result depends only on the residues of the integer values $\sum b_i[t] \cdot 2^{256-t}$ modulo n. As before, in scenarios where u_1 or u_2 must remain private the bits would have to be moved back to the witness and the bit-decomposition accumulator constraints reinstated.

8.3.2 UAIR$^+$ arithmetization of ECDSA

We now describe a UAIR$^+$ index–input–witness triple that belongs to $\mathsf{REL}_{\mathsf{UAIR}+}$ if and only if a corresponding triple belongs to $\mathsf{REL}_{\mathsf{ECDSA}}$. Our arithmetization is relatively straightforward and, for the most part, uses standard elliptic curve operations over $\mathbb{F}_p$.

The dominant cost (in terms of witness size and number of constraints) comes from the multi-scalar multiplication $R = u_1 G + u_2 Q$. We let b_1 and b_2 be 256-length vectors containing the bits of u_1 and u_2, respectively. Below we discuss how we compute R in our arithmetization.

Shamir's trick, briefly. The standard double-and-add algorithm computes a single scalar multiplication uA, for u a k-bit scalar, by initializing an accumulator $P \leftarrow \infty$ and, for $j = k-1$ down to 0, performing $P \leftarrow 2P$ followed by $P \leftarrow P + A$ whenever bit j of u is 1; the total cost is k doublings and at most k additions. Applied naively to a two-term sum $u_1G + u_2Q$, this would use two such loops for a total of $2k$ doublings and up to $2k$ additions. The so-called *Shamir's trick* merges the two loops into one: first one precomputes the four points

$$T_{00} = \infty, \quad T_{10} = G, \quad T_{01} = Q, \quad T_{11} = G + Q,$$

and then it suffices to run a single double-and-add loop over an accumulator P, initialized to ∞. Concretely, at step j, read the bit-pair $(b_1[j], b_2[j])$ of (u_1, u_2), set $P \leftarrow 2P$, and then set $P \leftarrow P + T_{b_1[j]\,b_2[j]}$. After $k = 256$ steps, $P = u_1G + u_2Q$. The total cost is k doublings and at most k additions.

We assume the point $T_{11} = G + Q$ is given as part of the public input and that it is correctly computed.

Projective coordinates. We run this loop in Jacobian coordinates rather than affine coordinates so that each doubling and addition reduces to a handful of multiplications and squarings in $\mathbb{F}_p$, without field inversions (which would require adding extra witness entries); a single inversion at the end converts the output back to affine form. Concretely, a Jacobian representative $(X : Y : Z)$ with $Z \neq 0$ denotes the affine point $(X/Z^2, Y/Z^3)$, and the point at infinity is any point with $Z = 0$.

Again, we consider UAIR$^+$ instances with extended semantics as discussed in Remarks 8.2 to 8.4.

UAIR$^+$ index. Fix a triple $(\mathbb{i}_{\mathsf{ECDSA}}, \mathbb{x}_{\mathsf{ECDSA}}; \mathbb{w}_{\mathsf{ECDSA}}) \in \mathsf{REL}_{\mathsf{ECDSA}}$. We construct a UAIR$^+$ index

$$\mathbb{i}_{\mathsf{UAIR}} = (n, c, \mathbf{q}, B, \mathbf{d}, \mathcal{C}, \mathcal{S}),$$

as follows:

Prime tuple $\mathbf{q}$. We set $\mathbf{q} = (p)$, so $|\mathbf{q}| = 1$. Accordingly, the UAIR$^+$ has two trace components: a $\mathbb{Q}[X]$-trace (component 0) and an $\mathbb{F}_p[X]$-trace (component 1). No $\mathbb{F}_n$ trace is needed: although u_1, u_2 are conceptually elements of $\mathbb{F}_n$, the arithmetization works directly with their integer bit decompositions and outputs $u_1G + u_2Q$ correctly because G has order n (cf. Remark 8.20 below).

Number of rows n. We set $n = 257$. Rows $t = 1, \ldots, 256$ run the double-and-add loop, with row t processing bit indices $256 - t$ of u_1 and u_2 and writing the updated accumulator P into row $t + 1$; row 257 holds the final accumulator P and performs the signature verification.

Number of columns c. We set $c_0 = 2$ for the $\mathbb{Q}[X]$-trace and $c_p = 9$ for the $\mathbb{F}_p[X]$-trace.

Bit-size B *and degree bounds* $\mathbf{d} = (d_0, d_p)$. We set $B = 256$ and $\mathbf{d} = (1, 1)$, so every trace entry (in both domains) is a field element (degree 0).

Witness. The witness $\mathbb{w}_{\mathsf{UAIR}} = (\mathbf{f}_0, \mathbf{f}_p)$ consists of $\mathbf{f}_0 \in (\mathbb{Q}^{<B,d_0}[X])^{n\times 2}$ and $\mathbf{f}_1 \in (\mathbb{F}_p)^{n\times 9}$. The two $\mathbb{Q}[X]$-columns of $\mathbf{f}_0$ are: the *final-x column* $\hat{x} \in \mathbb{Q}$, storing the integer representative of R_x in $[0\,..\,p-1]$ (enforced by a range-lookup) at entry $t = 257$; and the *quotient column* $k \in \{0, 1\}$, storing the quotient bit of the final signature check $R_x \equiv r \pmod{n}$ at $t = 257$. No constraint depends on the values of $\hat{x}$ or k at rows $t \neq 257$, so their entries at those rows are unconstrained; for concreteness we take them to be zero. The nine $\mathbb{F}_p$-columns of $\mathbf{f}_1$ contain: the Jacobian coordinates of the accumulator $P = (X, Y, Z)$ (3 columns); the coordinates of $2P$, denoted $X_{\mathsf{mid}}, Y_{\mathsf{mid}}, Z_{\mathsf{mid}}$ (3 columns); scratch columns H, R_a used to arithmetize elliptic curve additions (cf. (95)–(96)); and the *affine-conversion column* $Z_{\mathsf{inv}} \in \mathbb{F}_p$, storing $Z[257]^{-1}$ (zero except at $t = 257$).

Public input. The public input is split across both domains of the UAIR$^+$: a $\mathbb{Q}$-block $\mathbf{y}^{(0)}$ with 3 columns and an $\mathbb{F}_p$-block $\mathbf{y}^{(1)}$ with 4 columns. The values e, s from the relation (cf. (89)) are part of the verifier's public input but are not exposed as arithmetization columns since no in-circuit constraint references them; the verifier uses e, s only for the external check $u_1 \cdot s = e$, $u_2 \cdot s = r$ in $\mathbb{F}_n$ (cf. Remark 8.19).

$\mathbb{Q}$ *public-input columns*: The *signature column* $\mathbf{y}_r$ stores the signature element r as an integer in $[0\,..\,n-1]$ (zero except at $t = 257$); it is used in the final signature check (104). We keep $\mathbf{y}_r$ with entries in $\mathbb{Q}$ (rather than $\mathbb{F}_n$) because the final signature check is performed in $\mathbb{Q}$. The *bit columns* b_1, b_2 encode the verification scalars u_1, u_2: for $t \in \{1, \ldots, 256\}$, $b_i[t] \in \{0, 1\}$ is bit $256 - t$ of (the canonical integer representative in $[0\,..\,n-1]$ of) u_i, and $b_i[257] = 0$ (cf. Remark 8.20).

$\mathbb{F}_p$ *public-input columns*: The *public-key columns* Q_x, Q_y have entries in $\mathbb{F}_p$ and they store the affine coordinates of Q at each row (so, they are constant across rows). The *precomputed-point columns* $\bar{R}_x, \bar{R}_y$ store the affine coordinates of $G + Q$ in $\mathbb{F}_p$; they are supplied directly as public input, and, again, constant across rows.

Constraints. We now describe the constraints $\mathcal{C}$. The column-wise lookup constraints are: a range lookup $\hat{x} \in [0\,..\,p-1]$ on column $\hat{x}$, and a bit lookup $k \in \{0, 1\}$ on column k. Bit-membership of b_1, b_2 is checked externally on the public input (cf. Remark 8.20). The remaining constraints are as follows.

The next batch of constraints enforce correct computation of the update $P \leftarrow 2P + T$ in the double-and-add loop, for each $t \in [256]$. These are purely elliptic curve operations expressed natively over the base field $\mathbb{F}_p$, and so are not particularly amenable to the type of techniques introduced in this work. Because of this, and because these are well-known operations, we only provide a brief sketch of the constraints.

Jacobian doubling. The following constraints enforce $(X_{\mathsf{mid}} : Y_{\mathsf{mid}} : Z_{\mathsf{mid}}) = 2(X[t] : Y[t] : Z[t])$ over the secp256k1 curve. They apply for all $t \in [256]$, all in $\mathbb{F}_p$:

$$Z_{\mathsf{mid}}[t] - 2Y[t] \cdot Z[t] \in (0), \tag{90}$$

$$X_{\mathsf{mid}}[t] - \left(9X[t]^4 - 8X[t] \cdot Y[t]^2\right) \in (0), \tag{91}$$

$$Y_{\mathsf{mid}}[t] - \left(3X[t]^2 \cdot (4X[t] \cdot Y[t]^2 - X_{\mathsf{mid}}[t]) - 8Y[t]^4\right) \in (0). \tag{92}$$

Shamir addend selection. The following constraints compute the point $T \in \{\infty, G, Q, G{+}Q\}$ to be added in the update $P \leftarrow 2P{+}T$, for each iteration $t \in [256]$, depending on the bit-pair $(b_1[t], b_2[t])$. For $t \in [256]$, we denote the Jacobian coordinates of T by $(T_x[t], T_y[t], T_z[t])$, and these are obtained as follows. First define the following elements from $\mathbb{F}_p$:

$$\begin{aligned} s_{00}[t] &:- (1 - \phi_p(b_1[t])) \cdot (1 - \phi_p(b_2[t])), \quad & s_{10}[t] &:- \phi_p(b_1[t]) \cdot (1 - \phi_p(b_2[t])), \\ s_{01}[t] &:- (1 - \phi_p(b_1[t])) \cdot \phi_p(b_2[t]), \quad & s_{11}[t] &:- \phi_p(b_1[t]) \cdot \phi_p(b_2[t]), \end{aligned} \tag{93}$$

so that $s_{ij}[t] = 1$ iff $(b_1[t], b_2[t]) = (i, j)$, and then:

$$\begin{aligned} T_z[t] &= s_{10}[t] + s_{01}[t] + s_{11}[t], \\ T_x[t] &= s_{10}[t] \cdot G_x + s_{01}[t] \cdot Q_x[t] + s_{11}[t] \cdot \bar{R}_x[t], \\ T_y[t] &= s_{10}[t] \cdot G_y + s_{01}[t] \cdot Q_y[t] + s_{11}[t] \cdot \bar{R}_y[t]. \end{aligned} \tag{94}$$

The elements $s_{ij}[t]$ and $T_x[t], T_y[t], T_z[t]$ are not stored in the trace; they are just convenient shorthand for the constraints that follow.

Note that $T_z[t]$ is either 0 if $T[t] = \infty$ or 1 otherwise. It is crucially used as a selector for the addition formulas in the next step.

Addition. The following formulas compute the Jacobian coordinates of $2P + T$, for each $t \in [256]$, given the coordinates of P and T. They all occur in $\mathbb{F}_p$:

$$H[t] - \big(T_x[t] \cdot Z_{\mathsf{mid}}[t]^2 - X_{\mathsf{mid}}[t]\big) \in (0), \tag{95}$$
$$R_a[t] - \big(T_y[t] \cdot Z_{\mathsf{mid}}[t]^3 - Y_{\mathsf{mid}}[t]\big) \in (0). \tag{96}$$

When $2P \neq \infty$ and $T \neq \infty$, the point $2P + T$ is given by coordinates $(X_{\mathsf{reg}} : Y_{\mathsf{reg}} : Z_{\mathsf{reg}})$, where

$$\begin{aligned} Z_{\mathsf{reg}}[t] &= H[t] \cdot Z_{\mathsf{mid}}[t], \\ X_{\mathsf{reg}}[t] &= R_a[t]^2 - H[t]^3 - 2X_{\mathsf{mid}}[t] \cdot H[t]^2, \\ Y_{\mathsf{reg}}[t] &= R_a[t] \cdot (X_{\mathsf{mid}}[t] \cdot H[t]^2 - X_{\mathsf{reg}}[t]) - Y_{\mathsf{mid}}[t] \cdot H[t]^3. \end{aligned}$$

Again, the values $X_{\mathsf{reg}}[t], Y_{\mathsf{reg}}[t], Z_{\mathsf{reg}}[t]$ are not stored in the trace; they are shorthand for the constraints that follow.

The cases $2P = \infty$ or $T = \infty$ are handled using selector constraints, defined below. Note that in an honest execution $2P = \infty$ only when $t = 1$, since otherwise $2P$ belongs to the subgroup generated by G.

$$s_1[t] = 1 - T_z[t], \quad s_2[t] = T_z[t] \cdot \mathbf{1}[t = 1], \quad s_{\mathsf{reg}}[t] = 1 - s_1[t] - s_2[t], \tag{97}$$

Both elements $s_1[t], s_2[t]$ are necessarily bits; $s_1[t]$ is 1 iff $T[t] = \infty$; $s_2[t]$ is 1 iff $t = 1$ and $T[1] \neq \infty$; and $s_{\mathsf{reg}}[t]$ is 1 iff $T[t] \neq \infty$ and $t \geq 2$, i.e. the regular case where the addition formulas apply. Note that $2P = \infty$ only at $t = 1$, hence the usage of $\mathbf{1}[t = 1]$ in the definition of $s_2[t]$.

Then, the point $2P + T$ is obtained through the following constraints over $\mathbb{F}_p$:

$$Z[t{+}1] - \big(s_1 \cdot Z_{\mathsf{mid}}[t] + s_2 \cdot T_z[t] + s_{\mathsf{reg}} \cdot Z_{\mathsf{reg}}[t]\big) \in (0), \tag{98}$$
$$X[t{+}1] - \big(s_1 \cdot X_{\mathsf{mid}}[t] + s_2 \cdot T_x[t] + s_{\mathsf{reg}} \cdot X_{\mathsf{reg}}[t]\big) \in (0), \tag{99}$$
$$Y[t{+}1] - \big(s_1 \cdot Y_{\mathsf{mid}}[t] + s_2 \cdot T_y[t] + s_{\mathsf{reg}} \cdot Y_{\mathsf{reg}}[t]\big) \in (0). \tag{100}$$

Boundary constraints. Initialization (row $t = 1$). We set $P = (0 : 0 : 0)$ at $t = 1$. Concretely, we enforce:

$$X[1], Y[1], Z[1] \in (0) \subset \mathbb{F}_p. \tag{101}$$

Final verification (row $t = 257$). Here we enforce that $R = u_1 G + u_2 Q$ is not the point at infinity, and that the integer representative R_x in $[0, p-1]$ of the affine x-coordinate of R equals r modulo n.

Non-infinity and recovery of the affine x-coordinate $R_x = X[257] \cdot Z_{\mathsf{inv}}[257]^2$, binding it to the integer representative $\hat{x}[257] \in [0\,..\,p-1]$:

$$Z[257] \cdot Z_{\mathsf{inv}}[257] - 1 \in (0) \quad \text{in } \mathbb{F}_p, \tag{102}$$
$$X[257] \cdot Z_{\mathsf{inv}}[257]^2 - \phi_p(\hat{x}[257]) \in (0) \quad \text{in } \mathbb{F}_p. \tag{103}$$

The range $\hat{x}[257] \in [0\,..\,p-1]$ is enforced by a column-wise lookup on $\hat{x}$ (see Table 10). Finally, the signature check $r \equiv R_x \pmod{n}$; since $R_x < p < 2n$, the quotient bit $k[257] \in \{0, 1\}$ suffices:

$$\hat{x}[257] - \mathbf{y}_r[257] - k[257] \cdot n \in (0) \quad \text{in } \mathbb{Q}. \tag{104}$$

8.3.3 Arithmetization summary

Tables 10 to 12 summarize the trace and constraints of our ECDSA arithmetization. Selector polynomials are not made explicit; see Remark 8.18.

Lookup set $\subseteq$	Domain	#Rows	#Cols	Columns
$\mathbb{Q}[X]$-trace columns				
$[0\,..\,p-1]$	$\subseteq \mathbb{Q}$	257	1	$\hat{x}$ (zero except at $t = 257$)
$\{0,1\}$	$\subseteq \mathbb{Q}$	257	1	k (zero except at $t = 257$)
$\mathbb{F}_p[X]$-trace columns				
$\mathbb{F}_p$	$\subseteq \mathbb{F}_p$	257	3	X, Y, Z (accumulator P)
$\mathbb{F}_p$	$\subseteq \mathbb{F}_p$	257	3	$X_{\mathsf{mid}}, Y_{\mathsf{mid}}, Z_{\mathsf{mid}}$ (doubled accumulator $2P$)
$\mathbb{F}_p$	$\subseteq \mathbb{F}_p$	257	2	H, R_a (addition scratch)
$\mathbb{F}_p$	$\subseteq \mathbb{F}_p$	257	1	Z_{inv} (zero except at $t = 257$)
Public input columns				
$[0\,..\,n-1]$	$\subseteq \mathbb{Q}$	257	1	$\mathbf{y}_r$ (zero except at $t = 257$)
$\{0,1\}$	$\subseteq \mathbb{Q}$	257	2	b_1, b_2 (bits of u_1, u_2; $b_i[257] = 0$)
$\mathbb{F}_p$	$\subseteq \mathbb{F}_p$	257	2	Q_x, Q_y
$\mathbb{F}_p$	$\subseteq \mathbb{F}_p$	257	2	$\bar{R}_x, \bar{R}_y$ (coordinates of $G + Q$)

Table 10: Trace layout for the ECDSA UAIR$^+$ arithmetization. "Lookup set" indicates range constraint enforced on each column via a lookup PIOP (when the lookup set equals the domain, there is no need to prove such constraint at the PIOP level); "Domain" specifies the ambient set, enforced via the PCS/IOPP (cf. Section 2.1.2).

Domain	Ideal	Constraint polynomial	Rows t	Reference
$\mathbb{F}_p$	(0)	$Z_{\mathsf{mid}}[t] - 2Y[t]Z[t]$	[256]	doubling Z coord., (90)
$\mathbb{F}_p$	(0)	$X_{\mathsf{mid}}[t] - 9X[t]^4 + 8X[t]Y[t]^2$	[256]	doubling X coord., (91)
$\mathbb{F}_p$	(0)	$Y_{\mathsf{mid}}[t] - 3X[t]^2(4X[t]Y[t]^2 - X_{\mathsf{mid}}[t]) + 8Y[t]^4$	[256]	doubling Y coord., (92)
$\mathbb{F}_p$	(0)	$H[t] - T_x[t]Z_{\mathsf{mid}}[t]^2 + X_{\mathsf{mid}}[t]$	[256]	addition scratch H, (95)
$\mathbb{F}_p$	(0)	$R_a[t] - T_y[t]Z_{\mathsf{mid}}[t]^3 + Y_{\mathsf{mid}}[t]$	[256]	addition scratch R_a, (96)
$\mathbb{F}_p$	(0)	$Z[t{+}1] - s_1[t] \cdot Z_{\mathsf{mid}}[t] - s_2[t] \cdot T_z[t] - s_{\mathsf{reg}}[t] \cdot H[t] \cdot Z_{\mathsf{mid}}[t]$	[256]	addition Z coord., (98)
$\mathbb{F}_p$	(0)	$X[t{+}1] - s_1[t] \cdot X_{\mathsf{mid}}[t] - s_2[t] \cdot T_x[t] - s_{\mathsf{reg}}[t] \cdot (R_a[t]^2 - H[t]^3 - 2X_{\mathsf{mid}}[t]H[t]^2)$	[256]	addition X coord., (99)
$\mathbb{F}_p$	(0)	$Y[t{+}1] - s_1[t] \cdot Y_{\mathsf{mid}}[t] - s_2[t] \cdot T_y[t] - s_{\mathsf{reg}}[t] \cdot \big(R_a[t](X_{\mathsf{mid}}[t]H[t]^2 - X_{\mathsf{reg}}[t]) - Y_{\mathsf{mid}}[t]H[t]^3\big)$	[256]	addition Y coord., (100)

Table 11: Double-and-add constraints of the ECDSA arithmetization (applied on rows $t \in [256]$). The "Ideal" column specifies the ideal in which the constraint must lie; "Rows t" lists the row indices at which the constraint is enforced. The selectors $s_1[t], s_2[t], s_{\mathsf{reg}}[t]$ are defined in (97) and $T_x[t], T_y[t], T_z[t]$ in (94). The maximum variable degree is 6 (attained by the addition Y-coordinate constraint, via the monomial $s_{\mathsf{reg}} \cdot R_a \cdot X_{\mathsf{mid}} \cdot H^2$). Boundary constraints are listed separately in Table 12.

Domain	Ideal	Constraint polynomial	Rows t	Reference
$\mathbb{F}_p$	(0)	$X[1], Y[1], Z[1]$	$\{1\}$	init., (101)
$\mathbb{F}_p$	(0)	$Z[257]Z_{\mathsf{inv}}[257] - 1$	257	non-infinity, (102)
$\mathbb{F}_p$	(0)	$X[257]Z_{\mathsf{inv}}[257]^2 - \phi_p(\hat{x}[257])$	257	x-binding, (103)
$\mathbb{Q}$	(0)	$\hat{x}[257] - \mathbf{y}_r[257] - k[257] \cdot n$	257	sig., (104)

Table 12: Boundary constraints of the ECDSA arithmetization, enforced at the initial ($t = 1$) and final ($t = 257$) rows. Columns have the same meaning as in Table 11.

8.4 SHA-256 hashing followed by ECDSA verification

We sketch how to compose our UAIR$^+$ from Sections 8.2.3 and 8.3.2 so as to arithmetize the computation consisting of verifying an ECDSA signature on a SHA-256 hash of a message m. We let B be the number of 512-bit blocks in m. In a nutshell: we stack B copies of the SHA-256 compression arithmetization vertically, with feed-forward constraints enforcing correct chaining of the compressions. We then include the ECDSA UAIR$^+$, with its $\mathbb{Q}[X]$-trace being concatenated horizontally to the $\mathbb{Q}[X]$ trace of the stacked SHA-256 trace (adding zero padding appropriately). The resulting UAIR exposes the SHA-256 digest digest as a public value, and the bits b_1, b_2 of the ECDSA values u_1 and u_2 as public input columns (cf. Remarks 8.19 and 8.20); the verifier then checks externally that $u_1 \cdot s =$ digest and $u_2 \cdot s = r$ in $\mathbb{F}_n$. In settings where more privacy is required, one can keep u_1, u_2, the digest, and maybe the signature elements, as part of the witness, at the expense of adding extra constraints.

9 Further implementation and protocol details

We give a brief overview of the details of our protocols and their implementation. Full details will be provided in a separate further report. We remark that our implemented protocols use, at times, small straightforward deviations or extensions to the protocols described in this paper (all outlined below). Such variations will be included in the mentioned report. Our implementation is available at `github.com/NethermindEth/zinc-plus/tree/main-beta`.

Lookup PIOP. Our lookup constraints are all in the sets $\{0,1\}^{<32}[X]$ and small integer ranges. For the former, we prove the corresponding lookup constraints by proving that $v \cdot (v-1) = 0$ for every coefficient v of every entry of a column constrained to $\{0,1\}^{<32}[X]$. This is proved via a standard sumcheck approach over the random projected field.

The sumcheck outputs claims of the following form: let u be one of the columns whose entries are constrained to belong to $\{0,1\}^{<32}[X]$. Then $u = \sum_{i=0}^{31} X^i \cdot v_i$, where v_i is a column filled with bits. At the end of the aforementioned sumcheck, we are left with claims of the form $\mathsf{mle}[\psi(v_i)](\mathbf{r}) = c_i$ for random $\mathbf{r}$ and some values c_i, and where $\psi : \mathbb{Z}[X] \to \mathbb{F}$ is a projection to a finite field. The prover publishes points $\mathbf{r}'$ and values c_i' over $\mathbb{Z}$ such that $\psi(\mathbf{r}') = \mathbf{r}$ and $\psi(c_i') = c_i$ (and the verifier checks this), and the claim $\mathsf{mle}[v](\mathbf{r}') = \sum_{i=0}^{31} X^i \cdot c_i'$. Prover and verifier then use our instantiation of Zip+ to certify the latter claim.

For the columns constrained to small integer ranges (up to $[0\,..\,6]$ at most), we bit decompose the columns and we pack them into a single column lookup-constrained to $\{0,1\}^{<32}[X]$.

Our SHA-256 + ECDSA UAIR. Our UAIR is essentially the one described in Section 8, with the differences outlined below. Further details can be found in the preliminary documentation of our implementation (`https://github.com/NethermindEth/zinc-plus/tree/main-beta`).

The ECDSA only proves the multiscalar multiplication. As we mentioned, our ECDSA arithmetization includes constraints for the multi-scalar multiplication $R = u_1 G + u_2 Q$, which is the bulk of the signature verification: the rest would entail proving two multiplications modulo n and the constraint $\mathsf{int}(R_x) \equiv r \pmod{n}$ (cf. Section 8.3). In scenarios where privacy and zero-knowledge are necessary, one would need to include the full ECDSA verification in the UAIR. This is ongoing work in our implementation.

Simplification of boundary constraints, at the expense of 3 *more rows per compression.* We use 68 rows per SHA-256 compression (instead of 65) to allow for easier handling of boundary constraints; namely, the first 3 rows store initial register values for b, c, d in the a-column, and f, g, h in the e-column. This allows us to avoid all boundary constraints in Table 9 and boundary constraints 2 to 6 in Table 8, and eases the chaining of multiple compressions.

All the witnesses and constraints are over $\mathbb{Q}[X]$. We do not use witness columns nor constraints over $\mathbb{F}_q$ or $\mathbb{F}_2[X]$ due to limitations of our implementation. This requires slightly modifying the constraints that occur over $\mathbb{F}_2[X]$ (cf. Eqs. (64), (65), (74) and (75)) by adding, per constraint, an extra additive term of the form $2 \cdot w$ for some new witness column w, and similarly for the $\mathbb{F}_q$-constraints. We do so for the $\mathbb{F}_2[X]$-constraints, but for the $\mathbb{F}_q$-constraints, we instead modify our Zinc+ PIOP so that the verifier always sends the secp256k1 prime q instead of a random prime. Soundness of the scheme is still guaranteed because all columns corresponding to the SHA-256 arithmetization are lookup-constrained to $\{0,1\}^{<32}[X]$ or small range checks, and the SHA-256 constraints are so that there is never an overflow over the prime q (cf. Section 8.2.3). This allows us to avoid adding extra columns for the $\mathbb{F}_q$-constraints.

The aforementioned extra witness columns added for the $\mathbb{F}_2[X]$-constraints are kept public. This hinders zero-knowledge, but it reasonably keeps our performance closer to what a full implementation would look like.

Keeping most SHA constraints linear. The steps in our PIOR scheme (Steps 1, 2, and 3 of Section 2.2) are especially efficient for linear constraints. As specified in Section 8, all of the SHA-256 compression constraints are linear, except, potentially, for the presence of selector polynomials that ensure constraints only apply at certain rows, cf. Remark 8.18. Selector polynomials increase constraint degree by one, and thus we seek alternatives to them. Concretely, in this version of the implementation we added public *corrector columns* for linear non-boundary constraints. For each constraint that is meant to apply to a subset $S \subseteq [n]$ of all n rows, we linearly add a public column c_S to the constraint, with $c_S(i) = 0$ if $i \in S$ and $c_S(i)$ takes any value required for the constraint to hold in $[n] \setminus S$. This way, the constraint is satisfied at every row, but it is only meaningful at rows in S. The verifier checks that c_S is indeed zero at S. Boundary constraints are treated with selector polynomials in a standard way.
In the future, we plan to replace corrector columns with a more elaborate way of handling entry-wise shifts of witness vectors (columns), similarly as done in [Tho24]. This latter approach is amenable to zero-knowledge, unlike the above approach.

Packing SHA's integer columns as one single column lookup constrained to $\{0,1\}^{<32}[X]$. The integer-typed columns in the SHA-256 UAIR (which belong to small ranges, namely $[0\,..\,6], [0\,..\,5], [0\,..\,3]$) are bit-decomposed and packed together into a single column $\mathbf{u}$ lookup-constrained to $\{0,1\}^{<32}[X]$. We access MLE evaluations of the integer columns virtually, as these can be recovered from MLE evaluations of $\mathbf{u}$, in a similar manner as explained above in "Lookup PIOP".

Protocol optimization: Folded Zip+. To reduce proof sizes, we use the following extension of our Zip+ IOPP, which we call *Folded Zip+*. Say we want to commit to the MLE of a witness

vector v with entries in $\{0,1\}^{<32}[X]$. We write $v = v_1 + X^{16} \cdot v_2$, for vectors v_1, v_2 with entries in $\{0,1\}^{<16}[X]$. We then commit instead to the vector $v' = (v_1, v_2)$ resulting from concatenating v_1 and v_2. We get virtual access to $\mathsf{mle}[v](\mathbf{Y})$ through the identity $\mathsf{mle}[v'](Y_0, \mathbf{Y}) = (1 - Y_0) \cdot \mathsf{mle}[v_1](\mathbf{Y}) + Y_0 \cdot \mathsf{mle}[v_2](\mathbf{Y})$. Concretely, say we wish to prove an MLE evaluation claim for the original vector $\mathbf{v}$, say $\mathsf{mle}[v](\mathbf{r}) = c$ over $\mathbb{Q}[X]$. We can instead replace the claim by two claims $\mathsf{mle}[v_1](\mathbf{r}) = c_1, \mathsf{mle}[v_2](\mathbf{r}) = c_2$ over $\mathbb{Q}[X]$, with the verifier checking that $c = c_1 + X^{16} \cdot c_2$. Then the verifier sends a random challenge r_0 from a large subset of $\mathbb{Z}$, and then prover and verifier prove the claim $\mathsf{mle}[v'](r_0, \mathbf{r}) = (1 - r_0) \cdot c_1 + r_0 \cdot c_2$. The soundness of this process follows from standard Schwartz-Zippel arguments. This process can be iterated, e.g., we can write $v_1 = v_{11} + X^8 \cdot v_{12}$ and $v_2 = v_{21} + X^8 \cdot v_{22}$, and commit to $v'' = (v_{11}, v_{12}, v_{21}, v_{22})$.

This variation has a substantial effect on proof size reduction, with little expense on prover and verifier time. This is because the bit-size of the opened columns during the Zip+ IOPP, the main contributor to proof size, is essentially halved (or reduced by a factor of 4 in the iterated example above).

Details of our finite field PIOP and IOPP. As we mentioned in Section 2.5, our finite field PIOP is comprised of the following components: a sumcheck-based protocol similar to Hyperplonk's PIOP [CBBZ23]; the aforementioned Lookup PIOP for the set $\{0,1\}^{<32}[X]$; and a sumcheck-based protocol for reducing MLE evaluation claims of shifted vectors $\mathbf{v}^{\downarrow\Delta}$ to MLE evaluation claims of $\mathbf{v}$ (as in Section 4.3 of [DP25]), cf. Remark 8.18.

The MLE evaluation claims for the vectors $\phi_q(v_i)$ (where q is the projection prime and ϕ_q is reduction modulo q) are provided virtually through MLE claims of $\phi_q(\sum_i X^i v_i)$, as explained above. The latter are proved with the version Zip+ that allows committing to vectors $\mathbf{u}$ with entries in $\mathbb{Q}^{<d}[X]$ and proving MLE claims over $\mathbb{F}_q^{<d}[X]$ of the form $\mathsf{mle}[\mathbf{u}](\mathbf{r}) = c$ for evaluation points $\mathbf{r}$ over $\mathbb{F}_q$ and $c \in \mathbb{F}_q[X]$. Similarly, MLE evaluation claims for vectors of the form $\psi(\mathsf{SHR}^i(\mathbf{u}))$, $\mathbf{u} \in \{0,1\}^{<32}[X]$ (where $\psi : \mathbb{Z}_{(q)}[X] \to \mathbb{F}_q$ is our usual projection, cf. Definition 5.2, and SHR^i denotes entry-wise coefficient shift), we first lift the claim over $\mathbb{F}_q[X]$, and reduce the resulting to a claim about the MLE of $\phi_q(\mathbf{v})$ using Lemma 2.3. The latter is then proved with Zip+.

We use the iterated folded version of Zip+ described above.

AI components. The following parts of our implementation were completed with help from AI and careful posterior revision from us. These will be appropriately incorporated into the main branch in upcoming repository updates.

The parts are: the lookup PIOP; the folded Zip+ optimization; UAIR instance construction; virtual access to entry-wise SHIFT of columns typed in $\{0,1\}^{<32}[X]$; IPRS code configurations; and some optimizations[10].

We have not included several optimizations that we identified with the help of AI. These will be incorporated in future updates as we review them.

References

[ACCDE22] T. Attema, I. Cascudo, R. Cramer, I. B. Damgård, and D. Escudero. *Vector Commitments over Rings and Compressed Σ-Protocols*. Cryptology ePrint Archive, Report 2022/181. 2022. URL: https://eprint.iacr.org/2022/181 (cited on p. 11).

[10]These can be found in the following respective commits: Lookup: 70ee5cb, a9c2aab, de22384, 521d5bd; Folded Zip+: 0204c16, e0fe965; UAIR: d9a2665, 479933c, b0a8568, e0ef3af, f932f9c, ee045f9, 566ae45, c0a4770, 955f51d; Virtual SHIFT and ROTR: b8671c7, c399a0f; IPRS configs: be33e22; Optimizations: d97284d, 45d7ab6, 12b1abf, 5a49d8d, 31880d6, 6517386, ed245a9, 955f51d

[ACFY24] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. "STIR: Reed-Solomon Proximity Testing with Fewer Queries". In: *CRYPTO 2024, Part X*. Ed. by L. Reyzin and D. Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 380–413. DOI: 10.1007/978-3-031-68403-6_12 (cited on p. 125).

[ACFY25a] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. "WHIR: Reed-Solomon Proximity Testing with Super-Fast Verification". In: *EUROCRYPT 2025, Part IV*. Ed. by S. Fehr and P.-A. Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 214–243. DOI: 10.1007/978-3-031-91134-7_8 (cited on pp. 32, 125).

[ACFY25b] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. "WHIR: Reed-Solomon Proximity Testing with Super-Fast Verification". In: *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*. Ed. by S. Fehr and P. Fouque. Vol. 15604. Lecture Notes in Computer Science. Springer, 2025, pp. 214–243. DOI: 10.1007/978-3-031-91134-7_8. URL: https://eprint.iacr.org/2024/1586 (cited on p. 33).

[AHIV17] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *ACM CCS 2017*. Ed. by B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu. ACM Press, Oct. 2017, pp. 2087–2104. DOI: 10.1145/3133956.3134104 (cited on p. 9).

[AHIV22] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. *Ligero: Lightweight Sublinear Arguments Without a Trusted Setup*. Cryptology ePrint Archive, Paper 2022/1608. CCS 2017. 2022. DOI: 10.1145/3133956. URL: https://eprint.iacr.org/2022/1608 (cited on p. 33).

[AST24] A. Arun, S. T. V. Setty, and J. Thaler. "Jolt: SNARKs for Virtual Machines via Lookups". In: *EUROCRYPT 2024, Part VI*. Ed. by M. Joye and G. Leander. Vol. 14656. LNCS. Springer, Cham, May 2024, pp. 3–33. DOI: 10.1007/978-3-031-58751-1_1 (cited on p. 3).

[BBMRS21] C. Baum, L. Braun, A. Munch-Hansen, B. Razet, and P. Scholl. "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and Z2k". In: *ACM CCS 2021*. Ed. by G. Vigna and E. Shi. ACM Press, Nov. 2021, pp. 192–211. DOI: 10.1145/3460120.3484812 (cited on p. 11).

[BBMS22] C. Baum, L. Braun, A. Munch-Hansen, and P. Scholl. "Moz Z 2 k arella: efficient vector-OLE and zero-knowledge proofs over Z 2 k". In: *Annual International Cryptology Conference*. Springer. 2022, pp. 329–358 (cited on p. 11).

[BCFRRZ25] M. Brehm, B. Chen, B. Fisch, N. Resch, R. D. Rothblum, and H. Zeilberger. "Blaze: Fast SNARKs from Interleaved RAA Codes". In: *EUROCRYPT 2025, Part IV*. Ed. by S. Fehr and P.-A. Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 123–152. DOI: 10.1007/978-3-031-91134-7_5 (cited on pp. 10, 24).

[BCFW25] B. Bünz, A. Chiesa, G. Fenzi, and W. Wang. "Linear-Time Accumulation Schemes". In: *TCC 2025, Part I*. Ed. by B. Applebaum and H. (Lin. Vol. 16268. LNCS. Springer, Cham, Dec. 2025, pp. 369–399. DOI: 10.1007/978-3-032-12287-2_13 (cited on pp. 32, 39, 40, 50, 110).

[BCGM25] S. Bordage, A. Chiesa, Z. Guan, and I. Manzur. *All Polynomial Generators Preserve Distance with Mutual Correlated Agreement*. Cryptology ePrint Archive, Report 2025/2051. 2025. URL: https://eprint.iacr.org/2025/2051 (cited on pp. 28, 71).

[BCPS18] A. Bishnoi, P. L. Clark, A. Potukuchi, and J. R. Schmitt. "On zeros of a polynomial in a finite grid". In: *Combinatorics, Probability and Computing* 27.3 (2018), pp. 310–333. DOI: 10.1017/S0963548317000566. URL: https://arxiv.org/abs/1508.06020 (cited on p. 36).

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. "Interactive Oracle Proofs". In: *TCC 2016-B, Part II*. Ed. by M. Hirt and A. D. Smith. Vol. 9986. LNCS. Springer, Berlin, Heidelberg, Oct. 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2 (cited on pp. 40, 47).

[BCS21] J. Bootle, A. Chiesa, and K. Sotiraki. “Sumcheck Arguments and Their Applications”. In: *CRYPTO 2021, Part I*. Ed. by T. Malkin and C. Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 742–773. DOI: 10.1007/978-3-030-84242-0_26 (cited on p. 11).

[BCTV14] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. “Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture”. In: *USENIX Security 2014*. Ed. by K. Fu and J. Jung. USENIX Association, Aug. 2014, pp. 781–796. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson (cited on p. 3).

[BF22] B. Bünz and B. Fisch. *Multilinear Schwartz–Zippel mod N with Applications to Succinct Arguments*. Cryptology ePrint Archive, Paper 2022/458. TCC 2023. 2022. URL: https://eprint.iacr.org/2022/458 (cited on p. 11).

[BFKTWZ24] A. R. Block, Z. Fang, J. Katz, J. Thaler, H. Waldner, and Y. Zhang. “Field-Agnostic SNARKs from Expand-Accumulate Codes”. In: *CRYPTO 2024, Part X*. Ed. by L. Reyzin and D. Stebila. Vol. 14929. LNCS. Springer, Cham, Aug. 2024, pp. 276–307. DOI: 10.1007/978-3-031-68403-6_9 (cited on pp. 3, 10, 24).

[BHRRS21] A. R. Block, J. Holmgren, A. Rosen, R. D. Rothblum, and P. Soni. “Time- and Space-Efficient Arguments from Groups of Unknown Order”. In: *CRYPTO 2021, Part IV*. Ed. by T. Malkin and C. Peikert. Vol. 12828. LNCS. Virtual Event: Springer, Cham, Aug. 2021, pp. 123–152. DOI: 10.1007/978-3-030-84259-8_5 (cited on p. 11).

[BMNW25] B. Bünz, P. Mishra, W. Nguyen, and W. Wang. “Arc: Accumulation for Reed-Solomon Codes”. In: *CRYPTO 2025, Part VII*. Ed. by Y. T. Kalai and S. F. Kamara. Vol. 16006. LNCS. Springer, Cham, Aug. 2025, pp. 128–160. DOI: 10.1007/978-3-032-01907-3_5 (cited on pp. 5, 39, 40).

[BS23] W. Beullens and G. Seiler. “LaBRADOR: Compact Proofs for R1CS from Module-SIS”. In: *CRYPTO 2023, Part V*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14085. LNCS. Springer, Cham, Aug. 2023, pp. 518–548. DOI: 10.1007/978-3-031-38554-4_17 (cited on p. 11).

[CBBZ23] B. Chen, B. Bünz, D. Boneh, and Z. Zhang. “HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates”. In: *EUROCRYPT 2023, Part II*. Ed. by C. Hazay and M. Stam. Vol. 14005. LNCS. Springer, Cham, Apr. 2023, pp. 499–530. DOI: 10.1007/978-3-031-30617-4_17 (cited on pp. 34, 97).

[CH24] M. Campanelli and M. Hall-Andersen. *Fully Succinct Arguments over the Integers from First Principles*. Cryptology ePrint Archive, Report 2024/1548. 2024. URL: https://eprint.iacr.org/2024/1548 (cited on p. 11).

[CMS19] A. Chiesa, P. Manohar, and N. Spooner. “Succinct Arguments in the Quantum Random Oracle Model”. In: *TCC 2019, Part II*. Ed. by D. Hofheinz and A. Rosen. Vol. 11892. LNCS. Springer, Cham, Dec. 2019, pp. 1–29. DOI: 10.1007/978-3-030-36033-7_1 (cited on pp. 39, 40).

[Con26] O. Contributors. *SWIRL: Stacked WHIR with Interaction Reductions via LogUp*. Manuscript. Preprint, accessed 2026-02-12. Jan. 2026. URL: https://t.co/nazFilz84R (cited on p. 12).

[DP24] B. E. Diamond and J. Posen. *Polylogarithmic Proofs for Multilinears over Binary Towers*. Cryptology ePrint Archive, Report 2024/504. 2024. URL: https://eprint.iacr.org/2024/504 (cited on p. 11).

[DP25] B. E. Diamond and J. Posen. “Succinct Arguments over Towers of Binary Fields”. In: *EUROCRYPT 2025, Part IV*. Ed. by S. Fehr and P.-A. Fouque. Vol. 15604. LNCS. Springer, Cham, May 2025, pp. 93–122. DOI: 10.1007/978-3-031-91134-7_4 (cited on pp. 3, 11, 34, 75, 85, 97).

[FF25] S. Fehr and P.-A. Fouque, eds. *EUROCRYPT 2025, Part IV*. Vol. 15604. LNCS. Springer, Cham, May 2025.

[Fou] E. Foundation. *Ethereum Proof Benchmarks — CSP Benchmarks*. https://ethproofs.org/csp-benchmarks. Accessed: 2026-02-12 (cited on p. 11).

[Fs24] M. Frigo and a. shelat. *Anonymous credentials from ECDSA*. Cryptology ePrint Archive, Report 2024/2010. 2024. URL: https://eprint.iacr.org/2024/2010 (cited on p. 12).

[FZ25] G. Fenzi and Y. Zhang. *zip: Reducing Proof Sizes for Hash-Based SNARGs*. Cryptology ePrint Archive, Report 2025/1446. 2025. URL: https://eprint.iacr.org/2025/1446 (cited on p. 34).

[Gar+25] A. Garreta et al. "Zinc: Succinct Arguments with Small Arithmetization Overheads from IOPs of Proximity to the Integers". In: *Advances in Cryptology - CRYPTO 2025: 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VII*. Santa Barbara, CA, USA: Springer-Verlag, 2025, pp. 259–291. ISBN: 978-3-032-01906-6. DOI: 10.1007/978-3-032-01907-3_9. URL: https://doi.org/10.1007/978-3-032-01907-3_9 (cited on pp. 3, 9, 11, 13, 24, 45–47, 65).

[GLSTW23] A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, and R. S. Wahby. "Brakedown: Linear-Time and Field-Agnostic SNARKs for R1CS". In: *CRYPTO 2023, Part II*. Ed. by H. Handschuh and A. Lysyanskaya. Vol. 14082. LNCS. Springer, Cham, Aug. 2023, pp. 193–226. DOI: 10.1007/978-3-031-38545-2_7 (cited on pp. 9, 11, 33).

[GNS23] C. Ganesh, A. Nitulescu, and E. Soria-Vazquez. "Rinocchio: SNARKs for Ring Arithmetic". In: *Journal of Cryptology* 36.4 (Oct. 2023), p. 41. DOI: 10.1007/s00145-023-09481-3 (cited on p. 11).

[GPR21] L. Goldberg, S. Papini, and M. Riabzev. *Cairo – a Turing-complete STARK-friendly CPU architecture*. Cryptology ePrint Archive, Report 2021/1063. 2021. URL: https://eprint.iacr.org/2021/1063 (cited on p. 3).

[GWHD25] A. Garreta, H. Waldner, K. Hristova, and L. Dall'Ava. *Zinc: Succinct Arguments with Small Arithmetization Overheads from IOPs of Proximity to the Integers*. Cryptology ePrint Archive, Paper 2025/316. 2025. URL: https://eprint.iacr.org/2025/316 (cited on pp. 3, 9, 11, 13, 24, 45–47, 65).

[Hab22] U. Haböck. *Multivariate lookups based on logarithmic derivatives*. Cryptology ePrint Archive, Report 2022/1530. 2022. URL: https://eprint.iacr.org/2022/1530 (cited on p. 46).

[Hab25] U. Haböck. *A note on mutual correlated agreement for Reed-Solomon codes*. Cryptology ePrint Archive, Report 2025/2110. 2025. URL: https://eprint.iacr.org/2025/2110 (cited on pp. 28, 71).

[HMZ25] M.-Y. M. Huang, X. Mao, and J. Zhang. *Sublinear Proofs over Polynomial Rings*. Cryptology ePrint Archive, Report 2025/199. 2025. URL: https://eprint.iacr.org/2025/199 (cited on p. 11).

[Irr26] Irreducible Team. *M3: Multi-Multiset Matching*. Binius documentation. 2026. URL: https://www.binius.xyz/basics/binius-v0/m3 (cited on p. 6).

[KS25] D. Kaviani and S. Setty. *Vega: Low-Latency Zero-Knowledge Proofs over Existing Credentials*. Cryptology ePrint Archive, Report 2025/2094. 2025. URL: https://eprint.iacr.org/2025/2094 (cited on p. 12).

[KST22] A. Kothapalli, S. Setty, and I. Tzialla. "Nova: Recursive Zero-Knowledge Arguments from Folding Schemes". In: *CRYPTO 2022, Part IV*. Ed. by Y. Dodis and T. Shrimpton. Vol. 13510. LNCS. Springer, Cham, Aug. 2022, pp. 359–388. DOI: 10.1007/978-3-031-15985-5_13 (cited on p. 5).

[LXY24] F. Lin, C. Xing, and Y. Yao. "More Efficient Zero-Knowledge Protocols over $\mathbb{Z}_{2^k}$ via Galois Rings". In: *CRYPTO 2024, Part IX*. Ed. by L. Reyzin and D. Stebila. Vol. 14928. LNCS. Springer, Cham, Aug. 2024, pp. 424–457. DOI: 10.1007/978-3-031-68400-5_13 (cited on p. 11).

[OKMZ25] M. Orrù, G. Kadianakis, M. Maller, and G. Zaverucha. "Beyond the Circuit: How to minimize foreign arithmetic in ZKP circuits". In: *CiC* 2.1 (2025), p. 23. DOI: 10.62056/an-4c3c2h (cited on p. 3).

[Pai+23] S. Pailoor et al. *Automated Detection of Underconstrained Circuits for Zero-Knowledge Proofs*. Cryptology ePrint Archive, Report 2023/512. 2023. URL: https://eprint.iacr.org/2023/512 (cited on p. 3).

[RS24] L. Reyzin and D. Stebila, eds. *CRYPTO 2024, Part X*. Vol. 14929. LNCS. Springer, Cham, Aug. 2024.

[Sor22] E. Soria-Vazquez. "Doubly Efficient Interactive Proofs over Infinite and Non-commutative Rings". In: *TCC 2022, Part I*. Ed. by E. Kiltz and V. Vaikuntanathan. Vol. 13747. LNCS. Springer, Cham, Nov. 2022, pp. 497–525. DOI: 10.1007/978-3-031-22318-1_18 (cited on p. 11).

[SSPP26] A. Shirzad, S. Sridhar, D. Papadopoulos, and C. Papamanthou. *Relaxed Modular PCS from Arbitrary PCS and Applications to SNARKs for Integers*. Cryptology ePrint Archive, Paper 2026/347. 2026. URL: https://eprint.iacr.org/2026/347 (cited on p. 11).

[ST25] S. Setty and J. Thaler. *Twist and Shout: Faster memory checking arguments via one-hot addressing and increments*. Cryptology ePrint Archive, Report 2025/105. 2025. URL: https://eprint.iacr.org/2025/105 (cited on p. 46).

[Sta21] StarkWare. *ethSTARK Documentation*. Cryptology ePrint Archive, Report 2021/582. 2021. URL: https://eprint.iacr.org/2021/582 (cited on pp. 6, 47, 72).

[STW23] S. Setty, J. Thaler, and R. Wahby. *Customizable constraint systems for succinct arguments*. Cryptology ePrint Archive, Report 2023/552. 2023. URL: https://eprint.iacr.org/2023/552 (cited on p. 47).

[SZ25] S. Sridhar and Y.-N. Zhang. "FREPack: Improved SNARK Frontend for Highly Repetitive Computations". In: *ASIACRYPT 2025, Part V*. Ed. by G. Hanaoka and B.-Y. Yang. Vol. 16249. LNCS. Springer, Singapore, Dec. 2025, pp. 490–520. DOI: 10.1007/978-981-95-5116-3_16 (cited on p. 12).

[Tho24] T. W. Thomas Coratger. *Whirlaway*. GitHub repository. 2024. URL: https://github.com/TomWambsgans/Whirlaway/Whirlaway.pdf (cited on pp. 73, 87, 96).

[Wen+24] H. Wen et al. "Practical Security Analysis of Zero-Knowledge Proof Circuits". In: *USENIX Security 2024*. Ed. by D. Balzarotti and W. Xu. USENIX Association, Aug. 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/wen (cited on p. 3).

[WZD25] Y. Wei, X. Zhang, and Y. Deng. "Transparent SNARKs over Galois Rings". In: *PKC 2025, Part I*. Ed. by T. Jager and J. Pan. Vol. 15674. LNCS. Springer, Cham, May 2025, pp. 418–451. DOI: 10.1007/978-3-031-91820-9_14 (cited on p. 11).

[Zei24] H. Zeilberger. *Khatam: Reducing the Communication Complexity of Code-Based SNARKs*. Cryptology ePrint Archive, Report 2024/1843. 2024. URL: https://eprint.iacr.org/2024/1843 (cited on pp. 5, 32, 37, 71, 121, 123).

A Deferred Proofs

This appendix collects the deferred proofs from Section 5 (Zinc+ PIOR) and Section 6 (Zip+ IOPP), grouped by topic. Supplementary lemmas used only inside this appendix are placed near the proof that first cites them.

A.1 Proofs for the Zinc+ PIOR

This subsection proves the deferred Zinc+ PIOP results. Its subsubsections map onto the corresponding subsections of Section 5.

A.1.1 Ideal batching, MLE projection commutativity, and well-definedness detection

Ideal batching (Lemma 5.1).

Proof. Let $\mathcal{R} := \mathbb{K}[X]/\mathfrak{I}$ and let $\pi : \mathbb{K}[X] \to \mathcal{R}$ be the natural projection map. Since $\mathfrak{I} \cap \mathbb{K} = (0)$, the induced map $\mathbb{K} \hookrightarrow \mathcal{R}$ is injective.

For Item 1, assume $\mathbf{v}(\mathbf{x}) \in \mathfrak{I}$ for all $\mathbf{x} \in \{0,1\}^\mu$ and fix $\mathbf{r} \in \mathbb{K}^\mu$. By definition of the multilinear extension and closure of $\mathfrak{I}$ under $\mathbb{K}[X]$-linear combinations,

$$\widetilde{\mathbf{v}}(\mathbf{r}) \;=\; \sum_{\mathbf{x}\in\{0,1\}^\mu} \mathbf{v}(\mathbf{x}) \cdot \widetilde{\mathsf{eq}}(\mathbf{x};\mathbf{r}) \;\in\; \mathfrak{I},$$

because each coefficient $\widetilde{\mathsf{eq}}(\mathbf{x};\mathbf{r})$ lies in $\mathbb{K} \subseteq \mathbb{K}[X]$.

For Item 2, assume $\mathbf{v}(\mathbf{x}_0) \notin \mathfrak{I}$ for some $\mathbf{x}_0 \in \{0,1\}^\mu$. Define $\bar{\mathbf{v}} : \{0,1\}^\mu \to \mathcal{R}$ by $\bar{\mathbf{v}}(\mathbf{x}) := \pi(\mathbf{v}(\mathbf{x}))$. Then $\bar{\mathbf{v}}(\mathbf{x}_0) \neq 0$, so the multilinear polynomial $\widetilde{\bar{\mathbf{v}}}(\mathbf{Y}) \in \mathcal{R}[\mathbf{Y}]$ is nonzero.

Let $S \subseteq \mathbb{K} \subseteq \mathcal{R}$. For distinct $a, b \in S$, the difference $a - b$ is a nonzero element of the field $\mathbb{K}$, and hence a unit in $\mathcal{R}$. Therefore S is an exceptional set in $\mathcal{R}$, and Lemma 3.1 yields

$$\Pr_{\mathbf{r}\leftarrow S^\mu}\big[\widetilde{\bar{\mathbf{v}}}(\mathbf{r}) = 0\big] \;\le\; \frac{\mu}{|S|}.$$

Finally, for every $\mathbf{r} \in \mathbb{K}^\mu$, coefficient-wise application of π gives $\widetilde{\bar{\mathbf{v}}}(\mathbf{r}) = \pi(\widetilde{\mathbf{v}}(\mathbf{r}))$. Thus $\widetilde{\bar{\mathbf{v}}}(\mathbf{r}) = 0$ holds if and only if $\widetilde{\mathbf{v}}(\mathbf{r}) \in \mathfrak{I}$, which proves the claimed bound. □

MLE projection commutativity (Lemma 5.4).

Proof. Let $\mu := \log n$ and index the entries of $\mathbf{v}$ by points in the Boolean hypercube $\{0,1\}^\mu$, writing $\mathbf{v} = (v_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^\mu}$. By definition of multilinear extension (see Section 3.1), we have

$$\tilde{\mathbf{v}}(\mathbf{X}) \;=\; \sum_{\mathbf{b}\in\{0,1\}^\mu} v_{\mathbf{b}} \cdot \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{X}) \qquad \text{in } \mathcal{R}[\mathbf{X}],$$

and

$$\widetilde{\psi(\mathbf{v})}(\mathbf{X}) \;=\; \sum_{\mathbf{b}\in\{0,1\}^\mu} \psi(v_{\mathbf{b}}) \cdot \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{X}) \qquad \text{in } \mathcal{R}'[\mathbf{X}].$$

Let $\mathbf{u} \in \mathcal{R}'^\mu$ and let $\mathbf{r} = \psi^{-1}(\mathbf{u})$. Since ψ is a ring homomorphism and $\widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{X})$ is built from the ring operations and the constants $0, 1 \in \mathcal{R}$, we have $\psi\big(\widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{r})\big) = \widetilde{\mathsf{eq}}(\mathbf{b};\psi(\mathbf{r})) = \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{u})$ for every $\mathbf{b} \in \{0,1\}^\mu$. Hence,

$$\begin{aligned}\psi\big(\tilde{\mathbf{v}}(\psi^{-1}(\mathbf{u}))\big) &= \psi\Big(\sum_{\mathbf{b}\in\{0,1\}^\mu} v_{\mathbf{b}} \cdot \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{r})\Big) = \sum_{\mathbf{b}\in\{0,1\}^\mu} \psi(v_{\mathbf{b}}) \cdot \psi\big(\widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{r})\big) \\ &= \sum_{\mathbf{b}\in\{0,1\}^\mu} \psi(v_{\mathbf{b}}) \cdot \widetilde{\mathsf{eq}}(\mathbf{b};\mathbf{u}) = \widetilde{\psi(\mathbf{v})}(\mathbf{u}).\end{aligned}$$

Well-definedness detection (Theorem 5.5).

Since $\tilde{\mathbf{f}} \notin (\mathbb{Z}_{(p)}[X])[\mathbf{W}]$, there exist $\mathbf{b} \in \{0,1\}^\nu$ and an index k such that p divides the denominator of the coefficient $\tilde{f}_{\mathbf{b},k}$ (written in lowest terms). Let p^n be the largest power of p dividing the denominator of any such coefficient. Define $\tilde{\boldsymbol{g}} := p^n\tilde{\mathbf{f}} \in (\mathbb{Z}_{(p)}[X])[\mathbf{W}]$. By construction, at least one X-coefficient of $\tilde{\boldsymbol{g}}$ has numerator not divisible by p.

Suppose $\tilde{\mathbf{f}}(\mathbf{z}) \in \mathbb{Z}_{(p)}[X]$. Then every coefficient of $\tilde{\boldsymbol{g}}(\mathbf{z}) = p^n\tilde{\mathbf{f}}(\mathbf{z})$ lies in $\mathbb{Z}_{(p)}$ and has numerator divisible by p, so $\phi_p(\tilde{\boldsymbol{g}}(\mathbf{z})) = 0$.

Now, $\phi_p(\tilde{\boldsymbol{g}}) \neq 0$ in $(\mathbb{F}_p[X])[\mathbf{W}]$ by the choice of n. Since $\phi_p(S) = \mathbb{F}_p$ and the $z_{i,k}$ are sampled independently from S, the reductions $\phi_p(z_{i,k})$ are independent and uniform over $\mathbb{F}_p$. Hence $\phi_p(z_i) = \sum_k \phi_p(z_{i,k})\, X^k$ is uniform over $\mathbb{F}_p^{<\kappa}[X]$. Since $\mathbb{F}_p[X]$ is an integral domain, a nonzero affine polynomial in one variable has at most one root in $\mathbb{F}_p^{<\kappa}[X]$, and the multilinear Schwartz-Zippel bound gives

$$\Pr\big[\phi_p(\tilde{\boldsymbol{g}}(\mathbf{z})) = 0\big] \;\le\; \nu/p^\kappa.$$

□

A.1.2 $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ construction

Proof of Theorem 5.11. The construction uses the inner PESAT-PIOP Π as a black box, so Π''s knowledge state function and extractor invoke Π's as subroutines. Per Remark 3.10, Π''s knowledge state witness includes Π's witness as an auxiliary component alongside the candidate $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ witness.

Completeness. If $(\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}})$, constraint (i) ensures ψ_0 is defined on every entry of $\mathbf{f}_0$, so the projected oracle never returns $\perp$ on V_Π's queries to $\widetilde{[[\widetilde{\mathbf{f}}_0]]}$ and V' never rejects. By Lemma 5.4, the responses V' forwards to V_Π equal $\widetilde{\psi_0(\mathbf{f}_0)}(\mathbf{u})$ and $\widetilde{\psi_1(\mathbf{f}_1)}(\mathbf{u})$ on every query $\mathbf{u}$, and constraint (ii) gives $(\bar{\mathbb{i}}, \bar{\mathbb{x}}; (\psi_0(\mathbf{f}_0), \psi_1(\mathbf{f}_1))) \in \mathsf{PESAT}(\mathbb{F}_{\tilde{q}})$.

Round-by-round knowledge soundness. Let KState_Π and Ext_Π be a knowledge state function and extractor for Π. We write $\hat{\mathbf{f}}_0$ and $\hat{\mathbf{f}}_1$ for the Boolean-cube restrictions of the ring oracles $[[\widetilde{\mathbf{f}}_0]]$ and $[[\widetilde{\mathbf{f}}_1]]$. We call a transcript *post-query* if it contains at least $i^\star$ verifier messages (i.e., $\rho_{i^\star}$ has been revealed), and *pre-query* otherwise. For a post-query transcript, $\mathbf{u}^\star \in \mathbb{F}_{\tilde{q}}^\nu$ denotes the query point determined by $\rho_{i^\star}$.

Knowledge state witness. Following Remark 3.10, we take $\mathsf{w} = (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1, \mathsf{w}_0, \mathsf{w}_1)$, where $(\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)$ with $\hat{\mathsf{w}}_0 \in (\mathbb{Q}^{<d_0,B}[X])^m$ and $\hat{\mathsf{w}}_1 \in (\mathbb{F}_q^{<d_1}[X])^m$ is a candidate $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ witness, and $(\mathsf{w}_0, \mathsf{w}_1) \in (\mathbb{F}_{\tilde{q}}^m)^2$ is an auxiliary slot carrying a Π knowledge state witness.

Extractor. $\mathsf{Ext}_{\Pi'}(\mathbb{i}, \mathbb{x}, \mathrm{tr} \| \rho, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1, \mathsf{w}_0, \mathsf{w}_1)) := (\hat{\mathsf{w}}_0,\ \hat{\mathsf{w}}_1,\ \mathsf{Ext}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr} \| \rho, (\mathsf{w}_0, \mathsf{w}_1)))$.

Knowledge state function. We define $\mathsf{KState}_{\Pi'}$ by three cases, determined by $\hat{\mathbf{f}}_0$ and the transcript.
Case 1. ($\hat{\mathbf{f}}_0 \in (\mathbb{Z}_{(p)}[X])^m$, any transcript):

$$\mathsf{KState}_{\Pi'}(\mathbb{i}, \mathbb{x}, \mathrm{tr}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1, \mathsf{w}_0, \mathsf{w}_1)) := [\hat{\mathsf{w}}_0 = \hat{\mathbf{f}}_0] \cdot [\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1] \cdot \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, (\mathsf{w}_0, \mathsf{w}_1)). \tag{105}$$

Case 2. ($\hat{\mathbf{f}}_0 \notin (\mathbb{Z}_{(p)}[X])^m$, and either pre-query or $\widetilde{\mathbf{f}}_0(\psi_0^{-1}(\mathbf{u}^\star)) \notin \mathbb{Z}_{(p)}[X]$):

$$\mathsf{KState}_{\Pi'}(\mathbb{i}, \mathbb{x}, \mathrm{tr}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1, \mathsf{w}_0, \mathsf{w}_1)) := 0. \tag{106}$$

Case 3 ($\hat{\mathbf{f}}_0 \notin (\mathbb{Z}_{(p)}[X])^m$, post-query, and $\widetilde{\mathbf{f}}_0(\psi_0^{-1}(\mathbf{u}^\star)) \in \mathbb{Z}_{(p)}[X]$):

$$\mathsf{KState}_{\Pi'}(\mathbb{i}, \mathbb{x}, \mathrm{tr}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1, \mathsf{w}_0, \mathsf{w}_1)) := [\hat{\mathsf{w}}_0 \in (\mathbb{Z}_{(p)}[X])^m] \cdot [\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1] \cdot \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, (\mathsf{w}_0, \mathsf{w}_1)).$$

Cases 1 and 3 share the form $G(\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1) \cdot \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, (\mathsf{w}_0, \mathsf{w}_1))$, each for a different gate G that depends only on $(\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)$, the oracles, and $\rho_{i^\star}$ (which fixes $\mathbf{u}^\star$). In particular, G is independent of prover messages and of ρ_i for $i \neq i^\star$, and $\mathsf{Ext}_{\Pi'}$ preserves $(\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)$, so G takes the same value on both sides of the RBR-KS event (107).

Empty transcript. The empty transcript is pre-query, so only Cases 1 and 2 apply, depending on whether $\hat{\mathbf{f}}_0 \in (\mathbb{Z}_{(p)}[X])^m$.
Case 2 ($\hat{\mathbf{f}}_0 \notin (\mathbb{Z}_{(p)}[X])^m$*).* Then $\mathsf{KState}_{\Pi'} = 0$ by (106) for every w, and $(\mathbb{i}, \mathbb{x}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)) \notin \mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ for every $(\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)$ since the relation forces $\hat{\mathsf{w}}_0 = \hat{\mathbf{f}}_0 \notin (\mathbb{Z}_{(p)}[X])^m$. Both sides of the biconditional are uniformly false.
Case $\hat{\mathbf{f}}_0 \in (\mathbb{Z}_{(p)}[X])^m$ *(Case 1 applies).* Here $\mathsf{KState}_{\Pi'} = [\hat{\mathsf{w}}_0 = \hat{\mathbf{f}}_0] \cdot [\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1] \cdot \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \emptyset, (\mathsf{w}_0, \mathsf{w}_1))$ by (105).

($\Rightarrow$). $(\mathbb{i}, \mathbb{x}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)) \in \mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ implies $\mathsf{KState}_{\Pi'} = 1$ for some $(\mathsf{w}_0, \mathsf{w}_1)$: the relation forces $\hat{\mathsf{w}}_0 = \hat{\mathbf{f}}_0$ and $\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1$, so the first two factors are 1. Setting $\mathsf{w}_0 := \psi_0(\hat{\mathsf{w}}_0)$ and $\mathsf{w}_1 := \psi_1(\hat{\mathsf{w}}_1)$ gives $(\bar{\mathbb{i}}, \bar{\mathbb{x}}, (\mathsf{w}_0, \mathsf{w}_1)) \in \mathsf{PESAT}$ by Lemma 5.4, so the third factor is 1 by Π's empty-transcript property.

($\Leftarrow$). $\mathsf{KState}_{\Pi'} = 1$ for some $(\mathsf{w}_0, \mathsf{w}_1)$ implies $(\mathbb{i}, \mathbb{x}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)) \in \mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$: the first two factors force $\hat{\mathsf{w}}_0 = \hat{\mathbf{f}}_0 \in (\mathbb{Z}_{(p)}[X])^m$ and $\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1$. The third factor and Π's empty-transcript property give $(\bar{\mathbb{i}}, \bar{\mathbb{x}}, (\mathsf{w}_0, \mathsf{w}_1)) \in \mathsf{PESAT}$. MLE uniqueness then gives $\mathsf{w}_0 = \psi_0(\hat{\mathsf{w}}_0)$ and $\mathsf{w}_1 = \psi_1(\hat{\mathsf{w}}_1)$, so constraint (ii) of $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ holds; constraint (i) holds since $\hat{\mathsf{w}}_0 \in (\mathbb{Z}_{(p)}[X])^m$. Hence $(\mathbb{i}, \mathbb{x}, (\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1)) \in \mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$.

Prover moves. Appending a prover message does not change which case applies (no new verifier message, $\hat{\mathbf{f}}_0$ unchanged). In Case 2, $\mathsf{KState}_{\Pi'} = 0$ at both transcripts. In Cases 1 and 3, G is unchanged and KState_{Π} satisfies the prover-move property.

Complete transcript. The complete transcript $(\pi_1, \rho_1, \ldots, \pi_k, \rho_k)$ is post-query, so all three cases of $\mathsf{KState}_{\Pi'}$ are in scope. V' accepts iff the projected oracle's response at $\mathbf{u}^\star$ is not $\perp$ (equivalently, $\widetilde{\mathbf{f}}_0(\psi_0^{-1}(\mathbf{u}^\star)) \in \mathbb{Z}_{(p)}[X]$) and V_Π accepts.
Case 2. Then $\mathsf{KState}_{\Pi'} = 0$ for every w by (106), and the projected oracle returns $\perp$ at $\mathbf{u}^\star$, so V' rejects. Both sides of the biconditional are uniformly false.
Cases 1 and 3. The projected oracle's response at $\mathbf{u}^\star$ is not $\perp$, so V' accepts iff V_Π accepts. In both cases $\mathsf{KState}_{\Pi'} = G(\hat{\mathsf{w}}_0, \hat{\mathsf{w}}_1) \cdot \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, (\mathsf{w}_0, \mathsf{w}_1))$ for the gate G specific to that case.

($\Rightarrow$). V' accepts implies $\mathsf{KState}_{\Pi'} = 1$ for some w: V_Π accepts, so by Π's full-transcript property there exist $(\mathsf{w}_0, \mathsf{w}_1)$ with $\mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, (\mathsf{w}_0, \mathsf{w}_1)) = 1$, making the second factor 1. Taking $\hat{\mathsf{w}}_0 = \hat{\mathbf{f}}_0$ and $\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1$ in Case 1, or any $\hat{\mathsf{w}}_0 \in (\mathbb{Z}_{(p)}[X])^m$ and $\hat{\mathsf{w}}_1 = \hat{\mathbf{f}}_1$ in Case 3, makes $G = 1$. Hence $\mathsf{KState}_{\Pi'} = 1$.

($\Leftarrow$). $\mathsf{KState}_{\Pi'} = 1$ for some w implies V' accepts: the second factor is 1, so $\mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, (\mathsf{w}_0, \mathsf{w}_1)) = 1$ and V_Π accepts by Π's full-transcript property. The projected oracle's response at $\mathbf{u}^\star$ is not $\perp$ (in Case 1 because $\hat{\mathbf{f}}_0 \in (\mathbb{Z}_{(p)}[X])^m$ globally, and in Case 3 by the case condition $\widetilde{\mathbf{f}}_0(\psi_0^{-1}(\mathbf{u}^\star)) \in \mathbb{Z}_{(p)}[X]$), so V' accepts.

RBR-KS errors. For round $i \in [k]$, let $\mathrm{tr} = (\pi_1, \rho_1, \ldots, \pi_i)$ be a transcript where the verifier is about to send ρ_i. We bound

$$\Pr_{\rho_i}[\exists\, \mathsf{w}\colon\ \mathsf{KState}_{\Pi'}(\mathrm{tr}, \mathsf{Ext}_{\Pi'}(\mathbb{i}, \mathbb{x}, \mathrm{tr}\|\rho_i, \mathsf{w})) = 0\ \wedge\ \mathsf{KState}_{\Pi'}(\mathrm{tr}\|\rho_i, \mathsf{w}) = 1]\,. \tag{107}$$

When both tr and $\mathrm{tr}\|\rho_i$ are in the same case (1 or 3), G is shared between the two sides, so the event reduces to

$$\exists\,(\mathsf{w}_0, \mathsf{w}_1)\colon\ \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}, \mathsf{Ext}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}\|\rho_i, (\mathsf{w}_0, \mathsf{w}_1))) = 0\ \wedge\ \mathsf{KState}_\Pi(\bar{\mathbb{i}}, \bar{\mathbb{x}}, \mathrm{tr}\|\rho_i, (\mathsf{w}_0, \mathsf{w}_1)) = 1,$$

which is Π's RBR-KS event: probability $\leq \kappa_i$. When both sides are in Case 2, the event is empty.

The only remaining possibility is a transition from Case 2 (tr, pre-query) to Case 3 ($\mathrm{tr}\|\rho_i$, post-query with non-$\perp$ response). This requires $\hat{\mathbf{f}}_0 \notin (\mathbb{Z}_{(p)}[X])^m$, $i = i^\star$, and $\widetilde{\mathbf{f}}_0(\psi_0^{-1}(\mathbf{u}^\star)) \in \mathbb{Z}_{(p)}[X]$ for uniformly random $\mathbf{u}^\star$: probability $\leq \nu/|\mathbb{F}_{\tilde{q}}|$ by Theorem 5.5.

For $i \neq i^\star$, ρ_i does not affect which case applies, so both sides are in the same case: bound κ_i. For $i = i^\star$: Case 1 gives $\kappa_{i^\star}$, the Case 2 $\to$ 3 transition gives $\nu/|\mathbb{F}_{\tilde{q}}|$. These are mutually exclusive (determined by $\hat{\mathbf{f}}_0$): $\kappa'_{i^\star} \leq \max(\kappa_{i^\star}, \nu/|\mathbb{F}_{\tilde{q}}|)$. □

A.1.3 Zinc+ completeness, round-by-round knowledge soundness, and efficiency

We begin by introducing some supplementary lemmas.

Q_{0t}-evaluation bitlength bound.

Lemma A.1 (Evaluation bitlength bound, random-prime branch)**.** *Under the* $\mathsf{REL}_{\mathsf{UCS}}$ *typing, for* $i = 0$ *and* $t \in [|\mathcal{C}|]$*, the rational coefficients of the per-row evaluated polynomial* $Q_{0t}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0)$ *have bit-size strictly less than*

$$B_{0t}^{\text{eval}} \;:=\; B + \deg_{\mathbf{Y},\mathbf{Z}_0}(Q_{0t}) \cdot (B - 1 + \lceil \log d_0 \rceil) + \lceil \log \|Q_{0t}\|_0 \rceil.$$

Proof. Fix a monomial $M \in \operatorname{supp}(Q_{0t})$ with explicit X-polynomial coefficient c_M, of bit-size $< B$ and X-degree $< d_0$ per $\mathsf{REL}_{\mathsf{UCS}}$ typing. Evaluating M at $(\mathbf{b}, \mathbf{y}, \mathbf{f}_0)$ multiplies c_M by $\deg_{\mathbf{Y},\mathbf{Z}_0}(M)$ substituted polynomial-valued entries from $\mathbf{y}$ and $\mathbf{f}_0$, each of bit-size $< B$ and X-degree $< d_0$. The Boolean entries b_i contribute no bit-size growth.

Each pairwise multiplication $f \cdot g$ of two polynomials in $\mathbb{Q}^{<d_0,B}[X]$ has output coefficients each a sum of at most d_0 products of $< B$-bit coefficients. Each summand has absolute value $< 2^{B-1} \cdot 2^{B-1} = 2^{2B-2}$; the sum of at most d_0 such summands has absolute value $< d_0 \cdot 2^{2B-2} \le 2^{2B-2+\lceil \log d_0 \rceil}$, hence bit-size $< 2B - 1 + \lceil \log d_0 \rceil$. Each multiplication thus grows bit-size by at most $B - 1 + \lceil \log d_0 \rceil$. Iterating $\deg_{\mathbf{Y},\mathbf{Z}_0}(M)$ multiplications starting from c_M (bit-size $< B$) yields per-coefficient bit-size strictly less than

$$B + \deg_{\mathbf{Y},\mathbf{Z}_0}(M) \cdot (B - 1 + \lceil \log d_0 \rceil).$$

Summing $M(\mathbf{b}, \mathbf{y}, \mathbf{f}_0)$ over $M \in \operatorname{supp}(Q_{0t})$ recovers $Q_{0t}(\mathbf{b}, \mathbf{y}, \mathbf{f}_0)$. Each output coefficient is a sum of at most $\|Q_{0t}\|_0$ summands, each strictly bounded above; hence the sum has bit-size strictly less than the per-monomial bound plus $\lceil \log \|Q_{0t}\|_0 \rceil$. This also covers any post-substitution merging of monomials whose K-exponents differ but whose $(\mathbf{Y}, \mathbf{Z}_0)$-projections coincide, since $\|Q_{0t}\|_0$ counts pre-substitution monomials. Replacing $\deg_{\mathbf{Y},\mathbf{Z}_0}(M)$ by its upper bound $\deg_{\mathbf{Y},\mathbf{Z}_0}(Q_{0t})$ gives the claim. □

ϕ-collision bound.

Lemma A.2. *Let* $f, \mathsf{j} \in \mathbb{Q}^{<d,<B}[X]$*,* $d \ge 1$*,* $B_{\mathcal{P}} \ge 2$*. Then*

$$\Pr_{p \leftarrow \mathcal{P}} \left[\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle \;\wedge\; f \notin \langle \mathsf{j} \rangle \;\middle|\; f, \mathsf{j} \in \mathbb{Z}_{(p)} \right] < \frac{8d^2 B}{|\mathcal{P}|(B_{\mathcal{P}} - 1)}.$$

Proof. Let $(q, r) \in \mathbb{Q}[X] \times \mathbb{Q}[X]$ be the unique quotient–remainder pair given by Euclidean division in $\mathbb{Q}[X]$:

$$f = \mathsf{j}q + r, \qquad \deg(r) < \deg(\mathsf{j}).$$

Then $f \in \langle \mathsf{j} \rangle$ if and only if $r = 0$.

Write r in lowest terms as $r = n_r/d_r$, where $n_r \in \mathbb{Z}[X]$ and $d_r \in \mathbb{Z}^+$ satisfy $\gcd(d_r, c) = 1$, where $c = \mathsf{cont}(n_r)$ is the content of n_r (the greatest common divisor of its coefficients). Similarly write a, the leading coefficient of j, in lowest terms $a = n_a/d_a$. Define $N := c \cdot n_a$. We show that for every prime p,

$$\big(\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle \;\wedge\; f \notin \langle \mathsf{j} \rangle \;\wedge\; f, \mathsf{j} \in \mathbb{Z}_{(p)}\big) \Longrightarrow p \mid N.$$

Fix p and assume $\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle$, $f \notin \langle \mathsf{j} \rangle$, and $f, \mathsf{j} \in \mathbb{Z}_{(p)}$. Define $N := c \cdot n_a$. We show $p \mid N$. First assume $p \nmid n_a$. Then the leading coefficient of j is a unit in $\mathbb{Z}_{(p)}$, so Euclidean division of f by j can be performed in $\mathbb{Z}_{(p)}[X]$. Let $(q_p, r_p) \in \mathbb{Z}_{(p)}[X] \times \mathbb{Z}_{(p)}[X]$ be the resulting quotient–remainder pair, so

$$f = \mathsf{j}q_p + r_p, \qquad \deg(r_p) < \deg(\mathsf{j}).$$

By uniqueness of Euclidean division in $\mathbb{Q}[X]$, we have $r_p = r$, and in particular $r \in \mathbb{Z}_{(p)}[X]$. Applying ϕ_p gives

$$\phi_p(f) = \phi_p(\mathsf{j}) \cdot \phi_p(q_p) + \phi_p(r).$$

Since $p \nmid n_a$, we have $\deg(\phi_p(\mathsf{j})) = \deg(\mathsf{j})$, hence $\deg(\phi_p(r)) < \deg(\phi_p(\mathsf{j}))$. Together with $\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle$, this implies $\phi_p(r) = 0$. Because $r \in \mathbb{Z}_{(p)}[X]$, we have $\phi_p(r) = 0$ if and only if $p \mid \mathsf{cont}(n_r) = c$. Hence $p \mid c$, and therefore $p \mid N$.

Next assume $p \nmid c$. If also $p \nmid n_a$, then the same argument shows $r \in \mathbb{Z}_{(p)}[X]$ and $\phi_p(r) = 0 \iff p \mid c$, so $p \nmid c$ implies $\phi_p(r) \neq 0$ while $\deg(\phi_p(r)) < \deg(\phi_p(\mathsf{j}))$. A nonzero polynomial of degree strictly less than $\deg(\phi_p(\mathsf{j}))$ cannot lie in $\langle \phi_p(\mathsf{j}) \rangle$, contradicting $\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle$. Therefore $p \mid n_a$, and hence $p \mid N$.

Let $\mathcal{P}_{\text{bad}} = \{p \in \mathcal{P} : p \mid N\}$. The above implication yields

$$\Pr_{p \leftarrow \mathcal{P}} \left[\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle \ \wedge \ f \notin \langle \mathsf{j} \rangle \mid f, \mathsf{j} \in \mathbb{Z}_{(p)} \right] \leq \frac{|\mathcal{P}_{\text{bad}}|}{|\mathcal{P}|}.$$

Let $B_N = \lceil \log(|N| + 1) \rceil$. If $B_N < B_{\mathcal{P}}$ then $|\mathcal{P}_{\text{bad}}| = 0$. Otherwise, by Lemma 3.2,

$$|\mathcal{P}_{\text{bad}}| < \frac{B_N - 1}{B_{\mathcal{P}} - 1},$$

so

$$\Pr_{p \leftarrow \mathcal{P}} \left[\phi_p(f) \in \langle \phi_p(\mathsf{j}) \rangle \ \wedge \ f \notin \langle \mathsf{j} \rangle \mid f, \mathsf{j} \in \mathbb{Z}_{(p)} \right] < \frac{B_N - 1}{|\mathcal{P}|(B_{\mathcal{P}} - 1)}. \tag{108}$$

We now bound B_N. Since the coefficients of f and j have bitlength $< B$, we may write each coefficient in lowest terms with numerator and denominator of magnitude $< 2^B$. Let $\Delta \in \mathbb{Z}^+$ be the product of all coefficient denominators appearing in f and j. Then $\Delta < 2^{2dB}$, and the polynomials

$$F := \Delta f \in \mathbb{Z}[X], \qquad J := \Delta \mathsf{j} \in \mathbb{Z}[X]$$

satisfy $\|F\|_\infty, \|J\|_\infty < 2^B \Delta < 2^{(2d+1)B}$.

Let $m := \deg(F)$ and $n := \deg(J)$, so $m, n < d$. Let b be the leading coefficient of J. By pseudo-division, there exist $Q', R' \in \mathbb{Z}[X]$ with $\deg(R') < n$ such that

$$b^{m-n+1} F = JQ' + R'. \tag{109}$$

Dividing (109) by Δ and comparing with $f = \mathsf{j}q + r$ shows that

$$r = \frac{R'}{b^{m-n+1}\Delta}.$$

In particular, there exists $\alpha \in \mathbb{Z}^+$ such that $R' = \alpha n_r \in \mathbb{Z}[X]$ and $b^{m-n+1}\Delta = \alpha d_r \in \mathbb{Z} \setminus \{0\}$, hence

$$c = \mathsf{cont}(n_r) \leq \|n_r\|_\infty \leq \|R'\|_\infty.$$

Each coefficient of R' can be expressed as a determinant of a matrix of dimension at most $m+n < 2d$ whose entries are coefficients of F and J. By Hadamard's inequality,

$$\|R'\|_\infty \leq (2d)^d \cdot \left(\max(\|F\|_\infty, \|J\|_\infty)\right)^{2d} < (2d)^d \cdot \left(2^{(2d+1)B}\right)^{2d} = (2d)^d \cdot 2^{(4d^2+2d)B}.$$

Therefore $c < (2d)^d \cdot 2^{(4d^2+2d)B}$. Since $|n_a| < 2^B$, we have

$$|N| < (2d)^d \cdot 2^{(4d^2+2d+1)B}.$$

It follows that

$$B_N = \lceil \log(|N|+1) \rceil \le (4d^2+2d+1)B + d\log(2d) + 1 < 8d^2 B.$$

Substituting this bound into Eq. (108) completes the proof. □

Zinc+ PIOR round-by-round knowledge soundness (Theorem 5.13) .

Proof. We define an extractor and a state function, verify that the state function satisfies Definition 3.11, and then bound the two verifier-message transitions.

Write $\mathbb{i} = (m = 2^\nu, \ell, n, \mathbf{q}, B, d, \mathcal{C} = [(Q_{it}, \langle j_{it} \rangle)]_{i \in [0,|\mathbf{q}|],\, t \in [|\mathcal{C}|]})$ and $\mathbb{x} = (\mathbf{y})$.

Extractor. For any pre-challenge transcript tr, the extractor $\mathcal{E}_{\mathsf{rbr}}(\mathsf{tr})$ reads the oracle messages $[[\widetilde{\mathbf{f}}_0]], \ldots, [[\widetilde{\mathbf{f}}_{|\mathbf{q}|}]]$ already present in tr, queries each oracle at every point of the ν-dimensional Boolean hypercube, and outputs $\mathbb{w} := (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|})$. Thus the extracted witness is determined by the pre-challenge transcript and is independent of the verifier message about to be sampled.

State function. For transcript shapes not listed below, set $\mathsf{State} = 0$. For listed transcript shapes, define State as follows.

Transcript shape	$\mathsf{State} = 1$ iff
$\emptyset$	never
$[[\widetilde{\mathbf{f}}_0]], \ldots, [[\widetilde{\mathbf{f}}_{\|\mathbf{q}\|}]]$	never
above $+\, q_0, r_0, \ldots, r_{\|\mathbf{q}\|}$	$Q_{0t}, j_{0t}, \mathbf{y} \in \mathbb{Z}_{(q_0)}[X]$ for all t, $\mathbf{f}_0 \in (\mathbb{Z}_{(q_0)}[X])^m$, and (110) holds for all (i,t)
above $+\, e_{0t}, e_{it}$	same as previous
above $+\, a_0, \ldots, a_{\|\mathbf{q}\|}$	$Q_{0t}, j_{0t}, \mathbf{y} \in \mathbb{Z}_{(q_0)}[X]$ for all t, $\mathbf{f}_0 \in (\mathbb{Z}_{(q_0)}[X])^m$, $a_i \in \mathrm{Prim}(\mathbb{F}_{\widetilde{q}_i}/\mathbb{F}_{p_i})$ for all $i \in [\|\mathbf{q}\|]$, and (111) holds for all (i,t).

Here (110) denotes the random-row ideal checks

$$\begin{aligned} \sum_{b \in \{0,1\}^{\log n}} \psi_{q_0}(Q_{0t}(b, \mathbf{y}, \mathbf{f}_0)) \cdot \widetilde{\mathsf{eq}}(b, r_0) &\in \langle \psi_{q_0}(j_{0t}) \rangle \subseteq \mathbb{F}_{q_0}[X], \\ \sum_{b \in \{0,1\}^{\log n}} \psi_i(Q_{it})(b, \psi_i(\mathbf{y}), \psi_i(\mathbf{f}_0), \psi_i(\mathbf{f}_i)) \cdot \widetilde{\mathsf{eq}}(b, r_i) &\in \langle \psi_i(j_{it}) \rangle \subseteq \mathbb{F}_{\widetilde{q}_i}[X], \end{aligned} \tag{110}$$

for $i \in [|\mathbf{q}|]$. The final evaluation checks (111) are

$$\begin{aligned} \sum_{b \in \{0,1\}^{\log n}} \psi_{q_0}(Q_{0t})(b, \psi_{q_0}(\mathbf{y}), \psi_{q_0}(\mathbf{f}_0)) \cdot \widetilde{\mathsf{eq}}(b, r_0) - e_{0t}(a_0) &= 0 \quad \text{over } \mathbb{F}_{q_0}, \\ \sum_{b \in \{0,1\}^{\log n}} \psi_i(Q_{it})(b, \psi_i(\mathbf{y}), \psi_i(\mathbf{f}_0), \psi_i(\mathbf{f}_i)) \cdot \widetilde{\mathsf{eq}}(b, r_i) - e_{it}(a_i) &= 0 \quad \text{over } \mathbb{F}_{\widetilde{q}_i}, \end{aligned} \tag{111}$$

again for $i \in [|\mathbf{q}|]$.

State-function conditions. The empty transcript and the post-first-prover-move transcript both have state 0 by definition; the second prover message (e_{0t}, e_{it}) leaves the state unchanged. Hence prover moves preserve state 0. On a full transcript, Algorithm 1 outputs the per-branch instances of $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\widetilde{q}_i})$ (Definition 5.8). The $\mathbb{Z}_{(q_0)}$-membership and primitivity entries in row 4 are exactly Algorithm 1's well-definedness and extension-generator rejection conditions. Once these checks pass, the $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ constraints of the output instances are precisely the final projected equalities (111), together with the required typing condition on $\mathbf{f}_0$. Hence row 4 holds iff the verifier outputs an instance in $\mathsf{LANG}(\mathcal{R}')$, and State satisfies Definition 3.11.

Round 1: sampling $q_0, r_0, \ldots, r_{|\mathbf{q}|}$. Suppose the first verifier message causes a $0 \to 1$ transition and the extracted witness $\mathbb{w} = (\mathbf{f}_0, \ldots, \mathbf{f}_{|\mathbf{q}|})$ does not satisfy $\mathsf{REL}_{\mathsf{UCS}}(\mathbb{i}, \mathbb{x})$. Since the post-round state is 1, the well-definedness checks hold and (110) holds for every (i, t). Since $\mathbb{w} \notin \mathsf{REL}_{\mathsf{UCS}}(\mathbb{i}, \mathbb{x})$, there is a branch, constraint index, and row $b \in \{0,1\}^{\log n}$ such that the corresponding original-ring ideal membership fails. Let $(i^\star, t^\star, b^\star)$ be the lexicographically first such triple. At this fixed triple, the transition can occur only through one of two events.

First, if $i^\star = 0$, there can be a false projection: $Q_{0t^\star}(b^\star, \mathbf{y}, \mathbf{f}_0) \notin \langle j_{0t^\star} \rangle$ over $\mathbb{Q}[X]$, but $\phi_{q_0}(Q_{0t^\star}(b^\star, \mathbf{y}, \mathbf{f}_0)) \in \langle \phi_{q_0}(j_{0t^\star}) \rangle$ over $\mathbb{F}_{q_0}[X]$. By Eqs. (37) and (38) and Lemma A.1, the polynomial $Q_{0t^\star}(b^\star, \mathbf{y}, \mathbf{f}_0)$ has X-degree $< \delta_{0t^\star}$ and coefficient bit-size $< B^{\mathsf{Eval}}_{0t^\star}$. Applying Lemma A.2 to this fixed failed ideal membership and using the prime-set hypotheses,

$$\Pr_{q_0 \leftarrow \mathcal{P}}[\phi_{q_0}(Q_{0t^\star}(b^\star, \mathbf{y}, \mathbf{f}_0)) \in \langle \phi_{q_0}(j_{0t^\star}) \rangle] \;<\; \frac{8\delta^2_{0t^\star} B^{\mathsf{Eval}}_{0t^\star}}{|\mathcal{P}|(B_{\mathcal{P}} - 1)} \;\le\; \frac{8}{\lambda \cdot 2^\lambda}.$$

For $i^\star \in [|\mathbf{q}|]$, no analogous false projection is possible: the failed ideal membership has a nonzero Euclidean remainder over $\mathbb{F}_{q_{i^\star}}[X]$, and the coefficient-wise embedding $\psi_{i^\star}$ is injective, so the projected remainder is still nonzero and the projected row residue remains outside the projected ideal. Lemma A.2 is needed only at branch 0.

Second, conditioned on the complementary event that no false projection occurred, the multilinear batching over $r_{i^\star}$ may vanish. If $j_{i^\star t^\star}$ is a unit, no row-wise violation exists, contradicting our choice of triple; if it is zero, the constraint is equality and the residue analysis below applies in the ambient polynomial ring. Otherwise work in the quotient by the projected ideal: the residue vector indexed by $b \in \{0,1\}^{\log n}$ is nonzero at $b^\star$. By Lemma 3.1 and the field-size hypothesis $|\mathbb{F}_{\widetilde{q}_{i^\star}}| \ge 2^\lambda \cdot 2 \log n$,

$$\Pr_{r_{i^\star}}\left[\sum_{b \in \{0,1\}^{\log n}} \mathsf{eq}(b, r_{i^\star}) \cdot [Q_{i^\star t^\star}(b, \mathbf{y}, \mathbf{f}_0, \mathbf{f}_{i^\star})] = 0 \;\middle|\; \text{no false projection occurred}\right] \le \frac{\log n}{|\mathbb{F}_{\widetilde{q}_{i^\star}}|} \le 2^{-\lambda-1}.$$

where the bracketed term denotes the residue class modulo the projected ideal when $j_{i^\star t^\star} \neq 0$, and denotes the projected polynomial itself when $j_{i^\star t^\star} = 0$. Thus, since $\lambda \ge 80$,

$$\Pr[\text{round 1 } 0 \to 1 \text{ transition with extractor failure}] \;<\; \frac{8}{\lambda \cdot 2^\lambda} + 2^{-\lambda-1} \;=\; \left(\frac{8}{\lambda} + \frac{1}{2}\right) 2^{-\lambda} \;\le\; 2^{-\lambda}.$$

Round 2: sampling $a_0, \ldots, a_{|\mathbf{q}|}$. Suppose the second verifier message causes a $0 \to 1$ transition and the extracted witness does not satisfy $\mathsf{REL}_{\mathsf{UCS}}(\mathbb{i}, \mathbb{x})$. Since row 4 repeats the well-definedness conditions from row 3, these conditions hold both before and after the challenge. The only row-3 condition not repeated in row 4 is (110); hence, because the pre-round state is 0 and the post-round state is 1, there is some pair $(i^\star, t^\star)$ for which (110) fails before sampling $\mathbf{a}$, while (111) holds after sampling $\mathbf{a}$. Choose the lexicographically first such pair.

For this pair, the remainder modulo the projected ideal of the difference between the batched projected constraint polynomial and $e_{i^\star t^\star}$ is a nonzero polynomial in X of degree $< \delta_{i^\star t^\star}$: if it were zero, then (110) would hold. Schwartz–Zippel applied to this remainder polynomial evaluated at $a_{i^\star}$, sampled over the full field, together with the field-size hypothesis $|\mathbb{F}_{\widetilde{q}_{i^\star}}| \geq 2^\lambda \delta_{i^\star t^\star}$, gives

$$\Pr[(111) \text{ holds at } (i^\star, t^\star)] \;\leq\; \frac{\delta_{i^\star t^\star} - 1}{|\mathbb{F}_{\widetilde{q}_{i^\star}}|} \;<\; \frac{\delta_{i^\star t^\star}}{|\mathbb{F}_{\widetilde{q}_{i^\star}}|} \;\leq\; 2^{-\lambda}.$$

Sampling over the full field only increases this probability: non-primitive samples are rejected whenever primitivity is required, and therefore cannot contribute to a $0 \to 1$ transition. Thus the second verifier round has transition probability at most $2^{-\lambda}$. □

Zinc+ PIOR completeness (Corollary 5.14).

Proof. We show that unless the verifier rejects, i.e. outputs the unsatisfiable instance-index ($\perp_{\mathsf{UCS}}$, $\perp_{\mathsf{UCS}}$) (failing the public-quantities check, ideal-membership check, or primitivity check), the output satisfies $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}'$. Then we bound the probability $\mathsf{V}_{\mathsf{UCS}}$ outputs $(\perp_{\mathsf{UCS}}, \perp_{\mathsf{UCS}})$. We begin with $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathsf{REL}_{\mathsf{UCS}}$, where

$$\mathbb{i} = (m, \ell, n = 2^\mu, \mathbf{q}, B, \mathbf{d}, \mathcal{C} = [(Q_{it}, \langle j_{it} \rangle)]_{i \in [0, |\mathbf{q}|], t \in [|\mathcal{C}|]}), \;\; \mathbb{x} = (\mathbf{y}) \in (\mathbb{Z}_{(\mathbf{q})}^{<d_0, B}[X])^\ell$$
$$\mathbb{w} = (\mathbf{f}_0, \mathbf{f}_1, \ldots, \mathbf{f}_{|\mathbf{q}|}), \;\; \mathbf{f}_0 \in \mathbb{Q}^{<d_0, B}[X]^m, \;\; \mathbf{f}_i \in \mathbb{F}_{q_i}^{<d_i}[X]^m \text{ for } i \in [|\mathbf{q}|].$$

In step 1, the prover sends the oracles for $\widetilde{\mathbf{f}}_0 \in (\mathbb{Q}^{<d_0, B}[X])[\mathbf{W}]$ and $\widetilde{\mathbf{f}}_1, \ldots, \widetilde{\mathbf{f}}_{|\mathbf{q}|} \in (\mathbb{F}_{q_i}^{<d_i}[X])[\mathbf{W}]$, which are the MLEs of the corresponding witness vectors. Suppose step 2 passes (i.e., Q_{0t}, j_{0t}, $\mathbf{y} \in \mathbb{Z}_{(q_0)}[X]$ for all t) and that $\mathbf{f}_0 \in (\mathbb{Z}_{(q_0)}[X])^m$. In step 3, the prover sends

$$\begin{aligned} e_{0t} &= \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \phi_{q_0}(Q_{0t})(\mathbf{b}, \phi_{q_0}(\mathbf{y}), \phi_{q_0}(\mathbf{f}_0)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_0) && \in \mathbb{F}_{q_0}[X] \\ e_{it} &= \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \iota_{\tilde{q}_i}(Q_{it})(\mathbf{b}, \phi_{\tilde{q}_i}(\mathbf{y}), \phi_{\tilde{q}_i}(\mathbf{f}_0), \iota_{\tilde{q}_i}(\mathbf{f}_i)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i) && \in \mathbb{F}_{\tilde{q}_i}[X] \end{aligned}$$

for $i \in [|\mathbf{q}|], t \in [|\mathcal{C}|]$. By the X-degree bound in Section 5.1, the honest prover's e_{it} has X-degree at most $\delta_{it} - 1$, so the type restriction at Step 3 is automatically satisfied. Since $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathsf{REL}_{\mathsf{UCS}}$, the ideal membership conditions hold for each row, so by Lemma 5.1 the bundled error terms satisfy $e_{0t} \in \langle \phi_{q_0}(\mathsf{j}_{0t}) \rangle$ and $e_{it} \in \langle \iota_{\tilde{q}_i}(\mathsf{j}_{it}) \rangle$; thus step 4 passes. The verifier defines

$$\begin{aligned} Q'_{0t} &= \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \psi_{q_0, a_0}(Q_{0t})(\mathbf{b}, \mathbf{Y}, \mathbf{Z}_0) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_0) - e_{0t}(a_0) \in \mathbb{F}_{q_0}[\mathbf{Y}, \mathbf{Z}_0] \\ Q'_{it} &= \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \psi_{q_i, a_i}(Q_{it})(\mathbf{b}, \mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i) - e_{it}(a_i) \in \mathbb{F}_{\tilde{q}_i}[\mathbf{Y}, \mathbf{Z}_0, \mathbf{Z}_1] \end{aligned}$$

Then plugging in $\mathbf{y}'_i = \psi_{q_i, a_i}(\mathbf{y})$, $\mathbf{f}'_{i0} = \psi_{q_i, a_i}(\mathbf{f}_0)$, $\mathbf{f}'_{i1} = \psi_{q_i, a_i}(\mathbf{f}_i)$ we obtain

$$\begin{aligned} Q'_{0t} = \;& \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} (\psi_{q_0, a_0}(Q_{0t})(\mathbf{b}, \psi_{q_0, a_0}(\mathbf{y}), \psi_{q_0, a_0}(\mathbf{f}_0)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_0)) \\ & - \Big(\textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \phi_{q_0}(Q_{0t})(\mathbf{b}, \phi_{q_0}(\mathbf{y}), \phi_{q_0}(\mathbf{f}_0)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_0) \Big)(a_0) \\ = \;& \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_0) \cdot \big[\psi_{q_0, a_0}(Q_{0t})(\mathbf{b}, \psi_{q_0, a_0}(\mathbf{y}), \psi_{q_0, a_0}(\mathbf{f}_0)) \\ & \qquad - \phi_{q_0}(Q_{0t})(\mathbf{b}, \phi_{q_0}(\mathbf{y}), \phi_{q_0}(\mathbf{f}_0))(a_0) \big] = 0, \end{aligned}$$

and similarly

$$\begin{aligned} Q'_{it} = \;& \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} (\psi_{q_i, a_i}(Q_{it})(\mathbf{b}, \psi_{q_i, a_i}(\mathbf{y}), \psi_{q_i, a_i}(\mathbf{f}_0), \psi_{q_i, a_i}(\mathbf{f}_i)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i)) \\ & - \Big(\textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \iota_{\tilde{q}_i}(Q_{it})(\mathbf{b}, \phi_{\tilde{q}_i}(\mathbf{y}), \phi_{\tilde{q}_i}(\mathbf{f}_0), \iota_{\tilde{q}_i}(\mathbf{f}_i)) \cdot \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i) \Big)(a_i) \\ = \;& \textstyle\sum_{\mathbf{b} \in \{0,1\}^\mu} \widetilde{\mathsf{eq}}(\mathbf{b}, \mathbf{r}_i) \cdot \big[\psi_{q_i, a_i}(Q_{it})(\mathbf{b}, \psi_{q_i, a_i}(\mathbf{y}), \psi_{q_i, a_i}(\mathbf{f}_0), \psi_{q_i, a_i}(\mathbf{f}_i)) \\ & \qquad - \iota_{\tilde{q}_i}(Q_{it})(\mathbf{b}, \phi_{\tilde{q}_i}(\mathbf{y}), \phi_{\tilde{q}_i}(\mathbf{f}_0), \iota_{\tilde{q}_i}(\mathbf{f}_i))(a_i) \big] = 0. \end{aligned}$$

This shows constraint (ii) of $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}(\mathbb{F}_{\tilde{q}_i})$ holds for each branch. For constraint (i): since $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathsf{REL}_{\mathsf{UCS}}$, the well-definedness condition gives $\mathbf{f}_0 \in (\mathbb{Z}_{(q_1 \cdots q_{|\mathbf{q}|})}[X])^m \subseteq (\mathbb{Z}_{(p_i)}[X])^m$ for each i, where p_i is the characteristic of q_i. Assuming the verifier did not reject the primitivity check (i.e., $a_i \in \mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i})$ for every $i \in [|\mathbf{q}|]$ with $\kappa_i > 1$), the honest prover's oracles $[\![\widetilde{\mathbf{f}_0}]\!]$ and $[\![\widetilde{\mathbf{f}_i}]\!]$ are the MLEs of $\mathbf{f}_0$ and $\mathbf{f}_i$, as required by $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ (Definition 5.8). Hence $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}'$.

It remains to bound the probability of incompleteness events. The honest prover succeeds unless either (a) q_0 divides a denominator of some rational coefficient of $\mathbf{y}$, $\mathbf{f}_0$, j_{0t}, or Q_{0t}, or (b) $a_i \notin \mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i})$ for some $i \in [|\mathbf{q}|]$ with $\kappa_i > 1$ (causing the verifier to output $(\perp_{\mathsf{UCS}}, \perp_{\mathsf{UCS}})$ via the primitivity check). Case (a) causes the verifier to output $(\perp_{\mathsf{UCS}}, \perp_{\mathsf{UCS}})$ via the public-quantities check when the affected coefficient is among $\mathbf{y}$, j_{0t}, Q_{0t}, and via the ideal-membership check when it is in $\mathbf{f}_0$, since the honest prover cannot then compute $e_{0t} \in \langle \phi_{q_0}(\mathsf{j}_{0t}) \rangle$.

Bound on (a). The total number of rational coefficients we apply ϕ_{q_0} to is

$$N_{\mathrm{rat}} := d_0 \left(\ell + m + |\mathcal{C}| + \sum_{t \in [|\mathcal{C}|]} \|Q_{0t}\|_0 \right).$$

Each such coefficient has numerator and denominator absolute value $< 2^{B-1}$. Let $B_{\mathcal{P}}$ denote the minimum bit-length over $\mathcal{P}$, so $q_0 \geq 2^{B_{\mathcal{P}}-1}$ for all $q_0 \in \mathcal{P}$; the $i = 0$ case of the second hypothesis of Theorem 5.13 gives $B_{\mathcal{P}} \geq \lambda + 1$. By Lemma 3.2 and the union bound over the N_{rat} coefficients,

$$\Pr[\text{(a) fails}] \;<\; \frac{N_{\mathrm{rat}}(B-1)}{|\mathcal{P}|(B_{\mathcal{P}}-1)} \;\leq\; \frac{N_{\mathrm{rat}} \cdot B}{2^{\lambda} \cdot \lambda \cdot \max_t(\delta_{0t}^2 B_{0t}^{\mathrm{eval}})},$$

matching the corollary's first term.

Bound on (b). For $i \in [|\mathbf{q}|]$ with $\kappa_i > 1$ (where $\kappa_i = e_i \ell_i = [\mathbb{F}_{\tilde{q}_i} : \mathbb{F}_{p_i}]$, so $\kappa_i > 1$ iff $\tilde{q}_i \neq p_i$), the honest prover fails the primitive-element check only if $a_i \notin \mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i})$. Writing $\tilde{q}_i = p_i^{\kappa_i}$, this event occurs only when a_i lies in a proper intermediate subfield $\mathbb{F}_{p_i^d}$ with $d \mid \kappa_i$ and $d < \kappa_i$. By union bound over divisors,

$$\Pr[a_i \notin \mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i})] \;\leq\; \sum_{\substack{d \mid \kappa_i \\ d < \kappa_i}} p_i^{d-\kappa_i}.$$

(Möbius inclusion-exclusion gives the exact count $|\mathsf{Prim}(\mathbb{F}_{\tilde{q}_i}/\mathbb{F}_{p_i})| = \sum_{d|\kappa_i} \mu(\kappa_i/d) \cdot p_i^d$, yielding $\Pr[a_i \notin \mathsf{Prim}] = 1 - p_i^{-\kappa_i} \sum_{d|\kappa_i} \mu(\kappa_i/d) \cdot p_i^d$; we use the simpler union-bound form because it is tight up to negligible lower-order terms in the standard case where κ_i is a prime-power.)

When $\kappa_i = 1$, the extension is trivial and Step 5 imposes no primitivity check, so no completeness failure arises from this branch. For $i = 0$, no primitivity condition is needed for honest completeness, since $\tilde{q}_0 = q_0$ is the trivial extension and the honest polynomial identity evaluates correctly at every $a_0 \in \mathbb{F}_{q_0}$. Summing over $i \in [|\mathbf{q}|]$ with $\kappa_i > 1$ gives the second term of Corollary 5.14. □

A.1.4 Compiled Zinc+ IOP

Proof of Theorem 5.17. The protocol Π_Σ is the protocol from the theorem statement: it runs Algorithm 1 followed by the lifted protocols $\Pi'_0, \Pi'_1, \ldots, \Pi'_{|\mathbf{q}|}$ in parallel.

PIOR prefix. By Theorem 5.13, the protocol of Algorithm 1 has state-function-form RBR-KS with errors at most $2^{-\lambda}$ per round. Per [BCFW25, Theorem C.1], it thus has knowledge-state-form RBR-KS with the same per-round errors. Its output relation is $\mathcal{R}'$ (Definition 5.12).

Lifted CP-PIOPs. For each $i \in [0, |\mathbf{q}|]$, Theorem 5.11 gives Π'_i per-round error $\max(\kappa_{\Pi_i}, \nu/(|\mathbb{F}_{\tilde{q}_i}|))$ at the special query round and κ_{Π_i} otherwise. By Item 1, $\kappa_{\Pi_i} \le 2^{-\lambda'}$. By the field-size hypothesis, $\nu/(|\mathbb{F}_{\tilde{q}_i}|) \le 2^{-\lambda'}$. Hence every per-round error of Π'_i is at most $2^{-\lambda'}$. Since $2^{-\lambda'} \le 2^{-\lambda}/(|\mathbf{q}|+1)$, every per-round error of every lifted branch protocol is at most $2^{-\lambda}/(|\mathbf{q}|+1)$. Since the underlying Π_i is CP and Construction 5.9 simulates verifier queries to the projected oracle without introducing new oracle messages, Π'_i is also CP over Algorithm 1's oracle handles.

Parallel branch phase. After the protocol of Algorithm 1, the lifted protocols $\Pi'_0, \ldots, \Pi'_{|\mathbf{q}|}$ run in parallel. In global branch-phase round j, the verifier samples one uniform bitstring ρ_j. Every active branch receives ρ_j and applies its own deterministic verifier map to interpret ρ_j as its local round-j message. Branches whose local protocols have already terminated are idle. Thus the branch phase has $\max_{i \in [0,|\mathbf{q}|]} k_i$ rounds.

We use the auxiliary-slot convention of Remark 3.10. A branch-phase knowledge state witness has the form $\mathsf{w} = (\mathbb{w}', \zeta_0, \ldots, \zeta_{|\mathbf{q}|})$, where $\mathbb{w}'$ has the structured $\mathcal{R}'$-witness type and each ζ_i is an auxiliary knowledge state witness for the lifted protocol Π'_i. The auxiliary slots are not part of the $\mathcal{R}'$ witness; they are used only to evaluate the branch-local knowledge state functions and their extractors, exactly as in Theorem 5.11.

For each branch i, let $(\mathbb{i}_i, \bar{\mathbb{x}}_i; \bar{\mathbb{w}}_i)$ be the induced branch triple determined by $(\mathbb{i}', \mathbb{x}'; \mathbb{w}')$ in Definition 5.12. By Theorem 5.11, the lifted protocol Π'_i has a knowledge state function and extractor in which the branch-local knowledge state witness is $(\bar{\mathbb{w}}_i, \zeta_i)$, where $\bar{\mathbb{w}}_i$ is the candidate $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ witness for the induced branch triple and ζ_i is the auxiliary inner knowledge state witness. In the proof of Theorem 5.11, the extractor preserves the candidate $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ witness and updates only the auxiliary inner component. Write $\mathsf{KState}_i(\mathsf{tr}_i; \bar{\mathbb{w}}_i, \zeta_i)$ for this branch-local knowledge state function.

Define the branch-product predicate

$$\mathsf{KState}_{\mathrm{br}}(\mathsf{tr}, \mathsf{w}) := \bigwedge_{i \in [0,|\mathbf{q}|]} \mathsf{KState}_i(\mathsf{tr}_i; \bar{\mathbb{w}}_i, \zeta_i),$$

where tr_i is the branch-local transcript: it is empty before branch i starts, partial while branch i is running, and full after branch i has terminated.

The knowledge state function for the parallel branch phase is

$$\mathsf{KState}_{\|}(\mathsf{tr}, \mathsf{w}) := \begin{cases} 1 & \text{if } \mathsf{tr} = \emptyset \text{ and } (\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}', \\ 0 & \text{if } \mathsf{tr} = \emptyset \text{ and } (\mathbb{i}', \mathbb{x}'; \mathbb{w}') \notin \mathcal{R}', \\ 1 & \text{if } \mathsf{tr} \text{ is partial and } (\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}', \\ \mathsf{KState}_{\mathrm{br}}(\mathsf{tr}, \mathsf{w}) & \text{if } \mathsf{tr} \text{ is partial and } (\mathbb{i}', \mathbb{x}'; \mathbb{w}') \notin \mathcal{R}', \\ 1 & \text{if } \mathsf{tr} \text{ is full and all lifted branch protocols accept}, \\ 0 & \text{if } \mathsf{tr} \text{ is full and some lifted branch protocol rejects}. \end{cases}$$

Equivalently, at partial transcripts, $\mathsf{KState}_{\|}$ is the disjunction of $\mathcal{R}'$ source membership for $\mathbb{w}'$ and the branch-product predicate.

The branch-phase extractor is defined similarly. On input $(\mathsf{tr}\|\rho, \mathsf{w})$, if $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}'$, it returns w. Otherwise, it keeps $\mathbb{w}'$ fixed and applies the branch extractors componentwise to the active branch-local auxiliary data: $E_{\|}(\mathsf{tr}\|\rho, \mathsf{w}) = \big(\mathbb{w}', \zeta'_0, \ldots, \zeta'_{|\mathbf{q}|}\big)$.

where ζ'_i is the auxiliary component returned by the extractor for Π'_i on the local input $(\mathsf{tr}_i\|\rho_i; \bar{\mathbb{w}}_i, \zeta_i)$ for active branches, and $\zeta'_i = \zeta_i$ for idle branches. The induced branch source witnesses $\bar{\mathbb{w}}_i$ remain fixed because they are determined by the unchanged shared witness $\mathbb{w}'$.

The empty-transcript clause of Definition 3.8, with the auxiliary-slot convention of Remark 3.10, holds by the first two cases in the definition of $\mathsf{KState}_{\|}$. For the prover-move clause, suppose $\mathsf{KState}_{\|}(\mathsf{tr}, \mathsf{w}) = 0$ at a prover-move transcript. Then $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \notin \mathcal{R}'$. If $\mathsf{tr} = \emptyset$, then by Definition 5.12, at least one induced branch triple is not in its $\mathsf{PESAT}_{\mathbb{Q}[\mathsf{X}]}$ relation. By the empty-transcript property for the lifted branch protocol in Theorem 5.11, the corresponding branch-local knowledge state is 0 for all auxiliary choices: $\mathsf{KState}_i(\emptyset; \bar{\mathbb{w}}_i, \zeta_i) = 0$.

By the prover-move clause for that lifted branch protocol, it remains 0 after the prover message. If tr is partial, then $\mathsf{KState}_{\mathrm{br}}(\mathsf{tr}, \mathsf{w}) = 0$, so some component is already 0 and the same componentwise prover-move argument applies. Thus $\mathsf{KState}_{\|}$ remains 0 after any prover move. At a full branch-phase transcript, the target is $\mathcal{R}_{\perp}$, so by the accept/reject convention for $\mathcal{R}_{\perp}$, the full-transcript clause holds because $\mathsf{KState}_{\|} = 1$ exactly when all lifted branch protocols accept.

For the verifier-transition bound, fix a global pre-round transcript, a candidate branch-phase knowledge state witness $\mathsf{w} = (\mathbb{w}', \zeta_0, \ldots, \zeta_{|\mathbf{q}|})$, and a branch-phase round j. If $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \in \mathcal{R}'$, then the extractor returns w, so the previous state is already 1 and no bad transition occurs. Thus assume $(\mathbb{i}', \mathbb{x}'; \mathbb{w}') \notin \mathcal{R}'$. Let $A_j \subseteq [0, |\mathbf{q}|]$ be the set of branches that receive a verifier message in this global round. Branches outside A_j have unchanged local transcripts during this round. If

$$\mathsf{KState}_{\|}(\mathsf{tr}, E_{\|}(\mathsf{tr}\|\rho, \mathsf{w})) = 0 \qquad \text{and} \qquad \mathsf{KState}_{\|}(\mathsf{tr}\|\rho, \mathsf{w}) = 1,$$

then the source-membership disjunct is false for both w and $E_{\|}(\mathsf{tr}\|\rho, \mathsf{w})$, because the extractor leaves $\mathbb{w}'$ unchanged. Therefore the pre-state is controlled by the branch-product predicate. The post-state being 1 implies that all active branch-local post-states are 1; if a branch completes in this round, this is exactly the full local state, by the full-transcript clause for the lifted branch protocol. Inactive branch-local states do not change during this round. Hence some active branch $i \in A_j$ has a component transition from 0 to 1:

$$\mathsf{KState}_i(\mathsf{tr}_i; \bar{\mathbb{w}}_i, \zeta_i') = 0 \quad \text{and} \quad \mathsf{KState}_i(\mathsf{tr}_i\|\rho_i; \bar{\mathbb{w}}_i, \zeta_i) = 1.$$

For each active branch $i \in A_j$, the shared string ρ_j is distributed exactly as the verifier message in the standalone round-j execution of Π_i'. Hence the RBR knowledge soundness bound for that branch applies. A union bound over the active branches gives

$$\Pr_{\rho_j}\left[\mathsf{KState}_{\|}(\mathsf{tr}, E_{\|}(\mathsf{tr}\|\rho_j, \mathsf{w})) = 0 \ \wedge \ \mathsf{KState}_{\|}(\mathsf{tr}\|\rho_j, \mathsf{w}) = 1\right] \le \sum_{i \in A_j} \kappa_{i,j}^{\mathrm{br}}.$$

By the lifted-branch bound above, each $\kappa_{i,j}^{\mathrm{br}} \le 2^{-\lambda}/(|\mathbf{q}| + 1)$, and $|A_j| \le |\mathbf{q}| + 1$. Hence every branch-phase verifier round has error at most $2^{-\lambda}$ per round.

Sequential composition with the PIOR prefix. The protocol Π_{Σ} is the sequential composition of the PIOR prefix, which reduces $\mathsf{REL}_{\mathsf{UCS}}$ to $\mathcal{R}'$, and the parallel branch phase, whose source is $\mathcal{R}'$ and whose target is $\mathcal{R}_{\perp}$. At the boundary, we carry the auxiliary slots ζ_i using the convention of Remark 3.10; these slots are not part of the $\mathcal{R}'$ witness and are used only by the branch-local knowledge states. The composed knowledge state uses the PIOR-prefix knowledge state during the prefix and the parallel branch-phase knowledge state after the PIOR verifier outputs the structured $\mathcal{R}'$ instance. Hence the per-round error vector is the concatenation of the PIOR-prefix and branch-phase error vectors, whose entries are all at most $2^{-\lambda}$. Thus Π_{Σ} has knowledge-state-form RBR-KS with per-round error at most $2^{-\lambda}$, and round count $k_{\mathsf{PIOR}} + \max_{i \in [0, |\mathbf{q}|]} k_i$.

Compilation interface. Each lifted Π_i' is CP, so it sends no polynomial-oracle messages. The only polynomial oracles in Π_{Σ} are therefore Algorithm 1's first-round oracles $[[\widetilde{\mathbf{f}}_0]], \ldots, [[\widetilde{\mathbf{f}}_{|\mathbf{q}|}]]$. Hence Π_{Σ} satisfies the structural interface of Theorem B.2.

Compiled IOP. Applying Theorem B.2 to Π_Σ with the IOPPs of Item 2 produces a compiled IOP whose per-round RBR-KS error vector consists of the inherited PIOP-simulation errors from Π_Σ, the one-shot out-of-domain entry $\varepsilon_{\mathsf{ood}}$, and the final IOPP-round entries from Theorem B.2. The inherited PIOP entries are at most $2^{-\lambda}$ by the PIOR-prefix and parallel-branch arguments above. The out-of-domain entry is at most $2^{-\lambda}$ by hypothesis. Each final IOPP-round entry is at most $2^{-\lambda}$ by Item 2 and the error vector of Theorem B.2. Hence every entry of the compiled IOP's error vector is at most $2^{-\lambda}$. □

A.2 Proofs for the Zip+ IOPP

This subsection proves round-by-round knowledge soundness for the constrained interleaved-code IOPP and the batched multilinear IOPP from Section 6.

A.2.1 Round-by-round knowledge soundness for the constrained interleaved-code IOPP

Proof of Theorem 6.2, round-by-round knowledge soundness. We work against Definitions 3.8 and 3.9. Write $a = (a_1, \ldots, a_J) \in \mathbb{F}_m^J$. We first define the knowledge states and extractors associated with the two verifier messages, and then verify the required bounds for each round.

Knowledge states and extractors. We define knowledge states $\mathsf{KState}_0, \mathsf{KState}_1, \mathsf{KState}_2$. The initial state is

$$\mathsf{KState}_0(\mathtt{i}, \mathtt{x}, \emptyset, \mathtt{w}) = 1 \iff (\mathtt{i}, \mathtt{x}; \mathtt{w}) \in \mathsf{REL}_{\mathsf{CIC}}.$$

As required by Definition 3.8, the knowledge state is unchanged after prover messages.

After the verifier's first message in step 1, the transcript is

$$\tau_1 = (r_1, \ldots, r_J).$$

Set

$$v^* = \sum_{j=1}^{J} r_j v_j, \qquad \alpha^* \equiv \sum_{j=1}^{J} r_j a_j \pmod m.$$

We define

$$\mathsf{KState}_1(\mathtt{i}, \mathtt{x}, \tau_1, \mathsf{w}) = 1$$

if and only if $\mathsf{w} = w^\star \in \left(\mathbb{Q}^{<2J(B_0+K)+\log(m)}\right)^k$ and

$$\mathsf{dist}(v^*, Mw^\star) \le \beta \qquad \text{and} \qquad w^\star \cdot u \equiv \alpha^* \pmod m.$$

The extractor $\mathsf{E}_1(\mathtt{i}, \mathtt{x}, \tau_1, w^\star)$ is defined as follows.

(i) Compute the agreement set

$$S := \{\ell \in [n] : (Mw^\star)_\ell = (v^*)_\ell\}.$$

(ii) If $|S| < (1-\beta)n$, output $\perp$.

(iii) Otherwise, solve a linear system to find a tuple $(f_1, \ldots, f_J) \in \left((\mathbb{Q}^{<B})^k\right)^J$ such that

$$Mf_j|_S = v_j|_S \quad \forall j \in [J], \qquad \phi_m(f_j) \cdot u = a_j \forall \quad j \in [J],$$

and

$$\sum_{j=1}^{J} r_j f_j = w^\star.$$

If no such tuple exists, output $\perp$; otherwise output the tuple found.

We interpret $\perp$ as an invalid witness, so $\mathsf{KState}_0(\mathbb{i}, \mathbb{x}, \emptyset, \perp) = 0$.

After the verifier's second message in step 3, the transcript is

$$\tau_2 = \big((r_1, \ldots, r_J), w, (\ell_1, \ldots, \ell_C)\big).$$

We define

$$\mathsf{KState}_2(\mathbb{i}, \mathbb{x}, \tau_2, \mathsf{w}) = 1$$

if and only if the verifier accepts on transcript τ_2. Finally, the round-2 extractor simply returns the prover's message from step 2:

$$\mathsf{E}_2(\mathbb{i}, \mathbb{x}, \tau_2, \mathsf{w}) = w.$$

It is immediate that $\mathsf{KState}_0, \mathsf{KState}_1, \mathsf{KState}_2$ satisfy the requirements of Definition 3.8.

First round of interaction. Throughout, we will write $Err_{\mathsf{pg}} := Err_{\mathsf{pg}}(M, \beta, K)$ to denote the MCA error of the code generated by M at distance β with respect to $[2^K]$. Knowledge soundness for the first verifier message is established by the following lemma.

Lemma A.3. *Let $\tau_0 = \emptyset$. Then*

$$\Pr_{\rho=(r_1,\ldots,r_J)\in[2^K]^J}\Big[\exists \mathsf{w} \;:\; \mathsf{KState}_0\big(\mathbb{i}, \mathbb{x}, \tau_0, \mathsf{E}_1(\mathbb{i}, \mathbb{x}, \tau_0 \| \rho, \mathsf{w})\big) = 0 \\ \wedge\; \mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_0\|\rho, \mathsf{w}) = 1\Big] \le (J-1)\mathsf{err}_{\mathsf{pg}} + \frac{L}{2^{K-1}}.$$

Proof. Let Γ_β be the set of all tuples $(f_1, \ldots, f_J) \in (\mathbb{Q}^k)^J$ such that

$$\mathsf{dist}\big((Mf_1, \ldots, Mf_J), (v_1, \ldots, v_J)\big) \le \beta.$$

By the list-decoding property of $\mathcal{C}^J$, we have $|\Gamma_\beta| \le L$.

Fix $\rho = (r_1, \ldots, r_J)$, write $\tau_1 = \tau_0 \| \rho$, and suppose there exists $w^\star$ such that

$$\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_1, w^\star) = 1 \qquad \text{but} \qquad \mathsf{KState}_0\big(\mathbb{i}, \mathbb{x}, \tau_0, \mathsf{E}_1(\mathbb{i}, \mathbb{x}, \tau_1, w^\star)\big) = 0.$$

Let

$$S := \{\ell \in [n] : (Mw^\star)_\ell = (v^*)_\ell\}.$$

Since $\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_1, w^\star) = 1$, we have $|S| \ge (1-\beta)n$. The extractor can fail only if at least one of the following bad events occurs:

(a) There exists $w^\star \in \big(\mathbb{Q}^{<2J(B_0+K)+\log(m)}\big)^k$ such that $\mathsf{dist}(v^*, Mw^\star) \le \beta$, but either

- $w^\star \ne \sum_{j=1}^J r_j f_j$ for every $(f_1, \ldots, f_J) \in \Gamma_\beta$, or
- for the set S above there is no tuple $(f_1, \ldots, f_J) \in \Gamma_\beta$ such that $Mf_j|_S = v_j|_S$ for all $j \in [J]$ and $\sum_{j=1}^J r_j f_j = w^\star$.

(b) There exists $(f_1, \ldots, f_J) \in \Gamma_\beta$ with $f_j \notin (\mathbb{Q}^{<B})^k$ for some $j \in [J]$ and

$$\sum_{j=1}^{J} r_j f_j = w^\star$$

for some $w^\star \in \left(\mathbb{Q}^{<2J(B_0+K)+\log(m)}\right)^k$.

(c) There exists $(f_1, \ldots, f_J) \in \Gamma_\beta$ such that $f_j \cdot u \not\equiv a_j \pmod m$ for some $j \in [J]$, and

$$\sum_{j=1}^{J} r_j f_j = w^\star \qquad \text{with} \qquad \phi_m(w^\star) \cdot u = v\alpha^*$$

for some $w^\star \in \left(\mathbb{Q}^{<2J(B_0+K)+\log(m)}\right)^k$.

Indeed, if none of these events occurs and $\mathsf{KState}_1(\mathtt{i}, \mathtt{x}, \tau_1, w^\star) = 1$, then there is a tuple $(f_1, \ldots, f_J) \in \Gamma_\beta$ such that $Mf_j|_S = v_j|_S$ for all j, $\sum_{j=1}^J r_j f_j = w^\star$, each f_j has bitlength $< B$, and each $f_j \cdot u \equiv a_j \pmod m$. By definition of Γ_β, the interleaved codeword $(Mf_1, \ldots, Mf_J)$ is β-close to $(v_1, \ldots, v_J)$. Hence $(f_1, \ldots, f_J)$ is a valid witness for KState_0, and E_1 outputs such a tuple rather than $\perp$. Therefore the failure event in the lemma can only occur if at least one of the above items occur.

By Theorem 3.6, event (a) occurs with probability at most $(J-1)\mathsf{err}_{\mathsf{pg}}$.

For event (b), fix $(f_1, \ldots, f_J) \in \Gamma_\beta$ such that $f_j \notin (\mathbb{Q}^{<B})^k$ for some j, meaning that this tuple has at least one entry with large bitsize. By Lemma 6.11, the probability over the random choice of $r_1, \ldots, r_J$ that $\sum_{j=1}^J r_j f_j$ lies in the allowed message space $\left(\mathbb{Q}^{<2J(B_0+K)+\log(m)}\right)^k$ is at most 2^{-K}. Union bounding over the at most L tuples in Γ_β, we obtain

$$\Pr[\text{event (b)}] \le \frac{L}{2^K}.$$

For event (c), fix $(f_1, \ldots, f_J) \in \Gamma_\beta$ such that $f_j \cdot u \not\equiv a_j \pmod m$ for some $j \in [J]$. If all f_j lie in $\mathbb{Q}^k_{(m)}$, then

$$\left(\sum_{j=1}^{J} \phi_m(r_j)\phi_m(f_j)\right) \cdot u = \sum_{j=1}^{J} \phi_m(r_j) a_j$$

is a nontrivial linear equation in the random coefficients $r_1, \ldots, r_J$, so it holds with probability at most 2^{-K}. If some $f_j \notin \mathbb{Q}^k_{(m)}$, meaning it has an entry with denominator divisible by m, then by Lemma 6.13 the vector $\sum_{j=1}^J r_j f_j$ has denominator divisible by m with probability at least $1 - 2^{-K}$. On this event the congruence

$$\left(\sum_{j=1}^{J} \phi_m(r_j)\phi_m(f_j)\right) \cdot u = v\alpha^*$$

cannot hold. Thus, for each fixed tuple in Γ_β, event (c) occurs with probability at most 2^{-K}, and another union bound gives

$$\Pr[\text{event (c)}] \le \frac{L}{2^K}.$$

Combining the three bounds yields

$$\Pr_{\rho=(r_1,\ldots,r_J)}\left[\exists \mathsf{w} \;:\; \mathsf{KState}_0\big(\mathtt{i}, \mathtt{x}, \tau_0, \mathsf{E}_1(\mathtt{i}, \mathtt{x}, \tau_0 \| \rho, \mathsf{w})\big) = 0 \wedge \mathsf{KState}_1(\mathtt{i}, \mathtt{x}, \tau_0 \| \rho, \mathsf{w}) = 1\right] \le (J-1)\mathsf{err}_{\mathsf{pg}} + \frac{L}{2^{K-1}},$$

as claimed. □

Second round of interaction. We establish knowledge soundness for the second verifier message in the following lemma.

Lemma A.4. *For every partial transcript*

$$\tau = ((r_1, \ldots, r_J), w)$$

ending just before step 3, we have

$$\Pr_{\rho=(\ell_1,\ldots,\ell_C)}\Big[\exists \mathsf{w} \;:\; \mathsf{KState}_1\big(\mathbb{i}, \mathbb{x}, \tau, \mathsf{E}_2(\mathbb{i}, \mathbb{x}, \tau\|\rho, \mathsf{w})\big) = 0 \;\wedge\; \mathsf{KState}_2(\mathbb{i}, \mathbb{x}, \tau\|\rho, \mathsf{w}) = 1\Big] \le (1-\beta)^C.$$

Proof. Fix $\tau = ((r_1, \ldots, r_J), w)$, and let

$$v^* = \sum_{j=1}^{J} r_j v_j, \qquad \alpha^* \equiv \sum_{j=1}^{J} \phi_m(r_j) a_j.$$

By construction, $\mathsf{E}_2(\mathbb{i}, \mathbb{x}, \tau\|\rho, \mathsf{w}) = w$. Moreover, the message space in step 2 guarantees that

$$\phi_m(w) \cdot u = \alpha^*.$$

Hence the only way that

$$\mathsf{KState}_1\big(\mathbb{i}, \mathbb{x}, \tau, \mathsf{E}_2(\mathbb{i}, \mathbb{x}, \tau\|\rho, \mathsf{w})\big) = 0$$

can occur is if $\mathsf{dist}(v^*, Mw) > \beta$. If this happens, then Mw and v^* disagree on more than a β-fraction of the coordinates, so the probability that all C spot checks pass is at most $(1-\beta)^C$. Therefore

$$\Pr_{\rho=(\ell_1,\ldots,\ell_C)}\Big[\exists \mathsf{w} \;:\; \mathsf{KState}_1\big(\mathbb{i}, \mathbb{x}, \tau, \mathsf{E}_2(\mathbb{i}, \mathbb{x}, \tau\|\rho, \mathsf{w})\big) = 0 \;\wedge\; \mathsf{KState}_2(\mathbb{i}, \mathbb{x}, \tau\|\rho, \mathsf{w}) = 1\Big] \le (1-\beta)^C,$$

as required. □

□

A.2.2 Round-by-round knowledge soundness for the batched multilinear IOPP

Proof of Theorem 6.5, round-by-round knowledge soundness. We work in the simple one-constraint case.

For each $j \in [k_2]$, recall that we write

$$v_j = \sum_{i<d} v_j^{(i)} X^i, \qquad w_j = \sum_{i<d} w_j^{(i)} X^i, \qquad \alpha = \sum_{i<d} \alpha^{(i)} X^i.$$

For each $i \in [0 \ldots d-1]$, let $V^{(i)} \in \mathbb{Q}^{k_2 \times n}$ denote the k_2-interleaving

$$V^{(i)} = (v_1^{(i)}, \ldots, v_{k_2}^{(i)}),$$

and let $W^{(i)} \in \mathbb{Q}^{k_2 \times k_1}$ denote the coefficient matrix whose j-th row is $w_j^{(i)}$.

We let Γ_β denote the list of coefficient-tuples

$$(F^{(0)}, \ldots, F^{(d-1)}) \in \left(\mathbb{Q}^{k_2 \times k_1}\right)^d$$

such that there exists a set $S \subseteq [n]$ of size at least $(1-\beta)n$ for which

$$F^{(i)} M^T|_S = V^{(i)}|_S \qquad \forall i \in [0 \dots d-1].$$

Equivalently, Γ_β is the list of underlying messages in the β-list of the interleaved input. By the same list-decoding assumption of $\mathcal{C}$, we have

$$|\Gamma_\beta| \le L.$$

Throughout, we will write $Err_{\mathsf{pg}} := Err_{\mathsf{pg}}(M, \beta, K)$ to denote the MCA error of the code generated by M at distance β with respect to $[2^K]$.

Knowledge states and extractors.

We define knowledge states only after verifier moves. For transcripts ending in prover moves, the knowledge state is unchanged.

Initial state. We define

$$\mathsf{KState}_0(\mathbb{i}, \mathbb{x}, \emptyset, \mathbb{w}) = 1 \iff (\mathbb{i}, \mathbb{x}; \mathbb{w}) \in \mathsf{REL}_{\mathsf{BMLE},\mathbb{Q}}.$$

Round 1 knowledge state. After the verifier samples $\gamma_0, \dots, \gamma_{d-1} \in [2^K]$, the transcript is

$$\tau_1 = (\gamma_0, \dots, \gamma_{d-1}).$$

Set

$$V^* := \sum_{i<d} \gamma_i V^{(i)} \in (\mathbb{Q}^{k_2})^n \qquad \text{and} \qquad \alpha^* := \sum_{i<d} \gamma_i \alpha^{(i)} \in \mathbb{F}_m.$$

We define

$$\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_1, \mathsf{w}) = 1$$

if and only if $\mathsf{w} = W^* \in \left(\mathbb{Q}^{<B_{\mathsf{agg}}}\right)^{k_2 \times k_1}$ and

$$\mathsf{dist}(V^*, W^* M^T) \le \beta \qquad \text{and} \qquad q_2^T W^* q_1 \equiv \alpha^* \pmod m.$$

Round 1 extractor. The extractor $\mathsf{E}_1(\mathbb{i}, \mathbb{x}, \tau_1, W^*)$ works as follows.

(i) Compute the agreement set

$$S := \{\ell \in [n] : (W^* M^T)_\ell = V^*_\ell\}.$$

(ii) If $|S| < (1-\beta)n$, output $\perp$.

(iii) Otherwise, solve the linear system to find a tuple

$$(F^{(0)}, \dots, F^{(d-1)}) \in \left(\left(\mathbb{Q}^{<B}\right)^{k_2 \times k_1}\right)^d$$

satisfying

$$F^{(i)} M^T|_S = V^{(i)}|_S \quad \forall i < d,$$
$$q_2^T F^{(i)} q_1 \equiv \alpha^{(i)} \pmod m \quad \forall i < d,$$

and

$$\sum_{i<d} \gamma_i F^{(i)} = W^*.$$

If no such tuple exists, output $\perp$. Otherwise output the matrix

$$W := \sum_{i<d} F^{(i)} X^i \in \left(\mathbb{Q}^{<d,B}[X]\right)^{k_2 \times k_1}.$$

As before, $\perp$ is interpreted as an invalid witness.

The Π_{CIC} phase. After the prover sends $a \in \mathbb{F}_m^{k_2}$, the verifier checks

$$q_2 \cdot a \equiv \alpha^* \pmod{m}$$

and rejects otherwise. Conditioned on passing this check, define the derived $\mathsf{REL}_{\mathsf{CIC}}$-instance

$$\mathbb{i}_{\mathsf{CIC}} = (M, B_{\mathsf{code}}, n, k_1, k_2, B', \beta, (q_1, a, m)), \qquad \mathbb{x}_{\mathsf{CIC}} = ([[v]]_1^*, \ldots, [[v]]_{k_2}^*),$$

where $v_j^* = \sum_{i<d} \gamma_i v_j^{(i)}$. For all subsequent verifier messages, we use exactly the knowledge states and extractors from Appendix A.2.1 for the protocol Π_{CIC} on this derived instance.

First round of interaction (batched).

Lemma A.5. *Let $\tau_0 = \emptyset$. Then*

$$\Pr_{\rho=(\gamma_0,\ldots,\gamma_{d-1})}\left[\exists \mathsf{w} \ : \ \mathsf{KState}_0\big(\mathbb{i}, \mathbb{x}, \tau_0, \mathsf{E}_1(\mathbb{i}, \mathbb{x}, \tau_0 \| \rho, \mathsf{w})\big) = 0 \wedge \mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_0 \| \rho, \mathsf{w}) = 1\right] \leq (d-1)\mathsf{err}_{\mathsf{pg}} + \frac{L}{2^{K-1}}.$$

Proof. Fix $\rho = (\gamma_0, \ldots, \gamma_{d-1})$, let $\tau_1 = \tau_0 \| \rho$, and suppose there exists W^* such that

$$\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_1, W^*) = 1 \qquad \text{but} \qquad \mathsf{KState}_0\big(\mathbb{i}, \mathbb{x}, \tau_0, \mathsf{E}_1(\mathbb{i}, \mathbb{x}, \tau_1, W^*)\big) = 0.$$

Let

$$S := \{\ell \in [n] : (W^* M^T)_\ell = V_\ell^*\}.$$

Since $\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_1, W^*) = 1$, we have $|S| \geq (1-\beta)n$. The extractor can fail only if at least one of the following bad events occurs:

(a) There exists $W^* \in \left(\mathbb{Q}^{<B_{\mathsf{agg}}}\right)^{k_2 \times k_1}$ such that $\mathsf{dist}(V^*, W^* M^T) \leq \beta$, but either

- $W^* \neq \sum_{i<d} \gamma_i F^{(i)}$ for every $(F^{(0)}, \ldots, F^{(d-1)}) \in \Gamma_\beta$, or
- for the set S above there is no tuple $(F^{(0)}, \ldots, F^{(d-1)}) \in \Gamma_\beta$ such that

$$F^{(i)} M^T|_S = V^{(i)}|_S \quad \forall i < d \qquad \text{and} \qquad \sum_{i<d} \gamma_i F^{(i)} = W^*.$$

(b) There exists $(F^{(0)}, \ldots, F^{(d-1)}) \in \Gamma_\beta$ such that $F^{(i)} \notin \left(\mathbb{Q}^{<B}\right)^{k_2 \times k_1}$ for some $i < d$, and

$$\sum_{i<d} \gamma_i F^{(i)} = W^*$$

for some $W^* \in \left(\mathbb{Q}^{<B_{\mathsf{agg}}}\right)^{k_2 \times k_1}$.

(c) There exists $(F^{(0)}, \ldots, F^{(d-1)}) \in \Gamma_\beta$ such that

$$q_2^T F^{(i)} q_1 \not\equiv \alpha^{(i)} \pmod{m}$$

for some $i < d$, and

$$\sum_{i<d} \gamma_i F^{(i)} = W^* \qquad \text{with} \qquad q_2^T W^* q_1 \equiv \alpha^* \pmod{m}$$

for some $W^* \in \left(\mathbb{Q}^{<B_{\mathsf{agg}}}\right)^{k_2 \times k_1}$.

Indeed, if none of these events occurs and $\mathsf{KState}_1(\mathbb{i}, \mathbb{x}, \tau_1, W^*) = 1$, then the exhaustive search in E_1 finds a tuple

$$(F^{(0)}, \ldots, F^{(d-1)}) \in \left(\left(\mathbb{Q}^{<B}\right)^{k_2 \times k_1}\right)^d$$

with

$$F^{(i)} M^T|_S = V^{(i)}|_S \quad \forall i < d, \qquad q_2^T F^{(i)} q_1 \equiv \alpha^{(i)} \pmod{m} \quad \forall i < d,$$

and $\sum_{i<d} \gamma_i F^{(i)} = W^*$. Repacking these coefficient matrices into

$$W(X) = \sum_{i<d} F^{(i)} X^i$$

yields a valid witness for KState_0, contradiction.

We now bound the three bad events.

For event (a), apply the same list-preservation argument as in Appendix A.2.1, now to the d coefficient-blocks $V^{(0)}, \ldots, V^{(d-1)}$. By Theorem 3.6, event (a) occurs with probability at most

$$(d-1)\mathsf{err}_{\mathsf{pg}}.$$

For event (b), fix $(F^{(0)}, \ldots, F^{(d-1)}) \in \Gamma_\beta$ such that $F^{(i)} \notin \left(\mathbb{Q}^{<B}\right)^{k_2 \times k_1}$ for some i. Flattening matrices into vectors and applying Lemma 6.11, the probability that

$$\sum_{i<d} \gamma_i F^{(i)}$$

lands in the allowed message space $\left(\mathbb{Q}^{<B_{\mathsf{agg}}}\right)^{k_2 \times k_1}$ is at most 2^{-K}. Union bounding over the at most L tuples in Γ_β gives

$$\Pr[\text{event (b)}] \le \frac{L}{2^K}.$$

For event (c), fix $(F^{(0)}, \ldots, F^{(d-1)}) \in \Gamma_\beta$ such that

$$q_2^T F^{(i)} q_1 \not\equiv \alpha^{(i)} \pmod{m}$$

for some $i < d$. If all values $q_2^T F^{(i)} q_1$ lie in $\mathbb{Q}_{(m)}$, then

$$\sum_{i<d} \gamma_i \left(q_2^T F^{(i)} q_1 - \alpha^{(i)}\right) \equiv 0 \pmod{m}$$

is a nontrivial linear equation in $\gamma_0, \ldots, \gamma_{d-1}$, so it holds with probability at most 2^{-K}. If one of the values $q_2^T F^{(i)} q_1$ is not in $\mathbb{Q}_{(m)}$, then by Lemma 6.13 the sum

$$q_2^T \left(\sum_{i<d} \gamma_i F^{(i)}\right) q_1$$

has denominator divisible by m with probability at least $1 - 2^{-K}$, and hence cannot be congruent to α^* modulo m except with probability at most 2^{-K}. Union bounding over Γ_β yields

$$\Pr[\text{event (c)}] \leq \frac{L}{2^K}.$$

Combining the three bounds gives

$$\Pr_{\rho=(\gamma_0,\ldots,\gamma_{d-1})}\Big[\exists \mathsf{w} \;:\; \mathsf{KState}_0\big(\mathbb{i},\mathbb{x},\tau_0,\mathsf{E}_1(\mathbb{i},\mathbb{x},\tau_0\|\rho,\mathsf{w})\big)=0 \wedge \mathsf{KState}_1(\mathbb{i},\mathbb{x},\tau_0\|\rho,\mathsf{w})=1\Big] \leq (d-1)\mathsf{err}_{\mathsf{pg}}+\frac{L}{2^{K-1}},$$

as claimed. □

Transition to the Π_{CIC} phase.

The next message a is sent by the prover, so by definition the knowledge state does not change. To compose with Π_{CIC}, we only need to verify that any witness invalid for KState_1 is also invalid for the initial state of the derived Π_{CIC}-instance.

Lemma A.6. *Let*

$$\tau = ((\gamma_0,\ldots,\gamma_{d-1}),a)$$

be a transcript up to the end of Step 4, and assume that the verifier's check $q_2 \cdot a \equiv \alpha^* \pmod m$ passes. If

$$\mathsf{KState}_1(\mathbb{i},\mathbb{x},(\gamma_0,\ldots,\gamma_{d-1}),W^*)=0,$$

then W^* is not a valid witness for the initial relation of the derived Π_{CIC}-instance

$$(\mathbb{i}_{\mathsf{CIC}},\mathbb{x}_{\mathsf{CIC}}) = \big((M,n,k_1,k_2,B',\beta,(q_1,a,m)),([[v]]_1^*,\ldots,[[v]]_{k_2}^*)\big).$$

Consequently, for any subsequent prover messages in Π_{CIC}, the knowledge state remains invalid.

Proof. If $\mathsf{KState}_1(\mathbb{i},\mathbb{x},(\gamma_0,\ldots,\gamma_{d-1}),W^*)=0$, then either

$$\mathsf{dist}(V^*,W^*M^T) > \beta$$

or

$$q_2^T W^* q_1 \not\equiv \alpha^* \pmod m.$$

In the first case, W^* is not a valid witness for the derived $\mathsf{REL}_{\mathsf{CIC}}$-instance because the proximity condition fails.

In the second case, if W^* were a valid witness for the derived $\mathsf{REL}_{\mathsf{CIC}}$-instance, then it would satisfy

$$W^* q_1 \equiv a \pmod m$$

row-wise. Multiplying by q_2^T, we would obtain

$$q_2^T W^* q_1 \equiv q_2 \cdot a \equiv \alpha^* \pmod m,$$

where the last congruence is exactly the verifier's check in Step 4. This contradicts $q_2^T W^* q_1 \not\equiv \alpha^* \pmod m$. Hence W^* is not a valid witness for the derived instance in this case either.

The final statement follows from the prover-move monotonicity of knowledge states. □

Remaining rounds.

From this point onward the protocol is exactly an execution of Π_{CIC} on the derived instance $(\mathbb{i}_{\mathsf{CIC}}, \mathbb{x}_{\mathsf{CIC}})$. We therefore use the same knowledge states and extractors as in Appendix A.2.1 for that derived instance.

By Lemma A.6, any witness that is invalid for the previous BMLE knowledge state is also invalid for the initial knowledge state of the derived Π_{CIC}-execution. Hence the remaining round-by-round knowledge soundness errors are exactly those of Π_{CIC}, namely

$$\mathsf{err}_1^{\mathsf{CIC}} \qquad \text{and} \qquad \mathsf{err}_2^{\mathsf{CIC}}.$$

Combining this with Lemma A.5, we conclude that Algorithm 3 has round-by-round knowledge soundness with error tuple

$$\big((d-1)\mathsf{err}_{\mathsf{pg}} + L/2^{K-1},\ \mathsf{err}_1^{\mathsf{CIC}},\ \mathsf{err}_2^{\mathsf{CIC}}\big).$$

□

A.2.3 Proofs of proximity-gap theorems

This subsection proves the deferred proximity-gap results from Section 6, namely the basic mutual correlated agreement bound, its J-function generalization, and the corresponding list-preservation property.

Mutual correlated agreement up to the 1.5 Johnson bound. The rest of the section is dedicated to the proof of Theorem 3.4. Towards it, we first prove a slight variant of Theorem 3.4 wherein only one random coefficient is sent.

Theorem A.7. *Let $\mathcal{C} \subseteq \mathbb{Q}^n$ be a linear code with relative distance δ. Then, for any $\varepsilon, \beta \in [0,1)$ such that $\beta = a/n$ for some $a \in \mathbb{N}$ and*

$$\beta > (1 - \delta + \varepsilon)^{1/3}, \quad \varepsilon < 0.18,$$

and any $R \subseteq \mathbb{Q}$, the following holds for any $v_1, v_2 \in \mathbb{Q}^n$:

$$\Pr_r[\mathsf{AD}_{\mathcal{C},\beta}(v_1 + r \cdot v_2) \not\subseteq \cap_{i=1}^2 \mathsf{AD}_{\mathcal{C},\beta}(v_i)] \leq \frac{n}{\varepsilon|R|}.$$

The proof of Theorem A.7 is essentially the same as in [Zei24], but we include it in the next subsection for completeness. Before doing so, we show how Theorem 3.4, restated below, follows from Theorem A.7.

Theorem A.8. *Let $\mathcal{C} \subseteq \mathbb{Q}^n$ be a linear code with relative distance δ. Then, for any $\varepsilon, \beta \in [0,1)$ such that $\beta = a/n$ for some $a \in \mathbb{N}$ and*

$$\beta > (1 - \delta + \varepsilon)^{1/3}, \quad \varepsilon < 0.18,$$

and any $R \subseteq \mathbb{Q}$, the code $\mathcal{C}$ satisfies MCA with respect to R up to distance β with error $\frac{n}{\varepsilon|R|}$.

Proof. Fix any $v_1, v_2 \in \mathbb{Q}^n$. Choose $r_1 \in R$ uniformly and note that $\mathsf{AD}_{\mathcal{C},\beta}(r_1 v_1) = \mathsf{AD}_{\mathcal{C},\beta}(v_1)$. Now the result follows from applying Theorem A.7 to $r_1 v_1$ and v_2 with coefficients from R. □

Proof of Theorem A.7

Throughout this section, fix $v_1, v_2 \in \mathbb{Q}^n$, and a set of coefficients $R \subseteq \mathbb{Q}$. Our goal is to show that

$$\Pr_{r \in R}\left[\mathsf{AD}_{\mathcal{C},\beta}(v_1 + rv_2) \not\subseteq \cap_{i=1}^{2} \mathsf{AD}_{\mathcal{C},\beta}(v_i)\right] \leq \frac{n}{\varepsilon|R|}.$$

We use $\nu(\cdot)$ to denote measure in $[n]$.

Bad Coefficients. Let $\mathcal{A}$ denote the set of "bad coefficients" $r \in R$ such that the event in the probability of Theorem 3.4 occurs. That is, $r \in \mathcal{A}$ if and only if there exists $S \subseteq [n]$ of measure at least β such that $(v_1 + rv_2)|_S \in \mathcal{C}|_S$, $v_1|_S$ and $v_2|_S$ are not both in $\mathcal{C}|_{S'}$. Using this notation, Theorem 3.4 amounts to showing that $|\mathcal{A}| \leq \frac{n}{\varepsilon}$.

Bad Sets. We similarly define a collection of "bad sets", $\mathcal{S} \subseteq 2^{[n]}$. Let $S \subseteq [n]$ be in $\mathcal{S}$ if and only if $\nu(S) \geq \beta$ and there exists some $r \in R$ for which S satisfies the following two conditions:

- S is the largest set satisfying $(v_1 + rv_2)|_S \in \mathcal{C}|_S$.
- One of $v_1|_S$ or $v_2|_S$ is not in $\mathcal{C}|_S$.

Note that it is clear by definition that each bad coefficient $r \in \mathcal{A}$ corresponds to exactly one bad set $S \in \mathcal{S}$. We start by showing a converse statement, that for each bad set $S \in \mathcal{S}$, there is a unique bad coefficient $r \in R$ such that S satisfies the above two conditions with respect to r.

Claim A.9. *There is a bijection between $\mathcal{A}$ and $\mathcal{S}$.*

Proof. The forward direction of the bijection is clear by definition. For each $r \in \mathcal{A}$, we map r to S_r, the largest set $S \subseteq [n]$ with measure at least β such that $(v_1 + rv_2)|_S \in \mathcal{C}|_S$, but either $v_1 \notin \mathcal{C}|_S$ or $v_2 \notin \mathcal{C}|_S$.

We show the reverse direction of the bijection. Suppose that there are two distinct coefficients $r_1, r_2 \in R$ such that both satisfy the above two items for S. Then, we have that

$$(v_1 + r_1 v_2)|_S \in \mathcal{C}|_S \quad \text{and} \quad (v_1 + r_2 v_2)|_S \in \mathcal{C}|_S,$$

which implies, by linearity of $\mathcal{C}$, that both $v_1|_S$ and $v_2|_S$ are in $\mathcal{C}|_S$. But this contradicts the second item above in the definition of $\mathcal{S}$. □

By Claim A.9, we have $|\mathcal{A}| = |\mathcal{S}|$, so to prove Theorem 3.4, it suffices to upper bound $|\mathcal{S}|$. For each $S \in \mathcal{S}$, we let $r(S) \in R$ denote the unique coefficient associated with S such that S satisfies the two items in the definition of $\mathcal{S}$ with respect to $r(S)$.

Lemma A.10. *Fix $S_1, S_2 \in \mathcal{S}$ and suppose there are $m-2$ distinct sets $S_3, \ldots, S_m \in \mathcal{S}$ satisfying $\nu(S_1 \cap S_2 \cap S_i) \geq 1 - \delta$ for all $i \in \{3, \ldots, m\}$. Then, $m \leq n$.*

Proof. Let $S_1, \ldots, S_m \in \mathcal{S}$ be sets satisfying the above condition, and for each i, let $r_i = r(S_i)$ be the associated coefficient. Let $S' = S_1 \cap S_2$.

Note that we have

$$v_1|_{S'} + r_1 v_2|_{S'} \in \mathcal{C} \quad \text{and} \quad v_1|_{S'} + r_2 v_2|_{S'} \in \mathcal{C}.$$

Since $r_1 \neq r_2$ by [cite s to coeffs injective lemma], we have that both $v_1|_{S'}$ and $v_2|_{S'}$ are in $\mathcal{C}|_{S'}$, by linearity of $\mathcal{C}$. Let v_1^*, v_2^* be the codewords that agree with v_1, v_2 respectively on S'. Using the fact that $\nu(S') \geq 1 - \delta$ and for each $i \in [2]$

$$(v_1^* + r_i v_2^*)|_{S'} = (v_1 + r_i v_2)|_{S'},$$

we conclude by the distance of $\mathcal{C}$ and the fact that $(v_1 + r_i v_2)|_{S_i} \in \mathcal{C}|_{S_i}$, it follows that for each $i \in [2]$,

$$(v_1^* + r_i v_2^*)|_{S_i} = (v_1 + r_i v_2)|_{S_i}.$$

Now fix some $i \in \{3, \ldots, m\}$ and set $S'' = S' \cap S_i$, so that $\nu(S'') \geq 1 - \delta$. Then, we have that

$$(v_1^* + r_i v_2^*)|_{S''} = (v_1 + r_i v_2)|_{S''},$$

and $(v_1 + r_i v_2)|_{S_i} \in \mathcal{C}|_{S_i}$, so by the distance of $\mathcal{C}$ and the size of S'', it must be that

$$(v_1^* + r_i v_2^*)|_{S_i} = (v_1 + r_i v_2)|_{S_i}$$

actually holds for all $i \in [m]$.

Next, for each $i \in [m]$, let

$$\begin{aligned} T_i &:= \{i \in S_i \mid v_1(i) \neq v_1^*(i) \wedge v_2(i) \neq v_2^*(i)\} \\ &= \{i \in S_i \mid v_1(i) \neq v_1^*(i) \vee v_2(i) \neq v_2^*(i)\}, \end{aligned}$$

so that for each $i \in [2]$, $|T_i| \geq 1$. To conclude, we note that the T_i's are disjoint. Indeed, suppose that for some $i \neq j$, we have $x \in T_i \cap T_j$. Then,

$$\begin{aligned} v_1(x) + r_i v_2(x) &= v_1^*(x) + r_i v_2^*(x) \\ v_1(x) + r_j v_2(x) &= v_1^*(x) + r_j v_2^*(x), \end{aligned}$$

and $r_i \neq r_j$, so, $v_2(x) = v_2^*(x)$, contradicting $x \in T_j$. Since the T_i's are disjoint and each has measure at least one element, we have that $m \leq n$, as desired. □

Proof of Theorem 3.4. Consider the bipartite graph $G = (A, B, E)$ whose left vertices correspond to $A := \mathcal{S}$ and right vertices correspond to $B := [n]$. We connect a left vertex S to a right vertex i if and only if $i \in S$. Note that each left vertex has degree at least βn. We say that S_1, S_2, S_3, i form a 3-angle if S_1, S_2, S_3 are distinct and all connected to i. Note that the number of 3-angles in G is equal to

$$\sum_{i \in [n]} \binom{d(i)}{3} \geq n \binom{\sum_{i \in [n]} d(i)/n}{3} \geq n \binom{\beta |\mathcal{S}|}{3}$$

where the inequality is by Jensen's inequality.

Next, we upper bound the number of 3-angles. By the previous lemma, there are at most $n|\mathcal{S}|^2$ triples $S_1, S_2, S_3 \in \mathcal{S}$ such that $\nu(S_1 \cap S_2 \cap S_3) \geq 1 - \delta$, and for each such triple, $\nu(S_1 \cap S_2 \cap S_3) \leq \beta$. For all other triples, their threewise intersection has measure at most $(1 - \delta)$, and hence we can upper bound the number of 3-angles as

$$\binom{|\mathcal{S}|}{3} (\beta^3 - \varepsilon) n + |\mathcal{S}|^2 n^2 \beta.$$

Combining the two inequalities gives

$$\binom{\beta |\mathcal{S}|}{3} \leq \binom{|\mathcal{S}|}{3} (\beta^3 - \varepsilon) + |\mathcal{S}|^2 n \beta.$$

From here, we can follow the same work as in the Proof of Lemma 3 in [Zei24] to conclude that $|\mathcal{S}| \leq \frac{n}{\varepsilon}$ as desired. We note that η in [Zei24] is set to $1/n$ for our setting. □

Proofs of Theorem 3.5 and Theorem 3.6. Here, we prove Theorem 3.5 and Theorem 3.6 which are straightforward consequences of mutual correlated agreement.

Proof of Theorem 3.5. The theorem is proved inductively. The base case, where $J = 2$, follows from Theorem 3.4.

Now suppose that the theorem statement holds for J functions. We will show that it also holds for $J+1$. Fix $v_1, \ldots, v_{J+1} \in \mathbb{K}^n$. By the inductive hypothesis,

$$\Pr_{r_2,\ldots,r_J \in R}\left[\mathsf{AD}_{\mathcal{C},\beta}\left(v_1 + \sum_{i=2}^{J} r_i v_i\right) \not\subseteq \bigcap_{i=1}^{J} \mathsf{AD}_{\mathcal{C},\beta}(v_i)\right] \leq (J-1)\mathsf{err}_{\mathsf{pg}}. \tag{112}$$

Let the event in the probability above be E_J. We wish to bound,

$$\Pr_{r_2,\ldots,r_{J+1} \in R}\left[\mathsf{AD}_{\mathcal{C},\beta}\left(v_1 + \sum_{i=2}^{J+1} r_i v_i\right) \not\subseteq \cap_{i=1}^{J+1} \mathsf{AD}_{\mathcal{C},\beta}(v_i)\right].$$

For convenience, let the event in the probability above be E_{J+1}. Then, we have,

$$\Pr_{r_2,\ldots,r_{J+1}}[E_{J+1}] \leq \Pr_{r_2,\ldots,r_{J+1}}[E_{J+1} \mid \overline{E_J}] + \Pr_{r_2,\ldots,r_{J+1}}[E_J] \leq \Pr_{r_2,\ldots,r_{J+1}}[E_{J+1} \mid \overline{E_J}] + (J-1)\mathsf{err}_{\mathsf{pg}}, \tag{113}$$

where we use (112) in the last transition. After choosing $r_2, \ldots, r_J$, let $v' = v_1 + \sum_{i=2}^{J} r_i v_i$. Then,

$$\Pr_{r_2,\ldots,r_{J+1}}[E_{J+1} \mid \overline{E_J}] = \Pr_{r_{J+1} \in R}\left[\mathsf{AD}_{\mathcal{C},\beta}(v' + r_{J+1}v_{J+1}) \not\subseteq \bigcap_{i=1}^{J+1} \mathsf{AD}_{\mathcal{C},\beta}(v_i) \;\middle|\; \overline{E_J}\right].$$

However, since the probability is conditioned on the event $\overline{E_J}$, we have that

$$\mathsf{AD}_{\mathcal{C},\beta}(v') \subseteq \bigcap_{i=1}^{J} \mathsf{AD}_{\mathcal{C},\beta}(v_i),$$

so if $S \notin \bigcap_{i=1}^{J+1} \mathsf{AD}_{\mathcal{C},\beta}(v_i)$, then $S \notin \mathsf{AD}_{\mathcal{C},\beta}(v') \cap \mathsf{AD}_{\mathcal{C},\beta}(v_{J+1})$. Thus the probability above can be bounded by

$$\begin{aligned}
&\Pr_{r_{J+1} \in R}\left[\exists S \in \mathsf{AD}_{\mathcal{C},\beta}(v' + r_{J+1}v_{J+1}), S \notin \bigcap_{i=1}^{J+1} \mathsf{AD}_{\mathcal{C},\beta}(v_i) \;\middle|\; \overline{E_J}\right] \\
&\leq \Pr_{r_{J+1} \in R}\left[\exists S \in \mathsf{AD}_{\mathcal{C},\beta}(v' + r_{J+1}v_{J+1}), S \notin \mathsf{AD}_{\mathcal{C},\beta}(v') \cap \mathsf{AD}_{\mathcal{C},\beta}(v_{J+1})\right] \\
&\leq \mathsf{err}_{\mathsf{pg}}
\end{aligned}$$

where the last inequality follows from the assumption that $\mathcal{C}$ satisfies MCA up to distance β with error $\mathsf{err}_{\mathsf{pg}}$. Combining the above with (113) completes the proof. □

Proof of Theorem 3.6. The theorem follows straightforwardly from Theorem 3.5 by observing that if there exists $u \in \mathsf{List}_{\mathcal{C},\beta}(\sum_{j=1}^{J} r_j v_j)$ which is not in the set

$$\left\{\sum_{j=1}^{J} r_j u_j \mid (u_1, \ldots, u_J) \in \mathsf{List}_{\mathcal{C}^J,\beta}(v_1, \ldots, v_J)\right\},$$

then there must exist $S \subseteq [n]$ of fractional-size greater than $1-\beta$ such that $u|_S = (\sum_{j=1}^{J} r_j v_j)|_S$ but for every $(u_1, \ldots, u_J) \in \mathsf{List}_{\mathcal{C}^J,\beta}(v_1, \ldots, v_J)$ there is some $j \in [J]$ such that $u_j|_S \neq v_j|_S$. Indeed, if there were $(u_1, \ldots, u_J) \in \mathsf{List}_{\mathcal{C}^J,\beta}(v_1, \ldots, v_J)$ such that for all $j \in [J]$ we have $u_j|_S \neq v_j|_S$, then we would have $u|_S = (\sum_{j=1}^{J} r_j u_j)|_S = (\sum_{j=1}^{J} r_j v_j)|_S$, which implies $u = \sum_{j=1}^{J} r_j u_j$ because $\beta \leq \delta$ and $\mathcal{C}$ has distance δ. This is not possible though by the assumption that u is not in the set $\left\{\sum_{j=1}^{J} r_j u_j \mid (u_1, \ldots, u_J) \in \mathsf{List}_{\mathcal{C}^J,\beta}(v_1, \ldots, v_J)\right\}$.

This implies that $S \in \mathsf{AD}_{\mathcal{C},\beta}(\sum_{j=1}^{J} r_j v_j)$ but $S \notin \cap_{j=1}^{J} \mathsf{AD}_{\mathcal{C},\beta}(v_j)$, and the theorem follows from Theorem 3.5. □

B Compiling PIOPs for $\mathrm{REL}_{\mathrm{UCS}}$ into IOPs for $\mathrm{REL}_{\mathrm{UCS}}$

This appendix describes the PIOP-to-IOP compiler used in Theorem 5.16. The compiler replaces each polynomial oracle sent by the PIOP prover with an encoded string oracle, simulates all projected oracle queries explicitly, and then checks the resulting evaluation claims with IOPPs for projected multilinear evaluation. The proximity phase is intentionally *not batched across constraints*: for every evaluation constraint produced by the simulated PIOP, and for every out-of-domain constraint when out-of-domain sampling is enabled, the final phase runs a separate IOPP instance. These IOPPs are run in parallel, and the soundness analysis below accounts for the resulting multiplicity by a union bound over all final IOPP instances.

The construction is similar to the compilers of [ACFY24; ACFY25a]. As in those works, when the proximity parameter is beyond the unique-decoding radius we use out-of-domain samples to isolate a unique nearby message. In the unique-decoding regime the out-of-domain step may be omitted.

Branches and projections. Let

$$\mathcal{B} := \{0\} \cup [Q].$$

Branch 0 is the rational branch. For $b \in [Q]$, branch b is a finite-field branch with field $\mathbb{F}_b := \mathbb{F}_{q_b}$, where $q_b = p_b^{\kappa_b}$. For every branch $b \in \mathcal{B}$ we fix a degree bound d_b, a number of multilinear variables μ_b, and a split $k_{b,1} + k_{b,2} = \mu_b$.

In each branch, the first-round oracle is a projected polynomial oracle (Definition 5.3): $K = \mathbb{Q}$, X-degree bound d_0, bit-length bound B, $\nu = \mu_0$ for branch 0; $K = \mathbb{F}_b$, X-degree bound d_b, $\nu = \mu_b$ for $b \in [Q]$. In Appendix B's local notation, the primitive element a is fixed by the IOPP setup and elided from per-query notation: rational-branch queries are written (z, q) and finite-field-branch queries as z, abbreviating Definition 5.3's (u, q, a) form.

Structural form of the PIOP. Let R be the source relation. We assume a public-coin PIOP Π_{poly} for R with round complexity $\mathrm{rd}_{\mathrm{poly}}$, perfect completeness, and round-by-round knowledge-soundness errors

$$\varepsilon_1^{\mathrm{poly}}, \ldots, \varepsilon_{\mathrm{rd}_{\mathrm{poly}}}^{\mathrm{poly}}.$$

We assume the PIOP has the following commit-and-prove structure.

In the first prover message, for each branch $b \in \mathcal{B}$ and each $\ell \in [s_b]$, the prover sends a polynomial oracle

$$\widetilde{w}_{b,\ell} \in \begin{cases} \mathbb{Q}^{<d_0,B}[X][x_1, \ldots, x_{\mu_0}], & b = 0, \\ \mathbb{F}_b^{<d_b}[X][x_1, \ldots, x_{\mu_b}], & b \in [Q], \end{cases}$$

and $\widetilde{w}_{b,\ell}$ is multilinear in the x-variables. Over the rational branch, the honest prover uses coefficient bit-size at most $B_0 \le B$.

All later prover messages of Π_{poly} are explicit strings. The PIOP verifier may query only the first-round polynomial oracles, and all such queries are projected queries of the form described above. We write $T_{b,\ell}^{(i)}$ for an upper bound on the number of projected queries made in PIOP round i to oracle (b,ℓ).

Codes and IOPPs. For every branch $b \in \mathcal{B}$, let C_b be a linear code with generator matrix M_b, message length $2^{k_{b,1}}$, blocklength n_b, and proximity parameter β_b. We assume $C_b^{2^{k_{b,2}}}$ is (L_b, β_b)-list decodable. In the rational branch, the entries of M_0 have bit-size at most B_{code}. For finite-field branches, any bit-size parameters appearing in the formal syntax of Definition 6.6 are dummy parameters and are ignored.

For every projected multilinear evaluation constraint γ on a branch-b oracle, we assume an IOPP $\Pi^{\gamma}_{\mathrm{prox},b}$ for the corresponding relation $\mathrm{REL}_{\mathrm{PMLE},\mathbb{Q}}$ when $b = 0$, and $\mathrm{REL}_{\mathrm{PMLE},\mathbb{F}_b}$ when $b \in [Q]$, as in Definition 6.6. The IOPP proves that the committed encoded oracle is close to a codeword whose underlying message satisfies the single projected multilinear evaluation constraint γ. Its per-round RBR knowledge-soundness errors are denoted

$$\varepsilon^{\mathrm{prox}}_{b,\gamma,1}, \ldots, \varepsilon^{\mathrm{prox}}_{b,\gamma,\mathrm{rd}_{\mathrm{prox}}}.$$

If all constraints in branch b use a uniform error bound, we write this bound as $\varepsilon^{\mathrm{prox}}_{b,j}$.

The compiler. The compiled IOP, denoted $\mathrm{Compile}(\Pi_{\mathrm{poly}}, \{\Pi_{\mathrm{prox},b}\}_{b\in\mathcal{B}})$, proceeds as follows.

1. **Encoded first-round oracle messages.** For each branch $b \in \mathcal{B}$ and each $\ell \in [s_b]$, the prover sends a string oracle

$$[[V_{b,\ell}]], \qquad V_{b,\ell} \in \begin{cases} (\mathbb{Q}^{<d_0}[X])^{2^{k_{0,2}}\times n_0}, & b = 0, \\ (\mathbb{F}_b^{<d_b}[X])^{2^{k_{b,2}}\times n_b}, & b \in [Q]. \end{cases}$$

   The oracle is intended to be the interleaved encoding $V_{b,\ell} = W_{b,\ell} M_b^{\top}$, where the matrix $W_{b,\ell}$ represents the coefficient table of the multilinear polynomial $\widetilde{w}_{b,\ell}$.

2. **Optional out-of-domain challenge.** If β_b is within the unique-decoding radius for every branch used in the protocol, this step is skipped. Otherwise the verifier samples one out-of-domain point per branch. For every finite-field branch $b \in [Q]$, it samples

$$r_b \leftarrow \mathcal{R}_b^{\mu_b}$$

   from a finite exceptional challenge set $\mathcal{R}_b \subseteq \mathbb{F}_b$ (or from the chosen extension field used by the branch). For the rational branch, it samples

$$r_0 \leftarrow [2^K]^{\mu_0}, \qquad m_{\mathrm{ood}} \leftarrow \mathcal{P},$$

   where $\mathcal{P}$ is a finite set of primes satisfying $m > \max\{2^K, 2^{B_0}\}$ for every $m \in \mathcal{P}$. The projected rational point is $\bar{r}_0 := \varphi_{m_{\mathrm{ood}}}(r_0) \in \mathbb{F}_{m_{\mathrm{ood}}}^{\mu_0}$. The verifier sends $(r_0, m_{\mathrm{ood}}, r_1, \ldots, r_Q)$ to the prover.

3. **Optional out-of-domain answers.** If the previous step was skipped, set all out-of-domain constraint sets below to empty. Otherwise, for every rational-branch oracle $\ell \in [s_0]$, the prover sends

$$\alpha_{0,\ell}^{\mathrm{ood}} \in \mathbb{F}_{m_{\mathrm{ood}}},$$

intended to equal $\psi_{m_{\mathrm{ood}}}(\widetilde{w}_{0,\ell})(\bar{r}_0)$. Record

$$\Gamma_{0,\ell}^{\mathrm{ood}} := \{(\bar{r}_0, \alpha_{0,\ell}^{\mathrm{ood}}, m_{\mathrm{ood}})\}.$$

For every finite-field branch $b \in [Q]$ and every $\ell \in [s_b]$, the prover sends

$$\alpha_{b,\ell}^{\mathrm{ood}} \in \mathbb{F}_b,$$

intended to equal $\psi_b(\widetilde{w}_{b,\ell})(r_b)$. Record

$$\Gamma_{b,\ell}^{\mathrm{ood}} := \{(r_b, \alpha_{b,\ell}^{\mathrm{ood}})\}.$$

4. **Simulation of the PIOP.** The verifier now simulates Π_{poly}. Whenever the simulated PIOP verifier queries a first-round oracle, the compiled verifier asks the prover for the projected query answer and records the resulting evaluation constraint. Concretely, in PIOP round i, for a rational-branch query (z_t, q_t) to oracle $(0, \ell)$, the prover returns $\alpha_t \in \mathbb{F}_{q_t}$ and the verifier records

$$(z_t, \alpha_t, q_t) \in \Gamma_{0,\ell}^{(i)}.$$

For a finite-field query z_t to oracle (b, ℓ), $b \in [Q]$, the prover returns $\alpha_t \in \mathbb{F}_b$ and the verifier records

$$(z_t, \alpha_t) \in \Gamma_{b,\ell}^{(i)}.$$

The answers α_t are supplied to the simulated PIOP verifier, and the explicit prover and verifier messages of Π_{poly} are otherwise simulated unchanged. If the simulated verifier rejects, the compiled verifier records this rejection and will reject at the end.

For every branch and oracle, define the full set of projected evaluation constraints

$$\Gamma_{b,\ell} := \Gamma_{b,\ell}^{\mathrm{ood}} \cup \bigcup_{i=2}^{\mathrm{rd}_{\mathrm{poly}}} \Gamma_{b,\ell}^{(i)}.$$

5. **Parallel per-constraint IOPPs.** For every branch $b \in \mathcal{B}$, every $\ell \in [s_b]$, and every $\gamma \in \Gamma_{b,\ell}$, the parties run one independent IOPP instance $\Pi_{\mathrm{prox},b}^{\gamma}$ on the input oracle $[[V_{b,\ell}]]$. Equivalently, all these IOPPs are run in parallel. The honest prover uses the same underlying message matrix $W_{b,\ell}$ for every IOPP attached to $V_{b,\ell}$; each instance checks one constraint γ on that matrix.

In the rational branch, for $\gamma = (z, \alpha, q)$, the IOPP index is the corresponding instance of $\mathrm{REL}_{\mathrm{PMLE},\mathbb{Q}}$ from Definition 6.6,

$$(M_0, B_{\mathrm{code}}, n_0, 2^{k_{0,1}}, 2^{k_{0,2}}, d_0, B, \beta_0, (z, \alpha, q)).$$

In a finite-field branch $b \in [Q]$, for $\gamma = (z, \alpha)$, the IOPP index is the corresponding instance of $\mathrm{REL}_{\mathrm{PMLE},\mathbb{F}_b}$ from Definition 6.6,

$$(M_b, B_{\mathrm{code},b}, n_b, 2^{k_{b,1}}, 2^{k_{b,2}}, d_b, B_b, \beta_b, (z, \alpha, q_b)),$$

where the bit-size parameters are ignored in the finite-field case.

The compiled verifier accepts if and only if the simulated PIOP verifier accepts and every final IOPP instance accepts.

Remark B.1 (Unique-decoding regime)**.** If the proximity parameters are within the unique-decoding radius, the out-of-domain challenge and answer steps can be omitted. In that case $\Gamma^{\mathrm{ood}}_{b,\ell} = \emptyset$ for all b, ℓ, and the out-of-domain error term below is 0.

Theorem B.2 (PIOP-to-IOP compilation)**.** *Let $\mathcal{R}$ be an instance-witness relation. Let $\{\mathbb{K}_q\}_{q\in[Q]}$ be a collection of fields, and let the branch $q = 0$ denote the rational branch over $\mathbb{Q}$. For each $q \in \{0\} \cup [Q]$, let $\mathcal{R}_q$ be a challenge set, where $\mathcal{R}_0 = [2^K]$ for some $K \in \mathbb{N}$, and where for $q \in [Q]$ the set $\mathcal{R}_q$ is a subset of $\mathbb{K}_q$ or of an extension field of $\mathbb{K}_q$.*

Suppose we have the following ingredients.

- ***PIOP.*** *A PIOP Π_{Poly} for $\mathcal{R}$ satisfying the structural characteristics described above, with the following guarantees:*
  - *Round complexity: rd_p.*
  - *Round-by-round knowledge soundness:*
$$\left(\varepsilon_1^{\mathsf{poly}}, \ldots, \varepsilon_{\mathsf{rd}_p}^{\mathsf{poly}}\right).$$
  - *Completeness: If $(\mathbb{i}, \mathbf{x}, \mathbb{w}) \in \mathcal{R}$, there exists an honest prover who makes the verifier accept with probability 1. Moreover, over the rational branch, the honest prover sends only oracles corresponding to multilinear extensions of vectors in $\mathbb{Q}^{<d_0,B_0}[X]$, where B_0 is a bit-size bound.*
- ***Error-correcting codes.*** *For each $q \in [Q]$, let $k_{q,1}, k_{q,2} \in \mathbb{N}$ satisfy $k_{q,1} + k_{q,2} = \mu_q$. We assume there is an error-correcting code over $\mathbb{K}_q$ with message length $2^{k_{q,1}}$ and blocklength n_q, which is (L_q, β_q)-list decodable.*

  For the rational branch, let $k_{0,1}, k_{0,2} \in \mathbb{N}$ satisfy $k_{0,1} + k_{0,2} = \mu_0$. We assume there is an error-correcting code over $\mathbb{Q}$ with message length $2^{k_{0,1}}$ and blocklength n_0, which is (L_0, β_0)-list decodable and has a generator matrix with entries of bit-size at most B_{code}.
- ***IOPPs with projected multilinear evaluation constraints.*** *For each individual projected multilinear evaluation constraint, we assume the existence of a corresponding IOPP.*
  - ***Over $\mathbb{Q}$.*** *For any projected multilinear evaluation constraint (z, α, q'), where q' is a prime power, $z \in \mathbb{F}_{q'}^{\mu_0}$ is an evaluation point, and $\alpha \in \mathbb{F}_{q'}$ is an evaluation target, we assume that there is an IOPP $\Pi_{\mathsf{prox},0}$ for $\mathsf{REL}_{\mathsf{PMLE},\mathbb{Q}}$ with index*
$$\mathbb{i} = \left(M_0, B_{\mathsf{code}}, n_0, 2^{k_{0,1}}, 2^{k_{0,2}}, d_0, B, \beta_0, (z, \alpha, q')\right).$$
  - ***Over $\mathbb{K}_q$.*** *For any projected multilinear evaluation constraint (z, α), where $z \in \mathbb{K}_q^{\mu_q}$ is an evaluation point and $\alpha \in \mathbb{K}_q$ is an evaluation target, we assume that there is an IOPP $\Pi_{\mathsf{prox},q}$ for $\mathsf{REL}_{\mathsf{PMLE},\mathbb{K}_q}$ with index*
$$\mathbb{i} = \left(M_q, n_q, 2^{k_{q,1}}, 2^{k_{q,2}}, d_q, \beta_q, (z, \alpha)\right).$$

  We assume that each such IOPP has the following guarantees:
  - *Round complexity: $\mathsf{rd}_{\mathsf{prox}}$.*
  - *Completeness:*

  - *Over $\mathbb{Q}$, perfect completeness holds for valid witnesses over $\mathbb{Q}^{<d_0,B_0}[X]$, where $B_0 \le B$.*
  - *Over $\mathbb{K}_q$, perfect completeness holds for all valid witnesses.*
- *Round-by-round knowledge soundness. For an IOPP instance corresponding to branch q, oracle ℓ, and constraint γ, write its errors as*

$$\left(\varepsilon^{\mathsf{prox}}_{q,\ell,\gamma,1}, \ldots, \varepsilon^{\mathsf{prox}}_{q,\ell,\gamma,\mathsf{rd}_{\mathsf{prox}}}\right).$$

Let $\Gamma_{q,\ell}$ denote the set of projected multilinear evaluation constraints recorded by the compiled verifier for the ℓ-th first-round oracle in branch q. Namely,

$$\Gamma_{q,\ell} := \Gamma^{\mathsf{ood}}_{q,\ell} \cup \bigcup_{i=2}^{\mathsf{rd}_p} \Gamma^{(i)}_{q,\ell},$$

where $\Gamma^{\mathsf{ood}}_{q,\ell}$ is empty when the out-of-domain step is omitted. Let

$$N_{q,\ell} := |\Gamma_{q,\ell}| = |\Gamma^{\mathsf{ood}}_{q,\ell}| + \sum_{i=2}^{\mathsf{rd}_p} |\Gamma^{(i)}_{q,\ell}|.$$

The compiled protocol runs one independent IOPP instance for every $q \in \{0\} \cup [Q]$, every $\ell \in [s_q]$, and every $\gamma \in \Gamma_{q,\ell}$. All these IOPP instances are run in parallel.

Then there is an IOP for $\mathcal{R}$ with the following guarantees.

- ***Round complexity:***

$$1 + \mathsf{rd}_p + \mathsf{rd}_{\mathsf{prox}}.$$

  If the out-of-domain step is omitted, the first round is omitted as well.

- ***Completeness:*** *If $(\mathbb{i}, \mathbf{x}, \mathbb{w}) \in \mathcal{R}$, then there exists an honest prover who makes the verifier accept with probability 1.*

- ***Round-by-round knowledge soundness:***

$$\left(\varepsilon_{\mathsf{ood}}, \varepsilon^{\mathsf{poly}}_1, \ldots, \varepsilon^{\mathsf{poly}}_{\mathsf{rd}_p}, \varepsilon^{\mathsf{prox,par}}_1, \ldots, \varepsilon^{\mathsf{prox,par}}_{\mathsf{rd}_{\mathsf{prox}}}\right),$$

  where the first entry is omitted when the out-of-domain step is omitted, and where, for every IOPP round $j \in [\mathsf{rd}_{\mathsf{prox}}]$,

$$\varepsilon^{\mathsf{prox,par}}_j := \max_{q \in \{0\}\cup[Q],\, \ell \in [s_q],\, \gamma \in \Gamma_{q,\ell}} \varepsilon^{\mathsf{prox}}_{q,\ell,\gamma,j}.$$

  The out-of-domain error is

$$\varepsilon_{\mathsf{ood}} = \chi_{\mathsf{ood}} \left(s_0 L_0^2 \left(\varepsilon_{\mathcal{P}} + \frac{\mu_0}{2^K} \right) + \sum_{q=1}^{Q} s_q L_q^2 \frac{\mu_q}{|\mathcal{R}_q|} \right),$$

where $\chi_{\mathsf{ood}} = 1$ if the out-of-domain step is used and $\chi_{\mathsf{ood}} = 0$ otherwise. Here $\varepsilon_{\mathcal{P}}$ denotes the rational-branch random-prime collision error, i.e. the probability over the choice of the out-of-domain prime $m \leftarrow \mathcal{P}$ that two distinct rational-branch list candidates become identical after reduction modulo m. In particular, the prime set $\mathcal{P}$ is chosen so that every $m \in \mathcal{P}$ is larger than 2^K and satisfies the rational-branch IOPP hypotheses.

Proof. The protocol is the compiler described above. The round-complexity claim is immediate from the fact that the out-of-domain challenge, the simulated PIOP, and the parallel IOPPs are run sequentially, while all IOPPs in the final phase are run in parallel.

Completeness follows directly. If $(i, x; w) \in R$, the honest compiled prover encodes each first-round polynomial oracle that the honest Π_{poly} prover would have sent. It answers all out-of-domain and simulated PIOP queries with the corresponding projected multilinear evaluations. The simulated PIOP verifier accepts by completeness of Π_{poly}. For every recorded constraint $\gamma \in \Gamma_{b,\ell}$, the same underlying message $W_{b,\ell}$ is a valid witness for the corresponding REL_{PMLE} instance, so every IOPP accepts by its perfect completeness.

It remains to prove round-by-round knowledge soundness. We use the knowledge-state formulation of Definition 3.9 and the auxiliary witness-slot convention of Remark 3.10. The global knowledge witness contains: a candidate witness for the simulated PIOP, a candidate matrix $W_{b,\ell}$ for every encoded first-round oracle, and the auxiliary knowledge-state witnesses used internally by the final IOPP instances.

For an encoded oracle $V_{b,\ell}$, define its proximity list

$$\Lambda_{b,\ell}(V_{b,\ell}) := \{W : \text{dist}(W M_b^\top, V_{b,\ell}) \le \beta_b\},$$

with the additional coefficient bit-size condition in the rational branch. By the list-decoding hypothesis, every such list has size at most L_b.

Out-of-domain round. The out-of-domain knowledge state is live exactly when, for every branch $b \in \mathcal{B}$ and oracle $\ell \in [s_b]$, no two distinct elements of $\Lambda_{b,\ell}(V_{b,\ell})$ have the same projected multilinear evaluation at the sampled out-of-domain point. If this uniqueness condition fails, the state is permanently doomed. The extractor at this round is the identity on the candidate knowledge witness.

For a finite-field branch $b \in [Q]$, fix one oracle ℓ and two distinct candidates $W_1, W_2 \in \Lambda_{b,\ell}$. Their projected multilinear extensions differ by a nonzero multilinear polynomial in μ_b variables, so the Schwartz–Zippel bound over the exceptional set $\mathcal{R}_b$ gives collision probability at most $\mu_b/|\mathcal{R}_b|$. Union bounding over at most L_b^2 pairs and over the s_b oracles gives the finite-field contribution

$$s_b L_b^2 \frac{\mu_b}{|\mathcal{R}_b|}.$$

For the rational branch, fix one oracle ℓ and two distinct rational candidates. With probability at most $\varepsilon_{\mathcal{P}}$ over $m \leftarrow \mathcal{P}$, they become identical after coefficient-wise reduction modulo m. Conditioned on not becoming identical modulo m, their difference is a nonzero multilinear polynomial over $\mathbb{F}_m$. Since every $m \in \mathcal{P}$ is larger than 2^K, the map $[2^K] \hookrightarrow \mathbb{F}_m$ is injective, and Schwartz–Zippel gives collision probability at most $\mu_0/2^K$ over $r_0 \leftarrow [2^K]^{\mu_0}$. Union bounding over at most L_0^2 pairs and over s_0 rational oracles gives the rational contribution in ε_{ood}. This proves the stated out-of-domain RBR error.

Simulated PIOP rounds. After the out-of-domain state is live, each proximity list contains at most one candidate compatible with the out-of-domain constraint. Adding recorded query constraints can only shrink this compatible list. Therefore, for every branch and oracle, the compiled verifier has a uniquely determined candidate $W_{b,\ell}$ whenever the state remains live.

Given these unique candidates, the compiled transcript determines a genuine transcript of Π_{poly}: the first-round polynomial oracles are the multilinear extensions represented by the matrices $W_{b,\ell}$, and the recorded answers are exactly their projected evaluations. The compiled knowledge state

during PIOP round i is live if and only if the corresponding knowledge state of Π_{poly} is live on this induced transcript. The extractor invokes the extractor of Π_{poly} on the induced transcript and keeps the candidate matrices $W_{b,\ell}$ unchanged. Hence any doomed-to-live transition in the compiled protocol during a simulated PIOP round induces a doomed-to-live transition in Π_{poly}, and the error is at most $\varepsilon_i^{\text{poly}}$.

Entry into the final IOPP phase. At the end of the simulated PIOP, the state is live only if the simulated PIOP verifier has not rejected and the unique candidate $W_{b,\ell}$ for each encoded oracle satisfies every recorded projected evaluation constraint in $\Gamma_{b,\ell}$. If either the simulated verifier rejects, or some recorded constraint is not satisfied by the unique candidate, then at least one final IOPP instance has a false source statement. Since the final knowledge state is the conjunction of the knowledge states of all final IOPP instances, the global state is then doomed at the start of the final phase and the round-by-round soundness of the remainder of the protocol follows. □

Remark B.3 (Fiat–Shamir compilation)**.** The compiled protocol is a public-coin IOP. Although some messages encode rational or polynomial-ring elements, after the degree bounds, bit-size bounds, and field parameters are fixed, every message is represented over a finite alphabet: rational coefficients are encoded as bounded integer numerator and denominator data, and finite-field elements are encoded in the usual finite alphabet. Thus the BCS/Fiat–Shamir compilation results used in Section 3.3.2 apply to this setting, not only to IOPs whose messages are finite field elements. Applying the transformation to the IOP of Theorem 5.17 yields a non-interactive argument of knowledge in the random oracle model with error bounded by the sum of the per-round RBR errors above.