Zero-Knowledge IOPPs for Constrained Interleaved Codes

Alessandro Chiesa
alessandro.chiesa@epfl.ch
EPFL

Giacomo Fenzi
giacomo.fenzi@epfl.ch
EPFL

Guy Weissenberg
guy.weissenberg@epfl.ch
EPFL

November 16, 2026

Abstract

Succinct arguments based on interactive oracle proofs (IOPs) have achieved remarkable efficiency improvements and are now widely adopted in applications. State-of-the-art IOPs involve protocols for testing proximity to constrained interleaved linear codes, and enjoy essentially optimal parameters. However, recent IOP constructions provide no privacy guarantees, and adding zero-knowledge without sacrificing their efficiency remains a challenge.

We present a code-agnostic IOP of proximity for testing constrained interleaved linear codes that achieves (honest-verifier) zero-knowledge, while incurring only $o(1)$ relative communication overhead compared to the (non-zero-knowledge) state of the art.

Our approach is modular: we propose a definition of (honest-verifier) zero-knowledge for interactive oracle reductions (IORs) that accounts for subsequent queries to output implicit instances and is compatible with composition, and then obtain our main result by constructing and modularly composing several lightweight zero-knowledge IORs. Among these are a zero-knowledge sumcheck IOR and a zero-knowledge code-switching IOR that fit the strict efficiency requirements of our setting. Our components also generalize to codes over finite rings and modules.

Keywords: interactive oracle reductions, zero-knowledge, linear codes, dispersers

Contents

1 Introduction **4**
- 1.1 Our results . . . 5
- 1.2 Main technical contributions . . . 9
- 1.3 Practical evaluation . . . 11
- 1.4 Concurrent work . . . 12

2 Technical overview **15**
- 2.1 Basic notions . . . 15
- 2.2 Warmup: a simple zero-knowledge IOPP for constrained codes . . . 15
- 2.3 Composition for honest-verifier zero-knowledge IORs . . . 17
- 2.4 A "committed" relation for zero-knowledge . . . 18
- 2.5 An honest-verifier zero-knowledge sumcheck reduction . . . 19
- 2.6 HVZK code switching . . . 24
- 2.7 From building blocks to our main results . . . 27
- 2.8 Extension: distance-amplified codes . . . 30

3 Preliminaries **31**
- 3.1 Interactive oracle reductions . . . 31
- 3.2 Abstract algebra . . . 32
- 3.3 Error correcting codes . . . 34

4 Zero-knowledge for IORs and composition **40**

5 Succinct linear forms **45**

6 Zero-knowledge sumcheck IOR **47**
- 6.1 The mask polynomial . . . 47
- 6.2 Construction . . . 49
- 6.3 Zero-knowledge . . . 50
- 6.4 Round-by-round security . . . 52

7 A non-succinct zero-knowledge protocol for constrained codes **56**
- 7.1 Zero-knowledge . . . 57
- 7.2 Round-by-round knowledge soundness . . . 58

8 A sublinear zero-knowledge IOP for constrained codes **60**

9 Zero-knowledge code-switching **62**
- 9.1 Out-of-domain samples . . . 62
- 9.2 Private zero-evaders . . . 62
- 9.3 Code-switching complexity . . . 64
- 9.4 Code-switching IOR . . . 65
- 9.5 Zero-knowledge . . . 67
- 9.6 Round-by-round knowledge soundness . . . 68
- 9.7 Proof of Theorem 4 . . . 70

10 Zero-knowledge IOPP for constrained codes **72**
- 10.1 Reed–Solomon instantiation . . . 75

11 Full construction of the zero-knowledge IOPP **77**

12 Zero-knowledge reduction from R1CS **81**

13 High-distance codes from dispersers **87**
13.1 Preliminaries 87
13.2 Amplified Reed–Solomon code 88
13.3 Distance amplification and Merkle-compiled argument size 90

A A generic MCA bound for $(R, \mathcal{M})$-additive codes **92**

B Hiding Merkle commitments without salts **94**

Acknowledgments **96**

References **96**

1 Introduction

Succinct non-interactive arguments (SNARGs) are short cryptographic proofs that admit a fast-verification procedure. They are widely deployed across applications in decentralized systems and cloud computing. We study *hash-based SNARGs*, a class of constructions notable for their transparent setup (the choice of hash function), post-quantum security (in the quantum random oracle model), fast provers (they avoid public-key cryptography), and more. Such SNARGs are used, e.g., for scaling distributed systems [Sta; Pol; Ris; Suc] and for anonymous credentials [Fs26].

Hash-based SNARGs are constructed via the BCS transformation [BCS16] from an interactive oracle proof (IOP), and a beautiful line of research investigates efficient IOP constructions. In this paper we investigate efficient constructions of (honest-verifier) *zero-knowledge* IOPs with the goal of obtaining efficient constructions of (malicious-verifier) *zero-knowledge* hash-based SNARGs.

The efficiency of code-agnostic IOPs. Many constructions of efficient IOPs rely on polynomial codes [AHIV17; BBHR19; BCRSVW19; COS20; ACFY24; HK24; Dia25; MZ25]. The limitations of constructions based on polynomial codes (e.g., superlinear encoding time) have motivated the study of IOPs based only on combinatorial transformations of general linear codes (e.g., tensoring, interleaving), which has achieved "code-agnostic" IOPs [BCGGHJ17; BCG20; RR20; RR22; NA25; BMMS26a; DMR25; DL25; ARR25; BFRW26; BMMS26b]. This is reminiscent of similar efforts for PCPs (the non-interactive counterparts of IOPs) where researchers have sought to decouple PCPs from the properties of the underlying codes in order to understand the minimal structure that enables PCPs and related tools [GS00; DR04; Din07; IKW09; DM11; Mei13; Mei12].

Recent works have shown that general linear codes suffice to construct state-of-the-art IOPs in terms of efficiency [BCGGHJ17; BCFW25; ACFY25; BCFRRZ25; NA25; BMMS26a], *even outperforming* constructions tailored to an error-correcting code (such as [ACFY24]). This has provided a class of highly configurable and efficient IOP constructions.

Tensoring vs. interleaving. There are two fundamental operations on codes that are used to create a modicum of structure for designing "code-agnostic" protocols.

- *Code tensoring.* The tensor product of codes $\mathcal{C}_1 \subseteq \Sigma^{m_1}$ and $\mathcal{C}_2 \subseteq \Sigma^{m_2}$ is the code $\mathcal{C}_1 \otimes \mathcal{C}_2 \subseteq \Sigma^{m_1 \times m_2}$ of matrices $u \in \Sigma^{m_1 \times m_2}$ such that each column of u is in $\mathcal{C}_1$ and each row of u is in $\mathcal{C}_2$. The code $\mathcal{C}_1 \otimes \mathcal{C}_2$ is a "nice" code if $\mathcal{C}_1$ and $\mathcal{C}_2$ both have (even general forms of) linear structure.
- *Code interleaving.* The K-interleaving of a code $\mathcal{C} \subseteq \Sigma^m$ is the code $\mathcal{C}^{\equiv K} \subseteq (\Sigma^K)^m$ obtained by "stacking" K codewords from $\mathcal{C}$, that is, a word $u \in (\Sigma^K)^m$ is in $\mathcal{C}^{\equiv K}$ if and only if there exist words $v_1, \ldots, v_K \in \Sigma^m$ in $\mathcal{C}$ such that $u(z) = (v_1(z), \ldots, v_K(z)) \in \Sigma^K$ for every $z \in [m]$.

With either operation, we obtain linear-time IOP provers by way of linear-time encodable codes, but the tradeoffs differ. Tensor codes support local testing and tensor sumchecks [Mei13; RR20; BCG20; RR22; BCL22; RW24; BFRW26; BMMS26b], and query-optimal IOPs, for example, can be built from them. However, their rate and distance are worse than those of the original code, and their encoding is less efficient. Interleaved codes support alternative sumchecks and code-switching [AHIV17; BBHV22; ACFY25; NA25; BMMS26a] and have the same rate and distance as the original code, but a larger alphabet (which precludes query-optimal IOPs). They are also more efficient to encode (including with small space [BBHV22]), and every code can be interleaved, so a protocol has the most freedom in code choice.

The cost of zero-knowledge. Zero-knowledge for hash-based SNARGs is increasingly in demand [Fs26; Mid; Goo; Lig], which requires the underlying IOP to satisfy (honest-verifier) zero-

knowledge. Such an IOP is usually obtained by modifying an underlying (non-private) IOP. A key goal is minimizing the "zero-knowledge overhead", i.e., the cost incurred by the prover and verifier in using the zero-knowledge version of the protocol compared to its non-private equivalent.

The zero-knowledge overhead of concretely efficient IOPs is $\geq 2\times$ (in prover and verifier times), and they are specific to Reed–Solomon codes [RS60] (or other polynomial codes). This overhead deters developers and engineers from using the zero-knowledge version. Moreover, *none of the state-of-the-art code-agnostic IOPs in the literature provide zero-knowledge.*

Two notable works study zero-knowledge IOPs based on any "zero-knowledge" code (we define these shortly). [BCL22] constructs a tensor-code-based IOP with $\geq 2\times$ overhead. The IOP is not concretely efficient partly due to heavy PCP machinery. [RW24] constructs a tensor-code-based IOP whose communication can be made arbitrarily close to witness size for any fixed constant gap, but only with constant soundness. [BCL22] achieves honest-verifier zero-knowledge and [RW24] achieves malicious-verifier zero-knowledge. Honest-verifier zero-knowledge suffices for achieving (malicious-verifier) zero-knowledge via the BCS transformation in the explicitly programmable random oracle model [BCS16].

The main challenge. In a (non zero-knowledge) IOP, leakage typically takes two forms: (i) queries to oracle messages, and (ii) non-oracle messages. The former leakage can be mitigated via (low overhead) zero-knowledge codes, which hide the secret message from the verifier's codeword queries via randomized encodings. The latter is harder to protect against, as intermediate consistency checks in the protocol may reveal information (as demonstrated in e.g. [XZZPS19; dS25]). Simply adding an input-sized masking oracle would already incur communication proportional to the input length (and thus a $2\times$ overhead). The challenge is to hide the non-oracle messages of the (non-private) IOP at an asymptotically insignificant cost in communication.

1.1 Our results

Concretely-efficient IOPs for NP typically have two components: (a) a "preamble" that reduces the NP-problem, typically Rank-1-Constraint-Satisfaction (R1CS), to testing proximity to a code, and (b) an IOP of proximity (IOPP) that tests proximity to said code. Our main contribution is a concretely-efficient honest-verifier zero-knowledge (HVZK) IOPP for constrained interleaved codes, complemented by an HVZK preamble for R1CS (formally, an interactive oracle *reduction*). We match the communication efficiency of state-of-the-art (non-private) IOPs up to low-order factors, achieving $1 + o(1)$ multiplicative communication overhead (counting the initial codeword) in the parameter regimes below.

ZK encodings. We use *zero-knowledge encodings* to hide messages from the verifier's codeword queries. Codes equipped with these encodings are also known as zero-knowledge codes [ISVW13; BCGV16; BBHR19; BCL22; RW24]. For a finite field $\mathbb{F}$, we say that a code $\mathcal{C} \subseteq \Sigma^m$ is $\mathbb{F}$-*additive* if the alphabet Σ is an $\mathbb{F}$-linear vector space and $\mathcal{C}$ is closed under $\mathbb{F}$-linear combinations. When $\Sigma = \mathbb{F}$ this is an $\mathbb{F}$-linear code. For message length ℓ and randomness length r, we say that a map $\mathsf{Enc}_{\mathcal{C}} \colon \mathbb{F}^\ell \times \mathbb{F}^r \to \Sigma^m$, linear jointly in the message and encoding randomness, is a t-query zero-knowledge encoding for $\mathcal{C}$ if: (i) $\mathsf{Enc}_{\mathcal{C}}$ is injective and its image is $\mathcal{C}$ (i.e., $\mathsf{Enc}_{\mathcal{C}}(\mathbb{F}^\ell, \mathbb{F}^r) = \mathcal{C}$), and (ii) for every message $\mathsf{msg} \in \mathbb{F}^\ell$, if S is an ordered list of t locations sampled uniformly and independently from $[m]$ (allowing repetitions), then the distribution $\mathsf{Enc}_{\mathcal{C}}(\mathsf{msg}, \boldsymbol{r} \leftarrow \mathbb{F}^r)[S]$ can be simulated efficiently, up to some error ζ, given only S (but not msg). Intuitively, a *randomized* encoding of a message yields multiple possible encodings and the view of a certain number of

locations is approximately independent of the message. We ask only that the simulation succeed on average over independently uniform query lists, rather than for every fixed query list (the strong variant). The average-case requirement suffices for the uniformly random locations used by our honest verifiers. See Definition 3.21 for the formal definition. The encoding has p *spare coordinates* if it remains zero-knowledge when p of its random coordinates, the *mask*, are regarded as part of the message; the remaining random coordinates are the *seed* (Definition 3.25). Our sumcheck reduction uses two spare coordinates.

Our target: constrained interleaved codes with ZK encodings. We seek to construct concretely-efficient HVZK IOPPs for relations that involve interleaved zero-knowledge codes whose underlying messages are subject to succinct linear constraints. This is in line with state-of-the-art protocols that consider proximity testing to codes with linear constraints [ACFY25; BMNW25; BCFW25], which enables significantly more efficient (and simpler) reductions from NP compared to merely relying on proximity tests to unconstrained codes. Informally, given a zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}}$ for a code $\mathcal{C}$ we consider the relation of linear constraints defined as follows:

$$\mathcal{R}^{\mathsf{lin}}_{\mathcal{C},T} \coloneqq \left\{ \begin{pmatrix} \mathbb{x} = (\mu, \langle \boldsymbol{v} \rangle) \in \mathbb{F} \times \mathbb{F}^{\log \ell}, \\ \mathbb{y} = f \in \Sigma^m, \\ \mathbb{w} = (\boldsymbol{f}, \boldsymbol{r}) \in \mathbb{F}^{\ell} \times \mathbb{F}^{r} \end{pmatrix} \;\middle|\; \begin{array}{l} f = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r}) \\ \wedge \, \langle \boldsymbol{f}, \boldsymbol{v} \rangle = \mu \end{array} \right\}. \tag{1}$$

Above $\langle \boldsymbol{v} \rangle$ is some "succinct" description of the vector $\boldsymbol{v} \in \mathbb{F}^{\ell}$ from which $\boldsymbol{v}$ can be computed in time T and "k-folds of $\boldsymbol{v}$" (which we define later) can be computed in time $T^{(k)}$. (In fact $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C},T}$ is a simplification of the relation we use. In Section 2.4 we discuss and motivate a refined variant of it.)

HVZK IOPP for constrained interleaved codes. Our main result is an honest-verifier zero-knowledge IOP of proximity for the linear constraint relation $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}^{=2^k},T}$ (Equation 1). The folding parameter $k \geq 1$ regulates how much the problem size "shrinks": given an input of size N, the interaction uses $O(k)$ rounds to reduce to problem size $N/2^k$. We iterate this process multiple times, concluding with a direct test of a small instance. This gives an *HVZK generalization of WHIR* [ACFY25] to general zero-knowledge codes, and also an HVZK variant of Ligerito [NA25]. WHIR achieves state-of-the-art argument sizes [ACFY25] and has been implemented in e.g. OpenVM [Ope26], leanVM [lea26], Flock [BRW26], Succinct [Suc26], and World's ProveKit [Blo26]. It also serves as a component in FICS [BMMS26a] and the query-optimal IOPPs of [BMMS26b], and as the underlying Reed–Solomon proximity test in code-switching constructions such as Blaze [BCFRRZ25] and Bolt [GNR26].

Our IOPPs are zero-knowledge *on fresh inputs*: when the input oracle is an encoding of the message with freshly sampled randomness[1]. This is the guarantee that one needs when an honest prover has committed to the oracle. In our reduction from NP below, the prover sends this oracle itself, and the composed protocol is zero-knowledge in the standard sense (Section 2.3). For simplicity, we state the result for a single code $\mathcal{C}$, using the same code construction at smaller message lengths.[2]

Theorem 1 (informal, HVZK generalization of WHIR)**.** *Let $\mathcal{C} \subseteq \Sigma^{m_{\mathcal{C}}}$ be an $\mathbb{F}$-additive code with a $t_{\mathcal{C}}$-query zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon \mathbb{F}^{\ell_{\mathcal{C}}} \times \mathbb{F}^{r_{\mathcal{C}}} \to \Sigma^{m_{\mathcal{C}}}$ with two spare coordinates, error $\zeta_{\mathcal{C}}$,*

[1] This suffices, as in the composed IOP the prover will use fresh randomness for the encoding. The seeds of the rows are uniform, and their masks are determined by $O(k)$ uniform field elements (Section 2.5).

[2] We assume $\ell_{\mathcal{C}} \geq \lambda$ initially and $r_{\mathcal{C}} = O(\lambda)$ and the same rate, distance, and query zero-knowledge guarantee at every message length, with block length, encoding time, and prover code-switching time decreasing at least proportionally to the message length down to $\Theta(\lambda)$. The encoding time for messages of length $\Theta(\lambda)$ is $\widetilde{O}(\lambda)$.

and encoding time $\tau_{\mathcal{C}}$. Fix a folding parameter $k \geq 1$, security parameter $\lambda \in \mathbb{N}$, and proximity parameter $\delta \in (0, \delta(\mathcal{C}))$. Let $t := O(\frac{\lambda}{-\log(1-\delta)})$, $\ell := 2^k \ell_{\mathcal{C}}$, $K := \lceil \log(\ell/\lambda) \rceil$, and $\mathfrak{n} := 1 + \lceil K/k \rceil$. We construct an $O(k \cdot \mathfrak{n})$-round IOPP for $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}^{\equiv 2^k}, T}$. The protocol has the following features.

- *The prover sends $O(m_{\mathcal{C}} + \mathfrak{n} \cdot k)$ alphabet symbols. The verifier makes $O(\mathfrak{n} \cdot t)$ queries over alphabet Σ^{2^k}.*
- *The prover time is $O(T + 2^k \cdot (\ell_{\mathcal{C}} + r_{\mathcal{C}}) + \tau_{\mathcal{C}} + \mathfrak{t}^{\mathbf{P}}_{\mathcal{C}}) + \widetilde{O}(\mathfrak{n} \cdot (k + \lambda))$ and the verifier time is $O(T^{(K)} + \mathfrak{n} \cdot (2^k \cdot t + \mathfrak{t}^{\mathbf{V}}_{\mathcal{C}})) + \widetilde{O}(\mathfrak{n} \cdot (k + \lambda))$. Here $\mathfrak{t}^{\mathbf{P}}_{\mathcal{C}}$ and $\mathfrak{t}^{\mathbf{V}}_{\mathcal{C}}$ bound the time to compute the linear constraints introduced by code switching and to evaluate their folds, respectively (Section 2.6).*
- *The protocol has round-by-round knowledge soundness with a straightline extractor and error at most $2^{-\lambda}$ assuming $\mathbb{F}$ is large enough relative to the codes' combinatorial list-sizes and mutual correlated agreement at distance δ, as specified in Theorem 10.2.*
- *If $t_{\mathcal{C}} \geq t$, the protocol is honest-verifier zero-knowledge on fresh inputs with error $O(\mathfrak{n} \cdot 2^k \cdot \zeta_{\mathcal{C}})$.*

The notation $\widetilde{O}(\cdot)$ hides $\mathrm{poly}(\log \lambda)$ *factors (here and elsewhere in this introduction).*

The code $\mathcal{C}^{\equiv 2^k}$ has block length $m_{\mathcal{C}}$ over alphabet Σ^{2^k}, equivalently $N = 2^k m_{\mathcal{C}}$ coordinates over Σ. For common settings of the folding parameter k (say $\log \log N$), the prover sends $O(N/2^k + \log N) = o(N)$ base-alphabet coordinates. The additional communication of the zero-knowledge protocol is bounded by this $o(N)$ term, hence is lower order than the corresponding non-private communication when the initial N-coordinate codeword is included, yielding $1 + o(1)$ multiplicative communication overhead. For λ and 2^k polylogarithmic in N, the verifier time is polylogarithmic when the folded constraints and the code-switching linear forms can be evaluated in polylogarithmic time.

The more general construction in Theorem 5 below also permits the parameter settings implicit in [ACFY25] (including rate improvements) and those in [NA25] (with distinct codes in different stages). Compared with [BCL22], which achieves HVZK IOPPs for general linear codes with constant additional overhead and unique-decoding soundness, our protocols achieve $o(1)$ additional relative communication cost and list-decoding soundness in the parameter regimes described above.

Reed–Solomon codes. For Reed–Solomon codes, we obtain the following *perfect-HVZK version of WHIR.*

Corollary 2 (perfect-HVZK WHIR)**.** *Let $\mathcal{C}^{\equiv 2^k}$ be an interleaved Reed–Solomon code with message length ℓ, evaluation domain size m, where $k = \lfloor \log \log \ell \rfloor$, $\log m = O(\log \ell)$, $\log \lambda = O(\log \ell)$, and $1 + \lceil \log(\ell)/k \rceil = O(\lambda)$. Assume the rates and field size permit the mutual correlated agreement and list-size bounds required by Theorem 10.2 (see Section 10.1), with repetition parameters summing to $O(\lambda)$. There exists an IOPP for $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}^{\equiv 2^k}, T}$ with the following features.*

- *The prover sends $O(m + \lambda \log \ell)$ field elements, and the verifier makes $O(\lambda)$ queries over alphabet $\mathbb{F}^{\log \ell}$.*
- *The prover time is $O\left(T + m \cdot \log m + \frac{\lambda \cdot \ell}{\log \ell}\right) + \widetilde{O}(\lambda)$ and the verifier time is $O(T^{(\log \ell)} + \lambda \cdot \log \ell) + \widetilde{O}(\lambda \log \ell)$.*
- *The protocol has round-by-round knowledge soundness error at most $2^{-\lambda}$.*
- *The protocol is perfect honest-verifier zero-knowledge on fresh inputs.*

Sublinear-verifier HVZK IOPP. We can also use a single sumcheck reduction from the interleaved code to its base code, followed by a direct test of the remaining instance (Theorem 3). A notable setting is $2^k \approx m_{\mathcal{C}}$, so that for input length $N = 2^k m_{\mathcal{C}}$ the prover sends $O(\sqrt{N})$ alphabet

symbols. This is the "sublinear regime" of Ligero [AHIV17], here generalized to zero-knowledge codes with list-decoding soundness. With quasi-linear encoding time for the base code, λ and t polylogarithmic in N, and sublinear $T^{(k)}$, the verifier time is sublinear and the multiplicative communication overhead is $1 + o(1)$.

The protocol of Theorem 3 is the first HVZK IOPP for constrained interleaved general codes with list-decoding soundness and $1 + o(1)$ multiplicative communication overhead in the *sublinear verifier* regime. Ligero [AHIV17] considers constrained testing of interleaved Reed–Solomon codes including a zero-knowledge variant, while other works give IOPPs for interleaved general codes in the unique-decoding regime without providing zero-knowledge [BCG20; GLSTW23]. Sublinear protocols of this type are also concretely efficient and are used e.g. in anonymous credentials [Fs26; Goo], KYC solutions [Lig] and constructions such as [DP25] and [XZS22].

Theorem 3 (informal)**.** *Let $\mathcal{C} \subseteq \Sigma^{m_\mathcal{C}}$ be an $\mathbb{F}$-additive code with a $t_\mathcal{C}$-query zero-knowledge encoding $\mathsf{Enc}_\mathcal{C} \colon \mathbb{F}^{\ell_\mathcal{C}} \times \mathbb{F}^{r_\mathcal{C}} \to \Sigma^{m_\mathcal{C}}$ with two spare coordinates, error $\zeta_\mathcal{C}$, and encoding time $\tau_\mathcal{C}$. Fix a folding parameter $k \geq 1$, security parameter $\lambda \in \mathbb{N}$, and proximity parameter $\delta \in (0, \delta(\mathcal{C}))$. Let $t := O(\frac{\lambda}{-\log(1-\delta)})$. We construct an $O(k)$-round IOPP for $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}^{\equiv 2^k},T}$ with the following features.*

- *The prover sends $O(m_\mathcal{C} + k)$ alphabet symbols, and the verifier makes t queries over the alphabet Σ^{2^k} and $O(t)$ additional queries to the proof.*
- *The prover time is $O(T + 2^k \cdot (\ell_\mathcal{C} + r_\mathcal{C}) + \tau_\mathcal{C})$ and the verifier time is $O(T^{(k)} + 2^k \cdot t + \tau_\mathcal{C})$.*
- *The protocol has round-by-round knowledge soundness error at most $2^{-\lambda}$ assuming $\mathbb{F}$ is large enough (relative to $\mathcal{C}$'s combinatorial list-size and mutual correlated agreement at distance δ).*
- *If $t_\mathcal{C} \geq t$, the protocol is honest-verifier zero-knowledge on fresh inputs with error $O(2^k \cdot \zeta_\mathcal{C})$, and the simulator queries the input oracle at most t times.*

The simulation error comes from the encoding of the input oracle, which must hide the masks of the sumcheck together with the message (see Section 2.7 and Theorem 8.1).

HVZK code switching. The polylogarithmic-verifier protocol of Theorem 1 relies on an *HVZK code-switching* IOR from an interleaved code to another code. Code switching reduces proximity testing for one code to proximity testing for another, and was first introduced for tensor codes [RR20; RR22]. After reducing the message size via sumcheck (Section 2.5), the prover re-encodes the resulting message with a second code $\mathcal{C}_2$ and proves consistency of the two encodings. If $\mathcal{C}_2$ is itself an interleaved code, this process can be iterated. Typical choices make 2^k constant or logarithmic in the input length.

In Theorem 4, we use the composable HVZK notion for IORs described in Section 2.3.

Theorem 4 (informal, code switching with HVZK)**.** *For $b \in \{1, 2\}$, let $\mathcal{C}_b \subseteq \Sigma^{m_{\mathcal{C}_b}}$ be an $\mathbb{F}$-additive code that has a $t_{\mathcal{C}_b}$-query zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}_b} \colon \mathbb{F}^{\ell_{\mathcal{C}_b}} \times \mathbb{F}^{r_{\mathcal{C}_b}} \to \Sigma^{m_{\mathcal{C}_b}}$ with error $\zeta_{\mathcal{C}_b}$ and encoding time $\tau_{\mathcal{C}_b}$, where the encoding of $\mathcal{C}_1$ has two spare coordinates. Fix a folding parameter $k \geq 1$, security parameter $\lambda \in \mathbb{N}$, and proximity parameter $\delta \in (0, \delta(\mathcal{C}_1))$. Let $t := O(\frac{\lambda}{-\log(1-\delta)})$. Assuming that $\ell_{\mathcal{C}_2}$ is large enough, we construct an $O(k)$-round IOR of proximity from $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}_1^{\equiv 2^k},T_1}$ to $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}_2,T_2}$, where $T_2 = O(T_1 + T_{\mathcal{C}_1})$ and $T_{\mathcal{C}_1}$ bounds the time to form a linear combination of the t queried rows of $\mathcal{C}_1$'s encoding generator matrix, with the following features.*

- *The prover sends $O(m_{\mathcal{C}_2} + k)$ alphabet symbols, and the verifier makes $O(t)$ queries over alphabet Σ^{2^k}.*
- *The prover time is $O(T_1 + 2^k \cdot (\ell_{\mathcal{C}_1} + r_{\mathcal{C}_1}) + \tau_{\mathcal{C}_2})$ and the verifier time is $O(2^k \cdot t)$.*

- *The protocol has round-by-round knowledge soundness error at most $2^{-\lambda}$ assuming $\mathbb{F}$ is large enough (with respect to the codes' combinatorial list-decoding and mutual correlated agreement errors at distance δ).*
- *If $t_{\mathcal{C}_1} \geq t$, the protocol is honest-verifier zero-knowledge on fresh inputs with error $O(2^k \cdot \zeta_{\mathcal{C}_1}) + \zeta_{\mathcal{C}_2}$ against uniform-query distinguishers (Definition 4.9) that perform at most $t_{\mathcal{C}_2}$ queries. Its output oracle is again fresh, so the reduction can be iterated.*

We note that all our constructions achieve perfect HVZK when the ZK-encodings and zero-evaders have zero privacy error. We discuss the time bound $T_{\mathcal{C}_1}$ in Section 2.6.

HVZK IOR from NP to constrained code testing. As an additional contribution, we provide an interactive oracle reduction (IOR) from the NP-complete R1CS problem to proximity testing of $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}^{=2^k},T}$. The reduction incurs $o(1)$ additional relative communication cost compared with sending a single (interleaved) codeword, illustrating that the efficiency characteristics of the proximity test are also achievable in the reduction from an NP problem. The reduction adapts the Σ-IOP for R1CS in [ACFY25]. The prover hides the messages of its sumcheck with a mask polynomial that is determined by a logarithmic number of field elements, and it masks each of the three R1CS matrix-vector products. Prior reductions from R1CS [BCL22] incur constant additional relative communication cost by sending full masking oracles.

Our reduction can, in particular, be instantiated with the interleaving of a linear-time-encodable code (with zero-knowledge encodings, as shown in [BCL22]) and, together with our proximity test, leads to an HVZK IOP for R1CS with a linear-time prover, list-decoding soundness, and $o(1)$ additional relative communication cost. We note that our size comparison is against *non-zero-knowledge* protocols that use this style of reduction ("reduce R1CS to constrained code testing"), e.g. [ACFY24; ACFY25; MZ25]. Our linear-time instantiation improves upon the state-of-the-art for linear-time provers from [BCL22], which only achieves unique decoding soundness and constant zero-knowledge overhead.

Composing the IOR from NP with Theorem 3 (or its polylogarithmic-verifier version in Theorem 5) achieves an HVZK IOP for NP. The prover of this IOR sends the oracle with fresh randomness, so the composed protocol is zero-knowledge for every R1CS witness. The resulting R1CS verifier is non-succinct, since we omit the holographic/preprocessing step (Section 12). [Fs26] develop an anonymous credential solution using the sublinear hash-based zkSNARK in [AHIV17], which is based on zero-knowledge IOPs with zero-knowledge overhead $\geq 2\times$ (and the corresponding non-private IOPs are no longer state of the art). Our protocols provide an alternative with the communication overhead described above.

1.2 Main technical contributions

Our construction combines two new interactive oracle reductions (IORs), an HVZK sumcheck and an HVZK code switch, using a new HVZK definition and composition theorem for IORs.

- *Keeping sumcheck claims masked.* Sumchecks with small masks already appear in Libra [XZZPS19] and in the tensor-code protocols of [RW24], where the masks are committed separately. Revealing a mask's contribution would expose the corresponding linear form in the message. In our protocols, the masks are coordinates of the randomness with which the prover encodes the oracle itself. They lengthen the input of the encoding by two coordinates per row, and they

require no additional oracle or query. We keep their contribution in the linear constraint relation, to be checked in subsequent reductions (the "committed" sumcheck relation of Section 2.4). The prover samples $O(k)$ field elements, which determine the masks of all 2^k rows and a *mask polynomial*. With this polynomial the prover hides its sumcheck messages over every finite field $\mathbb{F}$, including characteristic two, where the sumcheck partial sums of Libra's mask can vanish.

- *Checking consistency between encodings.* Consistency checks between encodings involve the old encoding randomness as well as the message. We re-encode this randomness together with the message: we distribute it among the rows of the next interleaved code, which adds neither an oracle nor a variable to subsequent sumchecks. Every oracle of our protocols is thus a codeword of a single code, and the randomness with which the prover encodes it contains the masks of the next sumcheck. Moreover, we introduce *private zero-evaders* and use them to sample the additional linear checks used in list decoding (the "out-of-domain" samples) so that their answers hide the message (Section 2.6).

- *Preserving zero-knowledge under composition.* A later reduction may query oracles sent by an earlier one. We therefore include those subsequent queries in our HVZK definition for IORs and prove composition under the conditions in Section 2.3. Our constructions satisfy this definition for the bounded, uniformly sampled queries used by subsequent reductions and their simulators. Since the masks of a sumcheck are part of the randomness of its input oracle, our sumcheck reduction is zero-knowledge only when the prover has sampled this randomness as the reduction prescribes. We define zero knowledge on such *fresh* inputs and show that it composes when the prover of the preceding reduction sends the oracle and samples its randomness in this way.

On the choice of codes. Any $\mathbb{F}$-additive code with an appropriate zero-knowledge encoding can be used in Theorem 3 and Theorem 4. We also extend the construction using *distance amplification based on dispersers* to improve encoding time or argument size, at the cost of a larger alphabet (Section 2.8).

Generalizing the components. Our sumcheck and code-switching reductions generalize to $(R, \mathcal{M})$-additive codes. Such a code is specified by an injective homomorphism $R^\ell \to \mathcal{M}^m$ of right R-modules, where R is a finite ring and $\mathcal{M}$ is a right R-module (prior work includes [NKM97; GNW04; Woo09]). We obtain versions of Theorem 3 and Theorem 4 with soundness bounds depending on worst-case list sizes, mutual correlated agreement, and the polynomial-identity error of the challenge set (Theorems 8.1 and 9.12). This is conceptually similar to generalizing the GKR protocol and Reed–Solomon protocol to Galois rings and possibly infinite and non-commutative rings in [SV22; JLXYY25], with a number of technical differences arising in our setting. We further show that every $(R, \mathcal{M})$-additive code $\mathcal{C}$ has mutual correlated agreement below $\delta(\mathcal{C})/3$ for challenges from a strong sampling set (Lemma A.1), adapting the proof of [AHIV17]. Here a strong sampling set is a finite nonempty set of pairwise commuting ring elements whose differences between distinct elements are invertible (Definition 3.8). When the required code and challenge-set conditions hold, we can prove computations directly in R, without emulating its arithmetic over a field. The same goal motivates prior proof systems for CPU-word arithmetic [BBMHRS21; JLXYY25], ring arithmetic arising in homomorphic encryption [GNSV23], integer arithmetic with arbitrary or multiple moduli, as in Zinc [GWVHDČK25] or Zaratan [CH26], and lattice-based signature protocols [LSS24]. These applications motivate our generalization. Applying our protocols to them requires additional instantiation work.

In the technical overview (Section 2), we specialize to $R = \mathbb{F}$ a finite field and $\mathcal{M} \cong \mathbb{F}^d$ a finite-dimensional $\mathbb{F}$-vector space. In the technical sections we work over finite rings and modules.

1.3 Practical evaluation

We provide a Rust implementation of our main results based on Reed–Solomon codes over the binary field $\mathbb{F}_{2^{256}}$: a sublinear-verifier protocol for proving the evaluation of a multilinear polynomial (Theorem 3), and a polylogarithmic-verifier protocol for the same task (the variant of Theorem 5 in Section 11, whose last stage is the sublinear protocol). Our implementation is publicly available at *WizardOfMenlo/zo0k*.

We compare two communication measures: *IOP proof length* (the total oracle length, including the input oracle, plus the direct messages of the prover) and *argument size* (the size of the argument string after BCS compilation [BCS16]). The former is a proxy for argument prover time, while the latter is a proxy for argument verifier time. We use BLAKE3 as the hash function for compilation and use the Johnson bound MCA result [BCHKS25] to set parameters. The implementation follows the construction of Section 11 with Reed–Solomon codes and commits to its oracles without salts (see the optimization in Appendix B). All configurations target 100 bits of security. Every query round is preceded by 24 bits of proof-of-work grinding, which count towards this target.

We compare a *Pareto frontier* of nondominated configurations from our parameter search. We automatically explore pairs (l, s) of IOP proof length l and argument size s for different protocol configurations. In Figure 1 we plot these frontiers for the sublinear-verifier and polylogarithmic-verifier protocols, with and without zero-knowledge (ZK and non-ZK). As expected, larger IOP proofs correspond to smaller compiled arguments. We report two overheads of zero-knowledge. The *argument size overhead* compares our ZK configuration with the best non-ZK configuration under the same budget on the IOP proof length. The *IOP proof length overhead* compares it with the non-ZK protocol on the same schedule, that is, with the same folding parameters and evaluation domains.

- In Table 1 we tabulate the IOP proof length and argument size of the sublinear-verifier protocol for multilinear polynomials specified by evaluation tables of 2^{16}–2^{20} field elements. We minimize expected argument size subject to IOP proof length at most $5\times$ the input table's length in bytes. The argument size overhead is 3.2% at 2^{16} field elements and 1.2% at 2^{20}, where the argument has 544.0 KiB. The IOP proof length overhead decreases from 3.1% to 0.8% between these endpoints.
- In Table 2 we report the same comparison for the polylogarithmic-verifier protocol on tables of 2^{20}–2^{26} field elements. The argument size overhead decreases from 1.6% to 0.6%, with a 181.9 KiB argument and 0.1% IOP proof length overhead at 2^{26} field elements.

Since the initial publication of this paper, optimized open-source implementations of our protocols have appeared in Plonky3 [Plo] and in ProveKit's WHIR backend [Pro]. Moreover, Spartan-WHIR [Spa] builds on Plonky3's hiding-WHIR implementation and adapts our zero-knowledge R1CS reduction, and FlashProve [RACCDFMW26] implements our proximity test. These works are further evidence of the concrete efficiency of our techniques.

Memory efficiency. The protocol in Theorem 3 may also admit a prover with low memory use. An encoding of the interleaved code $\mathcal{C}^{\equiv 2^k}$ can be computed one base-code block at a time, using the space required for a single $\mathcal{C}$ encoding, as in [BBHV22] for Reed–Solomon codes. There are

also sumcheck algorithms with low time and space complexity [CTY11; VSBW13; BCFFMMZ25; NTZ25]. Adapting them to our masked reduction remains to be studied.

1.4 Concurrent work

Lightweight zero-knowledge. VEIL [DHRR26] describes how to add zero-knowledge with small overhead to a class of hash-based succinct arguments. There are similarities with our techniques: the use of zero-knowledge codes to ensure queries do not reveal information about the witness, and reducing the problem size until one can use a zero-knowledge proof system with $2\times$ overhead.

The main difference is that [DHRR26] applies a black-box transformation to the (non-oracle) transcript of the original IOP (roughly, each message is blinded via a previously committed one-time-pad), whose validity is then proved via a black-box IOP for NP (which resembles [AHIV17]). The (interleaved) proximity test of the original IOP is made ZK in a black-box way by appending an additional random row to the interleaved encoding. Conversely, we choose to white-box the components of state-of-the-art IOPs for NP: we modify the sumcheck protocol and proximity test to make them zero-knowledge. As a benefit, we significantly simplify the inner proof system (checking linear constraints suffices) and we achieve smaller argument sizes. The drawback is that our white-box approach requires delicate changes to the IOP (increasing implementation complexity).

Comparing efficiency with [DHRR26, Table 5.1] is not entirely fair in either direction. [DHRR26] applies their compiler to an IOP employing [ZCF24]'s proximity test in the unique decoding regime, while ours can be seen as an adaptation of [ACFY25] in the list-decoding regime. The latter (in the non-ZK version) has state-of-the-art argument size, and thus our ZK version inherits the better argument size. Our configurations also use 24 bits of proof-of-work grinding (Section 1.3). With these caveats, we compare at trace size 2^{25} with the numbers in [DHRR26, Table 5.1]. Their zero-knowledge argument has 872 KiB and ours has 172.9 KiB (Table 2), which is $\approx 5.0\times$ smaller. Our relative IOP proof length overhead is 0.1% while their prover time overhead is 4% (note this is prover time, not proof length). The relative argument size overhead of our construction is 0.8% while theirs is 17%. In absolute terms, our overhead is 1.3 KiB while theirs is 125 KiB.

Composition of zero-knowledge IOPs. [VW26] studies a general method for composing zero-knowledge IOPs against malicious verifiers making a bounded number of queries. Their analysis accounts for adaptive queries to earlier oracles during later sub-protocols, a difficulty that arises even when messages are exchanged sequentially. Their main results concern composition of multiple IOPs with perfect zero-knowledge and black-box straightline simulators. For two IOPs, they also prove statistical and computational zero-knowledge, assuming that the IOP called as a sub-protocol remains zero-knowledge when the verifier receives auxiliary input.

Our goal is to construct efficient zero-knowledge hash-based SNARGs via the BCS transformation [BCS16]. In the explicitly programmable random-oracle model, HVZK of the underlying IOP suffices to obtain malicious-verifier zero-knowledge of the resulting argument. Hence we prove HVZK for our oracle protocols. Both works compose simulators and account for their queries to preceding oracles: [VW26] allows arbitrary adaptive query locations within the stated bounds, whereas our constructions use uniformly sampled query locations. This allows us to use encodings whose zero-knowledge property need only hold on average over lists of uniformly and independently sampled query locations (Definition 3.21).

[VW26] establishes zero-knowledge for IOPs that use other IOPs as sub-protocols. In Theorem 4.7, the composed protocol is itself an HVZK IOR, whose guarantee accounts for subsequent

queries to its output implicit instance. In our constructions, we prove HVZK for a class of uniform-query distinguishers. An IOR with this guarantee can be followed by any IOR whose HVZK simulator is uniform-query and satisfies the required query bounds (Claim 4.11), without a separate HVZK analysis for each such choice. This allows us to apply the theorem repeatedly to a sequence of reductions, as in the proof of Theorem 10.2. Some of these reductions are zero-knowledge only when the randomness of their input oracle is uniform, and our theorem states when this condition is preserved along the sequence (Definitions 4.3 and 4.5). Iterating our theorem preserves HVZK but does not establish the malicious-verifier guarantee of [VW26]. Conversely, their theorems require malicious-verifier zero-knowledge of the constituent IOPs and do not apply directly under our HVZK assumptions. Thus, neither result subsumes the other as stated [VW26, Section 6.2].

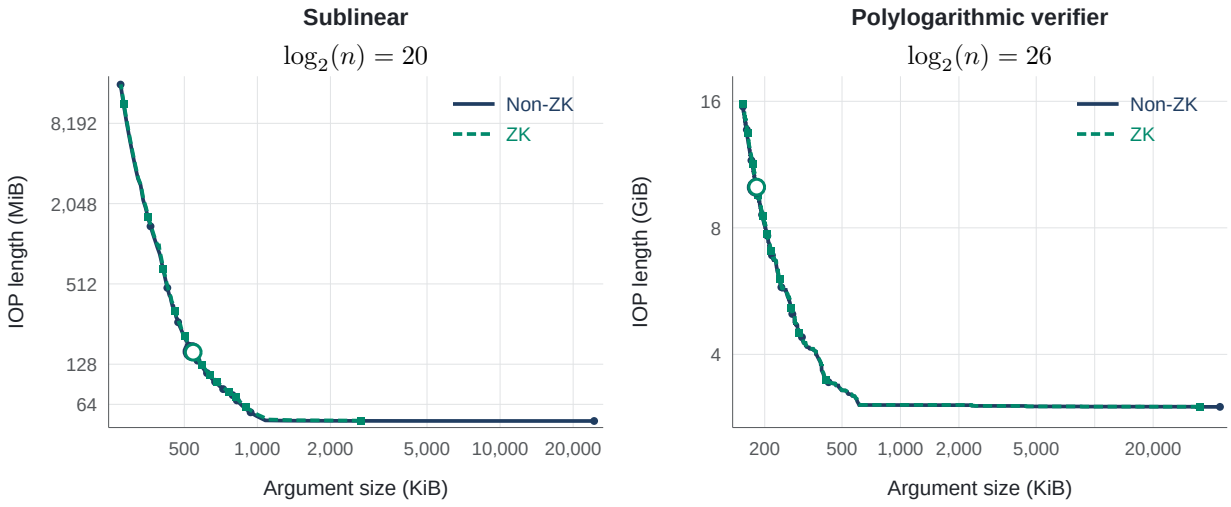

Figure 1: Pareto frontiers of IOP proof length and argument size (logarithmic axes). Open circles mark the configurations in Tables 1 and 2.

$\log_2 n$	IOP length (MiB)			Argument size (KiB)		
	Non-ZK	ZK	Overhead	Non-ZK	ZK	Overhead
16	9.63	9.93	3.1%	144.41	148.97	3.2%
17	19.38	19.98	3.1%	209.19	214.04	2.3%
18	38.63	39.23	1.6%	275.46	281.99	2.4%
19	78.25	79.47	1.6%	405.62	410.02	1.1%
20	156.25	157.47	0.8%	537.58	543.97	1.2%

Table 1: Sublinear protocol under a 5× input IOP budget. Argument overhead is relative to the best non-ZK configuration under the same budget; the non-ZK IOP proof length and the IOP overhead are for the non-ZK protocol on the same schedule.

$\log_2 n$	IOP length (GiB)			Argument size (KiB)		
	Non-ZK	ZK	Overhead	Non-ZK	ZK	Overhead
20	0.16	0.16	0.4%	123.48	125.47	1.6%
21	0.31	0.31	0.3%	134.26	136.03	1.3%
22	0.62	0.62	0.2%	144.12	145.91	1.2%
23	1.25	1.25	0.2%	154.27	155.67	0.9%
24	2.50	2.50	0.1%	163.59	165.06	0.9%
25	4.99	5.00	0.1%	171.57	172.86	0.8%
26	9.99	10.00	0.1%	180.74	181.85	0.6%

Table 2: Polylog protocol under a 5× input IOP budget. Argument overhead is relative to the best non-ZK configuration under the same budget; the non-ZK IOP proof length and the IOP overhead are for the non-ZK protocol on the same schedule.

2 Technical overview

We give an overview of our HVZK protocols for constrained interleaved codes. We begin with a simple, non-succinct protocol for linear constraints, and gradually introduce the main technical components (in particular, our HVZK sumcheck and code-switching reductions) and definitional framework needed to obtain a succinct HVZK protocol with small zero-knowledge overhead. For simplicity, this technical overview focuses on the special case of finite fields and vector spaces. The technical sections are stated for the general case of rings and modules.

2.1 Basic notions

We use the $\mathbb{F}$-additive codes and zero-knowledge encodings introduced in Section 1.1. Full definitions appear in Section 3.

ZK codes. For example, a Reed–Solomon encoding places t independent uniform random field elements in the lowest-degree coefficient positions and the message in the remaining positions. Evaluating this polynomial gives perfect t-query zero-knowledge for every fixed list S of at most t query locations (Lemma 3.29). Other zero-knowledge encodings are known, including some with linear-time encoding algorithms [BCL22] (Theorem 3.30).

IORs. Recent constructions of hash-based succinct arguments and accumulation schemes use a modular approach based on *interactive oracle reductions* (IORs) [BCGGRS19; BMNW25]. In an IOR (of proximity) from $\mathcal{R}$ to $\mathcal{R}'$, the prover $\mathbf{P}$ receives $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$, whereas the verifier $\mathbf{V}$ receives the *explicit instance* $\mathbb{x}$ and query access to the *implicit instance* $\mathbb{y}$. After interacting, $\mathbf{V}$ either rejects or outputs $(\mathbb{x}', \mathbb{y}')$, and $\mathbf{P}$ outputs a new witness $\mathbb{w}'$. Perfect completeness means that, for every input triple in $\mathcal{R}$, the output triple of an honest interaction always belongs to $\mathcal{R}'$.

RBR security. We target round-by-round security [CCHLRR18; CMS19; CY24; BMNW25; BCFW25] to construct *non-interactive* succinct arguments in the random oracle model. Informally, a *round error* bounds the probability of transitioning from a *bad* state for the attacker ("the statement is false") into a *good* state ("the next statement is true"). Specifically, we target *straightline RBR knowledge soundness* for IORs [BCFW25], which is suitable for code-agnostic protocols such as ours: the IOR is viewed as a sequence of one-round reductions, each with a straightline knowledge extractor and an extraction error. These errors depend on proximity parameters δ for $\mathcal{R}$ ("$\mathbb{y}$ is δ-far from $\mathcal{R}$") and δ' for $\mathcal{R}'$ ("$\mathbb{y}'$ is δ'-close to $\mathcal{R}'$").

List decoding and mutual correlated agreement. Our RBR knowledge soundness errors depend on the code's distance, worst-case list size, and mutual correlated agreement error.

- The worst-case list size at distance $\delta \in (0,1)$ is $\Lambda(\mathcal{C}, \delta) \coloneqq \max_f |\Lambda(\mathcal{C}, f, \delta)|$, where $\Lambda(\mathcal{C}, f, \delta) \coloneqq \{u \in \mathcal{C} : \Delta(u, f) \leq \delta\}$ is the set of codewords at relative distance at most δ from f.
- Mutual correlated agreement (MCA) [ACFY25] concerns agreement on a shared set of positions. Except with probability at most $\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta)$ over uniform $\gamma \in \mathbb{F}$, whenever $f_1 + \gamma \cdot f_2$ agrees with a codeword on a set S of at least $(1-\delta)m$ positions, each of f_1, f_2 agrees with some codeword on that same set S (Definition 3.19). We use the same error notation at distance $\delta \in (0,1)$ for $(1-\gamma) \cdot f_1 + \gamma \cdot f_2$, which has the same worst-case error.

2.2 Warmup: a simple zero-knowledge IOPP for constrained codes

We describe a simple honest-verifier zero-knowledge IOPP for linear constraints over any $\mathbb{F}$-additive code. It adds a masking oracle of the same length as the input oracle. We use variants as a "base

case" on small instances in our protocols.

An initial relation. Let $\mathcal{C} \subseteq \Sigma^m$ be an $\mathbb{F}$-additive code with an encoding $\mathsf{Enc}_{\mathcal{C}} \colon \mathbb{F}^\ell \times \mathbb{F}^r \to \Sigma^m$. We use the relation in Equation 1 with an explicit coefficient vector:

$$\bar{\mathcal{R}}_{\mathcal{C}} \coloneqq \left\{ \begin{pmatrix} \mathbb{x} = (\mu, \boldsymbol{v}) \in \mathbb{F} \times \mathbb{F}^\ell, \\ \mathbb{y} = f \in \Sigma^m, \\ \mathbb{w} = (\boldsymbol{f}, \boldsymbol{r}) \in \mathbb{F}^\ell \times \mathbb{F}^r \end{pmatrix} \;\middle|\; \begin{array}{l} f = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r}) \\ \wedge \langle \boldsymbol{f}, \boldsymbol{v} \rangle = \mu \end{array} \right\}. \tag{2}$$

Recall: HVZK for IOPPs. We recall the standard definition [BCFGRS17], in which the simulator, like the verifier, has query access to the implicit instance.

Definition 2.1 (informal)**.** *Let* $\mathsf{IOP} = (\mathbf{P}, \mathbf{V})$ *be an IOPP for a ternary relation* $\mathcal{R}$*. The* **view** *of* $(\mathbf{P}, \mathbf{V})$ *on* $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$ *is the random variable* $\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathbb{x}, \mathbb{y}, \mathbb{w})$ *that consists of* $\mathbf{V}$*'s randomness and the answers to* $\mathbf{V}$*'s queries in an interaction between* $\mathbf{P}(\mathbb{x}, \mathbb{y}, \mathbb{w})$ *and* $\mathbf{V}^{\mathbb{y}}(\mathbb{x})$*.* IOP *has* **honest-verifier zero-knowledge with error** ζ **and query complexity** b *if there exists a polynomial-time simulator* $\mathbf{S}$ *such that the random variables* $\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathbb{x}, \mathbb{y}, \mathbb{w})$ *and* $\mathbf{S}^{\mathbb{y}}(\mathbb{x})$ *are* ζ*-close (in statistical distance) and* $\mathbf{S}^{\mathbb{y}}(\mathbb{x})$ *makes at most* b *queries to* $\mathbb{y}$*.*

A simple protocol for linear constraints. We sketch a *non-succinct* HVZK IOPP for $\bar{\mathcal{R}}_{\mathcal{C}}$.

Construction 2.2. The inputs are as in Equation 2.

1. The prover samples $\boldsymbol{g} \in \mathbb{F}^\ell$ and $\boldsymbol{r'} \in \mathbb{F}^r$, and sends the oracle $g \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{g}, \boldsymbol{r'}) \in \Sigma^m$ and the message $\tilde{\mu} \coloneqq \langle \boldsymbol{g}, \boldsymbol{v} \rangle \in \mathbb{F}$.
2. The verifier samples and sends $\gamma \leftarrow \mathbb{F}$.
3. The prover sends the messages $\boldsymbol{f}^* \coloneqq \boldsymbol{g} + \boldsymbol{f} \cdot \gamma \in \mathbb{F}^\ell$ and $\boldsymbol{r}^* \coloneqq \boldsymbol{r'} + \boldsymbol{r} \cdot \gamma \in \mathbb{F}^r$.
4. The verifier samples and sends $x_1, \ldots, x_t \leftarrow [m]$. It computes $f^* \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*) \in \Sigma^m$ and checks that:
   - $\langle \boldsymbol{f}^*, \boldsymbol{v} \rangle = \tilde{\mu} + \mu \cdot \gamma$.
   - for $i \in [t]$, $f^*(x_i) = g(x_i) + f(x_i) \cdot \gamma$ (by querying f and g at $x_1, \ldots, x_t$).

Completeness follows by linearity of $\mathsf{Enc}_{\mathcal{C}}$. For straightline round-by-round knowledge soundness, we use MCA for the random combination, a union bound over the interleaved decoding list for the linear constraint, and a final spot-check bound of $(1-\delta)^t$. This analysis is similar to [BCFW25]. The generalized proof is in Section 7.

Lemma 2.3. *For every* $\delta \in (0, \delta(\mathcal{C}))$*, Construction 2.2 has round-by-round knowledge soundness with errors* $(\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) + \frac{\Lambda(\mathcal{C}^{=2}, \delta)}{|\mathbb{F}|},\ (1-\delta)^t)$*. It is a perfect honest-verifier zero-knowledge IOPP for* $\bar{\mathcal{R}}_{\mathcal{C}}$ *with query complexity* $\mathsf{b} = t$*.*

The simulator samples the verifier's challenge γ and query locations as in the protocol, and samples $\boldsymbol{f}^*$ and $\boldsymbol{r}^*$ independently and uniformly. It sets $\tilde{\mu} = \langle \boldsymbol{f}^*, \boldsymbol{v} \rangle - \mu \cdot \gamma$ and answers the query to g at x_i by $g(x_i) = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*)(x_i) - f(x_i) \cdot \gamma$. By linearity, this is exactly the honest joint distribution, using at most t queries to f. No zero-knowledge property of the encoding is needed.

Takeaway. The $2\times$ oracle communication overhead comes from sending g of the same size as f.

2.3 Composition for honest-verifier zero-knowledge IORs

Naively adapting prior HVZK definitions for IOPs (and IOPPs) to IORs does not ensure zero-knowledge under composition. We strengthen the definition to account for later queries and prove composition under this definition.

Definition 2.1 is insufficient for IORs. For a concrete counterexample, let $\mathcal{R} = \{(0,0),(0,1)\}$ be a binary relation of instance-witness pairs $(\mathbb{x}, \mathbb{w})$, so that the instance 0 has two distinct witnesses. The target relation is $\mathcal{R}' \coloneqq \{(\mathbb{x}', \mathbb{y}', \mathbb{w}' = \bot) : (\mathbb{x}', \mathbb{y}') \in \mathcal{R}\}$.

- In IOR_1, the prover sends $\mathbb{w}$ as an oracle. The verifier makes no queries and outputs $(\mathbb{x}', \mathbb{y}') \coloneqq (\mathbb{x}, \mathbb{w})$. This IOR from $\mathcal{R}$ to $\mathcal{R}'$ has perfect completeness and soundness, and satisfies Definition 2.1: the verifier's view is empty.
- In IOR_2, the prover does nothing. The verifier queries $\mathbb{y}'$ entirely and checks that $(\mathbb{x}', \mathbb{y}') \in \mathcal{R}$. This IOPP for $\mathcal{R}'$ has perfect completeness and soundness, and satisfies Definition 2.1: the simulator can query the entire implicit instance.

Their composition reveals whichever witness the prover receives. A simulator given only the instance 0 cannot reproduce both resulting views with statistical error less than $1/2$, so the composition cannot be HVZK. Definition 2.1 imposes no requirement on subsequent queries to the implicit instance output by IOR_1, and this leakage breaks zero-knowledge of the composition.

Defining HVZK for IORs. We strengthen Definition 2.1 by requiring the simulator to also simulate future queries to $\mathbb{y}'$, modeled by an oracle algorithm $\mathcal{D}$ called a *distinguisher*.

Definition 2.4 (informal)**.** *Let* $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ *be an IOR from* $\mathcal{R}$ *to* $\mathcal{R}'$*. The* **view** *of* IOR *on* $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$ *with respect to an oracle algorithm* $\mathcal{D}$ *is the random variable* $\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}, \mathbb{x}, \mathbb{y}, \mathbb{w})$ *that consists of* $\mathbf{V}$*'s randomness, the ordered queries of* $\mathbf{V}$ *and* $\mathcal{D}$ *to the input and proof oracles with their answers, and* $\mathcal{D}$*'s output* out *in this experiment:* $\mathbf{P}(\mathbb{x}, \mathbb{y}, \mathbb{w})$ *and* $\mathbf{V}^{\mathbb{y}}(\mathbb{x})$ *interact yielding output* $(\mathbb{x}', \mathbb{y}', \mathbb{w}')$*, and then* $\mathcal{D}^{\mathbb{y}'}(\mathbb{x}')$ *is run to produce* out*. Here each query of* $\mathcal{D}$ *to* $\mathbb{y}'$ *is represented by the queries it induces to the input and proof oracles.* IOR *has* **honest-verifier zero-knowledge with error** ζ **and query complexity** b **with respect to** $\mathcal{D}$ *if there exists a polynomial time simulator* $\mathbf{S}$ *such that the random variables* $\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}, \mathbb{x}, \mathbb{y}, \mathbb{w})$ *and* $\mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$ *are* ζ*-close (in statistical distance),* $\mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$ *makes at most* b *queries to* $\mathbb{y}$*, and the input-query transcript in its output consists of its actual ordered queries to* $\mathbb{y}$ *and their answers.*

The simulator $\mathbf{S}$ has black-box access to $\mathcal{D}$. It runs $\mathcal{D}$ on the simulated explicit output and simulates the answers to its oracle queries.

HVZK composition theorem. We informally state a composition theorem for HVZK IORs.

Lemma 2.5 (informal)**.** *Let* IOR_1 *be an IOR from* $\mathcal{R}_1$ *to* $\mathcal{R}_2$ *and* IOR_2 *be an IOR from* $\mathcal{R}_2$ *to* $\mathcal{R}_3$*, and let* $\mathsf{IOR} \coloneqq \mathsf{IOR}_1 \circ \mathsf{IOR}_2$ *be the IOR from* $\mathcal{R}_1$ *to* $\mathcal{R}_3$ *obtained by composing these. Suppose that:*

- IOR_2 *is HVZK with error* ζ_2 *and query complexity* b_2 *with respect to* $\mathcal{D}_2$*, with simulator* $\mathbf{S}_2$*.*
- IOR_1 *is HVZK with error* ζ_1 *and query complexity* b_1 *with respect to* $\mathcal{D}_1^{\mathbb{y}}(\mathbb{x}) \coloneqq \mathbf{S}_2^{\mathbb{y}, \mathcal{D}_2}(\mathbb{x})$*.*

Then IOR *is HVZK with error* $\zeta_1 + \zeta_2$ *and query complexity* b_1 *with respect to* $\mathcal{D}_2$*.*

Proof sketch. Let $\mathbf{S}_1$ be the simulator for IOR_1 and define $\mathbf{S}^{\mathbb{y}_1}(\mathbb{x}_1) \coloneqq \mathbf{S}_1^{\mathbb{y}_1, \mathcal{D}_1}(\mathbb{x}_1)$. By the honest-verifier zero-knowledge of IOR_1 with respect to $\mathcal{D}_1$, the distribution of $\mathbf{S}^{\mathbb{y}_1}(\mathbb{x}_1)$ has statistical distance at most ζ_1 from the experiment in which $(\mathbf{P}_1, \mathbf{V}_1)$ interact on $(\mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$, outputting $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$,

and then $\mathcal{D}_1^{\mathbb{y}_2}(\mathbb{x}_2)$ is run. Since $\mathcal{D}_1^{\mathbb{y}_2}(\mathbb{x}_2) = \mathbf{S}_2^{\mathbb{y}_2,\mathcal{D}_2}(\mathbb{x}_2)$, by honest-verifier zero-knowledge of IOR_2 with respect to $\mathcal{D}_2$, this latter part has statistical distance at most ζ_2 from running $(\mathbf{P}_2, \mathbf{V}_2)$ on $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$ to obtain $(\mathbb{x}_3, \mathbb{y}_3, \mathbb{w}_3)$ and then running $\mathcal{D}_2^{\mathbb{y}_3}(\mathbb{x}_3)$. Thus, the distribution of $\mathbf{S}^{\mathbb{y}_1}(\mathbb{x}_1)$ has statistical distance at most $\zeta_1 + \zeta_2$ from an honest execution of IOR followed by $\mathcal{D}_2$. □

Distinguisher classes in our constructions. We generalize Definition 2.4 to hold against distinguishers $\mathcal{D} \in \mathfrak{D}$ for a "distinguisher class" $\mathfrak{D}$. This allows us to apply Lemma 2.5 more flexibly. Consider for example the class $\mathfrak{D}^{\leq t}$ of uniform-query distinguishers, which make at most t queries to the implicit instance, at independent uniform locations (Definition 4.9). Suppose that IOR_2 is HVZK for $\mathfrak{D}^{\leq t}$ with a simulator $\mathbf{S}_2$ that makes at most b_2 queries, also at independent uniform locations. For every $\mathcal{D}_2 \in \mathfrak{D}^{\leq t}$, the algorithm $\mathcal{D}_1^{\mathbb{y}}(\mathbb{x}) \coloneqq \mathbf{S}_2^{\mathbb{y},\mathcal{D}_2}(\mathbb{x})$ makes at most b_2 queries and thus belongs to $\mathfrak{D}^{\leq \mathsf{b}_2}$ by Claim 4.11. We can therefore apply Lemma 2.5 if IOR_1 is HVZK for $\mathfrak{D}^{\leq \mathsf{b}_2}$. In our protocols the (honest) verifier samples query locations uniformly, and our encodings are only required to be zero-knowledge for such queries.

Fresh inputs. A zero-knowledge encoding hides its message only if its randomness is uniform. In our sumcheck reduction, the masks that hide the prover's messages are moreover part of the randomness of the input oracle (Section 2.4), so the reduction is zero-knowledge only if this randomness is uniform. An input is *fresh* if its oracle is an encoding of the message with uniform randomness (for an interleaved codeword, the randomness consists of the mask $\boldsymbol{\sigma}$ of Section 2.5 and the seeds of its rows), and an IOR is HVZK *on fresh inputs* if Definition 2.4 holds for fresh inputs, where the two distributions are also over the randomness of the input.

For this notion to compose, the output of the first reduction must be a fresh input of the second. An IOR *has fresh output* if its prover sends the output oracle and the randomness of this oracle is uniform given the view of the verifier and the output message. If IOR_1 has fresh output, then Lemma 2.5 holds with HVZK on fresh inputs in place of HVZK, and if IOR_2 is HVZK for every input, the condition on IOR_1 is not needed (Theorem 4.7). Our sumcheck reduction is HVZK on fresh inputs, and our code switch has fresh output and is HVZK for every input.
The formal definition and composition theorem appear in Section 4.

2.4 A "committed" relation for zero-knowledge

Adapting the relation for successive reductions. Reading the explicit coefficient vector $\boldsymbol{v} \in \mathbb{F}^\ell$ takes $\Omega(\ell)$ time. While the warmup treats μ as public, revealing new targets during reductions may leak information about $\boldsymbol{f}$ even if $\mathsf{Enc}_{\mathcal{C}}$ is zero-knowledge. We address these limitations with a relation similar to those in [ACFY25; BCFW25].

Modification 1: succinct linear constraints. We replace $\boldsymbol{v}$ in $\mathbb{x}$ with a succinct state st and fix in the relation a map sl from states to coefficient vectors in $\mathbb{F}^\ell$, which we call a *succinct linear form*. The resulting constraint is $\langle \boldsymbol{f}, \mathsf{sl}(\mathsf{st}) \rangle = \mu$. For example, sl can map the short state $\mathsf{st} = \alpha \in \mathbb{F}$ to the long vector $(\alpha^{i-1})_{i \in [\ell]}$. We use these forms to specify constraint coefficients succinctly, as with the sumcheck-based constraints in [ACFY25].

Modification 2: masks in the encoding. In a sumcheck, the prover's messages and the final target reveal information about the message of the oracle. To hide them, the prover needs a random mask that the verifier does not learn and that later reductions can still constrain. We use part of the randomness of the encoding of the input as this mask, and for this we require the encoding to have *spare coordinates*. An encoding $\mathsf{Enc}_{\mathcal{C}} \colon \mathbb{F}^\ell \times \mathbb{F}^{p+r} \to \Sigma^m$ has p spare coordinates if it remains

a t-query zero-knowledge encoding when the first p coordinates of its randomness are regarded as part of the message (Definition 3.25). Its randomness consists of a *mask* $\boldsymbol{s} \in \mathbb{F}^p$ and a *seed* $\boldsymbol{r} \in \mathbb{F}^r$. The code $\mathcal{C}$ with the encoding that regards the mask as part of the message is denoted $\mathcal{C}^{+p}$,

$$\mathsf{Enc}_{\mathcal{C}}^{+p}((\boldsymbol{f}, \boldsymbol{s}), \boldsymbol{r}) \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, (\boldsymbol{s}, \boldsymbol{r}))\,.$$

For example, a Reed–Solomon encoding with $t + p$ random coefficients is a t-query zero-knowledge encoding with p spare coordinates.

Committed sumcheck relation. We combine the two modifications in the following relation, for a code $\mathcal{C}$ with an encoding $\mathsf{Enc}_{\mathcal{C}}$.

$$\mathcal{R}_{\mathcal{C},\mathsf{sl}} \coloneqq \left\{ \begin{pmatrix} \mathbb{x} = (\mu, \mathsf{st}), \\ \mathbb{y} = f, \\ \mathbb{w} = (\boldsymbol{f}, \boldsymbol{r}) \end{pmatrix} \;\middle|\; \begin{array}{l} f = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r}) \\ \wedge\ \langle \boldsymbol{f}, \mathsf{sl}(\mathsf{st}) \rangle = \mu \end{array} \right\}. \tag{3}$$

We call this a **committed sumcheck relation**: the oracle f commits the prover to its message and, through the randomness of the encoding, to the mask of the next sumcheck. The relation depends on the encoding. If the encoding has spare coordinates, $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ imposes no constraints on the mask, while $\mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}'}$, for a linear form sl' on the message and the mask, also constrains the mask.

The verifiers of our reductions never compute the coefficient vector $\mathsf{sl}(\mathsf{st})$ itself. After each sumcheck, the constraint is on a fold of this vector, and only the verifier of the final test evaluates the fold, from the succinct state. For the linear forms of our Reed–Solomon instantiation, this evaluation takes polylogarithmic time.

2.5 An honest-verifier zero-knowledge sumcheck reduction

We present an HVZK sumcheck *reduction* for the relation in Equation 3. In it, the prover and the verifier reduce a linear constraint on the message of an interleaved codeword to a linear constraint on the message of its fold. The prover hides its sumcheck messages with a polynomial determined by the masks of the rows, that is, by the spare coordinates of their randomness.

Code interleaving and its folding. Code interleaving is defined in Section 1. We specify its zero-knowledge encoding for $K = 2^k$ and the corresponding folding operation. For simplicity, we assume power-of-two message lengths in this subsection; in general, only the number of rows must be a power of two (Section 6). Below, eq is the k-variable equality polynomial

$$\mathsf{eq}(\mathtt{X}_1, \ldots, \mathtt{X}_k, \mathtt{Y}_1, \ldots, \mathtt{Y}_k) \coloneqq \prod_{i=1}^{k} (\mathtt{X}_i \cdot \mathtt{Y}_i + (1 - \mathtt{X}_i) \cdot (1 - \mathtt{Y}_i))\,.$$

Definition 2.6. *For a code $\mathcal{C} \subseteq \Sigma^m$ with a zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon \mathbb{F}^\ell \times \mathbb{F}^r \to \Sigma^m$ and $k \in \mathbb{N}$, its* 2^k**-interleaving** *is the code $\mathcal{C}^{\equiv 2^k} \subseteq (\Sigma^{2^k})^m$ with the following zero-knowledge encoding. We split a message $\boldsymbol{f} \in \mathbb{F}^{2^k \ell}$ and randomness $\boldsymbol{r} \in \mathbb{F}^{2^k r}$ into 2^k blocks $(\boldsymbol{f_b})_{\boldsymbol{b} \in \{0,1\}^k}$ and $(\boldsymbol{r_b})_{\boldsymbol{b} \in \{0,1\}^k}$, indexed in binary order. The multilinear extension of the message satisfies*

$$\hat{\boldsymbol{f}}(\mathtt{X}_1, \ldots, \mathtt{X}_{k+\log \ell}) = \sum_{\boldsymbol{b} \in \{0,1\}^k} \mathsf{eq}(\boldsymbol{b}, (\mathtt{X}_1, \ldots, \mathtt{X}_k)) \cdot \hat{\boldsymbol{f}}_{\boldsymbol{b}}(\mathtt{X}_{k+1}, \ldots, \mathtt{X}_{k+\log \ell})\,. \tag{4}$$

The interleaved encoding is

$$\forall\, z \in [m],\ \mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}(\boldsymbol{f}, \boldsymbol{r})(z) \coloneqq (\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f_b}, \boldsymbol{r_b})(z))_{\boldsymbol{b} \in \{0,1\}^k} \in \Sigma^{2^k}\,.$$

The 2^k blocks are the *rows* of the interleaving. If the encoding of $\mathcal{C}$ has p spare coordinates, then each row has a mask and a seed, and the masks of the rows form a table $\boldsymbol{s} = (\boldsymbol{s_b})_{\boldsymbol{b}\in\{0,1\}^k} \in (\mathbb{F}^p)^{2^k}$.

We extend to zero-knowledge encodings the folding operations for arbitrary linear codes in [ACFY25; NA25; BCFW25].

Definition 2.7. *The* **interleaved fold** *of a function $f\colon [m] \to \Sigma^{2^k}$ at the point $\boldsymbol{\gamma} \in \mathbb{F}^i$ with $i \in [k]$ is the function $\mathsf{Fold}(f,\boldsymbol{\gamma})\colon [m] \to \Sigma^{2^{k-i}}$ defined as follows. We identify $\{0,1\}^k \cong \{0,1\}^i \times \{0,1\}^{k-i}$, and for every $z \in [m]$ we write $f(z) = \left(f_{(\boldsymbol{b},\boldsymbol{c})}(z)\right)_{\boldsymbol{b}\in\{0,1\}^i,\boldsymbol{c}\in\{0,1\}^{k-i}}$. The fold is defined coordinatewise by $\mathsf{Fold}(f,\boldsymbol{\gamma})(z) \in \Sigma^{2^{k-i}}$ such that*

$$\forall\, \boldsymbol{c} \in \{0,1\}^{k-i},\ (\mathsf{Fold}(f,\boldsymbol{\gamma})(z))_{\boldsymbol{c}} \coloneqq \sum_{\boldsymbol{b}\in\{0,1\}^i} \mathsf{eq}(\boldsymbol{b},\boldsymbol{\gamma}) \cdot f_{(\boldsymbol{b},\boldsymbol{c})}(z)\,.$$

The expressions are well-defined because Σ is an $\mathbb{F}$-linear vector space.

The key feature of the above definitions is that "encode then fold" equals "fold then encode".

Claim 2.8 (informal)**.** *For every $\boldsymbol{f} \in \mathbb{F}^{2^k\ell}$, $\boldsymbol{r} \in \mathbb{F}^{2^k r}$, $i \in [k]$, and $\boldsymbol{\gamma} \in \mathbb{F}^i$,*

$$\mathsf{Fold}\left(\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}\left(\boldsymbol{f},\boldsymbol{r}\right),\boldsymbol{\gamma}\right) = \mathsf{Enc}_{\mathcal{C}^{\equiv 2^{k-i}}}\left(\mathsf{Fold}(\boldsymbol{f},\boldsymbol{\gamma}),\mathsf{Fold}(\boldsymbol{r},\boldsymbol{\gamma})\right)\,.$$

The mask polynomial. A sumcheck protocol reveals partial sums, over the Boolean hypercube, of the polynomial being summed. For zero-knowledge, these partial sums must not reveal information about the message. As in prior work [CFS17], we accomplish this by masking the sumcheck claim via a random polynomial. We are interested in running a sumcheck protocol over a multivariate polynomial of individual degree 2. [CFS17] and other prior work would mask this protocol by sampling a uniform polynomial of the same size, unfortunately resulting in a $2\times$ overhead. Instead, as noticed by [XZZPS19], since the values revealed by the sumcheck protocol are structured, the full masking is not required, and the masking polynomial can be derandomized. In that work, they chose the masking polynomial to be $s_1(\mathtt{X}_1) + \cdots + s_k(\mathtt{X}_k)$ for random univariate polynomials s_i of degree ≥ 2. While that suffices over most fields, whenever $\mathrm{char}(\mathbb{F}) = 2$ the partial sums of that polynomial over the hypercube vanish in all but the last two rounds, so the corresponding prover messages are unmasked. The case $\mathrm{char}(\mathbb{F}) = 2$ is particularly interesting as it allows for exceptionally fast provers [BRW26].

We use a different polynomial, which is described by $(k+1)(d_s-1)$ field elements and hides the sumcheck messages over every finite field (and, as Lemma 6.2 shows, over finite rings and modules). Let $d_s \geq 3$ be a masking degree and let the encoding of $\mathcal{C}$ have $p = d_s - 1$ spare coordinates. For $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_0,\ldots,\boldsymbol{\sigma}_k) \in (\mathbb{F}^p)^{k+1}$ and $e \in \{0,\ldots,d_s-2\}$, let $\hat{s}_e(\mathtt{X}_1,\ldots,\mathtt{X}_k) \coloneqq \sum_{i=0}^{k} \boldsymbol{\sigma}_{i,e} \cdot \mathtt{X}_{i+1}\cdots\mathtt{X}_k$. The *mask polynomial* of $\boldsymbol{\sigma}$ is

$$\mathcal{U}^{\boldsymbol{\sigma}}(\mathtt{X}_1,\ldots,\mathtt{X}_k) \coloneqq \hat{s}_0(\mathtt{X}_1,\ldots,\mathtt{X}_k) + \sum_{e=1}^{d_s-2} \hat{s}_e(\mathtt{X}_1,\ldots,\mathtt{X}_k)\cdot(\mathtt{X}_1^e + \cdots + \mathtt{X}_k^e)\,.$$

For $d_s = 3$, the tuple $\boldsymbol{\sigma}$ has $2(k+1)$ coordinates and $\mathcal{U}^{\boldsymbol{\sigma}}(\mathtt{X}_1,\ldots,\mathtt{X}_k) = \hat{s}_0(\mathtt{X}_1,\ldots,\mathtt{X}_k) + \hat{s}_1(\mathtt{X}_1,\ldots,\mathtt{X}_k)\cdot(\mathtt{X}_1+\cdots+\mathtt{X}_k)$.

The rows of the input codeword share $\boldsymbol{\sigma}$: the mask of the row $\boldsymbol{b} \in \{0,1\}^k$ is $\boldsymbol{s_b} \coloneqq (\hat{s}_e(\boldsymbol{b}))_e$, so that $\hat{s}_e$ is the multilinear extension of the e-th mask coordinate across the rows. The prover that

encodes the oracle therefore samples only the $(k+1)(d_s-1)$ coordinates of $\boldsymbol{\sigma}$, and not one mask per row. In our reduction, the prover runs the sumcheck on the sum of $\mathcal{U}^{\boldsymbol{\sigma}}$ and a random multiple of the original polynomial (Construction 2.9). The mask polynomial has individual degree less than d_s and two properties that we use (Lemma 6.2).

The first property concerns its value at the point $\boldsymbol{\gamma}$ of the verifier's challenges:

$$\mathcal{U}^{\boldsymbol{\sigma}}(\boldsymbol{\gamma}) = \sum_{e=0}^{d_s-2} \mathsf{Fold}(\boldsymbol{s},\boldsymbol{\gamma})_e \cdot \mathsf{pow}_e(\boldsymbol{\gamma})\,,$$

where $\mathsf{pow}_0 := 1$ and $\mathsf{pow}_e(\boldsymbol{\gamma}) := \gamma_1^e + \cdots + \gamma_k^e$ for $e \geq 1$. The vector $\mathsf{Fold}(\boldsymbol{s},\boldsymbol{\gamma})$ is the mask of the fold of the input codeword (Claim 2.8), and the verifier can compute the coefficients $\mathsf{pow}_e(\boldsymbol{\gamma})$. The final claim of the sumcheck, which involves $\mathcal{U}^{\boldsymbol{\sigma}}(\boldsymbol{\gamma})$, is therefore a linear constraint on the message of the folded codeword under $\mathcal{C}^{+p}$. The prover does not reveal $\mathcal{U}^{\boldsymbol{\sigma}}(\boldsymbol{\gamma})$; the constraint becomes part of the output relation.

The second property concerns the messages of the sumcheck for $\mathcal{U}^{\boldsymbol{\sigma}}$ alone. For every choice of the verifier's challenges, if $\boldsymbol{\sigma}$ is uniform, then the claimed sum and the round polynomials are uniformly distributed among the tuples that pass the consistency checks of the sumcheck verifier. This holds over every finite field, including fields of characteristic two. A tuple that passes the checks is determined by the claimed sum and by d_s-1 coefficients of each round polynomial, because each check determines the linear coefficient. These $k(d_s-1)+1$ field elements are a surjective linear function of the $(k+1)(d_s-1)$ coordinates of $\boldsymbol{\sigma}$, and a surjective linear function of a uniform vector is uniform.

An HVZK committed-sumcheck IOR. We outline a queryless IOR from $\mathcal{R}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}$ to $\mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}'}$, where $\mathcal{C}$ is an $\mathbb{F}$-additive code whose encoding $\mathsf{Enc}_{\mathcal{C}}\colon \mathbb{F}^{\ell} \times \mathbb{F}^{p+r} \to \Sigma^m$ has $p = d_s - 1$ spare coordinates, $k \geq 1$, and $d_s \geq 3$. Each round fixes one interleaving variable. After k rounds, the output oracle is the fold of the input oracle, a virtual oracle. Figure 2 shows the rows of the input oracle and the row that results from folding them.

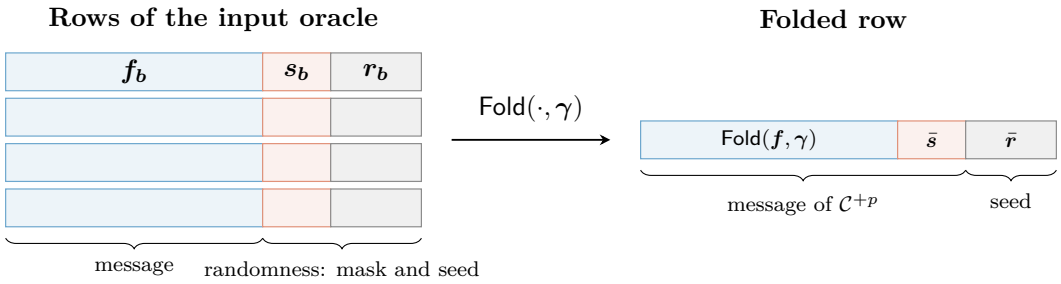

Figure 2: The reduction in Construction 2.9, pictorially. Every row of the input oracle encodes a message under a mask and a seed. The masks are determined by the tuple $\boldsymbol{\sigma}$, whose mask polynomial hides the sumcheck messages. The output relation counts the folded mask $\bar{\boldsymbol{s}}$ as part of the message and constrains it.

Construction 2.9. The verifier receives the explicit input $\mathbb{x} := (\mu, \mathsf{st})$ and oracle access to $\mathbb{y} := f$. In the honest case, the prover receives a witness $\mathbb{w} = (\boldsymbol{f}, (\boldsymbol{\sigma}, \boldsymbol{r}))$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}$, where the tuple $\boldsymbol{\sigma}$ determines the table $\boldsymbol{s}$ of the masks of the rows as above, and $\boldsymbol{r}$ consists of their seeds.
1. The prover sends $\tilde{\mu} \in \mathbb{F}$. The honest prover sets $\tilde{\mu} := \sum_{\boldsymbol{b}\in\{0,1\}^k} \mathcal{U}^{\boldsymbol{\sigma}}(\boldsymbol{b})$.

2. The verifier samples and sends $\varepsilon \leftarrow \mathbb{F}$.
3. The prover and the verifier run a k-round sumcheck protocol on the claim

$$\sum_{\boldsymbol{b}\in\{0,1\}^k} \left(\mathcal{U}^{\boldsymbol{\sigma}}(\boldsymbol{b}) + \varepsilon \cdot \widehat{G}(\boldsymbol{b})\right) = \tilde{\mu} + \varepsilon \cdot \mu$$

where

$$\widehat{G}(\mathtt{X}_1, \ldots, \mathtt{X}_k) \coloneqq \sum_{\boldsymbol{a}\in\{0,1\}^{\log \ell}} \hat{\boldsymbol{f}}(\mathtt{X}_1, \ldots, \mathtt{X}_k, \boldsymbol{a}) \cdot \mathsf{Fold}(\mathsf{sl}(\mathsf{st}), \mathtt{X}_1, \ldots, \mathtt{X}_k)[\boldsymbol{a}]$$

and Fold is defined as in Definition 2.7, viewing $\mathsf{sl}(\mathsf{st})$ as indexed by $(\boldsymbol{b}, \boldsymbol{a}) \in \{0,1\}^k \times \{0,1\}^{\log \ell}$. In round $j \in [k]$:
   (a) the prover sends $\hat{h}_j \in \mathbb{F}^{<d_s}[\mathtt{X}]$, and
   (b) the verifier checks $\hat{h}_1(0) + \hat{h}_1(1) = \tilde{\mu} + \varepsilon \cdot \mu$ when $j = 1$, and $\hat{h}_j(0) + \hat{h}_j(1) = \hat{h}_{j-1}(\gamma_{j-1})$ otherwise, rejecting if the check fails. The verifier then sends $\gamma_j \leftarrow \mathbb{F}$.

   Let $\boldsymbol{\gamma} \coloneqq (\gamma_1, \ldots, \gamma_k)$.
4. The verifier sets $\mu' \coloneqq \hat{h}_k(\gamma_k)$ and outputs a new instance for the target relation $\mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}'}$ with the joint constraint:

$$\langle \mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}), \varepsilon \cdot \mathsf{Fold}(\mathsf{sl}(\mathsf{st}), \boldsymbol{\gamma})\rangle + \sum_{e=0}^{d_s-2} \mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma})_e \cdot \mathsf{pow}_e(\boldsymbol{\gamma}) = \mu' \,.$$

   The new linear form sl' maps $(\mathsf{st}, \boldsymbol{\gamma}, \varepsilon)$ to $\varepsilon \cdot \mathsf{Fold}(\mathsf{sl}(\mathsf{st}), \boldsymbol{\gamma})$ on the folded message and to $(\mathsf{pow}_e(\boldsymbol{\gamma}))_e$ on the folded mask. The output implicit instance is $\mathbb{y}' \coloneqq \mathsf{Fold}(f, \boldsymbol{\gamma})$. A query to $\mathbb{y}'$ at x is answered by querying the column $f(x) \in \Sigma^{2^k}$ and folding it. In the honest case,

$$\mathbb{w}' \coloneqq ((\mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}), \mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma})), \mathsf{Fold}(\boldsymbol{r}, \boldsymbol{\gamma})) \,.$$

Completeness. Completeness follows from Claim 2.8: the folded oracle is the encoding under $\mathcal{C}^{+p}$ of the folded message and mask with the folded seed. By Equation 4,

$$\mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma})[a] = \hat{\boldsymbol{f}}(\boldsymbol{\gamma}, \mathsf{binary}(a)) \,,$$

where $\mathsf{binary}(a)$ is the representation of $a - 1$ in exactly $\log_2 \ell$ bits for $a \in [\ell]$. The same identity holds for the folding of $\mathsf{sl}(\mathsf{st})$ and of each mask coordinate, which gives the joint constraint above.

Honest-verifier zero-knowledge. We sketch why Construction 2.9 is HVZK when its input is *fresh*, that is, when the mask $\boldsymbol{\sigma}$ and the seeds of the input oracle are uniform (Section 2.3).

Lemma 2.10. *Let $t \in \mathbb{N}$. If $\mathsf{Enc}_{\mathcal{C}}$ is a t-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}}$, and $d_s \geq 3$, then Construction 2.9 is honest-verifier zero-knowledge on fresh inputs with error at most $2^k \cdot \zeta_{\mathcal{C}}$ for the class $\mathfrak{D}^{\leq t}$ of uniform-query distinguishers, which make at most t queries to $\mathbb{y}'$. The simulator is uniform-query and makes at most t column queries to $\mathbb{y}$.*

Proof sketch. On a fresh input the mask $\boldsymbol{\sigma}$ is uniform, so the mask polynomial acts as a one-time pad for the sumcheck messages. Its hypercube sum is uniform, and in each round it hides everything about the prover's polynomial except the sum $\hat{h}_j(0) + \hat{h}_j(1)$ that the verifier checks (Lemma 6.2). Thus the simulator can sample $\tilde{\mu}$ uniformly and, round by round, sample a uniform polynomial

satisfying that check, using the honest verifier's random challenges. These checks depend only on the public target and the preceding messages, so no witness is needed.

It remains to explain why queries to the output oracle do not undo this masking. The output oracle is a fold of the input oracle: the simulator queries the same input column and folds the answer itself. On a fresh input the seeds are uniform as well, and the encoding has spare coordinates. Hence t columns of the input oracle are independent of the message *and of the masks*, up to an error of $\zeta_{\mathcal{C}}$ per row, and therefore of the sumcheck messages. The simulator thus samples the sumcheck messages independently of the columns that it reads. The proof uses that the encoding is linear to bound the resulting error by $2^k \cdot \zeta_{\mathcal{C}}$, the error of the encoding of the 2^k rows. □

Round-by-round knowledge soundness. We sketch the straightline RBR knowledge soundness analysis [BCFW25]. As in [BMNW25; BCFW25], the "relaxed" relation for a proximity parameter δ is

$$\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}} := \left\{ \begin{pmatrix} \mathbb{x}, \\ \mathbb{y} = f, \\ (\mathbb{w}, \bar{\mathbb{y}} := \bar{f}) \end{pmatrix} \;\middle|\; \begin{array}{l} \Delta(f, \bar{f}) \leq \delta \\ \wedge\ (\mathbb{x}, \bar{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}} \end{array} \right\}.$$

We use the *same* proximity parameter δ to relax the initial and target relations.

Lemma 2.11 (informal)**.** *For every $\delta \in (0, \delta(\mathcal{C}))$, Construction 2.9 has round-by-round knowledge soundness with relaxation*

$$\left(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}, \tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{+p},\mathsf{sl}'}\right).$$

The error is controlled by the mutual correlated agreement errors and list sizes of the interleavings of $\mathcal{C}$, and is negligible as long as $\mathbb{F}$ is large enough. See Lemma 6.6 for the precise bounds.

Proof sketch. The overall strategy resembles that in [ACFY25; BCFW25]. We assume that no MCA failure occurs. Then, in each round, every codeword close to the folded oracle is the fold of a codeword close to the original oracle. A candidate codeword determines its message and its masks, and hence the mask polynomial that the prover should have used. Taking a union bound over the candidate codewords, one can use the soundness analysis of the original sumcheck protocol. The union bound must be over codewords close to the input word, so list-decoding bounds apply. Bounding the lists of the 2^k rows of f separately would require considering up to $\Lambda(\mathcal{C}, \delta)^{2^k}$ tuples, increasing the required field size to untenable regimes. We instead use the list of the interleaved code. The interleaved list size at any radius below the base code's minimum distance is polynomially bounded in the base-code list size, with coefficients and degree independent of the interleaving factor [GGR11]. □

Comparisons. The protocols in [CFS17; BCL22; Dia25] use a "full mask" as large as the message. The zero-knowledge PCP constructions in [GOS24; GOS25] also use sumcheck masking. In [BCL22], the construction has constant soundness error and does not use out-of-domain techniques for list-decoding soundness.

[XZZPS19] describes a zero-knowledge sumcheck, based on polynomial commitment schemes, that uses a small mask for each round rather than a full mask at the start. Our mask is also small, but we require a different interface for our reduction: the mask is part of the input *oracle*, and its contribution is retained in the joint constraint (as opposed to it being committed separately and opened once). Opening the mask as in [XZZPS19] would reveal an evaluation of it, and thus the corresponding linear form of the message.

The tensor-code protocol of [RW24] is malicious-verifier zero-knowledge and has constant soundness error with a polynomial-time prover. Our interleaved-code protocol is honest-verifier zero-knowledge and supports the list-decoding regime. The full protocol and proofs appear in Section 6.

2.6 HVZK code switching

We describe our HVZK code-switching reduction: an HVZK IOR from $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ to $\mathcal{R}_{\mathcal{C}',\mathsf{sl}'}$, switching the code $\mathcal{C}$ to another code $\mathcal{C}'$. Both are arbitrary $\mathbb{F}$-additive codes with zero-knowledge encodings and may themselves be interleaved. We only require that a message of $\mathcal{C}'$ is long enough to contain a message of $\mathcal{C}$, its encoding randomness, and an OOD pad. In our protocols, $\mathcal{C}$ is the code of the output relation of a sumcheck, whose messages contain a folded mask, and the encoding of $\mathcal{C}'$ has spare coordinates, which are the masks of the next sumcheck. We address encoding randomness and private OOD sampling, extending previous code switches (e.g. [RR20; RW24; ACFY24; ACFY25; BCFRRZ25; BMMS26a; BCFW25; BFRW26]).

Code-switching without ZK. We sketch the main idea behind previous (non-zero-knowledge) code switches. Let $\mathsf{Enc}_{\mathcal{C}}$ and $\mathsf{Enc}_{\mathcal{C}'}$ denote deterministic injective encoders for this sketch. The verifier has access to an oracle f that, in the honest case, is the encoding under $\mathcal{C}$ of a message $\boldsymbol{f}$ satisfying a given predicate Φ. The prover sends g, allegedly encoding the same message $\boldsymbol{f}$ under $\mathcal{C}'$. The verifier samples $x_1, \ldots, x_t \leftarrow [m]$ and queries f at these locations. The prover and verifier then engage in a proximity protocol for the claim "g encodes a message $\boldsymbol{g}$ satisfying $\Phi(\boldsymbol{g}) = 1$ and $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{g})[x_i] = f(x_i)$ for every $i \in [t]$".

Code-switching with ZK encodings. When using a zero-knowledge encoding, the codeword $f = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r})$ depends on both the message and the randomness, so a consistency check must account for both. We therefore re-encode the message *together with* a new auxiliary message, which contains the old randomness. Appending these coordinates to a message table of size 2^d and padding the result to a Boolean cube would increase the table size for subsequent sumchecks to at least 2^{d+1}, giving at least a $2\times$ overhead. We avoid this overhead when $\mathcal{C}'$ is an interleaved code. The prover divides the auxiliary message into equal blocks and appends one block to the message of each row. A row may then have any length; only the number of rows must be a power of two. In the next sumcheck, the prover folds the auxiliary blocks together with the main message. The consistency check at a point $x \in [m]$ is

$$f(x) = \langle \boldsymbol{f}, \boldsymbol{G}^{\#}_{\mathcal{C}}[x] \rangle + \langle \boldsymbol{r}, \boldsymbol{G}^{\$}_{\mathcal{C}}[x] \rangle \,,$$

where $\boldsymbol{G}_{\mathcal{C}} = [\boldsymbol{G}^{\#}_{\mathcal{C}}, \boldsymbol{G}^{\$}_{\mathcal{C}}]$ is the linear map associated with $\mathsf{Enc}_{\mathcal{C}}$. This check is a linear constraint on the message of $\mathcal{C}'$.

OOD sampling without ZK. Efficient protocols in the list-decoding regime typically rely on out-of-domain (OOD) sampling [BGKS20]. We first consider the non-private template. Informally, after sending an oracle f, the prover must "choose" a single codeword in $\Lambda(\mathcal{C}, f, \delta)$, which can contain multiple codewords when $\delta > \delta(\mathcal{C})/2$. Following [BCFW25], we fix a function $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathbb{F}^{t_{\mathsf{ood}} \times \ell}$. After the prover sends f, the verifier samples $\rho_{\mathsf{ood}} \leftarrow D_{\mathsf{ood}}$ and the prover sends $\boldsymbol{y} \in \mathbb{F}^{t_{\mathsf{ood}}}$, set to $\mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{f}$ in the honest case. The parties add the constraint $\mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{u} = \boldsymbol{y}$ for a candidate $u = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{u}) \in \Lambda(\mathcal{C}, f, \delta)$. Except with some small collision error, at most one candidate satisfies this constraint. However, the value $\boldsymbol{y}$ can (and typically does) leak information about $\boldsymbol{f}$.

OOD sampling with ZK. For private OOD sampling, we add fresh random coordinates. We use $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathbb{F}^{t_{\mathsf{ood}} \times (\ell + r_{\mathsf{ood}})}$, where r_{ood} is the length of an OOD pad. The revised OOD sample

is set as

$$\boldsymbol{y} \coloneqq \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{r}_{\mathsf{ood}} \end{bmatrix}, \tag{5}$$

where $\rho_{\mathsf{ood}} \leftarrow D_{\mathsf{ood}}$ and $\boldsymbol{r}_{\mathsf{ood}} \leftarrow \mathbb{F}^{r_{\mathsf{ood}}}$ are sampled independently. We require $\mathsf{ze}_{\mathsf{ood}}$ to satisfy two conditions: (i) $\mathsf{ze}_{\mathsf{ood}}$ must be suitable as an OOD sample, and (ii) $\mathsf{ze}_{\mathsf{ood}}$ must guarantee that $\boldsymbol{y}$ leaks no information on $\boldsymbol{f}$. The first condition, as shown in [BCFW25], is equivalent to $\mathsf{ze}_{\mathsf{ood}}$ being a zero-evader: for every $\boldsymbol{v} \neq \boldsymbol{0}$ it holds that $\Pr_{\rho_{\mathsf{ood}}}[\mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{v} = \boldsymbol{0}] \leq \varepsilon_{\mathsf{ood}}$. For the second condition, we introduce the notion of a $(r_{\mathsf{ood}}, \zeta_{\mathsf{ze}})$**-private zero-evader** for which $(\boldsymbol{y}, \rho_{\mathsf{ood}})$ in Equation 5 can be efficiently simulated from the public parameters without $\boldsymbol{f}$, with statistical error at most ζ_{ze}.

We show that every zero-evader can be made (perfectly) private with minor effect on the error.

Lemma 2.12. *Let $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathbb{F}^{t_{\mathsf{ood}} \times \ell}$ be a zero-evader with error $\varepsilon_{\mathsf{ood}}$. For every $r_{\mathsf{ood}} \geq t_{\mathsf{ood}}$ there exists a zero-evader $\mathsf{ze}'_{\mathsf{ood}} \colon D'_{\mathsf{ood}} \to \mathbb{F}^{t_{\mathsf{ood}} \times (\ell + r_{\mathsf{ood}})}$ with error $\approx \varepsilon_{\mathsf{ood}}$ that is $(r_{\mathsf{ood}}, 0)$-private.*

In the code switch, the input message and its encoding randomness are fixed. We append fresh random entries to the new auxiliary message to mask their OOD sample. The OOD sample is a function of the message of the new oracle alone, so it does not depend on the randomness of $\mathcal{C}'$, and hence not on the masks of the next sumcheck. Accordingly, we extend [BCFW25] by proving uniqueness of messages while allowing different encoding randomness.

Lemma 2.13. *Let ℓ' be the message length of $\mathcal{C}'$ and let $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathbb{F}^{t_{\mathsf{ood}} \times \ell'}$ be a zero-evader with error $\varepsilon_{\mathsf{ood}}$. For every g and δ', except with probability at most $\frac{1}{2}\Lambda(\mathcal{C}', \delta')^2 \cdot \varepsilon_{\mathsf{ood}}$ over $\rho_{\mathsf{ood}} \leftarrow D_{\mathsf{ood}}$, every two codewords in $\Lambda(\mathcal{C}', g, \delta')$ that encode distinct messages $\boldsymbol{u} \neq \boldsymbol{u}'$ satisfy $\mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{u} \neq \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{u}'$.*

This follows by a union bound over pairs of nearby codewords, as in Lemma 9.1 (which adapts the previous non-private version [BCFW25]). Each coordinate equation for the OOD sample is a linear constraint on the message encoded in g, and is thus compatible with the committed sumcheck relation.

Code-switching construction. We combine re-encoding, consistency checks on the message and the old randomness, and a private OOD sample in a single protocol, shown in Figure 3.

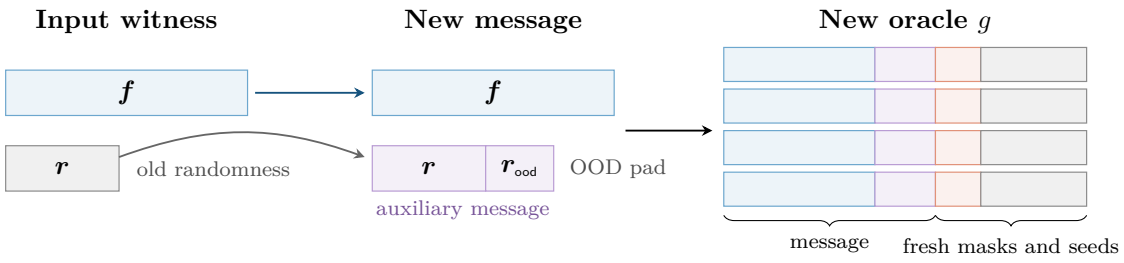

Figure 3: Code switching from $\mathcal{C}$ to an interleaved code $\mathcal{C}'$. The old randomness and an OOD pad form the auxiliary message, which is divided among the rows of the new oracle, where it follows the main message. The prover encodes the rows with fresh masks and seeds.

Construction 2.14. The verifier receives the explicit input $\mathbb{x} = (\mu, \mathsf{st})$ and oracle access to $\mathbb{y} = f$, and the prover receives $\mathbb{w} = (\boldsymbol{f}, \boldsymbol{r})$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}}$. Let r_{ood} be an OOD pad length, let $\mathcal{C}' \subseteq (\Sigma')^{m'}$ have an encoding $\mathsf{Enc}_{\mathcal{C}'} \colon \mathbb{F}^{\ell'} \times \mathbb{F}^{r'} \to (\Sigma')^{m'}$ with message length $\ell' \coloneqq \ell + r + r_{\mathsf{ood}}$, let

t and t_{ood} be the numbers of in-domain queries and of OOD samples, and let $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathbb{F}^{t_{\mathsf{ood}} \times \ell'}$ be a zero-evader.

1. The prover sends an oracle $g \in (\Sigma')^{m'}$. In the honest case, it independently samples $\boldsymbol{r'} \leftarrow \mathbb{F}^{r'}$ and $\boldsymbol{r}_{\mathsf{ood}} \leftarrow \mathbb{F}^{r_{\mathsf{ood}}}$, and sets
$$\boldsymbol{g} \coloneqq (\boldsymbol{f}, \boldsymbol{r}, \boldsymbol{r}_{\mathsf{ood}}), \qquad g \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{g}, \boldsymbol{r'}) \,.$$
2. The verifier samples and sends out-of-domain randomness $\rho_{\mathsf{ood}} \leftarrow D_{\mathsf{ood}}$.
3. The prover replies with $\boldsymbol{y} \in \mathbb{F}^{t_{\mathsf{ood}}}$. In the honest case, the prover sets $\boldsymbol{y} \coloneqq \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{g}$.
4. The verifier samples and sends in-domain query randomness $x_1, \ldots, x_t \leftarrow [m]$ and batching randomness $\boldsymbol{\nu} \leftarrow \mathbb{F}^{t_{\mathsf{ood}}+t}$.
5. The verifier queries $f(x_i)$ for $i \in [t]$, which we take to lie in $\mathbb{F}$ for simplicity, and sets
$$\mu' \coloneqq \mu + \sum_{i \in [t_{\mathsf{ood}}]} \boldsymbol{\nu}_i \cdot \boldsymbol{y}_i + \sum_{i \in [t]} \boldsymbol{\nu}_{t_{\mathsf{ood}}+i} \cdot f(x_i) \,.$$

   The verifier outputs a new instance for the target relation $\mathcal{R}_{\mathcal{C}',\mathsf{sl}'}$ with the joint constraint
$$\langle \boldsymbol{f}, \mathsf{sl}(\mathsf{st}) \rangle + \sum_{i \in [t_{\mathsf{ood}}]} \boldsymbol{\nu}_i \cdot \langle \boldsymbol{g}, \mathsf{ze}_{\mathsf{ood},i}(\rho_{\mathsf{ood}}) \rangle + \sum_{i \in [t]} \boldsymbol{\nu}_{t_{\mathsf{ood}}+i} \cdot \langle (\boldsymbol{f}, \boldsymbol{r}), \boldsymbol{G}_{\mathcal{C}}[x_i] \rangle = \mu' \,, \tag{6}$$

   where $\mathsf{ze}_{\mathsf{ood},i}$ is the i-th row of $\mathsf{ze}_{\mathsf{ood}}$. The output implicit instance is $\mathbb{y}' \coloneqq g$. The new succinct state is $(\mathsf{st}, \boldsymbol{\nu}, \rho_{\mathsf{ood}}, x_1, \ldots, x_t)$, and the new linear form sl' is obtained by grouping the terms of Equation 6. The honest output witness consists of the new message $\boldsymbol{g}$ and its randomness $\boldsymbol{r'}$.

Analysis of Construction 2.14.

Lemma 2.15. *Assume that $\mathsf{ze}_{\mathsf{ood}}$ is a $(r_{\mathsf{ood}}, \zeta_{\mathsf{ze}})$-private zero-evader and that the encoding of $\mathcal{C}'$ is t'-query zero-knowledge with error $\zeta_{\mathcal{C}'}$ (including any interleaving). Then, Construction 2.14 is HVZK with error $\zeta_{\mathcal{C}'} + \zeta_{\mathsf{ze}}$ against distinguishers making at most t' uniform queries to the output oracle. Its simulator makes at most t uniform queries to the input oracle. Moreover, the reduction has fresh output.*

Proof sketch. The simulator samples $x_1, \ldots, x_t$ uniformly and queries the input oracle at these positions. It uses the private zero-evader simulator for $(\boldsymbol{y}, \rho_{\mathsf{ood}})$ and the simulator of the encoding of $\mathcal{C}'$ for queries to g. The reduction has fresh output because the prover samples $\boldsymbol{r'}$ independently of everything else and uses it only to compute g, which the verifier does not query. When $\mathcal{C}'$ is an interleaved code, $\boldsymbol{r'}$ consists of a mask $\boldsymbol{\sigma}$ and the seeds of the rows (Section 2.5). □

Lemma 2.16 (informal)**.** *For every $\delta, \delta' \in (0,1)$, Construction 2.14 has round-by-round knowledge soundness with relaxation $(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}}, \tilde{\mathcal{R}}^{\delta'}_{\mathcal{C}',\mathsf{sl}'})$ and errors*
$$\left(\tfrac{1}{2} \Lambda(\mathcal{C}', \delta')^2 \cdot \varepsilon_{\mathsf{ood}},\ (1-\delta)^t + \frac{\Lambda(\mathcal{C}', \delta')}{|\mathbb{F}|} \right) .$$

Proof sketch. The round-by-round knowledge soundness proof is in Lemma 9.11.

Except with the error probability in Lemma 2.13, at most one message $\boldsymbol{g}$ in the output list satisfies the OOD condition. It determines the input message and its encoding randomness, and hence a unique input codeword.

If the input is outside the relaxed relation and the original constraint holds for this candidate, its input codeword has distance greater than δ from f. Otherwise the same witness would satisfy the relaxed input relation. Thus all consistency checks pass with probability at most $(1-\delta)^t$.

Outside these two failure events, every candidate fails the OOD condition, the original constraint, or a consistency check. Random batching accepts such a candidate with probability at most $1/|\mathbb{F}|$ (and never if only the original constraint fails). A union bound over the output list gives the remaining error term. □

Efficiency of Construction 2.14. The prover computes the target encoding and the OOD sample, and the verifier computes μ'. The new linear form used in subsequent reductions depends on the source code. We distinguish the two costs denoted by $T_{\mathcal{C}}$ in Theorem 4:

- The prover time will depend on the time $\mathsf{t}_{\mathcal{C}}^{\mathbf{P}}$ required to compute linear combinations of t rows of $\boldsymbol{G}_{\mathcal{C}}^{\#}$ and t rows of $\boldsymbol{G}_{\mathcal{C}}^{\$}$.
- The verifier time will depend on the time $\mathsf{t}_{\mathcal{C}}^{\mathbf{V}}$ required to evaluate linear combinations of t folded rows of $\boldsymbol{G}_{\mathcal{C}}^{\#}$ according to Definition 2.7, and t (non-folded) rows of $\boldsymbol{G}_{\mathcal{C}}^{\$}$.

The OOD rows contribute their evaluation costs.

2.7 From building blocks to our main results

We compose sumcheck, code switching, and the non-succinct base case using Lemma 2.5 and its extension to fresh inputs.

Theorem 3: sublinear HVZK IOPP for constrained interleaved codes. First we run the zero-knowledge sumcheck from Section 2.5 to reduce from $\mathcal{R}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}$ to $\mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}'}$. Then, we apply the non-succinct IOPP of Construction 2.2 to the folded instance. See Figure 4 for a pictorial description. Each query to the *folded* oracle is answered using a query to the original oracle over the alphabet Σ^{2^k}.

In the sublinear regime described in Section 1.1, the remaining message has sublinear size, so the base-case mask has $o(1)$ relative communication overhead, measured against the original oracle length. The interleaving factor can be optimized for concrete argument size [AHIV17; GLSTW23; BFRW26].

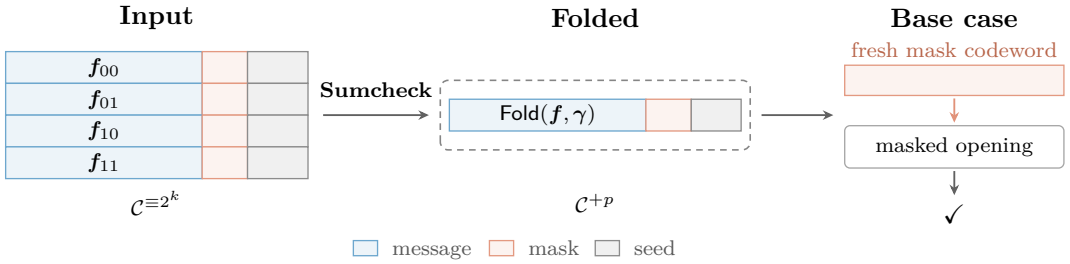

Figure 4: The sublinear HVZK IOPP of Theorem 3: the zero-knowledge sumcheck (Construction 2.9) folds the input oracle and moves the folded mask into the message, before Construction 2.2 is applied to the resulting virtual oracle.

Theorem 4: HVZK code-switching. We follow the same sumcheck IOR (Construction 2.9) with the code-switching IOR from Section 2.6, reducing to $\mathcal{R}_{\mathcal{C}',\mathsf{sl}'}$ (Figure 5). As above, queries to

the "virtual" folded oracle are answered via queries to its original interleaving. The composition is HVZK on fresh inputs and has fresh output, so it can be iterated.

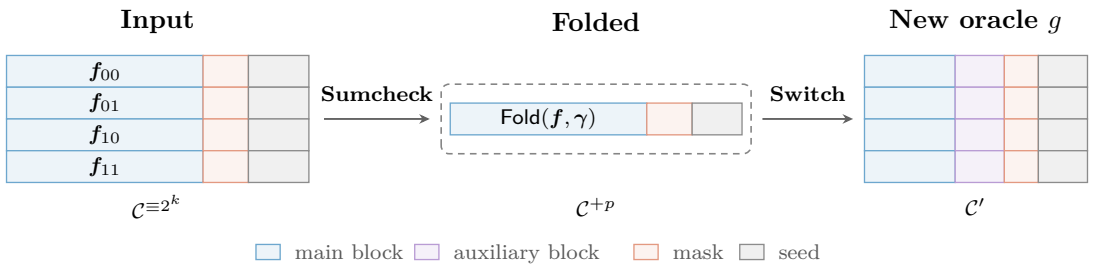

Figure 5: The HVZK IOR of Theorem 4: the zero-knowledge sumcheck (Construction 2.9) folds the input oracle, and Construction 2.14 re-encodes the folded message, mask, and seed. The rows of the new oracle g hold the main message and the auxiliary message, under fresh masks and seeds.

HVZK polylogarithmic-verifier protocol. To reduce the verifier time, we fix a folding parameter $k \in \mathbb{N}$ and a sequence of codes $\mathcal{C}_1, \ldots, \mathcal{C}_\mathfrak{n}$ with $\mathfrak{n} \geq 2$, selected so that, for every $\mathfrak{i} \in [\mathfrak{n}-2]$, a message of the 2^k interleaving of $\mathcal{C}_{\mathfrak{i}+1}$ is long enough to contain a message of (the non-interleaved) $\mathcal{C}_\mathfrak{i}$ together with its mask, its seed, and an OOD pad, and so is a message of $\mathcal{C}_\mathfrak{n}$ for $\mathcal{C}_{\mathfrak{n}-1}$. In every stage $\mathfrak{i} < \mathfrak{n}$, the prover and the verifier first run the sumcheck IOR from Construction 2.9 on an oracle of $\mathcal{C}_\mathfrak{i}^{\equiv 2^k}$ and then the code-switching IOR from Construction 2.14, in which the prover encodes the folded message, mask, and seed in the rows of $\mathcal{C}_{\mathfrak{i}+1}^{\equiv 2^k}$. The randomness of this encoding contains the masks of the next sumcheck. In stage $\mathfrak{n}-1$, the prover instead encodes them in the *uninterleaved* code $\mathcal{C}_\mathfrak{n}$. After each sumcheck, the main message and the auxiliary message are shorter by a factor of 2^k. We conclude with the base case of Construction 2.2. When $\mathfrak{n} = 2$, only the final reduction and this base case are needed. As in [ACFY24; ACFY25; NA25; BFRW26], choosing codes whose distances increase across the reductions improves query complexity. We state the result for a general sequence of codes.

Theorem 5 (informal, polylogarithmic HVZK)**.** *Let $\mathfrak{n} \geq 2$ and let:*

- *$k \in \mathbb{N}$ be a fixed folding parameter,*
- *$\mathcal{C}_1 \subseteq \Sigma^{m_{\mathcal{C}_1}}, \ldots, \mathcal{C}_\mathfrak{n} \subseteq \Sigma^{m_{\mathcal{C}_\mathfrak{n}}}$ be $\mathbb{F}$-additive codes such that, for $\mathfrak{i} \in [\mathfrak{n}]$, $\mathcal{C}_\mathfrak{i}$ has a $t_{\mathcal{C}_\mathfrak{i}}$-query zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}_\mathfrak{i}} \colon \mathbb{F}^{\ell_{\mathcal{C}_\mathfrak{i}}} \times \mathbb{F}^{r_{\mathcal{C}_\mathfrak{i}}} \to \Sigma^{m_{\mathcal{C}_\mathfrak{i}}}$ with error $\zeta_{\mathcal{C}_\mathfrak{i}}$ and encoding time $\tau_{\mathcal{C}_\mathfrak{i}}$, and with two spare coordinates if $\mathfrak{i} < \mathfrak{n}$. For each $\mathfrak{i} \in [\mathfrak{n}-2]$, let $\ell_{\mathcal{C}_\mathfrak{i}} := 2^k \cdot \ell_{\mathcal{C}_{\mathfrak{i}+1}}$, and let $\ell_{\mathcal{C}_{\mathfrak{n}-1}} := \ell_{\mathcal{C}_\mathfrak{n}}$. For $\mathfrak{i} \geq 2$, the messages of $\mathcal{C}_\mathfrak{i}$ have an additional auxiliary block, which holds the folded auxiliary block, mask, and seed of the preceding stage and its OOD pad. We assume $r_{\mathcal{C}_\mathfrak{i}} = O(\lambda)$, so that the auxiliary blocks of the rows of a stage have $O(\lambda)$ coordinates in total.*

For a security parameter $\lambda \in \mathbb{N}$ and proximity parameters $\delta_\mathfrak{i} \in (0, \delta(\mathcal{C}_\mathfrak{i}))$ for $\mathfrak{i} \in [\mathfrak{n}]$, let $t_\mathfrak{i} := O\left(\frac{\lambda}{-\log(1-\delta_\mathfrak{i})}\right)$. We construct an $O(k \cdot \mathfrak{n})$-round IOPP for $\mathcal{R}^{\mathsf{lin}}_{\mathcal{C}_1^{\equiv 2^k}, T}$, with the following features.

- *The prover sends $O\left(2^k \sum_{\mathfrak{i} \in [2,\mathfrak{n}-1]} m_{\mathcal{C}_\mathfrak{i}} + m_{\mathcal{C}_\mathfrak{n}}\right)$ symbols over Σ and $O\left(\mathfrak{n}k + \ell_{\mathcal{C}_\mathfrak{n}} + r_{\mathcal{C}_\mathfrak{n}} + \lambda\right)$ field elements. The verifier makes $t_\mathfrak{i}$ queries to the oracle of each stage $\mathfrak{i} \in [\mathfrak{n}]$, over alphabet Σ^{2^k} if $\mathfrak{i} < \mathfrak{n}$ and over Σ if $\mathfrak{i} = \mathfrak{n}$, and $t_\mathfrak{n}$ queries to the base-case mask.*

- *The prover time is*

$$O\left(T + 2^k \cdot (\ell_{\mathcal{C}_1} + r_{\mathcal{C}_1}) + 2^k \sum_{i \in [2,n-1]} \tau_{\mathcal{C}_i} + \tau_{\mathcal{C}_n} + \sum_{i \in [2,n]} \mathsf{t}^{\mathbf{P}}_{\mathcal{C}_{i-1}}\right)$$

  and the verifier time is

$$O\left(T^{(k\cdot(n-1))} + \tau_{\mathcal{C}_n} + \sum_{i \in [n-1]} \left(2^k \cdot t_i + \mathsf{t}^{\mathbf{V}}_{\mathcal{C}_i}\right) + t_n + n \cdot (k + \lambda)\right).$$

- *The protocol has round-by-round knowledge soundness error at most* $2^{-\lambda}$ *assuming* $\mathbb{F}$ *is sufficiently large compared to the list-decoding and mutual correlated agreement bounds of every code.*
- *If* $t_{\mathcal{C}_i} \geq t_i$ *for every* $i \in [n]$*, the protocol is honest-verifier zero-knowledge on fresh inputs with error* $O\left(2^k \cdot \sum_{i \in [n]} \zeta_{\mathcal{C}_i}\right)$.

The repeated construction and its final base case are illustrated in Figure 6. The Reed–Solomon instantiation yielding Corollary 2 is given in Section 10.1. The formal statement (Theorem 10.2) allows a different folding parameter in every stage, and the instantiation uses a smaller one in its last reduction.

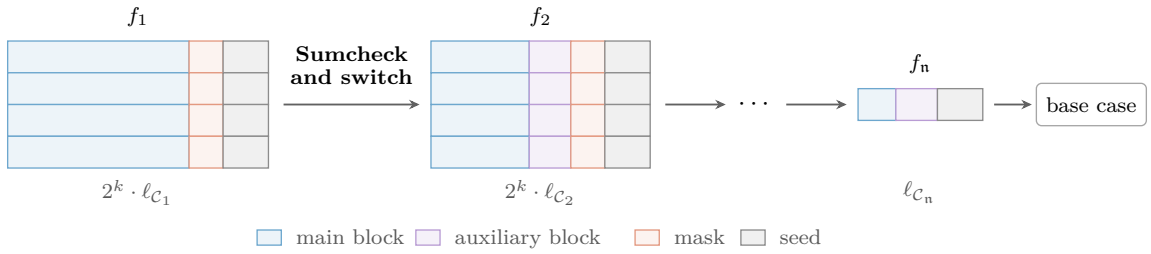

Figure 6: The protocol of Theorem 5: apply Theorem 4 $n-1$ times in sequence, followed by the base case. Every stage folds its oracle and re-encodes the result as in Figure 5, and the auxiliary blocks of the next oracle hold the folded mask and seed. Main-message lengths appear below.

A zero-knowledge reduction from R1CS to proximity-testing. We additionally provide a reduction with $1 + o(1)$ multiplicative communication overhead from R1CS to $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$, for any $\mathcal{C}$ (in particular, an interleaved one). Compared to prior work [BCL22], our additional masking communication is sublinear in the encoded codeword length. We omit the holographic/pre-processing step of [BCL22; RW24], so the verifier is non-succinct.

The reduction follows the Σ-IOP for R1CS of [ACFY25]. The prover sends one oracle, which encodes the witness together with a number of mask coordinates that is logarithmic in the number of constraints. The prover and the verifier then run a sumcheck over the constraints. At its end, the prover sends three values: the multilinear extensions of the three matrix-vector products of the R1CS instance, evaluated at the point of the verifier's challenges.

The sumcheck messages and these three values depend on the witness, and the prover hides both. It hides the sumcheck messages with the mask polynomial of Section 2.5. Here the summands are constraints, not rows of an oracle, so the prover places the tuple $\boldsymbol{\sigma}$ in the message of the oracle

that it sends. It hides each of the three values by adding to it one mask coordinate, multiplied by a nonzero factor that the verifier knows.

The remaining checks are linear in the witness and the masks. The verifier combines them, with random coefficients, into one linear constraint on the message of the oracle, which gives an instance of $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$. The prover encodes the oracle with fresh randomness and the verifier does not query it, so the reduction has fresh output (Section 2.3), and its composition with our IOPP is zero-knowledge for every R1CS witness.

2.8 Extension: distance-amplified codes

In this section, we propose **distance amplification via dispersers** [ABNNR92] as a way to improve encoding time or argument size (Section 13). The transformation groups $d = O(1/\gamma)$ base symbols into each amplified symbol according to a disperser graph, increasing any constant base code distance to at least $1 - \gamma$ while keeping the block length and incurring an alphabet blow-up from $\mathbb{F}$ to $\mathbb{F}^d$. The graph is chosen so that every set of base positions whose size is at least the base code's distance times its block length has neighbors in at least a $(1-\gamma)$ fraction of output positions. The disagreement set of any two distinct base codewords meets this hypothesis, so their amplified codewords differ in at least that fraction of positions. The resulting code remains $\mathbb{F}$-additive. Moreover, if the base code admits a strong t-query zero-knowledge encoding, then the amplified code has a strong $\lfloor t/d \rfloor$-query zero-knowledge encoding. We can therefore use amplified codes when the base encoding has query privacy for at least d times the required number of amplified queries.

We instantiate distance amplification with a Reed–Solomon code as the base code, yielding an *amplified Reed–Solomon code* (ARS code). For a message of length ℓ and target distance $1 - \gamma$, assuming evaluation domains supporting FFTs, encoding for plain RS takes time $O(\frac{\ell}{\gamma} \cdot \log \frac{\ell}{\gamma})$ while encoding for ARS takes time $O(\frac{\ell}{\gamma} + \ell \cdot \log \ell)$. ARS encoding uses an FFT of size $\Theta(\ell)$ and groups the resulting evaluations according to the disperser graph. Each symbol consists of one field element for plain RS and d field elements for ARS.

Relative to a fixed unamplified base code, distance amplification reduces the number of spot checks for a fixed soundness target, while each opened leaf contains d field symbols. In the parameter regime and cost model of Section 13.3, the communication due to Merkle openings decreases by $\approx 27\%$. Relative to the same base code, the prover performs the additional amplification and commits to larger symbols, increasing its runtime.

3 Preliminaries

We define objects and state results that we use in this paper. We use the following notation.

- We write $[n] \coloneqq \{1, \ldots, n\}$, and all logarithms are to base 2.
- Throughout, sampling from a set means uniform sampling and is used only for finite nonempty sets.
- For a function $f\colon [m] \to \Sigma$ we let $\mathrm{im}(f) = \{f(x) : x \in [m]\}$.
- For two functions $f, g\colon [m] \to \Sigma$ and a set $S \subseteq [m]$, we write $f(S) = g(S)$ to mean $f(x) = g(x)$ for every $x \in S$. Conversely, $f(S) \neq g(S)$ means there exists $x \in S$ with $f(x) \neq g(x)$.
- For two functions $f, g\colon [m] \to \Sigma$, $\Delta(f, g)$ is the fractional Hamming distance between f and g (the fraction of points in which they disagree).

Relations. A *relation* $\mathcal{R}$ consists of triples $(\mathbb{x}, \mathbb{y}, \mathbb{w})$ where $\mathbb{x}$ is the explicit instance, $\mathbb{y}$ the implicit instance, and $\mathbb{w}$ the witness.

3.1 Interactive oracle reductions

Interactive Oracle Reductions (IORs) [BCGGRS19; BMNW25] are a generalization of Interactive Oracle Proofs (IOPs) [BCS16; RRR16]. Below we describe the flavor of IOR relevant to this paper.

A public-coin IOR of proximity from relation $\mathcal{R}$ to relation $\mathcal{R}'$ is a tuple $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ that works as follows. The prover $\mathbf{P}$ is an interactive algorithm that receives as input $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$, and the verifier $\mathbf{V}$ is an interactive oracle algorithm that receives input $\mathbb{x}$ and query access to $\mathbb{y}$ (consisting of l_0 symbols over some alphabet Σ_0). They interact over k rounds: in each round $i \in [\mathsf{k}]$, $\mathbf{P}$ sends a proof string Π_i consisting of l_i symbols over some alphabet Σ_i, and $\mathbf{V}$ sends a uniformly random message $\rho_i \in \{0,1\}^{\mathsf{r}_i}$. After k rounds of interaction, $\mathbf{V}$ makes queries to $\mathbb{y}, \Pi_1, \ldots, \Pi_{\mathsf{k}}$ (each query to a symbol). After that: (i) $\mathbf{V}$ either rejects or outputs a new explicit instance $\mathbb{x}'$ and implicit instance $\mathbb{y}'$ (which is implicitly specified in terms of the oracles $\mathbb{y}$ and $\Pi_1, \ldots, \Pi_{\mathsf{k}}$), and (ii) $\mathbf{P}$ outputs a new witness $\mathbb{w}'$ such that $(\mathbb{x}', \mathbb{y}', \mathbb{w}') \in \mathcal{R}'$. The interaction of $\mathbf{P}, \mathbf{V}$ above is denoted $\langle \mathbf{P}, \mathbf{V} \rangle (\mathbb{x}, \mathbb{y}, \mathbb{w})$.

Definition 3.1. $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ *has* **(perfect) completeness** *if, for every* $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$*,*

$$\Pr\left[(\mathbb{x}', \mathbb{y}', \mathbb{w}') \in \mathcal{R}' \;\middle|\; (\mathbb{x}', \mathbb{y}', \mathbb{w}') \leftarrow \langle \mathbf{P}, \mathbf{V} \rangle (\mathbb{x}, \mathbb{y}, \mathbb{w}) \right] = 1 \,.$$

Efficiency measures. Efficiency measures are implicitly functions of $\mathbf{V}$'s inputs.

- *Rounds:* there are k rounds of interaction (a round is a prover message and verifier message).
- *Prover communication:* for each $i \in [\mathsf{k}]$, Π_i consists of l_i symbols over alphabet Σ_i.
- *Input queries:* $\mathsf{q}_{\mathbb{y}}$ is the number of alphabet symbols read by the verifier from $\mathbb{y}$.
- *Proof queries:* q_Π is the number of alphabet symbols read by the verifier from $\Pi_1, \ldots, \Pi_{\mathsf{k}}$.
- *Input query blowup:* $m_{\mathbb{y}' \to \mathbb{y}}$ is the number of queries that a query to $\mathbb{y}'$ induces to $\mathbb{y}$.
- *Proof query blowup:* $m_{\mathbb{y}' \to \Pi}$ is the number of queries that a query to $\mathbb{y}'$ induces to $\Pi_1, \ldots, \Pi_{\mathsf{k}}$.
- *Prover time:* pt is $\mathbf{P}$'s time measured in ring and module operations.
- *Verifier time:* vt is $\mathbf{V}$'s time measured in ring and module operations.

Remark 3.2. An IOP for a relation $\mathcal{R}$ is an IOR from the relation $\mathcal{R}$ to the trivial relation $\mathcal{R}_{\mathsf{triv}} \coloneqq \{(\mathbb{x} = \bot, \mathbb{y} = \bot, \mathbb{w} = \bot)\}$.

The IORs that we consider in this paper satisfy a strong notion of knowledge soundness, namely round-by-round knowledge soundness [BCFW25], extended to handle relaxed relations.

Definition 3.3. *Let* $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ *be an IOR from a relation* $\mathcal{R}$ *to* $\mathcal{R}'$*. A* **knowledge state function with relaxation** $(\tilde{\mathcal{R}}, \tilde{\mathcal{R}}')$ *for* IOR *is a (possibly inefficient) function* KState *that, on input a statement* $(\mathbb{x}, \mathbb{y})$*, 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* $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathsf{w}) = 1$ *if and only if* $(\mathbb{x}, \mathbb{y}, \mathsf{w}) \in \tilde{\mathcal{R}}$*.*
- *Prover moves: if* tr *is a transcript where the prover is about to move and* $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathsf{w}) = 0$*, then* $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \Pi, \mathsf{w}) = 0$ *for every prover message* Π*.*
- *Full transcript: if* $\mathsf{tr} = (\Pi_1, \rho_1, \ldots, \Pi_{\mathsf{k}}, \rho_{\mathsf{k}})$ *is a full transcript, then* $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathsf{w}) = 1$ *if and only if* $\mathbf{V}^{\mathbb{y}, \Pi_1, \ldots, \Pi_{\mathsf{k}}}(\mathbb{x}, \rho_1, \ldots, \rho_{\mathsf{k}})$ *outputs* $(\mathbb{x}', \mathbb{y}')$ *such that* $(\mathbb{x}', \mathbb{y}', \mathsf{w}) \in \tilde{\mathcal{R}}'$*.*

Definition 3.4. $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ *from a relation* $\mathcal{R}$ *to* $\mathcal{R}'$ *has* **round-by-round (RBR) knowledge soundness with relaxation** $(\tilde{\mathcal{R}}, \tilde{\mathcal{R}}')$**, errors** $(\varepsilon_1, \ldots, \varepsilon_{\mathsf{k}})$**, and extraction times** $(\mathsf{et}_1, \ldots, \mathsf{et}_{\mathsf{k}})$ *if there exist a knowledge state function* KState *with relaxation* $(\tilde{\mathcal{R}}, \tilde{\mathcal{R}}')$ *for* IOR *and a deterministic extractor* $\mathbf{E}_{\mathsf{rbr}}$ *with the following property: for every statement* $(\mathbb{x}, \mathbb{y})$*, round index* $i \in [\mathsf{k}]$*, and interaction transcript* $\mathsf{tr} = (\Pi_1, \rho_1, \ldots, \Pi_{i-1}, \rho_{i-1}, \Pi_i)$ *where the verifier is about to move,* $\mathbf{E}_{\mathsf{rbr}}$ *runs in time at most* et_i *and*

$$\Pr_{\rho_i}\left[\exists\, \mathsf{w}: \begin{array}{l} \mathsf{KState}\left(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \rho_i, \mathsf{w})\right) = 0 \\ \wedge\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \rho_i, \mathsf{w}) = 1 \end{array}\right] \leq \varepsilon_i(\mathbb{x}, \mathbb{y})\,.$$

The **total extraction time** *is* $\sum_{i \in [\mathsf{k}]} \mathsf{et}_i$*.*

3.2 Abstract algebra

Throughout the paper, we let R be a ring and $\mathcal{M}$ a right-module over R.[3] A ***generating set for*** $\mathcal{M}$ is a tuple $\boldsymbol{m} \in \mathcal{M}^{d_{\mathcal{M}}}$ such that $\{\sum_i \boldsymbol{m}_i \cdot \boldsymbol{r}_i \ :\ \boldsymbol{r} \in R^{d_{\mathcal{M}}}\} = \mathcal{M}$. We recall the notion of the ***center of a ring*** $\mathcal{Z}(R) = \{x \in R \mid \forall y \in R : xy = yx\}$ and let $\mathcal{M}^{<d}[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ denote polynomials with coefficients over $\mathcal{M}$ in the variables $\mathtt{X}_1, \ldots, \mathtt{X}_n$ (over R) of individual degree $< d$.[4] For $\boldsymbol{v} \in R^\ell$, $\boldsymbol{w} \in \mathcal{M}^\ell$ and $\boldsymbol{W} \in \mathcal{M}^{t \times \ell}$, we let:

$$\langle \boldsymbol{w}, \boldsymbol{v} \rangle = \sum_i \boldsymbol{w}_i \cdot \boldsymbol{v}_i \text{ and } \boldsymbol{W} \cdot \boldsymbol{v} = \begin{bmatrix} \langle \boldsymbol{W}_1, \boldsymbol{v} \rangle \\ \vdots \\ \langle \boldsymbol{W}_t, \boldsymbol{v} \rangle \end{bmatrix}.$$

We define zero-evaders over R and over $\mathcal{M}$.

Definition 3.5. *A* **zero-evader over** R *with error* $\varepsilon_{\mathsf{zero}}$ *is a function* $\mathsf{ze}\colon D_{\mathsf{ze}} \to \mathcal{M}^{t \times \ell}$ *such that*

$$\forall\, \boldsymbol{v} \in R^\ell \setminus \{0^\ell\}\,,\ \Pr_{\rho \leftarrow D_{\mathsf{ze}}}\left[\mathsf{ze}(\rho) \cdot \boldsymbol{v} = 0^t\right] \leq \varepsilon_{\mathsf{zero}}\,.$$

[3] The choice of using a right-module is arbitrary, and the notions coincide when R is commutative.

[4] We consider only polynomials with ordered monomials i.e. $\mathtt{X}_1 \cdot \mathtt{X}_2^3$ is valid while $\mathtt{X}_2 \cdot \mathtt{X}_1 \cdot \mathtt{X}_2^2$ is not. When R is commutative, the two are the same.

Definition 3.6. *A* **zero-evader over** $\mathcal{M}$ *with error* $\varepsilon_{\mathsf{zero}}$ *is a function* $\mathsf{ze}\colon D_{\mathsf{ze}} \to R^{\ell \times t}$ *such that*

$$\forall\, \boldsymbol{v} \in \mathcal{M}^{\ell} \setminus \{0^{\ell}\}\,,\ \Pr_{\rho \leftarrow D_{\mathsf{ze}}}\left[\boldsymbol{v}^{\mathsf{T}} \cdot \mathsf{ze}(\rho) = 0^{t}\right] \le \varepsilon_{\mathsf{zero}}\,.$$

Many examples of zero-evaders come from the polynomial identity lemma, which we generalize.

Definition 3.7. *Let* $\mathfrak{C} \subseteq R$ *be finite and nonempty, and let* $n, d \ge 1$*. The* **polynomial-identity error of** $\mathfrak{C}$ **over** $\mathcal{M}$ *is*

$$\epsilon_{\mathsf{pil}}^{\mathcal{M}}(\mathfrak{C}, n, d) \coloneqq \max_{\hat{p} \in \mathcal{M}^{<d}[\mathtt{X}_1, \ldots, \mathtt{X}_n] \setminus \{0\}} \Pr_{\boldsymbol{\tau} \leftarrow \mathfrak{C}^n}\left[\hat{p}(\boldsymbol{\tau}) = 0\right]\,.$$

The **polynomial-identity error of** $\mathfrak{C}$ **over** R*, denoted* $\epsilon_{\mathsf{pil}}^{R}(\mathfrak{C}, n, d)$*, is the same quantity for polynomials with coefficients in* R*, that is, for the module* $\mathcal{M} = R$*.*

Over a finite field every set of distinct points supports the polynomial identity lemma (also known as the Schwartz–Zippel lemma). Over rings, the relevant condition is that challenges commute and the difference of any two distinct challenges is invertible.

Definition 3.8. *A finite nonempty set* $\mathfrak{C} \subseteq R$ *is a* **strong sampling set** *if, for every distinct* $a, b \in \mathfrak{C}$*,* $a \cdot b = b \cdot a$ *and the difference* $a - b$ *is invertible in* R*.*

Strong sampling sets support the usual root bound for polynomials with coefficients in $\mathcal{M}$.

Lemma 3.9. *Let* $\mathfrak{C}$ *be a strong sampling set and let* $d \ge 1$*. Every nonzero* $\hat{p} \in \mathcal{M}^{<d}[\mathtt{X}]$ *has at most* $d - 1$ *roots in* $\mathfrak{C}$*.*

Proof. By induction on d. For $d = 1$, $\hat{p}$ is a nonzero constant and has no roots. Let $d \ge 2$ and write $\hat{p}(\mathtt{X}) = \sum_{j \in [d]} \boldsymbol{c}_j \cdot \mathtt{X}^{j-1}$. If $\hat{p}$ has no root in $\mathfrak{C}$ there is nothing to prove, so let $a \in \mathfrak{C}$ be a root. Define $\hat{q}(\mathtt{X}) \coloneqq \sum_{k \in [d-1]} \boldsymbol{e}_k \cdot \mathtt{X}^{k-1} \in \mathcal{M}^{<d-1}[\mathtt{X}]$, where $\boldsymbol{e}_k \coloneqq \sum_{j=k+1}^{d} \boldsymbol{c}_j \cdot a^{j-1-k}$. For every $b \in \mathfrak{C}$, since a and b commute, $b^{j-1} - a^{j-1} = \sum_{k=1}^{j-1} a^{j-1-k} \cdot b^{k-1} \cdot (b - a)$, and hence

$$\hat{p}(b) = \hat{p}(b) - \hat{p}(a) = \sum_{j=2}^{d} \boldsymbol{c}_j \cdot \left(b^{j-1} - a^{j-1}\right) = \sum_{k=1}^{d-1} \left(\sum_{j=k+1}^{d} \boldsymbol{c}_j \cdot a^{j-1-k} \right) \cdot b^{k-1} \cdot (b - a) = \hat{q}(b) \cdot (b - a)\,.$$

For $b \neq a$ the element $b - a$ is invertible, so every root of $\hat{p}$ in $\mathfrak{C} \setminus \{a\}$ is a root of $\hat{q}$. Moreover $\hat{q} \neq 0$: if j^* is the largest index with $\boldsymbol{c}_{j^*} \neq 0$ then $j^* \ge 2$ (otherwise $\hat{p}(a) = \boldsymbol{c}_1 \neq 0$), and $\boldsymbol{e}_{j^*-1} = \boldsymbol{c}_{j^*} \neq 0$. By induction $\hat{q}$ has at most $d - 2$ roots in $\mathfrak{C}$, so $\hat{p}$ has at most $d - 1$. □

Lemma 3.10. *If* $\mathfrak{C}$ *is a strong sampling set, then for every* $n, d \ge 1$*,*

$$\epsilon_{\mathsf{pil}}^{\mathcal{M}}(\mathfrak{C}, n, d) \le \frac{n \cdot (d - 1)}{|\mathfrak{C}|}\,,$$

and the same bound holds for $\epsilon_{\mathsf{pil}}^{R}(\mathfrak{C}, n, d)$*.*

Proof. The bound over R is the case $\mathcal{M} = R$ of the bound over $\mathcal{M}$, which we prove by induction on n. The case $n = 1$ is Lemma 3.9. For $n > 1$, a nonzero $\hat{p}$ can be written as

$\hat{p} = \sum_{j\in[d]} \hat{q}_j(\mathtt{X}_1, \ldots, \mathtt{X}_{n-1}) \cdot \mathtt{X}_n^{j-1}$ with $\hat{q}_j \in \mathcal{M}^{<d}[\mathtt{X}_1, \ldots, \mathtt{X}_{n-1}]$. Fix j^* with $\hat{q}_{j^*} \neq 0$. For $\boldsymbol{\tau} = (\boldsymbol{\tau}', x_n) \leftarrow \mathfrak{C}^n$,

$$\Pr\left[\hat{p}(\boldsymbol{\tau}) = 0\right] \leq \Pr\left[\hat{q}_{j^*}(\boldsymbol{\tau}') = 0\right] + \Pr\left[\hat{p}(\boldsymbol{\tau}', x_n) = 0 \mid \hat{q}_{j^*}(\boldsymbol{\tau}') \neq 0\right] .$$

The first term is at most $\frac{(n-1)(d-1)}{|\mathfrak{C}|}$ by induction. For the second, whenever $\hat{q}_{j^*}(\boldsymbol{\tau}') \neq 0$ the polynomial $\hat{p}(\boldsymbol{\tau}', \mathtt{X}_n) = \sum_{j\in[d]} \hat{q}_j(\boldsymbol{\tau}') \cdot \mathtt{X}_n^{j-1} \in \mathcal{M}^{<d}[\mathtt{X}_n]$ is nonzero, so the term is at most $\frac{d-1}{|\mathfrak{C}|}$ by Lemma 3.9. □

The equality polynomial eq for n variables is

$$\mathsf{eq}(\mathtt{X}_1, \ldots, \mathtt{X}_n, \mathtt{Y}_1, \ldots, \mathtt{Y}_n) \coloneqq \prod_{i=1}^{n} (\mathtt{X}_i \cdot \mathtt{Y}_i + (1 - \mathtt{X}_i) \cdot (1 - \mathtt{Y}_i)) .$$

Claim 3.11. *For every $\hat{f} \in \mathcal{M}^{<2}[\mathtt{X}_1, \ldots, \mathtt{X}_n]$ and $\boldsymbol{\tau} \in R^n$, $\hat{f}(\boldsymbol{\tau}) = \sum_{\boldsymbol{b}\in\{0,1\}^n} \hat{f}(\boldsymbol{b}) \cdot \mathsf{eq}(\boldsymbol{\tau}, \boldsymbol{b})$.*

3.3 Error correcting codes

Across the paper, we consider error correcting codes whose message space is R^ℓ.

Definition 3.12. *A* **code** $\mathcal{C}$ **over an alphabet** Σ **with message size** ℓ **and block length** m *is an injective map $\mathcal{C} \colon R^\ell \to \Sigma^m$. The* **minimum distance of** $\mathcal{C}$ *is*

$$\delta(\mathcal{C}) \coloneqq \min_{\substack{f,g\in\mathrm{im}(\mathcal{C}) \\ f\neq g}} \Delta(f, g) .$$

We often identify a code with its image $\mathcal{C}(R^\ell) \subseteq \Sigma^m$.

Definition 3.13. *An* $(R, \mathcal{M})$**-additive code** $\mathcal{C} \colon R^\ell \to \mathcal{M}^m$ *is an injective R-module homomorphism.*

Note that an $(R, \mathcal{M})$-additive code is a code over alphabet $\Sigma = \mathcal{M}$ with message size ℓ and block length m. When $R = \mathbb{F}$ is a finite field we call it an $\mathbb{F}$-*additive code* (or simply an additive code), and a linear code is an $(\mathbb{F}, \mathbb{F})$-additive code. We let $\mathsf{tenc}(\mathcal{C})$ be the number of ring operations required to encode a message $\boldsymbol{x} \in R^\ell$ into its corresponding codeword $u \coloneqq \mathcal{C}(\boldsymbol{x})$.

Definition 3.14. *A code $\mathcal{C} \subseteq \Sigma^m$ has* **erasure correction with time** $\mathsf{tcor}(\mathcal{C})$ *if there exists a deterministic algorithm $\mathbf{E}_\mathcal{C}$ that on input $f \in \Sigma^m$ and $S \subseteq [m]$: (i) if $|S| > (1 - \delta(\mathcal{C})) \cdot m$ and there exists a codeword $u \in \mathcal{C}$ such that $f(i) = u(i)$ for all $i \in S$, then $\mathbf{E}_\mathcal{C}(f, S) = u$ (the codeword u is unique), and (ii) otherwise, $\mathbf{E}_\mathcal{C}(f, S) = \perp$. Moreover, $\mathbf{E}_\mathcal{C}$ performs at most $\mathsf{tcor}(\mathcal{C})$ ring and module operations.*

Every additive code has efficient erasure correction (Lemma 3.15, see [GRS12]). The lemma does not necessarily hold for general $(R, \mathcal{M})$-additive codes, but similar statements hold for many R of interest (a sufficient condition is that R admits efficient algorithms to solve linear equations with coefficients in R).

Lemma 3.15. *Every additive code $\mathcal{C} \subseteq (\mathbb{F}^\iota)^m$ has erasure correction with time $\mathsf{tcor}(\mathcal{C}) = O((\iota \cdot m)^3)$.*

Interleaving. An interleaved code is obtained by "stacking" a code's codewords.

Definition 3.16. *Let $\mathcal{C} \subseteq \Sigma^m$ be a code and let $K \in \mathbb{N}$. The* **K-interleaved code** *$\mathcal{C}^{\equiv K}$ is the code over alphabet Σ^K with block length m defined as*

$$\mathcal{C}^{\equiv K} \coloneqq \{(u_1, \ldots, u_K) \mid \forall\, i \in [K] : u_i \in \mathcal{C}\} \subseteq (\Sigma^K)^m\,.$$

A codeword $u \in \mathcal{C}^{\equiv K}$ consists of $K \cdot m$ symbols in Σ viewed as m symbols in Σ^K.

If $\mathcal{C}$ is $(R, \mathcal{M})$-additive then $\mathcal{C}^{\equiv K}$ is $(R, \mathcal{M}^K)$-additive.

Lists. We recall the combinatorial list-decoding of a code.

Definition 3.17. *Let $\mathcal{C} \subseteq \Sigma^m$ be a code. The* **list around** *$f \in \Sigma^m$* **at distance** *$\delta \in [0,1]$ is*

$$\Lambda(\mathcal{C}, f, \delta) \coloneqq \{g \in \mathcal{C} \mid \Delta(f, g) \le \delta\}\,.$$

The **worst-case list size** *is $\Lambda(\mathcal{C}, \delta) \coloneqq \max_{f \in \Sigma^m} |\Lambda(\mathcal{C}, f, \delta)|$.*

For every $K \in \mathbb{N}$ and $\delta \in (0,1)$ it holds that $\Lambda(\mathcal{C}, \delta) \le \Lambda(\mathcal{C}^{\equiv K}, \delta) \le \Lambda(\mathcal{C}, \delta)^K$. For $\delta < \delta(\mathcal{C})$, the list size of the interleaved code is bounded by a polynomial in the base-code list size whose coefficients and degree are independent of K, as stated below.

Lemma 3.18 ([GGR11])**.** *Let $\mathcal{C} \subseteq \Sigma^m$ be a code and let $\delta \in [0, \delta(\mathcal{C}))$. Let $b \coloneqq \left\lceil \frac{\delta}{\delta(\mathcal{C}) - \delta} \right\rceil$ and $r \coloneqq \left\lceil \log \frac{\delta(\mathcal{C})}{\delta(\mathcal{C}) - \delta} \right\rceil$. For every $K \ge 1$, $\Lambda(\mathcal{C}^{\equiv K}, \delta) \le \binom{b+r}{r} \cdot \Lambda(\mathcal{C}, \delta)^r$.*

MCA. We recall the notion of mutual correlated agreement [ACFY25] (adapted to $(R, \mathcal{M})$-additive codes).

Definition 3.19. *Let $\mathcal{C} \subseteq \mathcal{M}^m$ be an $(R, \mathcal{M})$-additive code. The* **mutual correlated agreement (MCA)** *error of a function $\mathsf{PG}\colon D \to R^n$ for $\mathcal{C}$ at distance $\delta \in [0,1]$ is*

$$\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) \coloneqq \max_{f_1, \ldots, f_n \in \mathcal{M}^m} \Pr_{\substack{\alpha \leftarrow D \\ \gamma \coloneqq \mathsf{PG}(\alpha)}} \left[\exists\, S \subseteq [m] : \begin{array}{l} |S| \ge (1-\delta) \cdot m \\ \exists\, u \in \mathcal{C} : \sum_i f_i(S) \cdot \gamma_i = u(S) \\ \exists\, i \in [n], \forall\, u \in \mathcal{C} : f_i(S) \ne u(S) \end{array} \right]$$

Upper bounds on $\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta)$ are known for every $\mathbb{F}$-additive code (see [ABF26] for a survey). For completeness, we prove in Lemma A.1 the elementary bound for arbitrary $(R, \mathcal{M})$-additive codes at every $\delta < \delta(\mathcal{C})/3$ when the challenge is sampled from a strong sampling set. In this work we only consider functions $\mathsf{PG}\colon \mathfrak{C} \to R^2$ where $\gamma \mapsto (1-\gamma, \gamma)$ or $\gamma \mapsto (1, \gamma)$. These two have the same error, as can be seen by considering the identity $f_1 \cdot (1-\gamma) + f_2 \cdot \gamma = f_1 + (f_2 - f_1) \cdot \gamma$.

By a union bound, the MCA error of the interleaved code is at most K times the MCA error of its base code.

Lemma 3.20. *Let $\mathcal{C} \subseteq \mathcal{M}^m$ be an $(R, \mathcal{M})$-additive code and let $\mathsf{PG}\colon D \to R^n$ have MCA error ϵ_{mca}. For every $K \in \mathbb{N}$, $\epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv K}, \delta) \le K \cdot \epsilon_{\mathsf{mca}}(\mathcal{C}, \delta)$.*

Tighter bounds are known when $\mathcal{C}$ is $\mathbb{F}$-additive: $\epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv K}, \delta) \le 2 \cdot \epsilon_{\mathsf{mca}}(\mathcal{C}, \delta)$ (independent of K [Jo26]).

3.3.1 Zero-knowledge codes

We define zero-knowledge encoding for codes.

Definition 3.21. *Let $\mathcal{C}$ be an (R, Σ)-additive code with message size $\ell + r$. For message length ℓ and randomness length r, a* **t-query zero-knowledge encoding for $\mathcal{C}$ with error $\zeta_{\mathcal{C}}$** *is an injective R-module homomorphism $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell} \times R^{r} \to \Sigma^{m}$ with $\mathrm{im}(\mathsf{Enc}_{\mathcal{C}}) = \mathcal{C}$ and a polynomial-time computable inverse such that there exists a polynomial-time probabilistic simulator $\mathsf{Sim}_{\mathcal{C}}$ and, for every $\boldsymbol{v} \in R^{\ell}$, the following two random variables have statistical distance at most $\zeta_{\mathcal{C}}$:*

- *The random variable $(Q, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{v}, \boldsymbol{r})[Q])$ consisting of $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{v}, \boldsymbol{r})$ restricted to the locations of Q, where the randomness is over $Q \leftarrow [m]^{t}$ and $\boldsymbol{r} \leftarrow R^{r}$.*
- *The random variable $(Q, \mathsf{Sim}_{\mathcal{C}}(Q))$, where the randomness is over $Q \leftarrow [m]^{t}$ and $\mathsf{Sim}_{\mathcal{C}}$'s randomness.*

The encoding is **strong** *if the statistical distance above holds for every fixed Q.*

A strong zero-knowledge encoding is a zero-knowledge encoding, and a zero-knowledge encoding remains one, with the same error, for every smaller number of queries.

Claim 3.22. *A strong t-query zero-knowledge encoding for $\mathcal{C}$ with error $\zeta_{\mathcal{C}}$ is a t-query zero-knowledge encoding for $\mathcal{C}$ with error $\zeta_{\mathcal{C}}$. A t-query zero-knowledge encoding for $\mathcal{C}$ with error $\zeta_{\mathcal{C}}$ is a t'-query zero-knowledge encoding for $\mathcal{C}$ with error $\zeta_{\mathcal{C}}$ for every $t' \leq t$.*

We introduce notation for the generating matrix of a zero-knowledge code (that separates the message portion from the randomness).

Definition 3.23. *Let $\mathcal{C} \subseteq \mathcal{M}^{m}$ be an $(R, \mathcal{M})$-additive code with a zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell} \times R^{r} \to \mathcal{M}^{m}$. A* **generating matrix for $\mathsf{Enc}_{\mathcal{C}}$** *is a matrix $\boldsymbol{G}_{\mathcal{C}} = \begin{bmatrix} \boldsymbol{G}_{\mathcal{C}}^{\#} & \boldsymbol{G}_{\mathcal{C}}^{\$} \end{bmatrix} \in \mathcal{M}^{m \times (\ell + r)}$ such that*

$$\forall\, \boldsymbol{f} \in R^{\ell},\ \boldsymbol{r} \in R^{r} :\ \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r}) = \boldsymbol{G}_{\mathcal{C}} \cdot \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{r} \end{bmatrix} = \boldsymbol{G}_{\mathcal{C}}^{\#} \cdot \boldsymbol{f} + \boldsymbol{G}_{\mathcal{C}}^{\$} \cdot \boldsymbol{r}\,.$$

For $i \in [m]$, $\boldsymbol{G}_{\mathcal{C}}[i, \cdot]$ denotes the i-th row of $\boldsymbol{G}_{\mathcal{C}}$.

A generating matrix for $\mathsf{Enc}_{\mathcal{C}}$ exists because $\mathsf{Enc}_{\mathcal{C}}$ is an R-module homomorphism: its j-th column is the encoding of the j-th standard basis vector of $R^{\ell + r}$.

The interleaving of a code with a zero-knowledge encoding also has a zero-knowledge encoding: encode each block separately with independent randomness.

Claim 3.24. *Let $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell} \times R^{r} \to \Sigma^{m}$ be a t-query zero-knowledge encoding for $\mathcal{C} \subseteq \Sigma^{m}$ with error $\zeta_{\mathcal{C}}$, and let $K \in \mathbb{N}$. Split $\boldsymbol{f} \in R^{K\ell}$ and $\boldsymbol{r} \in R^{Kr}$ into K blocks $(\boldsymbol{f}_i)_{i \in [K]}$ and $(\boldsymbol{r}_i)_{i \in [K]}$ of lengths ℓ and r. Then*

$$\mathsf{Enc}_{\mathcal{C}^{\equiv K}}(\boldsymbol{f}, \boldsymbol{r}) \coloneqq (\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_i, \boldsymbol{r}_i))_{i \in [K]} \in (\Sigma^{K})^{m}$$

is a t-query zero-knowledge encoding for $\mathcal{C}^{\equiv K}$ with error $K \cdot \zeta_{\mathcal{C}}$.

Proof. Encoding and decoding blockwise, $\mathsf{Enc}_{\mathcal{C}^{\equiv K}}$ is an injective R-module homomorphism onto $\mathcal{C}^{\equiv K}$ with a polynomial-time inverse. Let $\mathsf{Sim}_{\mathcal{C}}$ be the simulator for $\mathsf{Enc}_{\mathcal{C}}$ and let $\mathsf{Sim}_{\mathcal{C}^{\equiv K}}(Q) \coloneqq$

$(\mathsf{Sim}_{\mathcal{C}}(Q))_{i\in[K]}$, with independent randomness in each invocation. The symbol of $\mathcal{C}^{\equiv K}$ at a location consists of the symbols of all blocks at that location, so

$$\mathsf{Enc}_{\mathcal{C}^{\equiv K}}(\boldsymbol{f},\boldsymbol{r})[Q] = (\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_i,\boldsymbol{r}_i)[Q])_{i\in[K]}\ ,$$

where the $\boldsymbol{r}_i$ are independent and uniform. Replace the blocks one at a time by $\mathsf{Sim}_{\mathcal{C}}(Q)$. Consecutive hybrids differ only in one block, whose randomness is independent of Q and of the other blocks, so by the zero-knowledge property of $\mathsf{Enc}_{\mathcal{C}}$ each replacement changes the statistical distance by at most $\zeta_{\mathcal{C}}$. Summing over the K blocks gives the bound. □

We define zero-knowledge encodings with spare coordinates.

Definition 3.25. *Let $\mathsf{Enc}_{\mathcal{C}}\colon R^{\ell}\times R^{p+r}\to\Sigma^m$ be an encoding for $\mathcal{C}$, whose randomness consists of a mask $\boldsymbol{s}\in R^p$ and a seed $\boldsymbol{r}\in R^r$. Let*

$$\mathsf{Enc}^{+p}_{\mathcal{C}}\colon R^{\ell+p}\times R^r\to\Sigma^m,\qquad \mathsf{Enc}^{+p}_{\mathcal{C}}((\boldsymbol{v},\boldsymbol{s}),\boldsymbol{r}) \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{v},(\boldsymbol{s},\boldsymbol{r}))$$

be the same map with the mask regarded as part of the message. The encoding $\mathsf{Enc}_{\mathcal{C}}$ is a **t-query zero-knowledge encoding with p spare coordinates** *and error $\zeta_{\mathcal{C}}$ if $\mathsf{Enc}^{+p}_{\mathcal{C}}$ is a t-query zero-knowledge encoding for $\mathcal{C}$ with error $\zeta_{\mathcal{C}}$. The code $\mathcal{C}$ equipped with the encoding $\mathsf{Enc}^{+p}_{\mathcal{C}}$ is denoted $\mathcal{C}^{+p}$.*

An encoding with spare coordinates is a zero-knowledge encoding, and its interleaving has spare coordinates.

Claim 3.26. *Let $\mathsf{Enc}_{\mathcal{C}}$ be a t-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}}$.*
1. *$\mathsf{Enc}_{\mathcal{C}}$ is a t-query zero-knowledge encoding with error $\zeta_{\mathcal{C}}$.*
2. *For every $K\in\mathbb{N}$, the interleaved encoding $\mathsf{Enc}_{\mathcal{C}^{\equiv K}}$ of Claim 3.24 is a t-query zero-knowledge encoding with Kp spare coordinates and error $K\cdot\zeta_{\mathcal{C}}$. Its mask consists of the masks of its rows, and $(\mathcal{C}^{\equiv K})^{+Kp} = (\mathcal{C}^{+p})^{\equiv K}$.*

Proof. For the first item, we use the simulator of $\mathsf{Enc}^{+p}_{\mathcal{C}}$. Fix a message $\boldsymbol{v}$ of $\mathsf{Enc}_{\mathcal{C}}$. The distribution of the queried symbols of its encoding is the average, over the mask $\boldsymbol{s}$, of the corresponding distributions for the messages $(\boldsymbol{v},\boldsymbol{s})$ of $\mathsf{Enc}^{+p}_{\mathcal{C}}$, and each of these is $\zeta_{\mathcal{C}}$-close to the simulated one. For the second item, regarding the masks of all rows as part of the message turns $\mathsf{Enc}_{\mathcal{C}^{\equiv K}}$ into the interleaving of $\mathsf{Enc}^{+p}_{\mathcal{C}}$, to which Claim 3.24 applies. □

Every zero-knowledge encoding with message length $\ell+p$ gives one with message length ℓ and p spare coordinates, by regarding p message coordinates as randomness.

We define Reed–Solomon codes over the ring R and their zero-knowledge encodings.

Definition 3.27. *Let $\mathcal{L}\subseteq R$ be finite and $\ell\in\mathbb{N}$. The* **Reed–Solomon code over R with evaluation domain $\mathcal{L}$ and degree bound ℓ** *is*

$$\mathsf{RS}[R,\mathcal{L},\ell] \coloneqq \left\{ f\colon \mathcal{L}\to R \ :\ \exists\, \hat{f}\in R^{<\ell}[\mathtt{X}] \ \textit{s.t.}\ \forall\, x\in\mathcal{L}\ \ \hat{f}(x)=f(x) \right\}.$$

When $\mathcal{L}\subseteq\mathcal{Z}(R)$ we have that $\mathsf{RS}[R,\mathcal{L},\ell]$ is a right R-submodule of $R^{\mathcal{L}}$.

For $R=\mathbb{F}$ a finite field this is the usual Reed–Solomon code. When the evaluation domain is a strong sampling set, the code is MDS.

Lemma 3.28. *Let $\mathcal{L} \subseteq R$ be a strong sampling set with $|\mathcal{L}| \geq \ell$. Then*

$$\delta(\mathsf{RS}[R, \mathcal{L}, \ell]) \geq 1 - \frac{\ell - 1}{|\mathcal{L}|}\,.$$

Proof. If $\hat{f}, \hat{g} \in R^{<\ell}[\mathtt{X}]$ agree on more than $\ell - 1$ points of $\mathcal{L}$, then $\hat{f} - \hat{g}$ has more than $\ell - 1$ roots in a strong sampling set, so $\hat{f} = \hat{g}$ by Lemma 3.9. Uniqueness follows since $|\mathcal{L}| \geq \ell$, and the distance bound since two distinct codewords agree on at most $\ell - 1$ points. □

Lemma 3.29. *Let R be finite, let $\mathcal{L} \subseteq \mathcal{Z}(R)$ be a strong sampling set, and let $t \leq \ell \leq |\mathcal{L}|$. The encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell - t} \times R^t \to R^{\mathcal{L}}$ mapping $(\boldsymbol{v}, \boldsymbol{r})$ to the evaluation over $\mathcal{L}$ of*

$$\hat{f}(\mathtt{X}) \coloneqq \sum_{j \in [t]} \boldsymbol{r}_j \cdot \mathtt{X}^{j-1} + \sum_{i \in [\ell - t]} \boldsymbol{v}_i \cdot \mathtt{X}^{t+i-1}$$

is a strong t-query zero-knowledge encoding for $\mathsf{RS}[R, \mathcal{L}, \ell]$ with message length $\ell - t$, randomness length t, and error $\zeta_{\mathcal{C}} = 0$.

Proof. By Lemma 3.9, a polynomial in $R^{<\ell}[\mathtt{X}]$ is determined by its values at any ℓ points of $\mathcal{L}$. Hence $\mathsf{Enc}_{\mathcal{C}}$ is injective with image $\mathsf{RS}[R, \mathcal{L}, \ell]$, and its inverse is computable in polynomial time by interpolation. It is a right R-module homomorphism since $\mathcal{L} \subseteq \mathcal{Z}(R)$.

Fix $\boldsymbol{v}$ and distinct points $x_1, \ldots, x_s \in \mathcal{L}$ with $s \leq t$. Separating the coefficients $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_s$ from the rest,

$$\left(\hat{f}(x_i)\right)_{i \in [s]} = \psi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_s) + \boldsymbol{w} \quad \text{where} \quad \psi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_s) \coloneqq \left(\sum_{j \in [s]} \boldsymbol{r}_j \cdot x_i^{j-1}\right)_{i \in [s]}$$

and $\boldsymbol{w} \in R^s$ depends only on $\boldsymbol{v}, \boldsymbol{r}_{s+1}, \ldots, \boldsymbol{r}_t$. The map $\psi \colon R^s \to R^s$ is injective by Lemma 3.9 (if $\psi(\boldsymbol{r}) = \psi(\boldsymbol{r}')$ then $\sum_{j \in [s]} (\boldsymbol{r}_j - \boldsymbol{r}'_j) \cdot \mathtt{X}^{j-1}$ has the s roots $x_1, \ldots, x_s$), hence a bijection of the finite set R^s. Since $(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_s)$ is uniform and independent of $\boldsymbol{w}$, the vector $\psi(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_s) + \boldsymbol{w}$ is uniform in R^s, whatever $\boldsymbol{v}$ is. The simulator therefore outputs uniformly random answers. □

We further recall a linear-time encodable zero-knowledge code.

Theorem 3.30 ([BCL22, Theorem 6], informal)**.** *For every $\varepsilon \in (0, 1), \ell, t \in \mathbb{N}$, there exists a randomized linear time encoding algorithm that maps messages in $\mathbb{F}^{\ell}$ to a code $\mathcal{C} \subseteq \mathbb{F}^{O_\varepsilon(\ell + t)}$ such that this encoding is strong t-query zero-knowledge with error ε.*

3.3.2 Interleaving and folding for zero-knowledge encodings

We fix the indexing of the rows of an interleaved encoding when their number is a power of two, and define the folding of interleaved codewords.

Definition 3.31. *Let $\mathcal{C} \subseteq \Sigma^m$ be a code with a zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell} \times R^r \to \Sigma^m$. For a folding parameter $k \in \mathbb{N}$, the* **2^k-interleaved zero-knowledge encoding** *$\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}} \colon R^{2^k \ell} \times R^{2^k r} \to (\Sigma^{2^k})^m$ is the interleaved encoding of Claim 3.24 with $K = 2^k$, whose blocks we index by $\{0,1\}^k$ in binary order:*

$$\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}(\boldsymbol{f}, \boldsymbol{r}) \coloneqq (\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_{\boldsymbol{b}}, \boldsymbol{r}_{\boldsymbol{b}}))_{\boldsymbol{b} \in \{0,1\}^k} \in (\Sigma^{2^k})^m\,,$$

where $(\boldsymbol{f}_{\boldsymbol{b}})_{\boldsymbol{b} \in \{0,1\}^k}$ and $(\boldsymbol{r}_{\boldsymbol{b}})_{\boldsymbol{b} \in \{0,1\}^k}$ are the blocks of $\boldsymbol{f}$ and $\boldsymbol{r}$, of lengths ℓ and r.

By Claim 3.24, this is a t-query zero-knowledge encoding for $\mathcal{C}^{\equiv 2^k}$ with error $2^k \cdot \zeta_{\mathcal{C}}$ whenever $\mathsf{Enc}_{\mathcal{C}}$ is one with error $\zeta_{\mathcal{C}}$.

Definition 3.32. *Let Σ be an R-module, and let $f \in (\Sigma^{2^k})^m$. For $i \in [k]$, the* **interleaved fold** *of f at $\boldsymbol{\gamma} \in R^i$ is the function $\mathsf{Fold}(f, \boldsymbol{\gamma}) \in (\Sigma^{2^{k-i}})^m$ defined as follows. We identify $\{0,1\}^k \cong \{0,1\}^i \times \{0,1\}^{k-i}$, and for every $z \in [m]$ we write $f(z) = \left(f_{(\boldsymbol{b},\boldsymbol{c})}(z)\right)_{\boldsymbol{b}\in\{0,1\}^i, \boldsymbol{c}\in\{0,1\}^{k-i}}$ to mean that $f_{(\boldsymbol{b},\boldsymbol{c})}(z) \in \Sigma$ is the $(\boldsymbol{b},\boldsymbol{c})$-th coordinate of $f(z) \in \Sigma^{2^k}$. Then we define*

$$\forall\, z \in [m],\ \mathsf{Fold}(f, \boldsymbol{\gamma})(z) \coloneqq \left(\sum_{\boldsymbol{b}\in\{0,1\}^i} f_{(\boldsymbol{b},\boldsymbol{c})}(z) \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \right)_{\boldsymbol{c}\in\{0,1\}^{k-i}} .$$

Remark 3.33. Let Σ, f, i be as in Definition 3.32. The interleaved fold of f is well-defined when $\boldsymbol{\gamma} \in R^i$ is replaced with indeterminates $(\mathtt{X}_1, \ldots, \mathtt{X}_i)$. Indeed, for every $z \in [m]$,

$$\mathsf{Fold}(f, (\mathtt{X}_1, \ldots, \mathtt{X}_i))(z) \coloneqq \left(\sum_{\boldsymbol{b}\in\{0,1\}^i} f_{(\boldsymbol{b},\boldsymbol{c})}(z) \cdot \mathsf{eq}(\boldsymbol{b}, \mathtt{X}_1, \ldots, \mathtt{X}_i) \right)_{\boldsymbol{c}\in\{0,1\}^{k-i}} \in (\Sigma^{<2}[\mathtt{X}_1, \ldots, \mathtt{X}_i])^{2^{k-i}} .$$

Lemma 3.34. *Let $\mathcal{C} \subseteq \Sigma^m$ be an (R, Σ)-additive code with a zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^\ell \times R^r \to \Sigma^m$ and generating matrix $\boldsymbol{G}_{\mathcal{C}}$ (as in Definition 3.23). For every $\boldsymbol{f} \in R^{2^k \ell}$, $\boldsymbol{r} \in R^{2^k r}$, $j \in [k]$, and $\boldsymbol{\gamma} \in R^j$,*

$$\mathsf{Fold}\left(\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}(\boldsymbol{f}, \boldsymbol{r}), \boldsymbol{\gamma}\right) = \mathsf{Enc}_{\mathcal{C}^{\equiv 2^{k-j}}}\left(\mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}), \mathsf{Fold}(\boldsymbol{r}, \boldsymbol{\gamma})\right) .$$

Proof. We identify $\{0,1\}^k \cong \{0,1\}^j \times \{0,1\}^{k-j}$. By the definition of $\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}$ (Definition 3.31),

$$\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}(\boldsymbol{f}, \boldsymbol{r}) = \left(\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})}, \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})})\right)_{(\boldsymbol{b},\boldsymbol{c})\in\{0,1\}^j\times\{0,1\}^{k-j}} .$$

By the definition of Fold over Σ (Definition 3.32) and the above equality,

$$\mathsf{Fold}\left(\mathsf{Enc}_{\mathcal{C}^{\equiv 2^k}}(\boldsymbol{f}, \boldsymbol{r}), \boldsymbol{\gamma}\right) = \left(\sum_{\boldsymbol{b}\in\{0,1\}^j} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})}, \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})}) \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \right)_{\boldsymbol{c}\in\{0,1\}^{k-j}} .$$

Using Definition 3.23 for each summand, for every $\boldsymbol{b} \in \{0,1\}^j$ and $\boldsymbol{c} \in \{0,1\}^{k-j}$,

$$\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})}, \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})}) = \boldsymbol{G}_{\mathcal{C}} \cdot \begin{bmatrix} \boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})} \\ \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})} \end{bmatrix} = \boldsymbol{G}^{\#}_{\mathcal{C}} \cdot \boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})} + \boldsymbol{G}^{\$}_{\mathcal{C}} \cdot \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})} .$$

Therefore,

$$\begin{aligned}
&\left(\sum_{\boldsymbol{b}\in\{0,1\}^j} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})}, \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})}) \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \right)_{\boldsymbol{c}\in\{0,1\}^{k-j}} \\
&= \boldsymbol{G}^{\#}_{\mathcal{C}} \cdot \left(\sum_{\boldsymbol{b}\in\{0,1\}^j} \boldsymbol{f}_{(\boldsymbol{b},\boldsymbol{c})} \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \right)_{\boldsymbol{c}\in\{0,1\}^{k-j}} + \boldsymbol{G}^{\$}_{\mathcal{C}} \cdot \left(\sum_{\boldsymbol{b}\in\{0,1\}^j} \boldsymbol{r}_{(\boldsymbol{b},\boldsymbol{c})} \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \right)_{\boldsymbol{c}\in\{0,1\}^{k-j}} \\
&\qquad\qquad \text{(by linearity)} \\
&= \boldsymbol{G}^{\#}_{\mathcal{C}} \cdot \mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}) + \boldsymbol{G}^{\$}_{\mathcal{C}} \cdot \mathsf{Fold}(\boldsymbol{r}, \boldsymbol{\gamma}) \qquad \text{(by definition of } \mathsf{Fold}) \\
&= \mathsf{Enc}_{\mathcal{C}^{\equiv 2^{k-j}}}\left(\mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}), \mathsf{Fold}(\boldsymbol{r}, \boldsymbol{\gamma})\right) .
\end{aligned}$$

□

4 Zero-knowledge for IORs and composition

Definition 4.1. *Let* $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ *be an IOR from* $\mathcal{R}$ *to* $\mathcal{R}'$ *and let* $\mathcal{D}$ *be an algorithm. The* **$\mathcal{D}$-extended view for** IOR *on input* $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$ *is the random variable*

$$\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}, \mathbb{x}, \mathbb{y}, \mathbb{w}) = ((\rho_1, \ldots, \rho_{\mathsf{k}}), (Q_{\mathbb{y}}, Q_1, \ldots, Q_{\mathsf{k}}), (\boldsymbol{a}_{\mathbb{y}}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{\mathsf{k}}), \mathsf{out})$$

obtained by running the experiment

$$\left[\begin{array}{r} (\mathbb{x}', \mathbb{y}', \mathbb{w}') \leftarrow \langle \mathbf{P}, \mathbf{V} \rangle (\mathbb{x}, \mathbb{y}, \mathbb{w}) \\ \mathsf{out} \leftarrow \mathcal{D}^{\mathbb{y}'}(\mathbb{x}') \end{array}\right]$$

where out *may be any output string of* $\mathcal{D}$*, and making the following definitions for every* $i \in [\mathsf{k}]$*:*

- $\rho_i \in \{0,1\}^{\mathsf{r}_i}$ *is the verifier randomness in round* i*,*
- $Q_{\mathbb{y}}$ *is the ordered list of queries to* $\mathbb{y}$ *made by* $\mathbf{V}$ *or induced by the queries of* $\mathcal{D}$ *to* $\mathbb{y}'$ *(allowing repetitions), and* $\boldsymbol{a}_{\mathbb{y}}$ *is the list of corresponding answers, and*
- Q_i *is the ordered list of queries to the proof oracle* Π_i *sent in round* i*, made by* $\mathbf{V}$ *or induced by the queries of* $\mathcal{D}$ *to* $\mathbb{y}'$ *(allowing repetitions), and* $\boldsymbol{a}_i$ *is the list of corresponding answers.*

We view non-oracle messages as oracle messages that the verifier queries entirely.

From now on a witness is a pair $\mathbb{w} = (\mathbb{m}, \mathbb{r})$ of a *message* and *randomness*, where the randomness may be empty. For a relation $\mathcal{R}$, an explicit instance $\mathbb{x}$, and a message $\mathbb{m}$, let

$$\mathcal{R}[\mathbb{x}, \mathbb{m}] \coloneqq \{\mathbb{r} \ : \ (\mathbb{x}, \mathbb{y}, (\mathbb{m}, \mathbb{r})) \in \mathcal{R} \text{ for some } \mathbb{y}\} \ .$$

Definition 4.2. *Let* $\mathbb{x}$ *and* $\mathbb{m}$ *be such that* $\mathcal{R}[\mathbb{x}, \mathbb{m}] \neq \emptyset$*. A* **fresh input for** $(\mathbb{x}, \mathbb{m})$ *is a random triple* $(\mathbb{x}, \mathbb{y}, (\mathbb{m}, \mathbb{r}))$ *in* $\mathcal{R}$ *whose randomness* $\mathbb{r}$ *is uniformly distributed over* $\mathcal{R}[\mathbb{x}, \mathbb{m}]$*.*

Definition 4.3. *Let* $\mathsf{IOR} = (\mathbf{P}, \mathbf{V})$ *be an IOR from* $\mathcal{R}$ *to* $\mathcal{R}'$ *and let* $\mathfrak{D}$ *be a set of algorithms.* IOR *is* **honest-verifier zero-knowledge on fresh inputs for** $\mathfrak{D}$ **with error** ζ **and query complexity** b *if there exists a polynomial-time probabilistic simulator* $\mathbf{S}$ *such that, for every* $\mathcal{D} \in \mathfrak{D}$ *and every fresh input* $(\mathbb{x}, \mathbb{y}, \mathbb{w})$*, the distributions of*

$$\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}, \mathbb{x}, \mathbb{y}, \mathbb{w}) \qquad \text{and} \qquad \mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$$

have statistical distance at most $\zeta(\mathcal{D})$*,* $\mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$ *makes at most* $\mathsf{b}(\mathcal{D})$ *queries to* $\mathbb{y}$*, and the components* $(Q_{\mathbb{y}}, \boldsymbol{a}_{\mathbb{y}})$ *in the output of* $\mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$ *are its queries to* $\mathbb{y}$ *and their answers.* IOR *is* **honest-verifier zero-knowledge** *if the above holds for every* $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}$ *in place of every fresh input.*

Remark 4.4. When the IOR is from $\mathcal{R}$ to the trivial relation $\mathcal{R}_{\mathsf{triv}}$ (i.e., the IOR is an IOP), Definition 4.3 implies honest-verifier zero-knowledge for the IOP.

We define IORs with fresh output.

Definition 4.5. *An IOR from* $\mathcal{R}$ *to* $\mathcal{R}'$ **has fresh output** *if its output implicit instance* $\mathbb{y}'$ *is a proof string sent by its prover and, for every input in* $\mathcal{R}$*, the output* $(\mathbb{x}', \mathbb{y}', (\mathbb{m}', \mathbb{r}'))$ *of an honest execution satisfies the following: conditioned on the view of the verifier and on* $\mathbb{m}'$*, the randomness* $\mathbb{r}'$ *is uniformly distributed over* $\mathcal{R}'[\mathbb{x}', \mathbb{m}']$*.*

Definition 4.6. *Given algorithms* $\mathbf{S}$ *and* $\mathcal{D}$*, we let* $\mathcal{D}_{\mathbf{S}}$ *be the algorithm such that* $\mathcal{D}_{\mathbf{S}}^{\mathbb{y}}(\mathbb{x}) \coloneqq \mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$*. For a set of algorithms* $\mathfrak{D}$ *we define the* **S-wise distinguisher class** $\mathfrak{D}_{\mathbf{S}} \coloneqq \{\mathcal{D}_{\mathbf{S}} \ : \ \mathcal{D} \in \mathfrak{D}\}$*.*

Theorem 4.7. *Let* IOR_1 *be an IOR from* $\mathcal{R}_{\mathsf{start}}$ *to* $\mathcal{R}_{\mathsf{int}}$ *and let* IOR_2 *be an IOR from* $\mathcal{R}_{\mathsf{int}}$ *to* $\mathcal{R}_{\mathsf{fin}}$ *that have the following parameters (respectively).*

- *Round complexities* k_1 *and* k_2*.*
- *Prover communication* l_1 *and* l_2*.*
- *Input query complexities* $\mathsf{q}_{\mathbb{y},1}$ *and* $\mathsf{q}_{\mathbb{y},2}$*.*
- *Proof query complexities* q_1 *and* q_2*.*
- *Prover times* pt_1 *and* pt_2*.*
- *Verifier times* vt_1 *and* vt_2*.*
- IOR_1 *has input query blowup* $m_{\mathbb{y}' \to \mathbb{y}}$ *and proof query blowup* $m_{\mathbb{y}' \to \Pi}$*.*

Construction 4.8 is an IOR IOR *from* $\mathcal{R}_{\mathsf{start}}$ *to* $\mathcal{R}_{\mathsf{fin}}$ *with the following parameters.*

- *Round complexity* $\mathsf{k}_1 + \mathsf{k}_2$*.*
- *Prover communication* $\mathsf{l}_1 + \mathsf{l}_2$*.*
- *Input query complexity* $\mathsf{q}_{\mathbb{y},1} + m_{\mathbb{y}' \to \mathbb{y}} \cdot \mathsf{q}_{\mathbb{y},2}$*.*
- *Proof query complexity* $\mathsf{q}_1 + \mathsf{q}_2 + m_{\mathbb{y}' \to \Pi} \cdot \mathsf{q}_{\mathbb{y},2}$*.*
- *Prover time* $\mathsf{pt}_1 + \mathsf{pt}_2$*.*
- *Verifier time* $\mathsf{vt}_1 + \mathsf{vt}_2$*, plus the cost of evaluating the queried intermediate symbols.*

Moreover:

- ***RBR security.*** *If* IOR_1 *has RBR knowledge soundness with relaxation* $(\tilde{\mathcal{R}}_{\mathsf{start}}, \tilde{\mathcal{R}}_{\mathsf{int}})$ *and errors* $(\varepsilon_1^{(1)}, \ldots, \varepsilon_{\mathsf{k}_1}^{(1)})$ *and* IOR_2 *has RBR knowledge soundness with relaxation* $(\tilde{\mathcal{R}}_{\mathsf{int}}, \tilde{\mathcal{R}}_{\mathsf{fin}})$ *and errors* $(\varepsilon_1^{(2)}, \ldots, \varepsilon_{\mathsf{k}_2}^{(2)})$*, then* IOR *has RBR knowledge soundness with relaxation* $(\tilde{\mathcal{R}}_{\mathsf{start}}, \tilde{\mathcal{R}}_{\mathsf{fin}})$ *and errors* $(\varepsilon_1^{(1)}, \ldots, \varepsilon_{\mathsf{k}_1}^{(1)}, \varepsilon_1^{(2)}, \ldots, \varepsilon_{\mathsf{k}_2}^{(2)})$*.*
- ***Zero-knowledge.*** *Suppose that* IOR_2 *is HVZK on fresh inputs for* $\mathfrak{D}$ *with error* ζ_2 *via the simulator* $\mathbf{S}_2$*, and that* IOR_1 *has fresh output and is HVZK on fresh inputs for* $\mathfrak{D}_{\mathbf{S}_2}$ *with error* ζ_1 *and query complexity* b *via the simulator* $\mathbf{S}_1$*. Then* IOR *is HVZK on fresh inputs for* $\mathfrak{D}$ *with error* $\zeta_1 + \zeta_2$ *and query complexity* b*. The simulator for* IOR *queries* $\mathbb{y}_1$ *exactly as* $\mathbf{S}_1$ *does. If* IOR_2 *has fresh output, then so does* IOR*. If* IOR_2 *is HVZK for* $\mathfrak{D}$ *with error* ζ_2 *via* $\mathbf{S}_2$*, then* IOR_1 *need not have fresh output.*

Construction 4.8. The IOR consists of the concatenation of IOR_1 and IOR_2.

- **Inputs.** The verifier receives as explicit input $\mathbb{x}_1$ and query access to $\mathbb{y}_1$. The honest prover receives as input $\mathbb{w}_1$ such that $(\mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1) \in \mathcal{R}_{\mathsf{start}}$.
- **Protocol.**
  1. **First IOR.** The prover and verifier run IOR_1 for $(\mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$. After the interaction, the verifier outputs an explicit instance $\mathbb{x}_2$ and implicit instance $\mathbb{y}_2$ (defined via $\mathbb{y}_1$ and proof oracles sent in the execution of IOR_1). The honest prover outputs a witness $\mathbb{w}_2$ such that $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2) \in \mathcal{R}_{\mathsf{int}}$.
  2. **Second IOR.** The prover and verifier run IOR_2 for $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$. After the interaction, the verifier outputs an explicit instance $\mathbb{x}_3$ and implicit instance $\mathbb{y}_3$. The honest prover outputs a witness $\mathbb{w}_3$ such that $(\mathbb{x}_3, \mathbb{y}_3, \mathbb{w}_3) \in \mathcal{R}_{\mathsf{fin}}$.
- **Output.** The verifier outputs explicit instance $\mathbb{x}_3$ and implicit instance $\mathbb{y}_3$. The honest prover outputs a witness $\mathbb{w}_3$.

Proof. The efficiency parameters follow via inspection. For RBR knowledge soundness, define the knowledge state function of IOR to be that of IOR_1 on transcripts of IOR_1, and afterwards that of IOR_2 on the instance output by $\mathbf{V}_1$, or zero if $\mathbf{V}_1$ rejected. Both functions express membership in $\tilde{\mathcal{R}}_{\mathsf{int}}$ once the transcript of IOR_1 is complete, so this is well defined, and the extractors of IOR_1 and IOR_2 give the stated errors. We now establish HVZK of the composed IOR. If out_1 is not of the form below, we use fixed values of the appropriate types and set $(Q'_{\mathbb{y}_2}, \boldsymbol{a}'_{\mathbb{y}_2}) = (\emptyset, \emptyset)$. Consider the following simulator.

$\mathbf{S}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1)$:
1. Define the algorithm $\mathcal{D}_1^{\mathbb{y}_2}(\mathbb{x}_2) \coloneqq \mathbf{S}_2^{\mathbb{y}_2,\mathcal{D}_2}(\mathbb{x}_2)$.
2. Compute $(\boldsymbol{\rho}, \boldsymbol{Q}, \boldsymbol{a}, \mathsf{out}_1) \leftarrow \mathbf{S}_1^{\mathbb{y}_1,\mathcal{D}_1}(\mathbb{x}_1)$.
3. Parse out_1 as an extended view $(\boldsymbol{\rho}', (Q'_{\mathbb{y}_2}, \boldsymbol{Q}'), (\boldsymbol{a}'_{\mathbb{y}_2}, \boldsymbol{a}'), \mathsf{out}_2)$ of IOR_2.
4. Output $(\boldsymbol{\rho}\|\boldsymbol{\rho}', \boldsymbol{Q}\|\boldsymbol{Q}', \boldsymbol{a}\|\boldsymbol{a}', \mathsf{out}_2)$, an extended view of the composed IOR.

We argue that, for every $\mathcal{D}_2 \in \mathfrak{D}$ and every fresh input $(\mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$ of $\mathcal{R}_{\mathsf{start}}$, the following two distributions are $(\zeta_1 + \zeta_2)$-close in statistical distance:

$$\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}_2, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1) \quad \text{and} \quad \mathbf{S}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1)\,.$$

Note that $\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}_2, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$ is equivalent to the following experiment.

$\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}_2, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$:
1. Compute $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2) \xleftarrow{\mathsf{view}_1} \langle \mathbf{P}_1, \mathbf{V}_1\rangle(\mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$ where view_1 is $\mathbf{V}_1$'s view in the interaction.
2. Compute $(\mathbb{x}_3, \mathbb{y}_3, \mathbb{w}_3) \xleftarrow{\mathsf{view}_2} \langle \mathbf{P}_2, \mathbf{V}_2\rangle(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$ where view_2 is $\mathbf{V}_2$'s view in the interaction.
3. Compute $\mathsf{out}_2 \leftarrow \mathcal{D}_2^{\mathbb{y}_3}(\mathbb{x}_3)$.
4. Let view be $\mathbf{V}$'s view obtained by combining view_1 and view_2.
5. Output $(\mathsf{view}, \mathsf{out}_2)$.

We rely on a two-step hybrid argument. For that, we define a hybrid simulator $\widetilde{\mathbf{S}}$.

$\widetilde{\mathbf{S}}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1, \mathbb{w}_1)$:
1. Define the algorithm $\mathcal{D}_1^{\mathbb{y}_2}(\mathbb{x}_2) \coloneqq \mathbf{S}_2^{\mathbb{y}_2,\mathcal{D}_2}(\mathbb{x}_2)$.
2. Compute $(\boldsymbol{\rho}, \boldsymbol{Q}, \boldsymbol{a}, \mathsf{out}_1) \leftarrow \mathsf{View}(\mathbf{P}_1, \mathbf{V}_1, \mathcal{D}_1, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$.
3. Parse out_1 as an extended view $(\boldsymbol{\rho}', (Q'_{\mathbb{y}_2}, \boldsymbol{Q}'), (\boldsymbol{a}'_{\mathbb{y}_2}, \boldsymbol{a}'), \mathsf{out}_2)$ of IOR_2.
4. Output $(\boldsymbol{\rho}\|\boldsymbol{\rho}', \boldsymbol{Q}\|\boldsymbol{Q}', \boldsymbol{a}\|\boldsymbol{a}', \mathsf{out}_2)$, an extended view of the composed IOR.

We consider the two steps of the hybrid argument.

- $\Delta(\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}_2, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1), \widetilde{\mathbf{S}}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1, \mathbb{w}_1)) \le \zeta_2$. We prove this for every fixed input in $\mathcal{R}_{\mathsf{start}}$, which implies it for a fresh one. We condition on the view of $\mathbf{V}_1$ and on the message $\mathbb{m}_2$ of the output witness $\mathbb{w}_2 = (\mathbb{m}_2, \mathbb{r}_2)$. This fixes $\mathbb{x}_2$, and, since IOR_1 has fresh output, $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$ is then a fresh input for $(\mathbb{x}_2, \mathbb{m}_2)$. The execution of IOR_2 and the output of $\mathcal{D}_2$ depend on the execution of IOR_1 only through this triple. By IOR_2's HVZK property on fresh inputs, since $\mathcal{D}_2 \in \mathfrak{D}$, the statistical distance of $\mathsf{View}(\mathbf{P}_2, \mathbf{V}_2, \mathcal{D}_2, \mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$ and $\mathbf{S}_2^{\mathbb{y}_2,\mathcal{D}_2}(\mathbb{x}_2)$ is at most ζ_2 under this conditioning. Each of the two contains its queries to $\mathbb{y}_2$ and their answers. Since $\mathbb{y}_2$ is a proof string of IOR_1, these are queries to this proof string, and the two sides of this step are obtained from the two random variables by the same map, which combines them with the view of $\mathbf{V}_1$. Statistical distance does not increase under this map. The bound holds for every such conditioning and hence also after averaging.

If IOR_2 is HVZK, we condition instead on the entire execution of IOR_1. This fixes $(\mathbb{x}_2, \mathbb{y}_2, \mathbb{w}_2)$ and the answer to every query that a query to $\mathbb{y}_2$ induces, so the same argument applies without the assumption that IOR_1 has fresh output.

- $\Delta(\widetilde{\mathbf{S}}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1, \mathbb{w}_1), \mathbf{S}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1)) \le \zeta_1$. Since $\mathcal{D}_2 \in \mathfrak{D}$ we know that $\mathcal{D}_1 \in \mathfrak{D}_{\mathbf{S}_2}$ (by definition of $\mathcal{D}_1$). By IOR_1's HVZK property on fresh inputs, the statistical distance of $\mathsf{View}(\mathbf{P}_1, \mathbf{V}_1, \mathcal{D}_1, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1)$ and $\mathbf{S}_1^{\mathbb{y}_1,\mathcal{D}_1}(\mathbb{x}_1)$ is at most ζ_1. Items 3 and 4 apply the same map to the two random variables, and statistical distance does not increase under this map.

We conclude that

$$\begin{aligned}
&\Delta(\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}_2, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1), \mathbf{S}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1)) \\
&\le \Delta(\mathsf{View}(\mathbf{P}, \mathbf{V}, \mathcal{D}_2, \mathbb{x}_1, \mathbb{y}_1, \mathbb{w}_1), \widetilde{\mathbf{S}}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1, \mathbb{w}_1)) + \Delta(\widetilde{\mathbf{S}}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1, \mathbb{w}_1), \mathbf{S}^{\mathbb{y}_1,\mathcal{D}_2}(\mathbb{x}_1)) \\
&\le \zeta_1 + \zeta_2 \,.
\end{aligned}$$

The simulator is polynomial-time as a composition of polynomial-time algorithms, and its number of queries to $\mathbb{y}_1$ is exactly that of $\mathbf{S}_1$, i.e., at most b. The components $(Q_{\mathbb{y}_1}, \boldsymbol{a}_{\mathbb{y}_1})$ of $\mathbf{S}_1$'s output are unchanged in steps 3 and 4, so the output of $\mathbf{S}$ contains its queries to $\mathbb{y}_1$ and their answers, as Definition 4.3 requires.

It remains to show that IOR has fresh output if IOR_2 does. The output implicit instance $\mathbb{y}_3$ is a proof string of IOR_2 and hence of IOR. Fix an input of IOR and condition on the execution of IOR_1, which fixes the input of IOR_2. Since IOR_2 has fresh output, conditioned in addition on the view of $\mathbf{V}_2$ and on $\mathbb{m}_3$, the randomness $\mathbb{r}_3$ is uniform over $\mathcal{R}_{\mathsf{fin}}[\mathbb{x}_3, \mathbb{m}_3]$. These conditions determine the view of $\mathbf{V}$ and the message $\mathbb{m}_3$. The randomness $\mathbb{r}_3$ therefore remains uniform when we condition on these two alone. □

We introduce the class of uniform-query distinguishers, which are distinguishers that make a bounded number of uniformly random queries to the implicit instance (and are otherwise unbounded). We only consider distinguishers $\mathcal{D}$ that are *non-adaptive*, i.e., $\mathcal{D}$ is specified by a pair $(\mathcal{D}_0, \mathcal{D}_1)$ such that

$$\forall\, \mathbb{x}, \mathbb{y} :\ \mathcal{D}^{\mathbb{y}}(\mathbb{x}) = \left[\begin{array}{l} (\mathsf{st}, t') \leftarrow \mathcal{D}_0(\mathbb{x}) \\ Q \leftarrow [m]_{\mathbb{y}}^{t'} \\ \mathcal{D}_1(\mathsf{st}, Q, \mathbb{y}[Q]) \end{array}\right],$$

where $[m]_{\mathbb{y}}$ is the domain of $\mathbb{y}$. We require t' to be computed from the randomness of $\mathcal{D}_0$ alone. A distinguisher always stands for the remaining part of a protocol, whose verifier samples its query locations uniformly at random. This lets us rely on Definition 3.21 rather than on its strong variant.

Definition 4.9. *Let $t \in \mathbb{N}$. We define the class of* **uniform-query distinguishers** $\mathfrak{D}^{\le t}$ *as the set of algorithms of the above form with $t' \le t$.*

In our relations the implicit instance is a single oracle, so one query list suffices above.

Uniform-query simulators. Theorem 4.7 requires IOR_1 to be HVZK for $\mathfrak{D}_{\mathbf{S}_2}$, so we give a condition under which this class is again of the above form.

Definition 4.10. *A simulator $\mathbf{S}$ as in Definition 4.3, for a class $\mathfrak{D}$ of distinguishers, is* **uniform-query** *if, for every $\mathcal{D} \in \mathfrak{D}$, it is specified by two algorithms $(\mathbf{S}_0^{\mathcal{D}}, \mathbf{S}_1^{\mathcal{D}})$ such that*

$$\mathbf{S}^{\mathbb{y},\mathcal{D}}(\mathbb{x}) = \left[\begin{array}{l} (\mathsf{st}, t') \leftarrow \mathbf{S}_0^{\mathcal{D}}(\mathbb{x}) \\ Q \leftarrow [m]_{\mathbb{y}}^{t'} \\ \mathbf{S}_1^{\mathcal{D}}(\mathsf{st}, Q, \mathbb{y}[Q]) \end{array}\right].$$

The query count t' is computed from the randomness of $\mathbf{S}$ and $\mathcal{D}$ alone, and Q contains exactly the queries that $\mathbf{S}$ makes to $\mathtt{y}$.

Claim 4.11. *If $\mathbf{S}$ is a uniform-query simulator for a class $\mathfrak{D}$ with query complexity b, then $\mathfrak{D}_{\mathbf{S}} \subseteq \mathfrak{D}^{\leq \mathsf{b}}$.*

Proof. Let $\mathcal{D} \in \mathfrak{D}$. The two algorithms in Definition 4.10 also define $\mathcal{D}_{\mathbf{S}}$. Its first algorithm is $\mathbf{S}_0^{\mathcal{D}}$ and outputs the state and query count. The query list is then sampled as in the definition of $\mathfrak{D}^{\leq \mathsf{b}}$, and its second algorithm is $\mathbf{S}_1^{\mathcal{D}}$. These are exactly the queries of $\mathcal{D}_{\mathbf{S}}$ to its implicit instance, and there are at most b of them. □

5 Succinct linear forms

We define succinct linear forms, which compute the matrix of a linear claim from a short description of the claim.

Definition 5.1. *A* **succinct linear form** sl *is a map from a state* st *to a matrix* $\mathsf{sl}(\mathsf{st}) \in \mathcal{M}^{t\times\ell}$*. We use the notation* $\mathsf{sl} \in \langle \mathcal{M}^{t\times\ell}|$ *to indicate the dimensions of the matrix, and* $\mathsf{sl} \in \langle \mathcal{M}^{\ell}|$ *when* $t = 1$*. The maximum number (over the choice of* st*) of ring operations for computing* $\mathsf{sl}(\mathsf{st})$ *is denoted* t_{sl}*.*

A zero-evader induces a succinct linear form.

Definition 5.2. *Let* $\mathsf{ze}\colon D_{\mathsf{ze}} \to \mathcal{M}^{t\times\ell}$ *be a zero-evader over* R*. The* **ze-linear form** *is the linear form* $\mathsf{sl}_{\mathsf{ze}} \in \langle \mathcal{M}^{t\times\ell}|$ *that takes as input a state* $\rho \in D_{\mathsf{ze}}$ *and outputs* $\mathsf{ze}(\rho)$*. We write* $\mathsf{teval}(\mathsf{ze})$ *for the number of ring and module operations needed to compute* $\mathsf{ze}(\rho)$*, so* $t_{\mathsf{sl}_{\mathsf{ze}}} = \mathsf{teval}(\mathsf{ze})$*. When convenient (e.g., in batched linear forms below), we identify a zero-evader* ze *with its associated linear form* $\mathsf{sl}_{\mathsf{ze}}$*.*

Succinct linear forms can be combined and operated on.

Definition 5.3. *The* **scalar multiplied linear form for** $\mathsf{sl} \in \langle \mathcal{M}^{t\times\ell}|$ *is the linear form* $\times(\mathsf{sl}) \in \langle \mathcal{M}^{t\times\ell}|$ *mapping* $(\mathsf{st}, \varepsilon \in R)$ *to* $\mathsf{sl}(\mathsf{st}) \cdot \varepsilon$*. Note that* $t_{\times(\mathsf{sl})} \le t_{\mathsf{sl}} + O(t \cdot \ell)$*.*

Definition 5.4. *For* $k \in \mathbb{N}$*, the* k**-zero-padded linear form for** $\mathsf{sl} \in \langle \mathcal{M}^{t\times\ell}|$ *is the linear form* $0^k(\mathsf{sl}) \in \langle \mathcal{M}^{t\times(\ell+k)}|$ *mapping* st *to* $[\mathsf{sl}(\mathsf{st}), 0^{t\times k}]$*. Note that* $t_{0^k(\mathsf{sl})} \le t_{\mathsf{sl}}$*: the output may reuse the given state, so no copying is needed.*

Definition 5.5. *Let* $b \in \mathbb{N}$*,* $\mathsf{sl}_1, \ldots, \mathsf{sl}_b \in \langle \mathcal{M}^{t\times\ell}|$ *and let* $\mathsf{ze}\colon D_{\mathsf{ze}} \to R^b$ *be a zero-evader over* $\mathcal{M}$*. The* **ze-batched linear form** $\mathsf{ze}[\mathsf{sl}_1, \ldots, \mathsf{sl}_b] \in \langle \mathcal{M}^{t\times\ell}|$ *maps* $(\mathsf{st}_1, \ldots, \mathsf{st}_b, \rho)$ *to* $\sum_{i\in[b]} \mathsf{sl}_i(\mathsf{st}_i) \cdot \mathsf{ze}(\rho)_i$*. Note that* $t_{\mathsf{ze}[\mathsf{sl}_1,\ldots,\mathsf{sl}_b]} \le \mathsf{teval}(\mathsf{ze}) + \sum_{i\in[b]} t_{\mathsf{sl}_i} + O(b \cdot t \cdot \ell)$*.*

The most important operation that we apply to succinct linear forms is "fixing variables". This operation takes a succinct linear form of width 2^n and a vector of ring elements $\boldsymbol{\gamma} \in R^i$ (where $i \le n$), and outputs a succinct linear form of width 2^{n-i}, in a way that is compatible with the folding operation in Definition 3.32.

Definition 5.6. *Let* $i \in \{0, \ldots, n\}$ *and* $\mathsf{sl} \in \langle \mathcal{M}^{t\times 2^n}|$*. The* i**-fixing linear form associated to** sl *is the linear form* $\odot_i\mathsf{sl} \in \langle \mathcal{M}^{t\times 2^{n-i}}|$ *that maps* $(\mathsf{st}, \boldsymbol{\gamma} \in R^i)$ *to* $\boldsymbol{V}' \in \mathcal{M}^{t\times 2^{n-i}}$ *defined as follows:*
1. Let $\boldsymbol{V} = \mathsf{sl}(\mathsf{st}) \in \mathcal{M}^{t\times 2^n}$*.*
2. For each row $\boldsymbol{v}_j \in \mathcal{M}^{2^n}$ *of* $\boldsymbol{V}$*, set* $\boldsymbol{v}'_j \coloneqq \mathsf{Fold}(\boldsymbol{v}_j, \boldsymbol{\gamma}) \in \mathcal{M}^{2^{n-i}}$ *as in Definition 3.32 i.e.*

$$\forall\, \boldsymbol{c} \in \{0,1\}^{n-i},\ \boldsymbol{v}'_j[\boldsymbol{c}] \coloneqq \sum_{\boldsymbol{b}\in\{0,1\}^i} \boldsymbol{v}_j[\boldsymbol{b}, \boldsymbol{c}] \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma})\,.$$

3. Let $\boldsymbol{V}' \in \mathcal{M}^{t\times 2^{n-i}}$ *be the matrix whose rows are* $\boldsymbol{v}'_j$*.*
Note that $t_{\odot_i\mathsf{sl}} \le t_{\mathsf{sl}} + O(t \cdot 2^n)$*.*

The definition extends verbatim to linear forms whose width $2^i \cdot \ell$ is not a power of two: then $\boldsymbol{c}$ ranges over $[\ell]$ and $\odot_i\mathsf{sl} \in \langle \mathcal{M}^{t\times\ell}|$.

The fixing operation is used to reduce the instance size, and we illustrate this via an example. Consider the linear form $\mathsf{sl}_{\mathsf{ze}_{\mathsf{eq}}}$ associated to the zero-evader $\mathsf{ze}_{\mathsf{eq}}$ mapping $\boldsymbol{\alpha} \mapsto (\mathsf{eq}(\boldsymbol{\alpha}, \boldsymbol{b}))_{\boldsymbol{b} \in \{0,1\}^n}$. By [VSBW13], $t_{\mathsf{sl}_{\mathsf{zeeq}}} = O(2^n)$. Consider now $\odot_n \mathsf{sl}_{\mathsf{ze}_{\mathsf{eq}}}$, which by Claim 3.11 maps $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ to $\mathsf{eq}(\boldsymbol{\alpha}, \boldsymbol{\gamma})$. Note that $t_{\odot_n \mathsf{sl}_{\mathsf{zeeq}}} = O(n)$, which is logarithmic in the width 2^n of the original linear form.

Following Remark 3.33, the fixing linear form is well-defined when part of the input state consists of indeterminates. For $\mathsf{sl} \in \langle \mathcal{M}^{2^n} |$,

$$(\odot_i \mathsf{sl})(\mathsf{st}, (\mathtt{X}_1, \ldots, \mathtt{X}_i)) = \left(\sum_{\boldsymbol{b} \in \{0,1\}^i} \mathsf{sl}(\mathsf{st})[\boldsymbol{b}, \boldsymbol{c}] \cdot \mathsf{eq}(\boldsymbol{b}, \mathtt{X}_1, \ldots, \mathtt{X}_i) \right)_{\boldsymbol{c} \in \{0,1\}^{n-i}}, \tag{7}$$

and this extends naturally to $\mathsf{sl} \in \langle \mathcal{M}^{t \times 2^n} |$.

We introduce the main relation that we use in this work.

Definition 5.7. *Let $\mathcal{C} \subseteq \Sigma^m$ be a code with an encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^\ell \times R^r \to \Sigma^m$ and let $\mathsf{sl} \in \langle \mathcal{M}^\ell |$. Define:*

$$\mathcal{R}_{\mathcal{C},\mathsf{sl}} \coloneqq \left\{ \begin{pmatrix} \mathbb{x} = (\mu, \mathsf{st}), \\ \mathbb{y} = f, \\ \mathbb{w} = (\boldsymbol{f}, \boldsymbol{r}) \end{pmatrix} \;\middle|\; \begin{array}{l} f = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r}) \\ \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f} \rangle = \mu \end{array} \right\}.$$

For $\delta \in (0,1)$, define

$$\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}} \coloneqq \left\{ \begin{array}{l} (\mathbb{x}, \mathbb{y} = f, \\ (\mathbb{w}, \bar{\mathbb{y}} = \bar{f})) \end{array} \;\middle|\; \begin{array}{l} \Delta(\mathbb{y}, \bar{\mathbb{y}}) \leq \delta \\ (\mathbb{x}, \bar{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}} \end{array} \right\}.$$

A fresh input of this relation (Definition 4.2) is an encoding of the message with uniform randomness: if $\boldsymbol{f}$ satisfies the constraint of $\mathbb{x}$, then $\mathcal{R}_{\mathcal{C},\mathsf{sl}}[\mathbb{x}, \boldsymbol{f}] = R^r$.

6 Zero-knowledge sumcheck IOR

6.1 The mask polynomial

We define the polynomial that will be used to hide the sumcheck messages.

Definition 6.1. *Let $k \geq 1$ and $d_s \geq 2$, let $\boldsymbol{m} \in \mathcal{M}^{d_{\mathcal{M}}}$, and let $p \coloneqq d_{\mathcal{M}}(d_s - 1)$. We index the coordinates of R^p by pairs $(u, e) \in [d_{\mathcal{M}}] \times \{0, \dots, d_s - 2\}$. For $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_0, \dots, \boldsymbol{\sigma}_k) \in (R^p)^{k+1}$, let*

$$\hat{s}_{u,e}(\mathtt{X}_1, \dots, \mathtt{X}_k) \coloneqq \sum_{i=0}^{k} \boldsymbol{\sigma}_{i,u,e} \cdot \mathtt{X}_{i+1} \cdots \mathtt{X}_k \ \in \ R^{<2}[\mathtt{X}_1, \dots, \mathtt{X}_k]\,,$$

where the product is empty for $i = k$. The **mask polynomial** *of $\boldsymbol{\sigma}$ is*

$$\mathcal{U}^{\boldsymbol{\sigma},\boldsymbol{m}}(\mathtt{X}_1, \dots, \mathtt{X}_k) \coloneqq \sum_{u \in [d_{\mathcal{M}}]} \boldsymbol{m}_u \cdot \sum_{e=0}^{d_s-2} \hat{s}_{u,e}(\mathtt{X}_1, \dots, \mathtt{X}_k) \cdot \mathsf{pow}_e(\mathtt{X}_1, \dots, \mathtt{X}_k) \ \in \ \mathcal{M}^{<d_s}[\mathtt{X}_1, \dots, \mathtt{X}_k]\,,$$

where $\mathsf{pow}_0 \coloneqq 1$ and $\mathsf{pow}_e(\mathtt{X}_1, \dots, \mathtt{X}_k) \coloneqq \mathtt{X}_1^e + \cdots + \mathtt{X}_k^e$ for $e \geq 1$. Its linear form is $\mathsf{sl}_{\mathcal{U}} \in \langle \mathcal{M}^p |$, which maps $(\boldsymbol{\gamma}, \boldsymbol{m})$ to $(\boldsymbol{m}_u \cdot \mathsf{pow}_e(\boldsymbol{\gamma}))_{u,e}$.

The tuple $\boldsymbol{\sigma}$ consists of the coefficients of the polynomials $\hat{s}_{u,e}$, which have $k+1$ of the 2^k multilinear monomials. For $\mathcal{M} = R$, $\boldsymbol{m} = (1)$, and $d_s = 3$, the tuple $\boldsymbol{\sigma}$ has $2(k+1)$ coordinates and $\mathcal{U}^{\boldsymbol{\sigma}}(\mathtt{X}_1, \dots, \mathtt{X}_k) = \hat{s}_0(\mathtt{X}_1, \dots, \mathtt{X}_k) + \hat{s}_1(\mathtt{X}_1, \dots, \mathtt{X}_k) \cdot (\mathtt{X}_1 + \cdots + \mathtt{X}_k)$.

Lemma 6.2. *Let $\boldsymbol{\sigma}$ and $\boldsymbol{m}$ be as in Definition 6.1, let $\boldsymbol{s} \in (R^p)^{2^k}$ be the table of the values $\boldsymbol{s}_{\boldsymbol{b},u,e} \coloneqq \hat{s}_{u,e}(\boldsymbol{b})$ for $\boldsymbol{b} \in \{0,1\}^k$, and let $\boldsymbol{\gamma} \in \mathcal{Z}(R)^k$.*

1. *$\mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma}) = \sum_{i=0}^{k} \boldsymbol{\sigma}_i \cdot \gamma_{i+1} \cdots \gamma_k$ and $\mathcal{U}^{\boldsymbol{\sigma},\boldsymbol{m}}(\boldsymbol{\gamma}) = \langle \mathsf{sl}_{\mathcal{U}}(\boldsymbol{\gamma}, \boldsymbol{m}), \mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma}) \rangle$.*
2. *Suppose that R and $\mathcal{M}$ are finite and that $\boldsymbol{m}$ generates $\mathcal{M}$. If $\boldsymbol{\sigma}$ is uniformly distributed over $(R^p)^{k+1}$, then the sumcheck transcript*

$$\pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma}) \coloneqq \left(\sum_{\boldsymbol{b} \in \{0,1\}^k} \mathcal{U}^{\boldsymbol{\sigma},\boldsymbol{m}}(\boldsymbol{b}),\ \left(\sum_{\boldsymbol{b} \in \{0,1\}^{k-j}} \mathcal{U}^{\boldsymbol{\sigma},\boldsymbol{m}}(\boldsymbol{\gamma}_{<j}, \mathtt{X}, \boldsymbol{b}) \right)_{j \in [k]} \right)$$

is uniformly distributed over the set $\mathcal{T}_{\boldsymbol{\gamma}}$ of **consistent transcripts**, *that is, the tuples $(\tilde{\mu}, \hat{h}_1, \dots, \hat{h}_k)$ with $\tilde{\mu} \in \mathcal{M}$ and $\hat{h}_j \in \mathcal{M}^{<d_s}[\mathtt{X}]$ such that $\hat{h}_1(0) + \hat{h}_1(1) = \tilde{\mu}$ and $\hat{h}_{j+1}(0) + \hat{h}_{j+1}(1) = \hat{h}_j(\gamma_j)$ for every $j \in [k-1]$.*

Proof. Item 1. The polynomial $\hat{s}_{u,e}$ is multilinear. For central challenges, the (u,e)-th coordinate of $\mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma})$ is therefore $\sum_{\boldsymbol{b}} \boldsymbol{s}_{\boldsymbol{b},u,e} \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) = \hat{s}_{u,e}(\boldsymbol{\gamma})$ (Claim 3.11), which gives the first identity. The second follows from the definition of the mask polynomial, because the $\mathsf{pow}_e(\boldsymbol{\gamma})$ are central.

Item 2. The map $\pi_{\boldsymbol{\gamma}}$ is a homomorphism of abelian groups, and its image lies in $\mathcal{T}_{\boldsymbol{\gamma}}$ because $\pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma})$ consists of the sumcheck messages of a polynomial of individual degree less than d_s. Write $\hat{h}_j = \sum_{\tau < d_s} c_{j,\tau} \mathtt{X}^\tau$. Since $\hat{h}_j(0) + \hat{h}_j(1) = 2c_{j,0} + \sum_{\tau \geq 1} c_{j,\tau}$, each equation defining $\mathcal{T}_{\boldsymbol{\gamma}}$ determines the linear coefficient $c_{j,1}$ from the other coefficients of $\hat{h}_j$ and the preceding entries. Hence a tuple in $\mathcal{T}_{\boldsymbol{\gamma}}$ is given by a free choice of $\tilde{\mu}$ and of the $c_{j,\tau}$ with $\tau \neq 1$, and it suffices to show that the map

Φ that sends $\boldsymbol{\sigma}$ to these $1 + k(d_s - 1)$ entries of $\pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma})$ is onto. Indeed, a surjective homomorphism of finite groups maps the uniform distribution to the uniform distribution.

The map Φ treats each u separately: $\Phi(\boldsymbol{\sigma}) = \sum_u \boldsymbol{m}_u \cdot \Phi_0((\boldsymbol{\sigma}_{i,u,e})_{i,e})$, where Φ_0 is the map Φ for $\mathcal{M} = R$ and $\boldsymbol{m} = (1)$. As $\boldsymbol{m}$ generates $\mathcal{M}$, it suffices to show that Φ_0 is onto $R^{1+k(d_s-1)}$. We thus assume $\mathcal{M} = R$ and $\boldsymbol{m} = (1)$ and write $\hat{s}_e$ for $\hat{s}_{1,e}$ and $\boldsymbol{\sigma}_{i,e}$ for $\boldsymbol{\sigma}_{i,1,e}$.

For a multilinear polynomial P and $j \in [k]$, let $\lambda_j(P), \lambda'_j(P), \nu_j(P), \nu'_j(P) \in R$ be defined by

$$\sum_{\boldsymbol{b} \in \{0,1\}^{k-j}} P(\boldsymbol{\gamma}_{<j}, \mathtt{X}, \boldsymbol{b}) = \lambda_j(P) + \lambda'_j(P) \cdot \mathtt{X}, \qquad \sum_{\boldsymbol{b} \in \{0,1\}^{k-j}} |\boldsymbol{b}| \cdot P(\boldsymbol{\gamma}_{<j}, \mathtt{X}, \boldsymbol{b}) = \nu_j(P) + \nu'_j(P) \cdot \mathtt{X}\,,$$

where $|\boldsymbol{b}|$ is the Hamming weight of $\boldsymbol{b}$. For $e \geq 1$ we have $\mathsf{pow}_e(\boldsymbol{\gamma}_{<j}, \mathtt{X}, \boldsymbol{b}) = g_{j,e} + \mathtt{X}^e + |\boldsymbol{b}|$ with $g_{j,e} \coloneqq \sum_{i<j} \gamma_i^e$, so

$$\hat{h}_j(\mathtt{X}) = \lambda_j(\hat{s}_0) + \lambda'_j(\hat{s}_0)\mathtt{X} + \sum_{e=1}^{d_s-2} \Big(\big(\lambda_j(\hat{s}_e) + \lambda'_j(\hat{s}_e)\mathtt{X} \big) (g_{j,e} + \mathtt{X}^e) + \nu_j(\hat{s}_e) + \nu'_j(\hat{s}_e)\mathtt{X} \Big)\,.$$

Comparing coefficients, and setting $\hat{s}_{d_s-1} \coloneqq 0$,

$$\begin{aligned} c_{j,\tau} &= \lambda'_j(\hat{s}_{\tau-1}) + \lambda_j(\hat{s}_\tau) && \text{for } 2 \leq \tau \leq d_s - 1, \\ c_{j,0} &= \lambda_j(\hat{s}_0) + \kappa_j, \qquad \tilde{\mu} = \sum_{\boldsymbol{b} \in \{0,1\}^k} \hat{s}_0(\boldsymbol{b}) + \kappa_0\,, \end{aligned}$$

where $\kappa_0, \dots, \kappa_k$ depend only on $\hat{s}_1, \dots, \hat{s}_{d_s-2}$. Let $Q_i \coloneqq \mathtt{X}_{i+1} \cdots \mathtt{X}_k$ for $i \in \{0, \dots, k\}$, so that $Q_k = 1$ and $\hat{s}_e = \sum_i \boldsymbol{\sigma}_{i,e} \cdot Q_i$. We claim that the two maps

$$P \mapsto \Big(\lambda'_j(P)\Big)_{j \in [k]} \in R^k \qquad \text{and} \qquad P \mapsto \left(\sum_{\boldsymbol{b} \in \{0,1\}^k} P(\boldsymbol{b}), \big(\lambda_j(P)\big)_{j \in [k]} \right) \in R^{1+k}$$

are onto when restricted to the polynomials $P = \sum_i a_i \cdot Q_i$ with $a_0, \dots, a_k \in R$. Given a target, we then choose $(\boldsymbol{\sigma}_{i,d_s-2})_i, \dots, (\boldsymbol{\sigma}_{i,1})_i$ in this order so that the $c_{j,\tau}$ with $\tau \geq 2$ take their values, using the first map, and finally $(\boldsymbol{\sigma}_{i,0})_i$ so that $\tilde{\mu}$ and the $c_{j,0}$ take theirs, using the second.

For the claim, let $T \subseteq [k]$. The monomial $\prod_{i \in T} \mathtt{X}_i$ satisfies

$$\sum_{\boldsymbol{b} \in \{0,1\}^{k-j}} \left(\prod_{i \in T} \mathtt{X}_i \right) (\boldsymbol{\gamma}_{<j}, \mathtt{X}, \boldsymbol{b}) = 2^{k-j-|T \cap (j,k]|} \cdot \prod_{i \in T,\, i<j} \gamma_i \cdot \mathtt{X}^{[j \in T]}\,.$$

For $i \in [k]$, the monomial $Q_{i-1} = \mathtt{X}_i \mathtt{X}_{i+1} \cdots \mathtt{X}_k$ has $\lambda'_j(Q_{i-1}) = 0$ for $j < i$ and $\lambda'_i(Q_{i-1}) = 1$. For $i \in \{0, \dots, k\}$, the monomial Q_i has $\lambda_j(Q_i) = 0$ for $j > i$ and $\lambda_i(Q_i) = 1$ for $i \geq 1$, and $\sum_{\boldsymbol{b}} Q_0(\boldsymbol{b}) = 1$. In both cases the images of these monomials form a triangular matrix with unit diagonal, which is invertible over every ring. □

The prover computes the messages of $\pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma})$ with $O(kp)$ operations in R and $O(kd_{\mathcal{M}}d_s)$ applications of the module action. Indeed, for $P = \sum_{i=0}^k a_i \cdot \mathtt{X}_{i+1} \cdots \mathtt{X}_k$, the formula for monomials in the proof above gives $\lambda_k(P) = a_k$, $\nu_k(P) = 0$, $\lambda'_1(P) = a_0$, and

$$\begin{aligned} \lambda_j(P) &= a_j + 2\lambda_{j+1}(P), & \nu_j(P) &= (k-j) \cdot a_j + 2\nu_{j+1}(P) + \lambda_{j+1}(P), \\ \lambda'_{j+1}(P) &= \gamma_j \cdot \lambda'_j(P) + a_j, & \nu'_j(P) &= (k-j) \cdot \lambda'_j(P)\,. \end{aligned}$$

The prover computes λ_j and ν_j of every $\hat{s}_{u,e}$ before the first round and updates λ'_j in every round. It forms each round polynomial from these values with $O(d_{\mathcal{M}}d_s)$ operations.

6.2 Construction

Theorem 6.3. *Consider the following ingredients:*

- *a folding parameter $k \geq 1$, a masking degree $d_s \geq 3$, a tuple $\boldsymbol{m} \in \mathcal{M}^{d_{\mathcal{M}}}$ generating $\mathcal{M}$, and a finite nonempty challenge set $\mathfrak{C} \subseteq \mathcal{Z}(R)$,*
- *$\mathcal{C} \subseteq \Sigma^m$ with an encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^\ell \times R^{p+r} \to \Sigma^m$, where $p \coloneqq d_{\mathcal{M}}(d_s - 1)$,*
- *$\mathsf{sl} \in \langle \mathcal{M}^{2^k \ell} |$.*

Define the linear form $\mathsf{sl}' \in \langle \mathcal{M}^{\ell+p} |$ by

$$\mathsf{sl}'(\mathsf{st}', \mathsf{st}'_s) \coloneqq \left(\times(\odot_k \mathsf{sl})(\mathsf{st}'),\ \mathsf{sl}_{\mathcal{U}}(\mathsf{st}'_s)\right) .$$

Construction 6.4 is an IOR from $\mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$ to $\mathcal{R}_{\mathcal{C}^{+p}, \mathsf{sl}'}$ with the following properties.

- *Round complexity: $k+1$.*
- *Prover communication: $k \cdot d_s + 1$ elements of $\mathcal{M}$ sent directly, and no oracle.*
- *Verifier oracle queries: none. Each output-oracle query uses one input-oracle query.*
- *Prover time (in ring and module operations): $O\left(2^k(\ell + r) + t_{\mathsf{sl}} + kp\right)$.*
- *Verifier time (in ring and module operations): $O(kd_s)$.*
- **Round-by-round security:** *For every $\delta \in (0, \delta(\mathcal{C}))$, Construction 6.4 has RBR knowledge soundness with relaxation*

  $$\left(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}, \tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{+p}, \mathsf{sl}'}\right)$$

  and errors

  $$\left(\Lambda(\mathcal{C}^{\equiv 2^k}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2),\ \left(\epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv 2^{k-j}}, \delta) + \Lambda(\mathcal{C}^{\equiv 2^{k-j+1}}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, d_s)\right)_{j \in [k]}\right) .$$

  The total extraction time is $O\left(k \cdot 2^k m + \sum_{j \in [k]} \mathsf{tcor}(\mathcal{C}^{\equiv 2^{k-j}})\right)$.
- **Zero-knowledge:** *If R and $\mathcal{M}$ are finite and $\mathsf{Enc}_{\mathcal{C}}$ is a t-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}}$ (Definition 3.25), then Construction 6.4 is HVZK on fresh inputs (Definition 4.3) for $\mathfrak{D}^{\leq t}$ with error $2^k \cdot \zeta_{\mathcal{C}}$ and query complexity t. The simulator is uniform-query (Definition 4.10).*

Construction 6.4.

- **Inputs and notation.** The verifier receives explicit input $\mathbb{x} = (\mu, \mathsf{st})$ and oracle access to $\mathbb{y} = f$. In the honest case, the prover receives a witness $\mathbb{w} = (\boldsymbol{f}, (\boldsymbol{\sigma}, \boldsymbol{r}))$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$. Here we interpret the randomness of the encoding of $\mathcal{C}^{\equiv 2^k}$ as a mask $\boldsymbol{\sigma} \in (R^p)^{k+1}$ and the seeds $\boldsymbol{r}$ of the rows: the $\boldsymbol{b}$-th row of f is $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}_{\boldsymbol{b}}, (\boldsymbol{s}_{\boldsymbol{b}}, \boldsymbol{r}_{\boldsymbol{b}}))$, where $\boldsymbol{s}_{\boldsymbol{b}} \coloneqq (\hat{s}_{u,e}(\boldsymbol{b}))_{u,e}$ for the polynomials $\hat{s}_{u,e}$ of Definition 6.1. For $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_j)$ and $0 \leq i \leq j$, let $\boldsymbol{\gamma}_{\leq i} = (\gamma_1, \ldots, \gamma_i)$ and $\boldsymbol{\gamma}_{<i} = \boldsymbol{\gamma}_{\leq i-1}$, and let $\mathsf{Fold}(\boldsymbol{v}, \emptyset) = \boldsymbol{v}$.
- **Interaction phase.**
  1. **Mask sum.** The prover sends $\tilde{\mu} \in \mathcal{M}$. In the honest case,

     $$\tilde{\mu} \coloneqq \sum_{\boldsymbol{b} \in \{0,1\}^k} \mathcal{U}^{\boldsymbol{\sigma}, \boldsymbol{m}}(\boldsymbol{b}) .$$

  2. **Combination randomness.** The verifier samples and sends $\varepsilon \leftarrow \mathfrak{C}$.

3. **Sumcheck.** For $\mathtt{X} = (\mathtt{X}_1, \ldots, \mathtt{X}_k)$, define

$$H(\mathtt{X}) \coloneqq \mathcal{U}^{\boldsymbol{\sigma},m}(\mathtt{X}) + \langle \odot_k \mathsf{sl}(\mathsf{st}, \mathtt{X}), \boldsymbol{f}(\mathtt{X}, \cdot) \rangle \cdot \varepsilon \,.$$

Initially $\boldsymbol{\gamma} = \emptyset$. For $j \in [k]$:

(a) The prover sends $\hat{h}_j \in \mathcal{M}^{<d_s}[\mathtt{X}]$. In the honest case,

$$\hat{h}_j(\mathtt{X}) \coloneqq \sum_{\boldsymbol{b} \in \{0,1\}^{k-j}} H(\boldsymbol{\gamma}, \mathtt{X}, \boldsymbol{b}) \,.$$

(b) The verifier samples and sends $\gamma_j \leftarrow \mathfrak{C}$ and sets $\boldsymbol{\gamma} \coloneqq \boldsymbol{\gamma} \| \gamma_j$.

- **Decision phase.**

1. The verifier checks that $\hat{h}_1(0) + \hat{h}_1(1) = \tilde{\mu} + \mu \cdot \varepsilon$.
2. For every $j \in \{2, \ldots, k\}$, it checks that $\hat{h}_j(0) + \hat{h}_j(1) = \hat{h}_{j-1}(\gamma_{j-1})$.

- **Output claims.** The verifier sets

$$\mu' \coloneqq \hat{h}_k(\gamma_k), \qquad \mathsf{st}' \coloneqq ((\mathsf{st}, \boldsymbol{\gamma}), \varepsilon), \qquad \mathsf{st}'_s \coloneqq (\boldsymbol{\gamma}, \boldsymbol{m}) \,,$$

and outputs

$$\mathbb{x}' \coloneqq (\mu', (\mathsf{st}', \mathsf{st}'_s)), \qquad \mathbb{y}' \coloneqq \mathsf{Fold}(f, \boldsymbol{\gamma}) \,.$$

A query to $\mathbb{y}'$ at $x \in [m]$ is answered by querying $f(x)$ and folding it by $\boldsymbol{\gamma}$. In the honest case, the prover outputs

$$\mathbb{w}' \coloneqq ((\mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}), \mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma})) \,, \mathsf{Fold}(\boldsymbol{r}, \boldsymbol{\gamma})) \,.$$

Completeness. Fix $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$. The polynomial H has individual degree less than d_s and

$$\sum_{\boldsymbol{b} \in \{0,1\}^k} H(\boldsymbol{b}) = \tilde{\mu} + \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f} \rangle \cdot \varepsilon = \tilde{\mu} + \mu \cdot \varepsilon \,.$$

Thus the honest round polynomials satisfy every decision-phase check, and

$$\mu' = H(\boldsymbol{\gamma}) = \langle \mathsf{sl}'(\mathsf{st}', \mathsf{st}'_s), (\mathsf{Fold}(\boldsymbol{f}, \boldsymbol{\gamma}), \mathsf{Fold}(\boldsymbol{s}, \boldsymbol{\gamma})) \rangle \,,$$

by Lemma 6.2 and the centrality of the challenges. By Lemma 3.34, the folded oracle is the encoding under $\mathsf{Enc}_{\mathcal{C}}$ of the folded message with the folded mask and seed, that is, the encoding under $\mathsf{Enc}_{\mathcal{C}}^{+p}$ of the message of $\mathbb{w}'$ with its seed. Thus $(\mathbb{x}', \mathbb{y}', \mathbb{w}') \in \mathcal{R}_{\mathcal{C}^{+p}, \mathsf{sl}'}$.

6.3 Zero-knowledge

Lemma 6.5. *If R and $\mathcal{M}$ are finite and $\mathsf{Enc}_{\mathcal{C}}$ is a t-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}}$ (Definition 3.25), then Construction 6.4 is HVZK on fresh inputs (Definition 4.3) for $\mathfrak{D}^{\leq t}$ with error $2^k \cdot \zeta_{\mathcal{C}}$ and query complexity t. The simulator is uniform-query (Definition 4.10).*

Proof. We define the simulator $\mathbf{S}$ for the protocol.

$\mathbf{S}^{\mathsf{y},\mathcal{D}}(\mathbb{x})$:

1. Sample $\varepsilon \leftarrow \mathfrak{C}$ and $\boldsymbol{\gamma} \leftarrow \mathfrak{C}^k$.
2. Sample $(\tilde{\mu}, \hat{h}_1, \ldots, \hat{h}_k)$ uniformly from $\mathcal{M} \times \left(\mathcal{M}^{<d_s}[\mathtt{X}]\right)^k$ conditioned on the decision-phase checks of Construction 6.4. (Sample $\tilde{\mu} \leftarrow \mathcal{M}$ and all coefficients of each $\hat{h}_j$ except the linear one, which is then determined by the corresponding check.)
3. Define $\mathbb{x}'$ from $(\mathbb{x}, \varepsilon, \boldsymbol{\gamma}, (\hat{h}_j)_{j\in[k]})$ as in the output-claims phase of Construction 6.4.
4. Write $\mathcal{D}$ as a pair $(\mathcal{D}_0, \mathcal{D}_1)$ as in Definition 4.9. Compute $(\mathsf{st}, t') \leftarrow \mathcal{D}_0(\mathbb{x}')$ and sample $Q \leftarrow [m]^{t'}$.
5. Query $\mathsf{y}[Q]$, denote the answers by $\boldsymbol{a}_{\mathsf{y}}$, and form $\boldsymbol{a}_{\mathsf{y}'}$ by folding them at $\boldsymbol{\gamma}$. Set $\mathsf{out} \coloneqq \mathcal{D}_1(\mathsf{st}, Q, \boldsymbol{a}_{\mathsf{y}'})$.
6. Identify each $\hat{h}_j$ with its list of d_s coefficients, including zero coefficients, and output

$$\left((\varepsilon, (\gamma_j)_{j\in[k]}), (Q, [1], ([d_s])_{j\in[k]}), (\boldsymbol{a}_{\mathsf{y}}, \tilde{\mu}, (\hat{h}_j)_{j\in[k]}), \mathsf{out}\right) .$$

For Definition 4.10, the first algorithm samples the variables in steps 1–3, runs $\mathcal{D}_0$, and outputs its state and query count. After Q is sampled, the second performs the remaining steps using the given answers $\boldsymbol{a}_{\mathsf{y}}$. Thus the simulator is uniform-query and makes exactly $t' \le t$ queries to y. Its output contains these queries and their answers, as Definition 4.3 requires.

Fix $\mathcal{D}$ in the stated class and $(\mathbb{x}, \boldsymbol{f})$, and let $(\mathbb{x}, f, (\boldsymbol{f}, (\boldsymbol{\sigma}, \boldsymbol{r})))$ be a fresh input for $(\mathbb{x}, \boldsymbol{f})$, so that $\boldsymbol{\sigma}$ and $\boldsymbol{r}$ are uniform and independent, and let $\boldsymbol{s}$ consist of the masks of the rows. We condition on ε, $\boldsymbol{\gamma}$, and the randomness of $\mathcal{D}_0$, which have the same distribution in both experiments and fix t'. Let $\mathcal{T}^* \subseteq \mathcal{M} \times \left(\mathcal{M}^{<d_s}[\mathtt{X}]\right)^k$ be the set of tuples that satisfy the decision-phase checks.

- *Sumcheck transcript.* The messages of the honest prover form the tuple $\tau \coloneqq \pi_\gamma(\boldsymbol{\sigma}) + \boldsymbol{u}$, where $\boldsymbol{u}$ is the tuple of honest messages for $\boldsymbol{\sigma} = \mathbf{0}$. The tuple $\boldsymbol{u}$ depends only on $\mathbb{x}$, $\boldsymbol{f}$, and the challenges, its polynomials have degree at most $2 < d_s$, and $\mathcal{T}^* = \boldsymbol{u} + \mathcal{T}_\gamma$. By Lemma 6.2, τ is uniform over $\mathcal{T}^*$, and so is the tuple τ' sampled by the simulator.

- *Two triples.* For uniform $Q \in [m]^{t'}$, the real view is obtained from $(\tau, Q, f[Q])$ and the simulated view from $(\tau', Q, f[Q])$ by the same randomized map, where τ' is uniform over $\mathcal{T}^*$ and independent of $(Q, \boldsymbol{\sigma}, \boldsymbol{r})$. It thus suffices to show that the statistical distance between these two triples is at most $2^k \cdot \zeta_{\mathcal{C}}$. In both triples, Q is uniform and independent of the other variables. The distance between the triples is therefore the average, over Q, of the distance between the pairs $(\tau, f[Q])$ and $(\tau', f[Q])$ for this Q. We fix Q in the next two items.

- *The answers for a fixed mask.* The encoding is R-linear. Hence there are R-linear maps A and B with values in the group $G \coloneqq (\Sigma^{2^k})^{t'}$ such that

$$f[Q] = A(\boldsymbol{f}, \boldsymbol{s}) + B(\boldsymbol{r}) .$$

  Let $V \subseteq G$ be the image of B. It is a subgroup of G, and $B(\boldsymbol{r})$ is uniform over V because $\boldsymbol{r}$ is uniform. Hence, for every value of $\boldsymbol{\sigma}$, the answers $f[Q]$ are uniform over the set

$$W(\boldsymbol{\sigma}) \coloneqq A(\boldsymbol{f}, \boldsymbol{s}) + V ,$$

  where $\boldsymbol{s}$ consists of the masks that $\boldsymbol{\sigma}$ determines. Two sets of this form are equal or disjoint. Let $W_1, \ldots, W_N$ be the distinct sets $W(\boldsymbol{\sigma})$, over all $\boldsymbol{\sigma} \in (R^p)^{k+1}$, and let E be the set of the pairs $(\tau, W(\boldsymbol{\sigma}))$, over all $\boldsymbol{\sigma}$.

The pair $(\tau, W(\boldsymbol{\sigma}))$ is uniform over E, and the set $W(\boldsymbol{\sigma})$ is uniform over $\{W_1, \ldots, W_N\}$. Indeed, $\tau = \pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma}) + \boldsymbol{u}$ and $W(\boldsymbol{\sigma}) = A(0, \boldsymbol{s}) + A(\boldsymbol{f}, 0) + V$, where $\boldsymbol{\sigma} \mapsto \pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma})$ and $\boldsymbol{\sigma} \mapsto A(0, \boldsymbol{s})$ are homomorphisms of groups. Two tuples thus give the same pair if and only if their difference $\boldsymbol{\sigma}''$, with masks $\boldsymbol{s}''$, satisfies $\pi_{\boldsymbol{\gamma}}(\boldsymbol{\sigma}'') = 0$ and $A(0, \boldsymbol{s}'') \in V$. The tuples with these two properties form a subgroup, so every pair in E is given by the same number of tuples. The same argument applies to the set $W(\boldsymbol{\sigma})$ alone. Moreover, $|E| \geq |\mathcal{T}^*|$, because τ takes every value in $\mathcal{T}^*$.

- *The distance for a fixed query list.* The pair $(\tau, f[Q])$ is distributed as follows: sample (τ, W) uniformly from E, and then sample $f[Q]$ uniformly from W. The pair $(\tau', f[Q])$ is distributed in the same way, except that (τ', W) is uniform over $\mathcal{T}^* \times \{W_1, \ldots, W_N\}$, because τ' is independent of $\boldsymbol{\sigma}$. The second step is the same for both pairs. The distance between the two pairs is therefore at most the distance between the uniform distribution over E and the uniform distribution over $\mathcal{T}^* \times \{W_1, \ldots, W_N\}$, which contains E. This distance is

$$1 - \frac{|E|}{|\mathcal{T}^*| \cdot N} \leq 1 - \frac{1}{N}\,.$$

- *The error of the encoding.* Let Sim be the simulator of the encoding of $(\mathcal{C}^{+p})^{\equiv 2^k}$, which has error $2^k \cdot \zeta_{\mathcal{C}}$ by Claim 3.26, also for $t' \leq t$ queries by Claim 3.22. The simulator $\mathbf{S}$ does not run it. For every value of $\boldsymbol{\sigma}$, the oracle f is the encoding of the message $(\boldsymbol{f}, \boldsymbol{s})$ of this code under the uniform seeds $\boldsymbol{r}$. By Definition 3.21, the pairs $(Q, f[Q])$ and $(Q, \mathsf{Sim}(Q))$, for uniform Q, then have distance at most $2^k \cdot \zeta_{\mathcal{C}}$. This distance is the average, over Q, of the distance $\delta_Q(\boldsymbol{\sigma})$ between $f[Q]$ and $\mathsf{Sim}(Q)$ for this Q and this $\boldsymbol{\sigma}$. Hence the average of $\delta_Q(\boldsymbol{\sigma})$ over Q and over a uniform $\boldsymbol{\sigma}$ is at most $2^k \cdot \zeta_{\mathcal{C}}$.

  Fix Q again. For every $\boldsymbol{\sigma}$, the answers $f[Q]$ lie in $W(\boldsymbol{\sigma})$, so $\delta_Q(\boldsymbol{\sigma}) \geq 1 - \Pr[\mathsf{Sim}(Q) \in W(\boldsymbol{\sigma})]$. For uniform $\boldsymbol{\sigma}$, the set $W(\boldsymbol{\sigma})$ is uniform over the N pairwise disjoint sets $W_1, \ldots, W_N$, and the probabilities $\Pr[\mathsf{Sim}(Q) \in W_i]$ sum to at most 1. The average of $\delta_Q(\boldsymbol{\sigma})$ over $\boldsymbol{\sigma}$ is therefore at least $1 - 1/N$, which is at least the distance between the pairs $(\tau, f[Q])$ and $(\tau', f[Q])$.

We now average over Q. The distance between the two triples is at most the average of $\delta_Q(\boldsymbol{\sigma})$ over Q and $\boldsymbol{\sigma}$, and hence at most $2^k \cdot \zeta_{\mathcal{C}}$. This bound holds for every conditioning, and hence also after averaging. □

6.4 Round-by-round security

Lemma 6.6. *For every $\delta \in (0, \delta(\mathcal{C}))$, Construction 6.4 has RBR knowledge soundness with relaxation*

$$\left(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}, \tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{+p}, \mathsf{sl}'}\right)$$

and errors

$$\left(\Lambda(\mathcal{C}^{\equiv 2^k}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2),\ \left(\epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv 2^{k-j}}, \delta) + \Lambda(\mathcal{C}^{\equiv 2^{k-j+1}}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, d_s)\right)_{j \in [k]}\right)\,.$$

The total extraction time is $O\left(k \cdot 2^k m + \sum_{j \in [k]} \mathsf{tcor}(\mathcal{C}^{\equiv 2^{k-j}})\right)$.

Proof. We give notation, describe the extractors and state function, and bound the RBR errors.

Notation. Let $\mathbb{x} = (\mu, \mathsf{st})$ and $\mathbb{y} = f$. A complete transcript has the form $\mathsf{tr} = \left(\tilde{\mu}, \varepsilon, ((\hat{h}_j, \gamma_j))_{j \in [k]}\right)$. For a partial transcript, define

- $\boldsymbol{\gamma}^{(j)} = \boldsymbol{\gamma}_{\le j}$ and $f^{(j)} = \mathsf{Fold}(f, \boldsymbol{\gamma}^{(j)})$, and
- $\mu^{(j)} = \hat{h}_j(\gamma_j)$ for $j > 0$, and $\mu^{(0)} = \tilde{\mu} + \mu \cdot \varepsilon$.

A witness for $j \in \{0, \dots, k\}$ has the form

$$\mathsf{w}^{(j)} = \left(\left(\boldsymbol{f}^{(j)}, (\boldsymbol{s}^{(j)}, \boldsymbol{r}^{(j)})\right), \bar{f}^{(j)}\right), \qquad \text{where} \quad \bar{f}^{(j)} = \mathsf{Enc}_{\mathcal{C}^{\equiv 2^{k-j}}}\left(\boldsymbol{f}^{(j)}, (\boldsymbol{s}^{(j)}, \boldsymbol{r}^{(j)})\right).$$

Here every row has its own mask, because a malicious prover need not follow the interpretation of Construction 6.4. For $j = k$ we regard it as a witness for $\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{+p},\mathsf{sl}'}$ with message $(\boldsymbol{f}^{(k)}, \boldsymbol{s}^{(k)})$ and seed $\boldsymbol{r}^{(k)}$.

For $0 \le j \le k$, a prefix $\boldsymbol{\gamma}_{\le j} \in \mathcal{Z}(R)^j$, and a table $\boldsymbol{v} = (\boldsymbol{v_b})_{\boldsymbol{b} \in \{0,1\}^{k-j}} \in (R^p)^{2^{k-j}}$, let

$$\Psi_j(\boldsymbol{\gamma}_{\le j}, \boldsymbol{v}) \coloneqq \sum_{\boldsymbol{b} \in \{0,1\}^{k-j}} \langle \mathsf{sl}_{\mathcal{U}}((\boldsymbol{\gamma}_{\le j}, \boldsymbol{b}), \boldsymbol{m}), \boldsymbol{v_b} \rangle \ .$$

For $1 \le j \le k$ and a table $\boldsymbol{v}$ with 2^{k-j+1} rows and halves $\boldsymbol{v}_0, \boldsymbol{v}_1$, put

$$P_j^{\boldsymbol{v}}(\mathtt{X}) \coloneqq \Psi_j\left((\boldsymbol{\gamma}_{<j}, \mathtt{X}), \boldsymbol{v}_0 \cdot (1 - \mathtt{X}) + \boldsymbol{v}_1 \cdot \mathtt{X}\right) . \tag{8}$$

This polynomial has degree less than d_s: the entries of the table are affine in $\mathtt{X}$ and the weights have degree at most $d_s - 2$. Its values at 0 and 1 are the parts of the sum $\Psi_{j-1}(\boldsymbol{\gamma}_{<j}, \boldsymbol{v})$ over the rows of $\boldsymbol{v}_0$ and of $\boldsymbol{v}_1$, so $P_j^{\boldsymbol{v}}(0) + P_j^{\boldsymbol{v}}(1) = \Psi_{j-1}(\boldsymbol{\gamma}_{<j}, \boldsymbol{v})$.

Extractors. We describe the round-by-round extractors from the last round to the first. If parsing or erasure correction fails, the extractor returns $\perp$.

1. **Sumcheck round** $j \in [k]$**.** $\mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma_j, \mathsf{w}^{(j)})$:
   (a) Compute $f^{(j)}$ and the two halves $f_0^{(j-1)}, f_1^{(j-1)}$ of $f^{(j-1)}$ from $\mathbb{y}$ and tr.
   (b) Read $\bar{f}^{(j)}$ from $\mathsf{w}^{(j)}$ and let S be its agreement set with $f^{(j)}$. For $a \in \{0, 1\}$, apply $\mathbf{E}_{\mathcal{C}^{\equiv 2^{k-j}}}$ to $(f_a^{(j-1)}, S)$ and invert the encoding of its output to obtain its message, mask, and seed.
   (c) Output the witness $\mathsf{w}^{(j-1)}$ whose codeword consists of the two recovered codewords.
2. **Combination randomness.** $\mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \varepsilon, \mathsf{w}^{(0)})$ outputs $\mathsf{w}^{(0)}$.

The extractors spend $O(k \cdot 2^k m)$ operations on folds in total. The remaining costs are included in the stated erasure-correction times.

The state function. We define KState as follows, unchanged on prover moves and zero on witnesses not of the indicated form.

0. **Initial transcript:** $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \emptyset, \mathsf{w}) = 1$ if and only if $(\mathbb{x}, \mathbb{y}, \mathsf{w}) \in \tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}$.
1. **Combination randomness:** For $\mathsf{tr} = (\tilde{\mu})$, $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \varepsilon, \mathsf{w}^{(0)}) = 1$ if and only if
   - $\Delta(f, \bar{f}^{(0)}) \le \delta$, and
   - $\Psi_0(\emptyset, \boldsymbol{s}^{(0)}) + \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}^{(0)} \rangle \cdot \varepsilon = \mu^{(0)}$.
2. **Sumcheck, round** $j \in [k]$**:** For $\mathsf{tr} = \left(\tilde{\mu}, \varepsilon, (\hat{h}_{j'}, \gamma_{j'})_{j' < j}, \hat{h}_j\right)$, $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma_j, \mathsf{w}^{(j)}) = 1$ if and only if

- $\Delta(f^{(j)}, \bar{f}^{(j)}) \le \delta$,
- $\hat{h}_1(0) + \hat{h}_1(1) = \mu^{(0)}$ and $\hat{h}_{j'+1}(0) + \hat{h}_{j'+1}(1) = \mu^{(j')}$ for every $j' \in [j-1]$, and
- $\Psi_j(\boldsymbol{\gamma}^{(j)}, \boldsymbol{s}^{(j)}) + \langle \odot_j \mathsf{sl}(\mathsf{st}, \boldsymbol{\gamma}^{(j)}), \boldsymbol{f}^{(j)} \rangle \cdot \varepsilon = \mu^{(j)}$.

When $j = k$, we have $\Psi_k(\boldsymbol{\gamma}, \boldsymbol{s}^{(k)}) = \langle \mathsf{sl}_{\mathcal{U}}(\boldsymbol{\gamma}, \boldsymbol{m}), \boldsymbol{s}^{(k)} \rangle$, so these conditions are the verifier's checks and membership in $\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{+p},\mathsf{sl}'}$.

Bounding the errors.

1. **Combination randomness.** Let $\mathsf{tr} = (\tilde{\mu})$. We show that

$$\Pr_{\varepsilon \leftarrow \mathfrak{C}} \left[\exists\, \mathsf{w} : \begin{array}{l} \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \varepsilon, \mathsf{w})) = 0 \\ \wedge\, \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \varepsilon, \mathsf{w}) = 1 \end{array} \right] \le \Lambda(\mathcal{C}^{\equiv 2^k}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2)\,.$$

   There are at most $\Lambda(\mathcal{C}^{\equiv 2^k}, \delta)$ witnesses satisfying the encoding and distance conditions, independently of ε. If the extracted witness has state zero, then $\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}^{(0)} \rangle \neq \mu$. The two sides of the last condition in Item 1, viewed as polynomials in ε, are therefore distinct of degree at most one. Their agreement has probability at most $\epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2)$. A union bound proves the claim.

2. **Sumcheck randomness.** Let $\mathsf{tr} = \left(\tilde{\mu}, \varepsilon, (\hat{h}_{j'}, \gamma_{j'})_{j'<j}, \hat{h}_j\right)$. We show that

$$\Pr_{\gamma_j \leftarrow \mathfrak{C}} \left[\exists\, \mathsf{w} : \begin{array}{l} \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma_j, \mathsf{w})) = 0 \\ \wedge\, \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma_j, \mathsf{w}) = 1 \end{array} \right] \le \epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv 2^{k-j}}, \delta) + \Lambda(\mathcal{C}^{\equiv 2^{k-j+1}}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, d_s)\,.$$

   Let E be the event that, for every $S \subseteq [m]$ with $|S| \ge (1-\delta)m$, agreement of $f^{(j)}$ with a codeword of $\mathcal{C}^{\equiv 2^{k-j}}$ on S implies that both $f^{(j-1)}_a$, $a \in \{0,1\}$, agree with codewords of this code on S. By Definition 3.19, $\Pr[\neg E] \le \epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv 2^{k-j}}, \delta)$.

   Assume that E holds and fix a post-challenge witness of state one. The erasure corrections succeed, and the recovered codewords are unique because $\delta < \delta(\mathcal{C})$. The fold of the recovered codewords agrees with $\bar{f}^{(j)}$ on S, so the two coincide. Since encodings are injective, Lemma 3.34 implies that the fold of the recovered message, mask, and seed equals the message, mask, and seed in $\mathsf{w}^{(j)}$. The extracted witness therefore satisfies the distance and encoding conditions of round $j-1$, and its codeword belongs to the fixed list $\Lambda(\mathcal{C}^{\equiv 2^{k-j+1}}, f^{(j-1)}, \delta)$.

   Fix a codeword in this list, and denote the halves of its message by $\boldsymbol{f}_0, \boldsymbol{f}_1$ and its mask by $\boldsymbol{v}$. If the extracted witness has state zero, then

$$\hat{h}_j(\mathtt{X}) \neq P^{\boldsymbol{v}}_j(\mathtt{X}) + \left\langle \odot_j \mathsf{sl}(\mathsf{st}, (\boldsymbol{\gamma}^{(j-1)}, \mathtt{X})), \boldsymbol{f}_0 \cdot (1 - \mathtt{X}) + \boldsymbol{f}_1 \cdot \mathtt{X} \right\rangle \cdot \varepsilon\,.$$

   Indeed, by the identity $P^{\boldsymbol{v}}_j(0) + P^{\boldsymbol{v}}_j(1) = \Psi_{j-1}(\boldsymbol{\gamma}_{<j}, \boldsymbol{v})$, summing the right-hand side over $\mathtt{X} \in \{0,1\}$ gives the left-hand side of the last condition of Item 1 (if $j = 1$) or Item 2 (if $j > 1$) for the extracted witness. This witness satisfies the other conditions of that item, so the sum differs from $\mu^{(j-1)} = \hat{h}_j(0) + \hat{h}_j(1)$.

Both polynomials have degree less than d_s, and the post-challenge state condition makes them agree at γ_j by the folding identities above and the centrality of the challenges. For each fixed codeword this has probability at most $\epsilon_{\mathsf{pil}}^{\mathcal{M}}(\mathfrak{C}, 1, d_s)$. A union bound over the list and the MCA error concludes the proof.

□

7 A non-succinct zero-knowledge protocol for constrained codes

We describe a zero-knowledge IOP for the relation $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ (Definition 5.7). The IOP is *non-succinct*, and is used on small instances as a base case.

Theorem 7.1. *Fix the following:*
- *a finite ring R and an integer $t \geq 1$,*
- *a code $\mathcal{C} \subseteq \Sigma^m$ with injective R-linear encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^\ell \times R^r \to \mathcal{C}$,*
- *a finite nonempty challenge set $\mathfrak{C} \subseteq \mathcal{Z}(R)$,*
- $\mathsf{sl} \in \langle \mathcal{M}^\ell |$.

Construction 7.2 is a 2-round IOPP for $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ with the following properties:
- *Prover communication: m symbols of Σ sent as an oracle, and one element of $\mathcal{M}$ together with $\ell + r$ elements of R sent as non-oracle messages.*
- *Query complexity: t queries to the input oracle and t queries to the proof oracle.*
- *Prover time (in ring and module operations): $O(\ell + r + \mathsf{tenc}(\mathcal{C}) + t_{\mathsf{sl}})$.*
- *Verifier time (in ring and module operations): $O(\ell + t + \mathsf{tenc}(\mathcal{C}) + t_{\mathsf{sl}})$.*
- **Round-by-round security:** *For every $\delta \in (0, \delta(\mathcal{C}))$, Construction 7.2 has RBR knowledge soundness with relaxation $\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}}$ and errors*

$$\left(\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) + \Lambda(\mathcal{C}^{\equiv 2}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2), \quad (1-\delta)^t\right) .$$

  The total extraction time is $O\left(m + \mathsf{tcor}(\mathcal{C})\right)$.
- **Zero-knowledge:** *Construction 7.2 is perfect HVZK for every class of distinguishers, with query complexity t, and its simulator is uniform-query (Definition 4.10).*

Construction 7.2.

- **Inputs.** The verifier receives explicit input $\mathbb{x} = (\mu, \mathsf{st})$ and oracle access to $\mathbb{y} = f$. In the honest case, the prover receives a witness $\mathbb{w} = (\boldsymbol{f}, \boldsymbol{r})$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}}$.

- **Interaction phase.**

  1. **New mask.** The prover sends $g \in \Sigma^m$ as an oracle and $\tilde{\mu} \in \mathcal{M}$. In the honest case, it samples $\boldsymbol{g} \leftarrow R^\ell$ and $\boldsymbol{r}' \leftarrow R^r$ independently, and sets

  $$g \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{g}, \boldsymbol{r}'), \qquad \tilde{\mu} \coloneqq \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{g} \rangle .$$

  2. **Combination randomness.** The verifier samples and sends $\gamma \leftarrow \mathfrak{C}$.
  3. **Answer.** The prover sends $\boldsymbol{f}^* \in R^\ell$ and $\boldsymbol{r}^* \in R^r$. In the honest case,

  $$\boldsymbol{f}^* \coloneqq \boldsymbol{g} + \boldsymbol{f} \cdot \gamma, \qquad \boldsymbol{r}^* \coloneqq \boldsymbol{r}' + \boldsymbol{r} \cdot \gamma .$$

  4. **Spotcheck randomness.** The verifier samples and sends $Q \leftarrow [m]^t$.

- **Decision phase.** The verifier computes $f^* \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*)$ and performs the following checks:

  – Target check: $\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}^* \rangle = \tilde{\mu} + \mu \cdot \gamma$.
  – Spot checks: $f^*[Q] = g[Q] + \mathbb{y}[Q] \cdot \gamma$.

  The verifier may compute only the coordinates of f^* used in these checks when this is faster than computing its full encoding.

7.1 Zero-knowledge

Lemma 7.3. *Construction 7.2 is perfect HVZK for every class of distinguishers, with query complexity t, and its simulator is uniform-query (Definition 4.10).*

Proof. We define the simulator.

$\mathbf{S}^{\mathbb{y},\mathcal{D}}(\mathbb{x})$:

1. Parse $\mathbb{x}$ as (μ, st), and recall that $\mathbb{y} = f$.
2. Sample $\gamma \leftarrow \mathfrak{C}$ and $Q \leftarrow [m]^t$.
3. Query $\mathbb{y}[Q]$ and denote the answers by $\boldsymbol{a}_{\mathbb{y}}$.
4. Sample $\boldsymbol{f}^* \leftarrow R^\ell$ and $\boldsymbol{r}^* \leftarrow R^r$.
5. Set
$$\boldsymbol{a}_g \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*)[Q] - \boldsymbol{a}_{\mathbb{y}} \cdot \gamma\,.$$
6. Set $\tilde{\mu} \coloneqq \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}^* \rangle - \mu \cdot \gamma$ and compute $\mathsf{out} \leftarrow \mathcal{D}^{\perp}(\perp)$.
7. Output
$$\begin{pmatrix} (\gamma, Q), \\ (Q, (Q, [1]), [\ell + r])\,, \\ (\boldsymbol{a}_{\mathbb{y}}, (\boldsymbol{a}_g, \tilde{\mu}), (\boldsymbol{f}^*, \boldsymbol{r}^*))\,, \\ \mathsf{out} \end{pmatrix}.$$

The simulator runs in polynomial time and queries $\mathbb{y}$ exactly as the honest verifier does: t independent uniform queries. For Definition 4.10, its first algorithm samples γ, $\boldsymbol{f}^*$, and $\boldsymbol{r}^*$ and outputs these with the query count. Its second performs the remaining steps from Q and $\boldsymbol{a}_{\mathbb{y}}$. The output contains these queries and answers, as required by Definition 4.3. Since the target relation is trivial, $\mathcal{D}^{\perp}(\perp)$ makes no queries, so the simulator can run it directly for every class of distinguishers.

We show that for every $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}}$ the simulated view is distributed identically to the real view. Both views are the image, under the same deterministic function, of

$$(\gamma, Q, \boldsymbol{f}^*, \boldsymbol{r}^*)$$

and the oracle answers $\mathbb{y}[Q]$, together with the randomness of $\mathcal{D}$:

- In the real execution, linearity of the encoding and $\gamma \in \mathcal{Z}(R)$ give
$$\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*) = g + \mathbb{y} \cdot \gamma\,.$$
Thus $g[Q]$ is given by the simulator's formula. Likewise, the input relation gives
$$\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}^* \rangle = \tilde{\mu} + \mu \cdot \gamma\,,$$
so $\tilde{\mu}$ is given by the simulator's formula as well.
- The tuple has the same distribution in both experiments: (γ, Q) is uniform in both and, in the real execution,
$$(\boldsymbol{f}^*, \boldsymbol{r}^*) = (\boldsymbol{g}, \boldsymbol{r}') + (\boldsymbol{f}, \boldsymbol{r}) \cdot \gamma$$
is uniform conditional on every (γ, Q), since it is a translate of the uniform pair $(\boldsymbol{g}, \boldsymbol{r}')$.

Therefore the two views coincide and the simulation is perfect. □

7.2 Round-by-round knowledge soundness

Lemma 7.4. *For every $\delta \in (0, \delta(\mathcal{C}))$, Construction 7.2 has RBR knowledge soundness with relaxation $\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}}$ and errors*

$$\left(\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) + \Lambda(\mathcal{C}^{\equiv 2}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2), \quad (1-\delta)^t\right) .$$

The total extraction time is $O\left(m + \mathsf{tcor}(\mathcal{C})\right)$.

Proof. We give notation, describe the extractors and state function, and bound the RBR errors.

Notation. Let $\mathbb{x} = (\mu, \mathsf{st})$ and $\mathbb{y} = f$. A complete transcript has the form

$$\mathsf{tr} = (g, \tilde{\mu}, \gamma, \boldsymbol{f}^*, \boldsymbol{r}^*, Q) .$$

Extractor. We describe the round-by-round extractors from the last round to the first. If parsing or erasure correction fails, the extractor returns $\perp$.

1. **Spotcheck randomness.** $\mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| Q, \emptyset)$:
   (a) Parse the transcript to obtain $\boldsymbol{f}^*$ and $\boldsymbol{r}^*$.
   (b) Compute $\bar{\mathbb{y}}^* \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*)$.
   (c) Output $\mathsf{w} \coloneqq ((\boldsymbol{f}^*, \boldsymbol{r}^*), \bar{\mathbb{y}}^*)$.

2. **Combination randomness.** $\mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma, \mathsf{w})$:
   (a) Parse w as $((\boldsymbol{f}^*, \boldsymbol{r}^*), \bar{\mathbb{y}}^*)$.
   (b) Let $S \subseteq [m]$ be the agreement set of $\bar{\mathbb{y}}^*$ and $g + \mathbb{y} \cdot \gamma$.
   (c) Apply $\mathbf{E}_{\mathcal{C}}$ to $(\mathbb{y}, S)$, obtaining $\bar{f}$, and invert its encoding to obtain $(\boldsymbol{f}, \boldsymbol{r})$.
   (d) Output $\left((\boldsymbol{f}, \boldsymbol{r}), \bar{f}\right)$.

The extractor computes the agreement set in $O(m)$ operations. The remaining costs are included in $\mathsf{tcor}(\mathcal{C})$.

The state function. We define KState as follows, unchanged on prover moves and zero on witnesses not of the indicated form.

0. **Initial transcript:** $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \emptyset, \mathsf{w}) = 1$ if and only if $(\mathbb{x}, \mathbb{y}, \mathsf{w}) \in \tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}}$.

1. **Combination randomness:** For $\mathsf{tr} = (g, \tilde{\mu})$, $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma, \mathsf{w}) = 1$ if and only if $\mathsf{w} = ((\boldsymbol{f}^*, \boldsymbol{r}^*), \bar{\mathbb{y}}^*)$ satisfies
   (a) $\bar{\mathbb{y}}^* = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*)$,
   (b) $\Delta(g + \mathbb{y} \cdot \gamma, \bar{\mathbb{y}}^*) \leq \delta$, and
   (c) $\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}^* \rangle = \tilde{\mu} + \mu \cdot \gamma$.

2. **Spotcheck randomness:** For $\mathsf{tr} = (g, \tilde{\mu}, \gamma, \boldsymbol{f}^*, \boldsymbol{r}^*)$, $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| Q, \mathsf{w}) = 1$ if and only if $\mathsf{w} = \emptyset$ and the verifier accepts.

Bounding the errors.

1. **Combination randomness.** For $\mathsf{tr} = (g, \tilde{\mu})$, we show that

$$\Pr_{\gamma \leftarrow \mathfrak{C}} \left[\exists\, \mathsf{w} : \begin{array}{l} \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma, \mathsf{w})) = 0 \\ \wedge\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| \gamma, \mathsf{w}) = 1 \end{array} \right] \leq \epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) + \Lambda(\mathcal{C}^{\equiv 2}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2)\,.$$

   Let E be the event that, for every $S \subseteq [m]$ with $|S| \geq (1-\delta)m$, agreement of $g + \mathbb{y} \cdot \gamma$ with a codeword of $\mathcal{C}$ on S implies that $\mathbb{y}$ agrees with a codeword of $\mathcal{C}$ on S. By Definition 3.19,

$$\Pr[\neg E] \leq \epsilon_{\mathsf{mca}}(\mathcal{C}, \delta)\,.$$

   Assume E holds and fix a post-challenge witness of state one. Since $\delta < \delta(\mathcal{C})$, the erasure correction succeeds, and the extracted witness satisfies the encoding and distance conditions of the input relation. Let

$$(\boldsymbol{g}, \boldsymbol{r}') \coloneqq (\boldsymbol{f}^*, \boldsymbol{r}^*) - (\boldsymbol{f}, \boldsymbol{r}) \cdot \gamma, \qquad \bar{g} \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{g}, \boldsymbol{r}')\,.$$

   Linearity gives

$$\bar{g}(S) = \bar{\mathbb{y}}^*(S) - \bar{f}(S) \cdot \gamma = g(S)\,.$$

   Consequently,

$$(\bar{f}, \bar{g}) \in \Lambda\left(\mathcal{C}^{\equiv 2}, (f, g), \delta\right)\,,$$

   a list fixed before γ is sampled. The encoding is injective, so the message and randomness pairs have at most $\Lambda(\mathcal{C}^{\equiv 2}, \delta)$ possible values.

   If the extracted witness has state zero, then $\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f} \rangle \neq \mu$. Substituting $\boldsymbol{f}^* = \boldsymbol{g} + \boldsymbol{f} \cdot \gamma$ into the post-challenge target condition gives

$$\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{g} \rangle + \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f} \rangle \cdot \gamma = \tilde{\mu} + \mu \cdot \gamma\,.$$

   For each fixed pair in the displayed list, the two sides are distinct polynomials of degree at most one. Their agreement has probability at most $\epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2)$. A union bound over the list and the MCA error proves the claim.

2. **Spotcheck randomness.** We show that

$$\Pr_{Q \leftarrow [m]^t} \left[\begin{array}{l} \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| Q, \emptyset)) = 0 \\ \wedge\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| Q, \emptyset) = 1 \end{array} \right] \leq (1-\delta)^t\,.$$

   Let $\bar{\mathbb{y}}^* = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}^*, \boldsymbol{r}^*)$. If the target check fails, the verifier rejects for every Q. Otherwise, a state-zero extracted witness implies

$$\Delta(g + \mathbb{y} \cdot \gamma, \bar{\mathbb{y}}^*) > \delta\,.$$

   All t independent uniform queries fall in the agreement set with probability at most $(1-\delta)^t$, proving the claim.

□

8 A sublinear zero-knowledge IOP for constrained codes

We compose the IOR of Theorem 6.3 and the 2-round IOPP of Theorem 7.1.

Theorem 8.1. *Consider the following ingredients:*

- *a finite ring R, a folding parameter $k \geq 1$, and an integer $t \geq 1$,*
- *a masking degree $d_s \geq 3$, a tuple $\boldsymbol{m} \in \mathcal{M}^{d_{\mathcal{M}}}$ generating $\mathcal{M}$, and a finite nonempty challenge set $\mathfrak{C} \subseteq \mathcal{Z}(R)$,*
- *$\mathcal{C} \subseteq \Sigma^m$ with an encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^\ell \times R^{p+r} \to \Sigma^m$, where $p \coloneqq d_{\mathcal{M}}(d_s - 1)$,*
- *$\mathsf{sl} \in \langle \mathcal{M}^{2^k \ell} |$.*

We use the notation of Theorem 6.3. Construction 8.2 is an IOPP for $\mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$ with the following parameters.

- *Round complexity: $k + 3$.*
- *Prover communication: m symbols of Σ sent as an oracle (the mask of the non-succinct IOPP), and $kd_s + 2$ elements of $\mathcal{M}$ and $\ell + p + r$ elements of R sent as non-oracle messages.*
- *Verifier queries: t queries to the input oracle and t queries to the oracle sent by the non-succinct IOPP.*
- *Prover time (in ring and module operations): $O\left(2^k(\ell + r) + \mathsf{tenc}(\mathcal{C}) + t_{\mathsf{sl}} + kp\right)$.*
- *Verifier time (in ring and module operations): $O\left(t \cdot 2^k + \ell + kp + \mathsf{tenc}(\mathcal{C}) + t_{\odot_k \mathsf{sl}}\right)$.*
- **Round-by-round security:** *For every $\delta \in (0, \delta(\mathcal{C}))$, Construction 8.2 has RBR knowledge soundness with relaxation $\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$ and errors*

$$\begin{pmatrix} \Lambda(\mathcal{C}^{\equiv 2^k}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2), \\ \left(\epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv 2^{k-j}}, \delta) + \Lambda(\mathcal{C}^{\equiv 2^{k-j+1}}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, d_s)\right)_{j \in [k]}, \\ \epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) + \Lambda(\mathcal{C}^{\equiv 2}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2), \quad (1 - \delta)^t \end{pmatrix}.$$

  The total extraction time is $O\left(k \cdot 2^k m + \mathsf{tcor}(\mathcal{C}) + \sum_{j \in [k]} \mathsf{tcor}(\mathcal{C}^{\equiv 2^{k-j}})\right)$.
- **Zero-knowledge:** *If $\mathcal{M}$ is finite and $\mathsf{Enc}_{\mathcal{C}}$ is a t-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}}$, then Construction 8.2 is HVZK on fresh inputs, for every class of distinguishers, with error $2^k \cdot \zeta_{\mathcal{C}}$ and query complexity t. The simulator is uniform-query (Definition 4.10).*

Construction 8.2.

- **Inputs.** The verifier receives explicit input $\mathbb{x} = (\mu, \mathsf{st})$ and oracle access to $\mathbb{y} = f$. In the honest case, the prover receives a witness $\mathbb{w} = (\boldsymbol{f}, (\boldsymbol{\sigma}, \boldsymbol{r}))$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$.
- **Interaction phase.**
  - **Sumcheck IOR.** The prover and verifier run Construction 6.4 from $\mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$ to $\mathcal{R}_{\mathcal{C}^{+p}, \mathsf{sl}'}$, producing $(\mathbb{x}', \mathbb{y}', \mathbb{w}')$ with $\mathbb{y}' = \mathsf{Fold}(f, \boldsymbol{\gamma})$.
  - **Non-succinct IOPP.** The prover and verifier run Construction 7.2 for $\mathcal{R}_{\mathcal{C}^{+p}, \mathsf{sl}'}$ on $(\mathbb{x}', \mathbb{y}', \mathbb{w}')$, with challenge set $\mathfrak{C}$ and query count t.

Proof of Theorem 8.1. Let IOR_1 be Construction 6.4 and let IOR_2 be Construction 7.2 instantiated on $\mathcal{R}_{\mathcal{C}^{+p}, \mathsf{sl}'}$. Construction 8.2 is the concatenation $\mathsf{IOR}_1 \circ \mathsf{IOR}_2$.

The round complexity, communication, and RBR errors follow from Theorems 6.3 and 7.1 by Theorem 4.7, using distance δ for the common intermediate relation. The codes $\mathcal{C}^{+p}$ and $\mathcal{C}$ have the same codewords, and hence the same distance, list sizes, MCA errors, and erasure correction. Each query to y' uses one query to y and $O(2^k)$ operations. The code $\mathcal{C}^{+p}$ has message length $\ell + p$, randomness length r, and encoding cost $\mathsf{tenc}(\mathcal{C})$. The prover and the verifier evaluate its linear form sl' from the fixed linear form and the linear form of the mask polynomial; the latter costs $O(kd_s + p)$ operations, which is $O(kp)$. Combining these costs gives the stated query and running-time bounds.

We establish HVZK on fresh inputs:

- By Theorem 7.1, there is a uniform-query simulator $\mathbf{S}_2$ for IOR_2 with error $\zeta_2 = 0$ and query complexity t.
- By Claim 4.11, $\mathfrak{D}_{\mathbf{S}_2} \subseteq \mathfrak{D}^{\leq t}$ for every class $\mathfrak{D}$ of distinguishers. By Theorem 6.3, IOR_1 is HVZK on fresh inputs for $\mathfrak{D}^{\leq t}$ via a simulator $\mathbf{S}_1$ with error $\zeta_1 = 2^k \cdot \zeta_{\mathcal{C}}$ and query complexity t.
- We apply Theorem 4.7 with $\mathcal{R}_{\mathsf{start}} \coloneqq \mathcal{R}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}$, $\mathcal{R}_{\mathsf{int}} \coloneqq \mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}'}$, and $\mathcal{R}_{\mathsf{fin}} \coloneqq \mathcal{R}_{\mathsf{triv}}$. The theorem does not require IOR_1 to have fresh output, because IOR_2 is HVZK. We conclude that the composed protocol is HVZK on fresh inputs with query complexity t and error at most $\zeta_1 + \zeta_2$. Its simulator queries the input oracle exactly as $\mathbf{S}_1$ does, and $\mathbf{S}_1$ is uniform-query, so it is uniform-query.

□

9 Zero-knowledge code-switching

We introduce an honest-verifier zero-knowledge code-switching protocol. To do so, we first describe the variant of out-of-domain samples that we use in Section 9.1, the notion of private zero-evaders in Section 9.2, and a measure of the code-switching efficiency in Section 9.3.

9.1 Out-of-domain samples

We refine the notion of out-of-domain samples for a linear code (see [BGKS20] and their later uses in [ACFY24; ACFY25; BCFW25]) to account for zero-knowledge encodings. Specifically, two codewords in the list may encode the same message with different randomness, and we only need to separate codewords that encode distinct messages.

Lemma 9.1. *Let $\mathcal{C} \subseteq \Sigma^m$ be a code with an injective R-linear encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^\ell \times R^r \to \mathcal{C}$, let $f \in \Sigma^m$, and let $\delta \in [0,1]$. Let $\mathsf{ze} \colon D_{\mathsf{ze}} \to \mathcal{M}^{t\times\ell}$ be a zero-evader over R with error $\varepsilon_{\mathsf{zero}}$ (Definition 3.5). Then,*

$$\Pr_{\rho \leftarrow D_{\mathsf{ze}}}\left[\begin{array}{l} \exists\, u, v \in \Lambda(\mathcal{C}, f, \delta) \text{ s.t.} \\ \boldsymbol{u} \neq \boldsymbol{v} \ \wedge\ \mathsf{ze}(\rho)\cdot \boldsymbol{u} = \mathsf{ze}(\rho)\cdot \boldsymbol{v} \\ \text{where } (\boldsymbol{u}, \boldsymbol{r}) \coloneqq \mathsf{Enc}_{\mathcal{C}}^{-1}(u),\ (\boldsymbol{v}, \boldsymbol{r}') \coloneqq \mathsf{Enc}_{\mathcal{C}}^{-1}(v) \end{array}\right] \leq \frac{\Lambda(\mathcal{C},\delta)^2}{2} \cdot \varepsilon_{\mathsf{zero}}\,.$$

Proof. Let $u, v \in \Lambda(\mathcal{C}, f, \delta)$ be distinct codewords and $(\boldsymbol{u}, \boldsymbol{r}) \coloneqq \mathsf{Enc}_{\mathcal{C}}^{-1}(u)$ and $(\boldsymbol{v}, \boldsymbol{r}') \coloneqq \mathsf{Enc}_{\mathcal{C}}^{-1}(v)$. We consider two cases: (i) if $\boldsymbol{u} = \boldsymbol{v}$, the pair does not satisfy the event, and (ii) if $\boldsymbol{u} \neq \boldsymbol{v}$, the zero-evader guarantee for $\boldsymbol{u} - \boldsymbol{v} \neq 0$ gives $\mathsf{ze}(\rho)\cdot\boldsymbol{u} = \mathsf{ze}(\rho)\cdot\boldsymbol{v}$ with probability at most $\varepsilon_{\mathsf{zero}}$. The bound follows by taking a union bound over the at most $\binom{\Lambda(\mathcal{C},\delta)}{2}$ distinct pairs of codewords in $\Lambda(\mathcal{C}, f, \delta)$. □

9.2 Private zero-evaders

We consider zero-evaders with a *privacy guarantee*: the inner product with a vector consisting of a private portion and a random portion reveals little or no information about the private portion.

Definition 9.2. *A zero-evader $\mathsf{ze} \colon D_{\mathsf{ze}} \to \mathcal{M}^{t\times\ell}$ over R is (r, ζ_{ze})-**private** if there exists a simulator $\mathbf{S}_{\mathsf{ze}}$ such that for every $\boldsymbol{f} \in R^{\ell-r}$ the following two distributions have statistical distance at most ζ_{ze}:*

$$\left\{ (\rho, \boldsymbol{y}) \;\middle|\; \begin{array}{l} \rho \leftarrow D_{\mathsf{ze}}, \boldsymbol{r} \leftarrow R^r \\ \boldsymbol{y} = \mathsf{ze}(\rho)\cdot \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{r} \end{bmatrix} \end{array} \right\} \text{ and } \mathbf{S}_{\mathsf{ze}}\,.$$

The all-zero zero-evader is perfectly private but has zero-evading error 1, so privacy is meaningful only together with a small zero-evading error. Any zero-evader can be modified to have *perfect* privacy at the cost of a small increase in its zero-evading error (Lemma 9.4).

Definition 9.3. *Let $t, r \in \mathbb{N}$. We define:*

$$\mathcal{M}_{t,r} \coloneqq \left\{ \boldsymbol{M} \in \mathcal{M}^{t\times r} \;:\; \text{the columns of } \boldsymbol{M} \text{ generate } \mathcal{M}^t \text{ as an } R\text{-module} \right\}.$$

For $t \in \mathbb{N}$, let $\varepsilon_{\mathsf{mat}}^{t,0} \coloneqq 0$. For $r \geq 1$, define

$$\varepsilon_{\mathsf{mat}}^{t,r} \coloneqq \max_{\boldsymbol{r}\in R^r\setminus\{\boldsymbol{0}\}} \max_{\boldsymbol{t}\in\mathcal{M}^t} \Pr_{\boldsymbol{M}\leftarrow\mathcal{M}_{t,r}}[\boldsymbol{M}\cdot\boldsymbol{r} = \boldsymbol{t}]\,.$$

Lemma 9.4. *Let R and $\mathcal{M}$ be finite, and let $\mathsf{ze}\colon D_{\mathsf{ze}} \to \mathcal{M}^{t\times\ell}$ be a zero-evader over R with error $\varepsilon_{\mathsf{zero}}$. For every $r \geq t \cdot d_{\mathcal{M}}$ (where $d_{\mathcal{M}}$ denotes the size of a minimal generating set for $\mathcal{M}$), there exists a zero-evader $\mathsf{ze}'\colon D'_{\mathsf{ze}} \to \mathcal{M}^{t\times(\ell+r)}$ with error $\max\{\varepsilon_{\mathsf{zero}}, \varepsilon^{t,r}_{\mathsf{mat}}\}$ that is $(r,0)$-private.*

Proof. Define $D'_{\mathsf{ze}} \coloneqq D_{\mathsf{ze}} \times \mathcal{M}_{t,r}$ and

$$\mathsf{ze}'(\rho, \boldsymbol{M}) \coloneqq [\mathsf{ze}(\rho), \boldsymbol{M}]\,.$$

If $r = 0$, the bound $r \geq t \cdot d_{\mathcal{M}}$ implies $\mathcal{M}^t = \{0\}$. The claims follow directly in this case. We may therefore assume that $r \geq 1$. Note that $\mathcal{M}_{t,r}$ is nonempty since $r \geq t \cdot d_{\mathcal{M}}$: let $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{d_{\mathcal{M}}}$ generate $\mathcal{M}$ and take, for every $i \in [t]$ and $d \in [d_{\mathcal{M}}]$, a column whose i-th entry is $\boldsymbol{m}_d$ and whose remaining entries are 0 (these columns generate $\mathcal{M}^t$, and any leftover columns can be set to $\mathbf{0}$).

Privacy error. Fix any $\boldsymbol{f} \in R^\ell$. Let $\mathbf{S}_{\mathsf{ze}'}$ be the simulator that samples $(\rho, \boldsymbol{M}) \leftarrow D'_{\mathsf{ze}}$ and $\boldsymbol{y} \leftarrow \mathcal{M}^t$ and outputs $((\rho, \boldsymbol{M}), \boldsymbol{y})$. Let $\boldsymbol{t}_\rho \coloneqq \mathsf{ze}(\rho) \cdot \boldsymbol{f} \in \mathcal{M}^t$. The sample from the real distribution is

$$((\rho, \boldsymbol{M}), \boldsymbol{y}) \quad \text{where} \quad \boldsymbol{y} = \mathsf{ze}'(\rho, \boldsymbol{M}) \cdot \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{r} \end{bmatrix} = \boldsymbol{t}_\rho + \boldsymbol{M} \cdot \boldsymbol{r}\,.$$

Since $\boldsymbol{M} \in \mathcal{M}_{t,r}$, the map $\boldsymbol{r} \mapsto \boldsymbol{M} \cdot \boldsymbol{r}$ is a surjective homomorphism of abelian groups from R^r onto $\mathcal{M}^t$. Therefore, when $\boldsymbol{r}$ is uniform in R^r, the value $\boldsymbol{M} \cdot \boldsymbol{r}$ is uniform in $\mathcal{M}^t$, and hence $\boldsymbol{y} = \boldsymbol{t}_\rho + \boldsymbol{M} \cdot \boldsymbol{r}$ is uniform in $\mathcal{M}^t$, even conditioned on $(\rho, \boldsymbol{M})$. Thus the joint distribution of $((\rho, \boldsymbol{M}), \boldsymbol{y})$ is identical to the distribution produced by the simulator.

Zero-evader error. Let $\boldsymbol{v} \neq 0$ and write $\boldsymbol{v} = (\boldsymbol{s}, \boldsymbol{r})^\mathsf{T}$ for $\boldsymbol{s} \in R^\ell$ and $\boldsymbol{r} \in R^r$. Note that $\mathsf{ze}'(\rho, \boldsymbol{M}) \cdot \boldsymbol{v} = \mathsf{ze}(\rho) \cdot \boldsymbol{s} + \boldsymbol{M} \cdot \boldsymbol{r}$. We consider two cases.

- If $\boldsymbol{r} = \mathbf{0}$, then $\boldsymbol{s} \neq \mathbf{0}$ and

$$\Pr[\mathsf{ze}'(\rho, \boldsymbol{M}) \cdot \boldsymbol{v} = \mathbf{0}] = \Pr[\mathsf{ze}(\rho) \cdot \boldsymbol{s} = \mathbf{0}] \leq \varepsilon_{\mathsf{zero}}\,.$$

- If $\boldsymbol{r} \neq \mathbf{0}$, write $\boldsymbol{t}_\rho \coloneqq \mathsf{ze}(\rho) \cdot \boldsymbol{s}$. Since $\boldsymbol{M}$ is independent of ρ, conditioning on any value of ρ gives

$$\Pr[\mathsf{ze}'(\rho, \boldsymbol{M}) \cdot \boldsymbol{v} = \mathbf{0}] = \Pr_{\boldsymbol{M} \leftarrow \mathcal{M}_{t,r}}[\boldsymbol{M} \cdot \boldsymbol{r} = -\boldsymbol{t}_\rho] \leq \varepsilon_{\mathsf{mat}}\,,$$

  by the definition of $\varepsilon_{\mathsf{mat}}$.

□

Lemma 9.5. *Let $R = \mathcal{M} = \mathbb{F}$ be a finite field and let $1 \leq t \leq r$. Then,*

$$\varepsilon^{t,r}_{\mathsf{mat}} \leq \frac{1}{|\mathbb{F}|^t - 1}\,.$$

Proof. In this setting, $\mathcal{M}_{t,r}$ is the set of matrices whose columns span $\mathbb{F}^t$. Fix $\boldsymbol{r} \neq \mathbf{0}$ and $\boldsymbol{t} \in \mathbb{F}^t$. We bound $\Pr_{\boldsymbol{M}}[\boldsymbol{M}\boldsymbol{r} = \boldsymbol{t}]$. Since $\boldsymbol{r} \neq \mathbf{0}$, there exists an invertible matrix $\boldsymbol{R} \in \mathbb{F}^{r\times r}$ such that $\boldsymbol{R} \cdot \boldsymbol{r} = \boldsymbol{e}_1$. Right-multiplication by $\boldsymbol{R}^{-1}$ preserves the property that the columns span $\mathbb{F}^t$, hence the map $\boldsymbol{M} \mapsto \boldsymbol{M}\boldsymbol{R}^{-1}$ is a bijection on $\mathcal{M}_{t,r}$ and therefore

$$\Pr[\boldsymbol{M}\boldsymbol{r} = \boldsymbol{t}] = \Pr[\boldsymbol{M}\boldsymbol{e}_1 = \boldsymbol{t}]\,.$$

Denote by $\boldsymbol{m}_1$ the first column of $\boldsymbol{M}$, so $\boldsymbol{M}\boldsymbol{e}_1 = \boldsymbol{m}_1$. We upper bound $\Pr[\boldsymbol{m}_1 = \boldsymbol{t}]$.

If $\boldsymbol{t} \neq \boldsymbol{0}$, then for every $\boldsymbol{u} \in \mathbb{F}^t \setminus \{\boldsymbol{0}\}$ the probability $\Pr[\boldsymbol{m}_1 = \boldsymbol{u}]$ is the same by left-invariance: for any invertible $\boldsymbol{L} \in \mathbb{F}^{t\times t}$, the map $\boldsymbol{M} \mapsto \boldsymbol{LM}$ is a bijection on $\mathcal{M}_{t,r}$ and sends $\boldsymbol{m}_1$ to $\boldsymbol{L} \cdot \boldsymbol{m}_1$. These $|\mathbb{F}|^t - 1$ equal probabilities sum to at most 1, and therefore

$$\Pr[\boldsymbol{m}_1 = \boldsymbol{t}] \leq \frac{1}{|\mathbb{F}|^t - 1}\,.$$

If $\boldsymbol{t} = \boldsymbol{0}$ and $r - 1 < t$, no matrix with first column $\boldsymbol{0}$ has columns that span $\mathbb{F}^t$. Hence the probability is 0 and the bound follows. Suppose that $r - 1 \geq t$. Then

$$\Pr_{\boldsymbol{M}\leftarrow\mathcal{M}_{t,r}}[\boldsymbol{m}_1 = \boldsymbol{0}] = \frac{|\mathcal{M}_{t,r-1}|}{|\mathcal{M}_{t,r}|} = \prod_{i=0}^{t-1} \frac{|\mathbb{F}|^{r-1} - |\mathbb{F}|^i}{|\mathbb{F}|^r - |\mathbb{F}|^i} \leq |\mathbb{F}|^{-t}\,,$$

where the equality follows by observing that fixing $\boldsymbol{m}_1 = \boldsymbol{0}$ leaves a uniformly random matrix in $\mathbb{F}^{t\times(r-1)}$ whose columns span $\mathbb{F}^t$, and the inequality holds since for all i, $\frac{|\mathbb{F}|^{r-1}-|\mathbb{F}|^i}{|\mathbb{F}|^r-|\mathbb{F}|^i} \leq \frac{|\mathbb{F}|^{r-1}}{|\mathbb{F}|^r} = |\mathbb{F}|^{-1}$.

Combining the two subcases, for all $\boldsymbol{r} \neq \boldsymbol{0}$ and $\boldsymbol{t} \in \mathbb{F}^t$ we obtain

$$\Pr_{\boldsymbol{M}\leftarrow\mathcal{M}_{t,r}}[\boldsymbol{Mr} = \boldsymbol{t}] \leq \max\left\{\frac{1}{|\mathbb{F}|^t - 1},\, |\mathbb{F}|^{-t}\right\} = \frac{1}{|\mathbb{F}|^t - 1}\,,$$

and hence $\varepsilon_{\mathsf{mat}} \leq \frac{1}{|\mathbb{F}|^t-1}$. □

9.3 Code-switching complexity

We define the linear form whose values are the rows of the generating matrix of an encoding.

Definition 9.6. *Let $\mathcal{C} \subseteq (\mathcal{M}^w)^m$ be a code with an R-linear encoding $\mathsf{Enc}_{\mathcal{C}}\colon R^\ell \times R^r \to \mathcal{C}$, and let $\boldsymbol{G}_{\mathcal{C}} = \begin{bmatrix}\boldsymbol{G}^{\#}_{\mathcal{C}} & \boldsymbol{G}^{\$}_{\mathcal{C}}\end{bmatrix}$ be its generating matrix (Definition 3.23), with rows indexed by $[m] \times [w]$. The* **generator form** *of $\mathcal{C}$ is the linear form $\mathsf{sl}_{\mathcal{C}} = (\mathsf{sl}^{\#}_{\mathcal{C}}, \mathsf{sl}^{\$}_{\mathcal{C}})$ with*

$$\mathsf{sl}^{\#}_{\mathcal{C}}(x, j) \coloneqq \boldsymbol{G}^{\#}_{\mathcal{C}}[(x, j), \cdot] \in \mathcal{M}^\ell, \qquad \mathsf{sl}^{\$}_{\mathcal{C}}(x, j) \coloneqq \boldsymbol{G}^{\$}_{\mathcal{C}}[(x, j), \cdot] \in \mathcal{M}^r\,.$$

For $w = 1$ we omit j.

For Reed–Solomon codes over explicit domains, the two parts of the generator form can be computed in $O(\ell + \log m + r)$ operations and, once all message variables are fixed, in $O(\log m + r)$ operations (this fact is implicit in [ACFY25; NA25] and explicit in [BFRW26]).

Lemma 9.7. *Let R be finite, let $r \geq 1$, and let $\mathcal{L} \subseteq \mathcal{Z}(R)$ be a strong sampling set with $|\mathcal{L}| = m$ and $\ell + r \leq m$. Let $\mathcal{C} \coloneqq \mathsf{RS}[R, \mathcal{L}, \ell + r]$ be the Reed–Solomon code over R (Definition 3.27), viewed as an (R, R)-additive code (i.e., $\mathcal{M} \coloneqq R$ as a right module over itself), and suppose that there exists a bijection $\varphi\colon [m] \to \mathcal{L}$ computable in $O(\log m)$ ring operations.[5] There exists an r-query perfect zero-knowledge encoding for $\mathcal{C}$ such that: (i) if ℓ is a power of two, for every $i \in \{0, \ldots, \log \ell\}$, $t_{\odot_i(\mathsf{sl}^{\#}_{\mathcal{C}})} = O(\ell/2^i + i + \log m + \log r)$, and (ii) $t_{\mathsf{sl}^{\$}_{\mathcal{C}}} = O(r + \log m)$.*

[5] This class includes many commonly used Reed–Solomon codes, but excludes (for example) Reed–Solomon codes where $\mathcal{L}$ is a random subset of the ring.

Proof. Consider the encoding map $\mathsf{Enc}_{\mathcal{C}}\colon R^{\ell} \times R^{r} \to \mathcal{C}$ that takes as input $(\boldsymbol{f}, \boldsymbol{r})$, interprets them as coefficients of polynomials $\hat{\boldsymbol{f}}, \hat{\boldsymbol{r}} \in R[\mathtt{X}]$ and outputs $(\hat{\boldsymbol{r}} + \hat{\boldsymbol{f}} \cdot \mathtt{X}^{r})(\mathcal{L})$. By Lemma 3.29, $\mathsf{Enc}_{\mathcal{C}}$ is an r-query perfect zero-knowledge encoding for $\mathcal{C}$. The generating matrix of $\mathsf{Enc}_{\mathcal{C}}$ is $\boldsymbol{G}_{\mathcal{C}} = [\boldsymbol{G}^{\#}_{\mathcal{C}}, \boldsymbol{G}^{\$}_{\mathcal{C}}]$, where $\boldsymbol{G}^{\#}_{\mathcal{C}} = [\varphi(x)^{r+j-1}]_{x\in[m],j\in[\ell]}$ and $\boldsymbol{G}^{\$}_{\mathcal{C}} = [\varphi(x)^{j-1}]_{x\in[m],j\in[r]}$. In particular, $\boldsymbol{G}_{\mathcal{C}}$ is a column permutation of the Vandermonde matrix $[\varphi(x)^{j-1}]_{x\in[m],j\in[\ell+r]}$.

Assume that ℓ is a power of two, $x \in [m]$, $\boldsymbol{\gamma} \in \mathcal{Z}(R)^{i}$, and consider $\odot_i \mathsf{sl}^{\#}_{\mathcal{C}}(x, \boldsymbol{\gamma})$. All the quantities below lie in the commutative subring of R generated by $\mathcal{L} \cup \{\gamma_1, \ldots, \gamma_i\} \subseteq \mathcal{Z}(R)$, so we may manipulate them as in the commutative case. By definition, for every $\boldsymbol{c} \in \{0,1\}^{\log \ell - i}$,

$$
\begin{aligned}
(\odot_i \mathsf{sl}^{\#}_{\mathcal{C}}(x, \boldsymbol{\gamma}))[\boldsymbol{c}] &= \sum_{\boldsymbol{b}\in\{0,1\}^i} \boldsymbol{G}^{\#}_{\mathcal{C}}[x, \boldsymbol{b}, \boldsymbol{c}] \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \\
&= \sum_{\boldsymbol{b}\in\{0,1\}^i} \varphi(x)^{r + \sum_{t\in[i]} b_t \cdot 2^{t-1} + \sum_{t\in[\log \ell - i]} c_t \cdot 2^{i+t-1}} \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \\
&= \sum_{\boldsymbol{b}\in\{0,1\}^i} \varphi(x)^{r} \cdot \prod_{t\in[i]} \varphi(x)^{b_t \cdot 2^{t-1}} \cdot \prod_{t\in[\log \ell - i]} \varphi(x)^{c_t \cdot 2^{i+t-1}} \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \\
&= \left(\varphi(x)^{r} \cdot \prod_{t\in[\log \ell - i]} \varphi(x)^{c_t \cdot 2^{i+t-1}} \right) \\
&\qquad \cdot \left(\sum_{\boldsymbol{b}\in\{0,1\}^i} \prod_{t\in[i]} ((1-b_t) \cdot 1 + b_t \cdot \varphi(x)^{2^{t-1}}) \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) \right) \\
&= \left(\varphi(x)^{r} \cdot \prod_{t\in[\log \ell - i]} \varphi(x)^{c_t \cdot 2^{i+t-1}} \right) \cdot \prod_{t\in[i]} (1 - \gamma_t + \gamma_t \cdot \varphi(x)^{2^{t-1}}) \,.
\end{aligned}
$$

Here we used that $\boldsymbol{b} \mapsto \prod_{t\in[i]}((1-b_t) + b_t \varphi(x)^{2^{t-1}})$ is multilinear and $\sum_{\boldsymbol{b}} \hat{p}(\boldsymbol{b}) \cdot \mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}) = \hat{p}(\boldsymbol{\gamma})$ for any multilinear polynomial $\hat{p}$ (Claim 3.11). This yields an algorithm to compute $\odot_i \mathsf{sl}^{\#}_{\mathcal{C}}(x, \boldsymbol{\gamma})$:

1. Compute $\varphi(x)$ in $O(\log m)$ ring operations, and then $\varphi(x)^{r}$ in $O(\log r)$ ring operations by repeated squaring.
2. Compute $\prod_{t\in[i]}(1 - \gamma_t + \gamma_t \cdot \varphi(x)^{2^{t-1}})$ in $O(i)$ ring operations.
3. Compute $\left(\prod_{t\in[\log \ell - i]} \varphi(x)^{c_t \cdot 2^{i+t-1}}\right)_{\boldsymbol{c}\in\{0,1\}^{\log \ell - i}}$ in $O(\ell/2^i)$ ring operations using a dynamic programming algorithm (e.g., [VSBW13; BCFFMMZ25]).
4. Compute the final expression in $O(\ell/2^i)$ ring operations.

For the second item, for every $x \in [m]$, the row $\boldsymbol{G}^{\$}_{\mathcal{C}}[x, \cdot] = [\varphi(x)^{j-1}]_{j\in[r]}$ can be computed in $O(\log m + r)$ ring operations. □

9.4 Code-switching IOR

We describe an IOR from $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ to $\mathcal{R}_{\mathcal{C}',\mathsf{sl}'}$ that re-encodes the input message and randomness in $\mathcal{C}'$.

Theorem 9.8. *Let:*

- *R be a finite ring, $\mathcal{M}$ a right R-module, and $w, t, t_{\mathsf{ood}} \geq 1$ be integers,*
- *$\mathcal{C} \subseteq (\mathcal{M}^{w})^{m}$ have an injective R-linear encoding $\mathsf{Enc}_{\mathcal{C}}\colon R^{\ell} \times R^{r} \to \mathcal{C}$,*
- *$\mathcal{C}' \subseteq (\Sigma')^{m'}$ have an injective R-linear encoding $\mathsf{Enc}_{\mathcal{C}'}\colon R^{\ell'} \times R^{r'} \to \mathcal{C}'$, where $\ell' = \ell + r + r_{\mathsf{ood}}$ and $r_{\mathsf{ood}} \in \mathbb{N}$,*

- $\mathsf{sl} \in \langle \mathcal{M}^{\ell} |$,
- $\mathsf{ze} \colon D_{\mathsf{ze}} \to \mathcal{Z}(R)^{1+t_{\mathsf{ood}}+wt}$ *be a zero-evader over* $\mathcal{M}$ *with error* $\varepsilon_{\mathsf{zero}}$,
- $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathcal{M}^{t_{\mathsf{ood}} \times \ell'}$ *be a zero-evader over* R *with error* $\varepsilon_{\mathsf{ood}}$. *For* $i \in [t_{\mathsf{ood}}]$, *let* $\mathsf{ze}_{\mathsf{ood},i}$ *denote its* i*-th row.*

Using $\mathsf{sl}_{\mathcal{C}}$ *from Section 9.3, define*

$$\mathsf{sl}' \coloneqq \mathsf{ze}\left[0^{r+r_{\mathsf{ood}}}(\mathsf{sl}), (\mathsf{ze}_{\mathsf{ood},i})_{i\in[t_{\mathsf{ood}}]}, (0^{r_{\mathsf{ood}}}(\mathsf{sl}_{\mathcal{C}}))_{(i,j)\in[t]\times[w]}\right] .$$

Construction 9.9 is an IOR from $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ *to* $\mathcal{R}_{\mathcal{C}',\mathsf{sl}'}$ *with the following properties.*

- *Round complexity:* 2.
- *Prover communication:* m' *symbols of* Σ' *sent as an oracle, and* t_{ood} *elements of* $\mathcal{M}$ *sent as a non-oracle message.*
- *Query complexity:* t *queries to the input oracle. The output oracle is the new oracle.*
- *Prover time (in ring and module operations):* $O\left(\mathsf{tenc}(\mathcal{C}') + \mathsf{teval}(\mathsf{ze}_{\mathsf{ood}}) + t_{\mathsf{ood}}\ell'\right)$.
- *Verifier time (in ring and module operations):* $O\left(\mathsf{teval}(\mathsf{ze}) + t_{\mathsf{ood}} + wt\right)$.
- **Round-by-round security:** *For every* $\delta, \delta' \in (0,1)$, *Construction 9.9 has RBR knowledge soundness with relaxation*

  $$\left(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}}, \tilde{\mathcal{R}}^{\delta'}_{\mathcal{C}',\mathsf{sl}'}\right)$$

  and errors

  $$\left(\frac{\Lambda(\mathcal{C}',\delta')^2}{2} \cdot \varepsilon_{\mathsf{ood}}, \quad (1-\delta)^t + \Lambda(\mathcal{C}',\delta') \cdot \varepsilon_{\mathsf{zero}}\right) .$$

  The total extraction time is $O\left(\mathsf{tenc}(\mathcal{C}) + wm\right)$.
- **Zero-knowledge:** *If* $\mathsf{ze}_{\mathsf{ood}}$ *is* $(r_{\mathsf{ood}}, \zeta_{\mathsf{ze}})$*-private and* $\mathsf{Enc}_{\mathcal{C}'}$ *is a* t'*-query zero-knowledge encoding with error* $\zeta_{\mathcal{C}'}$, *then Construction 9.9 is HVZK for* $\mathfrak{D}^{\leq t'}$ *with error* $\zeta_{\mathcal{C}'} + \zeta_{\mathsf{ze}}$ *and query complexity* t. *The simulator is uniform-query (Definition 4.10), and the reduction has fresh output (Definition 4.5).*

Construction 9.9.

- **Inputs.** The verifier receives explicit input $\mathbb{x} = (\mu, \mathsf{st})$ and oracle access to $\mathbb{y} = f$. In the honest case, the prover receives a witness $\mathbb{w} = (\boldsymbol{f}, \boldsymbol{r})$ such that $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}}$.
- **Interaction phase.**
  1. **Send witness.** The prover sends $g \in (\Sigma')^{m'}$ as an oracle. In the honest case, it samples $\boldsymbol{r}_{\mathsf{ood}} \leftarrow R^{r_{\mathsf{ood}}}$ and $\boldsymbol{r}' \leftarrow R^{r'}$ independently and sets

     $$\boldsymbol{g} \coloneqq (\boldsymbol{f}, \boldsymbol{r}, \boldsymbol{r}_{\mathsf{ood}}), \qquad g \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{g}, \boldsymbol{r}') .$$

  2. **Out-of-domain samples.** The verifier samples and sends $\rho_{\mathsf{ood}} \leftarrow D_{\mathsf{ood}}$.
  3. **Out-of-domain answers.** The prover sends $\boldsymbol{y} \in \mathcal{M}^{t_{\mathsf{ood}}}$. In the honest case,

     $$\boldsymbol{y} \coloneqq \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{g} .$$

  4. **In-domain queries and batching.** The verifier samples and sends $x_1, \ldots, x_t \leftarrow [m]$ and $\rho \leftarrow D_{\mathsf{ze}}$. Let $\boldsymbol{\nu} \coloneqq \mathsf{ze}(\rho)$.

- **Decision phase.** The verifier queries $\mathbb{y}(x_1), \ldots, \mathbb{y}(x_t) \in \mathcal{M}^w$ and computes

$$\mu' \coloneqq \mu \cdot \boldsymbol{\nu}_1 + \sum_{i \in [t_{\mathsf{ood}}]} y_i \cdot \boldsymbol{\nu}_{1+i} + \sum_{i \in [t]} \sum_{j \in [w]} \mathbb{y}(x_i)_j \cdot \boldsymbol{\nu}_{1+t_{\mathsf{ood}}+w(i-1)+j} \,.$$

It sets

$$\mathsf{st}' \coloneqq \Big(\mathsf{st}, (\rho_{\mathsf{ood}})_{i \in [t_{\mathsf{ood}}]}, ((x_i, j))_{(i,j) \in [t] \times [w]}, \rho\Big)$$

and outputs

$$\mathbb{x}' \coloneqq (\mu', \mathsf{st}'), \qquad \mathbb{y}' \coloneqq g \,. \tag{9}$$

In the honest case, the prover outputs $\mathbb{w}' \coloneqq (\boldsymbol{g}, \boldsymbol{r}')$.

Completeness. In an honest execution,

$$\begin{aligned} \mu &= \Big\langle 0^{r+r_{\mathsf{ood}}}(\mathsf{sl})(\mathsf{st}), \boldsymbol{g} \Big\rangle \,, \\ y_i &= \langle \mathsf{ze}_{\mathsf{ood},i}(\rho_{\mathsf{ood}}), \boldsymbol{g} \rangle && \text{for } i \in [t_{\mathsf{ood}}], \\ \mathbb{y}(x_i)_j &= \langle 0^{r_{\mathsf{ood}}}(\mathsf{sl}_{\mathcal{C}})(x_i, j), \boldsymbol{g} \rangle && \text{for } (i,j) \in [t] \times [w] \,. \end{aligned}$$

By linearity and centrality of $\boldsymbol{\nu}$, $\mu' = \langle \mathsf{sl}'(\mathsf{st}'), \boldsymbol{g} \rangle$. Thus $(\mathbb{x}', \mathbb{y}', \mathbb{w}') \in \mathcal{R}_{\mathcal{C}', \mathsf{sl}'}$.

9.5 Zero-knowledge

Lemma 9.10. *If $\mathsf{ze}_{\mathsf{ood}}$ is $(r_{\mathsf{ood}}, \zeta_{\mathsf{ze}})$-private and $\mathsf{Enc}_{\mathcal{C}'}$ is a t'-query zero-knowledge encoding with error $\zeta_{\mathcal{C}'}$, then Construction 9.9 is HVZK for $\mathfrak{D}^{\leq t'}$ with error $\zeta_{\mathcal{C}'} + \zeta_{\mathsf{ze}}$ and query complexity t. The simulator is uniform-query (Definition 4.10), and the reduction has fresh output (Definition 4.5).*

Proof. We describe the simulator $\mathbf{S}$ for Construction 9.9.

$\mathbf{S}^{\mathbb{y}, \mathcal{D}}(\mathbb{x})$:

1. Write $\mathcal{D} = (\mathcal{D}_0, \mathcal{D}_1)$ as in Definition 4.9. Sample the randomness of $\mathcal{D}_0$ and its query list Q' of length at most t'.
2. Independently sample $Q = (x_1, \ldots, x_t) \leftarrow [m]^t$, $\rho \leftarrow D_{\mathsf{ze}}$, and $(\rho_{\mathsf{ood}}, \boldsymbol{y}) \leftarrow \mathbf{S}_{\mathsf{ze}_{\mathsf{ood}}}$.
3. Query $\mathbb{y}[Q]$ and compute $\mathbb{x}'$ as in Equation 9.
4. Sample $\boldsymbol{a}' \leftarrow \mathsf{Sim}_{\mathcal{C}'}(Q')$.
5. Run $\mathcal{D}_0(\mathbb{x}')$ with the sampled randomness to obtain st, and set $\mathsf{out} \coloneqq \mathcal{D}_1(\mathsf{st}, Q', \boldsymbol{a}')$.
6. Output

$$\begin{pmatrix} (\rho_{\mathsf{ood}}, (Q, \rho)), \\ (Q, Q', [t_{\mathsf{ood}}]), \\ (\mathbb{y}[Q], \boldsymbol{a}', \boldsymbol{y}), \\ \mathsf{out} \end{pmatrix} .$$

The simulator makes exactly t independent uniform queries to $\mathbb{y}$. These queries and their answers are included in its output, as required by Definitions 4.3 and 4.10.

The proof follows by hybrids H_0, H_1, H_2, starting with the real experiment H_0. In H_1, the answers $g[Q']$ are replaced with $\mathsf{Sim}_{\mathcal{C}'}(Q')$. We condition on the input witness, the padding $\boldsymbol{r}_{\mathsf{ood}}$, the verifier randomness, and the randomness of $\mathcal{D}_0$, so that $\mathbb{x}'$ and the number of output queries

are fixed. Their locations remain uniform and independent of $\boldsymbol{r}'$, which is used only to encode g. By the zero-knowledge property of $\mathsf{Enc}_{\mathcal{C}'}$ and statistical-distance contraction,

$$\Delta(H_0, H_1) \le \zeta_{\mathcal{C}'} .$$

In H_2, the pair $(\rho_{\mathsf{ood}}, \boldsymbol{y})$ is further replaced with a sample from $\mathbf{S}_{\mathsf{ze}_{\mathsf{ood}}}$. After the first replacement, all remaining components are computed from this pair, the input witness, and randomness independent of $\boldsymbol{r}_{\mathsf{ood}}$. The privacy of $\mathsf{ze}_{\mathsf{ood}}$ gives

$$\Delta(H_1, H_2) \le \zeta_{\mathsf{ze}} .$$

The distribution H_2 is the simulator's output.

Finally, the reduction has fresh output. Its output implicit instance is the proof string g, which the verifier does not query. The view of the verifier is determined by the input, its randomness, and $\boldsymbol{y} = \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}}) \cdot \boldsymbol{g}$, and hence by the input, its randomness, and the output message $\boldsymbol{g}$. The prover samples $\boldsymbol{r}'$ uniformly and independently of these, and uses it only to compute $g = \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{g}, \boldsymbol{r}')$. Conditioned on the view of the verifier and on $\boldsymbol{g}$, the randomness $\boldsymbol{r}'$ is therefore uniform, as Definition 4.5 requires. □

9.6 Round-by-round knowledge soundness

Lemma 9.11. *For every $\delta, \delta' \in (0,1)$, Construction 9.9 has RBR knowledge soundness with relaxation*

$$\left(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}}, \tilde{\mathcal{R}}^{\delta'}_{\mathcal{C}',\mathsf{sl}'}\right)$$

and errors

$$\left(\frac{\Lambda(\mathcal{C}', \delta')^2}{2} \cdot \varepsilon_{\mathsf{ood}}, \quad (1-\delta)^t + \Lambda(\mathcal{C}', \delta') \cdot \varepsilon_{\mathsf{zero}}\right) .$$

The total extraction time is $O\left(\mathsf{tenc}(\mathcal{C}) + wm\right)$.

Proof. We give notation, describe the extractors and state function, and bound the round-by-round errors. We split the final verifier message (Q, ρ) into two substeps, first Q and then ρ, and add their errors.

Notation. Let $\mathbb{x} = (\mu, \mathsf{st})$ and $\mathbb{y} = f$. A complete transcript has the form

$$\mathsf{tr} = (g, \rho_{\mathsf{ood}}, \boldsymbol{y}, (Q, \rho)), \qquad Q = (x_1, \ldots, x_t) .$$

Let $\mathsf{Coll}(g, \rho_{\mathsf{ood}})$ denote the event

$$\left\{ \begin{array}{l} \exists\, u, v \in \Lambda(\mathcal{C}', g, \delta') : \\ \boldsymbol{u} \neq \boldsymbol{v} \quad \wedge \quad \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}})\boldsymbol{u} = \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}})\boldsymbol{v} \end{array} \right\} ,$$

where $(\boldsymbol{u}, \boldsymbol{r}') = \mathsf{Enc}^{-1}_{\mathcal{C}'}(u)$ and similarly for v.

Extractors. The batching substep and the out-of-domain round use the identity extractor. The extractor for the in-domain samples parses

$$\mathsf{w} = ((\boldsymbol{g}, \boldsymbol{r}'), \bar{\mathbb{y}}'), \qquad \boldsymbol{g} = (\boldsymbol{f}, \boldsymbol{r}, \boldsymbol{r}_{\mathsf{ood}}) ,$$

and outputs

$$((\boldsymbol{f}, \boldsymbol{r}), \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r})) .$$

It outputs $\perp$ if parsing fails. Composing the two substep extractors gives the final-round extractor. Their total cost is $O(\mathsf{tenc}(\mathcal{C}) + wm)$.

The state function. We define KState, unchanged on prover moves.

0. **Initial transcript.** $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \emptyset, \mathsf{w}) = 1$ if and only if

$$(\mathbb{x}, \mathbb{y}, \mathsf{w}) \in \tilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}} .$$

1. **Out-of-domain samples.** After ρ_{ood}, the state is 1 if and only if either
   (a) $\mathsf{KState}(\mathbb{x}, \mathbb{y}, \emptyset, \mathsf{w}) = 1$, or
   (b) $\mathsf{Coll}(g, \rho_{\mathsf{ood}})$ holds.

2. **In-domain samples.** After Q, the state is 1 if and only if $\mathsf{w} = ((\boldsymbol{g}, \boldsymbol{r}'), \bar{\mathbb{y}}')$, with $\boldsymbol{g} = (\boldsymbol{f}, \boldsymbol{r}, \boldsymbol{r}_{\mathsf{ood}})$, satisfies

$$\begin{aligned} \bar{g} \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{g}, \boldsymbol{r}') \in \Lambda(\mathcal{C}', g, \delta'), \qquad & \bar{\mathbb{y}}' = \bar{g}, \\ \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f} \rangle = \mu, \qquad & \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}})\boldsymbol{g} = \boldsymbol{y}, \\ \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r})[Q] = \mathbb{y}[Q] . & \end{aligned}$$

3. **Combination randomness.** After ρ, the state is 1 if and only if

$$(\mathbb{x}', g, \mathsf{w}) \in \tilde{\mathcal{R}}^{\delta'}_{\mathcal{C}',\mathsf{sl}'} .$$

Bounding the errors. We bound the error of the out-of-domain round and of the two substeps of the final round.

1. **Out-of-domain samples.** The initial state is independent of ρ_{ood}. By Lemma 9.1,

$$\Pr_{\rho_{\mathsf{ood}} \leftarrow D_{\mathsf{ood}}} \left[\begin{array}{l} \exists\, \mathsf{w} : \\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, g, \mathsf{w}) = 0 \wedge \\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, (g, \rho_{\mathsf{ood}}), \mathsf{w}) = 1 \end{array} \right] \leq \Pr[\mathsf{Coll}(g, \rho_{\mathsf{ood}})] \leq \frac{\Lambda(\mathcal{C}', \delta')^2}{2} \cdot \varepsilon_{\mathsf{ood}} .$$

2. **In-domain samples.** Fix the transcript $\mathsf{tr} = (g, \rho_{\mathsf{ood}}, \boldsymbol{y})$ before the queries. If $\mathsf{Coll}(g, \rho_{\mathsf{ood}})$ holds, the preceding state accepts every witness, and extraction cannot fail. Otherwise, there is at most one message $\boldsymbol{g}$ represented in $\Lambda(\mathcal{C}', g, \delta')$ satisfying

$$\mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}})\boldsymbol{g} = \boldsymbol{y} .$$

   If such a message $\boldsymbol{g} = (\boldsymbol{f}, \boldsymbol{r}, \boldsymbol{r}_{\mathsf{ood}})$ exists, let $\bar{\mathbb{y}} = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r})$. This word is fixed before Q is sampled. If the target equation fails, no post-state accepts. Otherwise, extraction can fail only if $\Delta(\mathbb{y}, \bar{\mathbb{y}}) > \delta$ and all queried positions agree. Hence

$$\Pr_{Q \leftarrow [m]^t} \left[\begin{array}{l} \exists\, \mathsf{w} : \\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr}, \mathbf{E}_{\mathsf{rbr}}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| Q, \mathsf{w})) = 0 \wedge \\ \mathsf{KState}(\mathbb{x}, \mathbb{y}, \mathsf{tr} \| Q, \mathsf{w}) = 1 \end{array} \right] \leq (1 - \delta)^t .$$

3. **Combination randomness.** Fix the transcript through Q. For a codeword $\bar{g} \in \Lambda(\mathcal{C}', g, \delta')$ with message $\boldsymbol{g} = (\boldsymbol{f}, \boldsymbol{r}, \boldsymbol{r}_{\mathsf{ood}})$, let $\boldsymbol{d}_{\bar{g}} \in \mathcal{M}^{1+t_{\mathsf{ood}}+wt}$ consist, in order, of the discrepancies

$$\begin{pmatrix} \langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f} \rangle - \mu \\ \mathsf{ze}_{\mathsf{ood}}(\rho_{\mathsf{ood}})\boldsymbol{g} - \boldsymbol{y} \\ (\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{f}, \boldsymbol{r})(x_i)_j - \mathbb{y}(x_i)_j)_{i \in [t],\, j \in [w]} \end{pmatrix}.$$

The last three equations defining the state after Q hold if and only if $\boldsymbol{d}_{\bar{g}} = 0$. The final target equation is their combination with coefficients $\mathsf{ze}(\rho)$. By the zero-evader guarantee and a union bound over the fixed list,

$$\Pr_{\rho \leftarrow D_{\mathsf{ze}}} \left[\begin{array}{l} \exists\, \bar{g} \in \Lambda(\mathcal{C}', g, \delta') : \\ \boldsymbol{d}_{\bar{g}} \neq 0 \ \wedge\ \sum_i (\boldsymbol{d}_{\bar{g}})_i \mathsf{ze}(\rho)_i = 0 \end{array} \right] \leq \Lambda(\mathcal{C}', \delta') \cdot \varepsilon_{\mathsf{zero}}\,.$$

This bounds the batching substep's extraction error. Adding the last two substep errors proves the final-round bound.

□

9.7 Proof of Theorem 4

We combine Construction 6.4 and Construction 9.9 to give a formal statement and proof of Theorem 4.

Theorem 9.12. *Let:*

- R *be a finite ring,* $\mathcal{M}$ *a right* R*-module,* $k \geq 1$ *a folding parameter, and* $w, t, t_{\mathsf{ood}} \geq 1$ *integers,*
- $\boldsymbol{m} \in \mathcal{M}^{d_{\mathcal{M}}}$ *generate* $\mathcal{M}$*,* $\mathfrak{C} \subseteq \mathcal{Z}(R)$ *be a finite nonempty challenge set, and* $d_s \geq 3$ *be a masking degree, with* $p := d_{\mathcal{M}}(d_s - 1)$*,*
- $\mathcal{C} \subseteq (\mathcal{M}^w)^m$ *have an encoding* $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell} \times R^{p+r} \to (\mathcal{M}^w)^m$*,*
- $\mathsf{sl} \in \langle \mathcal{M}^{2^k \ell} |$*,*
- $\mathcal{C}' \subseteq (\Sigma')^{m'}$ *have an injective* R*-linear encoding* $\mathsf{Enc}_{\mathcal{C}'} \colon R^{\ell'} \times R^{r'} \to \mathcal{C}'$*, where*

$$\ell' := \ell + p + r + r_{\mathsf{ood}}, \qquad r_{\mathsf{ood}} \in \mathbb{N}\,,$$

- $\mathsf{ze} \colon D_{\mathsf{ze}} \to \mathcal{Z}(R)^{1+t_{\mathsf{ood}}+wt}$ *be a zero-evader over* $\mathcal{M}$ *with error* $\varepsilon_{\mathsf{zero}}$*,*
- $\mathsf{ze}_{\mathsf{ood}} \colon D_{\mathsf{ood}} \to \mathcal{M}^{t_{\mathsf{ood}} \times \ell'}$ *be a zero-evader over* R *with error* $\varepsilon_{\mathsf{ood}}$*. For* $i \in [t_{\mathsf{ood}}]$*, let* $\mathsf{ze}_{\mathsf{ood},i}$ *denote its* i*-th row.*

Let $\mathsf{sl}_{\mathsf{sc}}$ *denote the output linear form of Theorem 6.3, and define*

$$\mathsf{sl}' := \mathsf{ze}\left[0^{r+r_{\mathsf{ood}}}(\mathsf{sl}_{\mathsf{sc}}), (\mathsf{ze}_{\mathsf{ood},i})_{i \in [t_{\mathsf{ood}}]}, (0^{r_{\mathsf{ood}}}(\mathsf{sl}_{\mathcal{C}^{+p}}))_{(i,j) \in [t] \times [w]}\right].$$

There exists an IOR from $\mathcal{R}_{\mathcal{C}^{\equiv 2^k}, \mathsf{sl}}$ *to* $\mathcal{R}_{\mathcal{C}', \mathsf{sl}'}$ *with the following properties:*

- *Round complexity:* $k + 3$*.*
- *Prover communication:* m' *symbols of* Σ' *sent as an oracle, and* $kd_s + t_{\mathsf{ood}} + 1$ *elements of* $\mathcal{M}$ *sent as non-oracle messages.*
- *Query complexity:* t *queries to the input oracle. The output oracle is the new oracle.*
- *Prover time (in ring and module operations):*

$$O\left(2^k(\ell + r) + t_{\mathsf{sl}} + kp + \mathsf{tenc}(\mathcal{C}') + \mathsf{teval}(\mathsf{ze}_{\mathsf{ood}}) + t_{\mathsf{ood}}\ell'\right).$$

- *Verifier time (in ring and module operations):* $O\left(kd_s + \mathsf{teval}(\mathsf{ze}) + t_{\mathsf{ood}} + t \cdot 2^k w\right)$.
- **Round-by-round security:** *For $\delta \in (0, \delta(\mathcal{C}))$ and $\delta' \in (0,1)$, the protocol has RBR knowledge soundness with relaxation*
$$\left(\tilde{\mathcal{R}}^{\delta}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}}, \tilde{\mathcal{R}}^{\delta'}_{\mathcal{C}',\mathsf{sl}'}\right)$$
and errors
$$\begin{pmatrix} \Lambda(\mathcal{C}^{\equiv 2^k}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, 2), \\ \left(\epsilon_{\mathsf{mca}}(\mathcal{C}^{\equiv 2^{k-j}}, \delta) + \Lambda(\mathcal{C}^{\equiv 2^{k-j+1}}, \delta) \cdot \epsilon^{\mathcal{M}}_{\mathsf{pil}}(\mathfrak{C}, 1, d_s)\right)_{j \in [k]}, \\ \frac{\Lambda(\mathcal{C}', \delta')^2}{2} \cdot \varepsilon_{\mathsf{ood}}, \quad (1-\delta)^t + \Lambda(\mathcal{C}', \delta') \cdot \varepsilon_{\mathsf{zero}} \end{pmatrix}.$$
The total extraction time is $O\left(k \cdot 2^k m + wm + \mathsf{tenc}(\mathcal{C}) + \sum_{j \in [k]} \mathsf{tcor}(\mathcal{C}^{\equiv 2^{k-j}})\right)$.
- **Zero-knowledge:** *Suppose that:*
  - $\mathsf{ze}_{\mathsf{ood}}$ *is $(r_{\mathsf{ood}}, \zeta_{\mathsf{ze}})$-private,*
  - $\mathsf{Enc}_{\mathcal{C}'}$ *is a t'-query zero-knowledge encoding with error $\zeta_{\mathcal{C}'}$,*
  - $\mathsf{Enc}_{\mathcal{C}}$ *is a t-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}}$.*

  Then the protocol is HVZK on fresh inputs for $\mathfrak{D}^{\leq t'}$ with error $\zeta_{\mathcal{C}'} + \zeta_{\mathsf{ze}} + 2^k \cdot \zeta_{\mathcal{C}}$ and query complexity t. The simulator is uniform-query (Definition 4.10), and the protocol has fresh output (Definition 4.5).

Proof. Let $\mathsf{IOR}_{\mathsf{sc}} \colon \mathcal{R}_{\mathcal{C}^{\equiv 2^k},\mathsf{sl}} \to \mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}_{\mathsf{sc}}}$ be the IOR from Theorem 6.3, and let $\mathsf{IOR}_{\mathsf{cs}} \colon \mathcal{R}_{\mathcal{C}^{+p},\mathsf{sl}_{\mathsf{sc}}} \to \mathcal{R}_{\mathcal{C}',\mathsf{sl}'}$ be the IOR from Theorem 9.8. Define $\mathsf{IOR} \coloneqq \mathsf{IOR}_{\mathsf{sc}} \circ \mathsf{IOR}_{\mathsf{cs}}$.

The code $\mathcal{C}^{+p}$ has message length $\ell + p$ and randomness length r, and the same codewords as $\mathcal{C}$. Thus the code switch encodes the folded message, mask, and seed, together with r_{ood} fresh OOD coordinates, in $\mathcal{C}'$, and its output form is sl'.

The round and communication bounds follow by adding the bounds of the two theorems. Each query to the intermediate oracle uses one input-oracle query and $O(2^k w)$ operations. Adding the wt batching cost gives the verifier bound. The prover bound is the sum of the two prover bounds, and the extraction bound is the sum of the two extraction bounds. The RBR errors follow from Theorem 4.7, using distance δ for the common intermediate relation.

For zero-knowledge, $\mathcal{M}$ is finite since R is finite and $\boldsymbol{m}$ generates $\mathcal{M}$. Fix $\mathcal{D} \in \mathfrak{D}^{\leq t'}$, let $\mathbf{S}_{\mathsf{cs}}$ be the simulator from Theorem 9.8, and let $\mathcal{D}_* \coloneqq \mathcal{D}_{\mathbf{S}_{\mathsf{cs}}}$. Since $\mathbf{S}_{\mathsf{cs}}$ is uniform-query and makes t queries to the intermediate oracle, Claim 4.11 gives $\mathcal{D}_* \in \mathfrak{D}^{\leq t}$. By Theorem 6.3, there is a simulator $\mathbf{S}_{\mathsf{sc}}$ for $\mathsf{IOR}_{\mathsf{sc}}$ on fresh inputs with error $2^k \cdot \zeta_{\mathcal{C}}$ and query complexity t to the original implicit instance. Since the composed simulator queries the implicit instance only through $\mathbf{S}_{\mathsf{sc}}$, it is uniform-query.

Since $\mathsf{IOR}_{\mathsf{cs}}$ is HVZK (Theorem 9.8), Theorem 4.7 applies although $\mathsf{IOR}_{\mathsf{sc}}$ does not have fresh output. This gives total simulation error $2^k \cdot \zeta_{\mathcal{C}} + (\zeta_{\mathcal{C}'} + \zeta_{\mathsf{ze}})$ on fresh inputs. The protocol has fresh output because $\mathsf{IOR}_{\mathsf{cs}}$ does. □

10 Zero-knowledge IOPP for constrained codes

In this section, we formally state and prove Theorem 5. The resulting protocol can be seen as a zero-knowledge variant (and generalization) of both WHIR [ACFY25] and Ligerito [NA25].

Batched linear forms. We define the cost of evaluating a batch of rows of a linear form.

Definition 10.1. *Let* $\mathsf{sl} \in \langle \mathcal{M}^{t\times\ell}|$ *and let* $b \in \mathbb{N}$. *We let* $\mathsf{b}(\mathsf{sl}, b)$ *denote the number of ring and module operations needed to compute*

$$\sum_{i\in[b]} \mathsf{sl}(\mathsf{st}_i)\cdot \boldsymbol{\nu}_i \in \mathcal{M}^{t\times\ell}$$

given $(\mathsf{st}_1,\dots,\mathsf{st}_b)$ *and central coefficients* $\boldsymbol{\nu} \in \mathcal{Z}(R)^b$. *In particular,*

$$\mathsf{b}(\mathsf{sl}, b) \le b\cdot t_{\mathsf{sl}} + O(b\cdot t\cdot \ell)$$

ring and module operations.

For many linear forms (e.g., the ones in Section 9.3) the cost to compute such batches can be much smaller than the naive bound above. To see this, fix a matrix $\boldsymbol{M} \in \mathcal{M}^{m\times\ell}$ and let $|\boldsymbol{M}|$ denote the number of module and ring operations to compute $\boldsymbol{M}\cdot\boldsymbol{v}$ for any $\boldsymbol{v}\in R^\ell$. Let $\mathsf{sl}_{\boldsymbol{M}}$ map $x\in[m]$ to $\boldsymbol{M}[x,\cdot]\in\mathcal{M}^\ell$. Given $x_1,\dots,x_b$ and $\boldsymbol{\nu}$, the batch

$$\sum_{i\in[b]} \boldsymbol{M}[x_i,\cdot]\cdot\boldsymbol{\nu}_i = \boldsymbol{M}^{\mathsf{T}}\cdot\left(\sum_{i\in[b]} \boldsymbol{e}_{x_i}\cdot\boldsymbol{\nu}_i\right)$$

can be computed in $O(b+|\boldsymbol{M}^{\mathsf{T}}|) = O(b+|\boldsymbol{M}|)$ operations by the transposition principle [Bor57; Ber].

Main theorem. The main theorem in this section follows by iteratively applying the zero-knowledge sumcheck IOR in Section 6, the zero-knowledge code-switching IOR in Section 9, and finally the non-succinct base case in Section 7.

Theorem 10.2. *Fix the following:*

- *a finite ring* R, *a right* R*-module* $\mathcal{M}$ *generated by* $\boldsymbol{m}\in\mathcal{M}^{d_{\mathcal{M}}}$, *a finite nonempty challenge set* $\mathfrak{C}\subseteq\mathcal{Z}(R)$, *and integers* $\mathfrak{n}\ge 2$ *and* $d_s\ge 3$. *Let* $p_{\mathfrak{i}} := d_{\mathcal{M}}(d_s-1)$ *for* $\mathfrak{i}\in[\mathfrak{n}-1]$ *and* $p_{\mathfrak{n}} := 0$.
- *for every* $\mathfrak{i}\in[\mathfrak{n}]$, *an* $(R,\mathcal{M})$*-additive code* $\mathcal{C}_{\mathfrak{i}}\subseteq\mathcal{M}^{m_{\mathfrak{i}}}$ *with an encoding*

$$\mathsf{Enc}_{\mathcal{C}_{\mathfrak{i}}}\colon R^{\ell_{\mathfrak{i}}+a_{\mathfrak{i}}}\times R^{p_{\mathfrak{i}}+r_{\mathfrak{i}}}\to\mathcal{C}_{\mathfrak{i}}\,,$$

  whose messages consist of a main block of length $\ell_{\mathfrak{i}}$ *followed by an auxiliary block of length* $a_{\mathfrak{i}}$, *and whose randomness consists of a mask of length* $p_{\mathfrak{i}}$ *and a seed of length* $r_{\mathfrak{i}}$.
- *for every* $\mathfrak{i}\in[\mathfrak{n}-1]$, *integers* $k_{\mathfrak{i}}\ge 1$, $t_{\mathsf{ood},\mathfrak{i}}\ge 1$, *and* $r_{\mathsf{ood},\mathfrak{i}}\in\mathbb{N}$.
- *query counts* $t_{\mathfrak{i}}\ge 1$ *for* $\mathfrak{i}\in[\mathfrak{n}]$, *and* $k_{\mathfrak{n}} := 0$.
- *for every* $\mathfrak{i}\in[\mathfrak{n}-1]$, *assume*

$$2^{k_{\mathfrak{i}+1}}\ell_{\mathfrak{i}+1} = \ell_{\mathfrak{i}},$$
$$2^{k_{\mathfrak{i}+1}}a_{\mathfrak{i}+1} = a_{\mathfrak{i}} + p_{\mathfrak{i}} + r_{\mathfrak{i}} + r_{\mathsf{ood},\mathfrak{i}}\,.$$

- $\mathsf{sl}_0 = (\mathsf{sl}, \mathsf{sl}_a)$, *where* $\mathsf{sl} \in \langle \mathcal{M}^{2^{k_1}\ell_1}|$ *and* $\mathsf{sl}_a \in \langle \mathcal{M}^{2^{k_1}a_1}|$.
- *for every* $\mathsf{i} \in [\mathfrak{n}-1]$, *a zero-evader* $\mathsf{ze}_\mathsf{i} \colon D_{\mathsf{ze},\mathsf{i}} \to \mathcal{Z}(R)^{1+t_{\mathsf{ood},\mathsf{i}}+t_\mathsf{i}}$ *over* $\mathcal{M}$ *with error* $\varepsilon_{\mathsf{zero},\mathsf{i}}$, *and a zero-evader*

$$\mathsf{ze}_{\mathsf{ood},\mathsf{i}} \colon D_{\mathsf{ood},\mathsf{i}} \to \mathcal{M}^{t_{\mathsf{ood},\mathsf{i}} \times \left(\ell_\mathsf{i} + 2^{k_{\mathsf{i}+1}} a_{\mathsf{i}+1}\right)}$$

  over R *with error* $\varepsilon_{\mathsf{ood},\mathsf{i}}$.

Let $\mathcal{C}_{\mathrm{in},\mathsf{i}} \coloneqq \mathcal{C}_\mathsf{i}^{\equiv 2^{k_\mathsf{i}}}$ *for* $\mathsf{i} \in [\mathfrak{n}]$, *with the interleaved encodings, whose randomness is interpreted as in Construction 6.4, and let* $K_0 \coloneqq 0$, $K_j \coloneqq \sum_{\mathsf{i} \in [j]} k_\mathsf{i}$, *and* $K \coloneqq K_{\mathfrak{n}-1}$. *We write the messages of* $\mathcal{C}_{\mathrm{in},\mathsf{i}}$ *as pairs* $(\boldsymbol{f}, \boldsymbol{v}) \in R^{2^{k_\mathsf{i}}\ell_\mathsf{i}} \times R^{2^{k_\mathsf{i}}a_\mathsf{i}}$ *whose* $\boldsymbol{b}$*-th blocks* $(\boldsymbol{f_b}, \boldsymbol{v_b})$ *form the message of the* $\boldsymbol{b}$*-th row, and linear forms on them as pairs of linear forms. Folding and fixing act on each component. The restrictions of* $\mathsf{ze}_{\mathsf{ood},\mathsf{i}}$ *to the first* ℓ_i *and the remaining coordinates are denoted* $\mathsf{ze}^{\leftarrow}_{\mathsf{ood},\mathsf{i}}$ *and* $\mathsf{ze}^{\rightarrow}_{\mathsf{ood},\mathsf{i}}$, *respectively. Similarly,* $\mathsf{sl}^{\#}_{\mathcal{C}_\mathsf{i}}$ *and* $\mathsf{sl}^{\$}_{\mathcal{C}_\mathsf{i}}$ *denote the restrictions of the generator form of* $\mathcal{C}_\mathsf{i}$ *(Definition 9.6) to the* ℓ_i *columns of the main block and to the remaining* $a_\mathsf{i} + p_\mathsf{i} + r_\mathsf{i}$ *columns, respectively. There exists an IOPP for* $\mathcal{R}_{\mathcal{C}_{\mathrm{in},1},\mathsf{sl}_0}$ *with the following properties.*

- *Round complexity:* $K + 3(\mathfrak{n}-1) + 2$ *rounds.*
- *Prover communication:*
  - *for every* $\mathsf{i} \in [\mathfrak{n}-1]$, $m_{\mathsf{i}+1}$ *symbols of* $\mathcal{M}^{2^{k_{\mathsf{i}+1}}}$ *are sent as an oracle.*
  - $m_\mathfrak{n}$ *symbols of* $\mathcal{M}$ *are sent as an oracle in the base case.*
  - $\sum_{\mathsf{i} \in [\mathfrak{n}-1]} (k_\mathsf{i} d_s + t_{\mathsf{ood},\mathsf{i}} + 1) + 1$ *elements of* $\mathcal{M}$ *and* $\ell_\mathfrak{n} + a_\mathfrak{n} + r_\mathfrak{n}$ *elements of* R *are sent as non-oracle messages.*
- *Query complexity:* t_i *queries to the oracle of each stage* $\mathsf{i} \in [\mathfrak{n}]$, *and* $t_\mathfrak{n}$ *queries to the base-case proof oracle.*
- *Prover time (in ring and module operations):*

$$O\left(\begin{array}{l} t_\mathsf{sl} + t_{\mathsf{sl}_a} + \displaystyle\sum_{\mathsf{i} \in [\mathfrak{n}-1]} \left(\begin{array}{l} 2^{k_\mathsf{i}}(\ell_\mathsf{i} + a_\mathsf{i} + p_\mathsf{i} + r_\mathsf{i}) + \mathsf{tenc}(\mathcal{C}_{\mathrm{in},\mathsf{i}+1}) + k_\mathsf{i} p_\mathsf{i} \\ + \mathsf{teval}(\mathsf{ze}_\mathsf{i}) + \mathsf{teval}(\mathsf{ze}_{\mathsf{ood},\mathsf{i}}) + t_{\mathsf{ood},\mathsf{i}} \left(\ell_\mathsf{i} + 2^{k_{\mathsf{i}+1}} a_{\mathsf{i}+1}\right) \\ + \mathsf{b}(\mathsf{sl}^{\#}_{\mathcal{C}_\mathsf{i}}, t_\mathsf{i}) + \mathsf{b}(\mathsf{sl}^{\$}_{\mathcal{C}_\mathsf{i}}, t_\mathsf{i}) \end{array}\right) \\ + \mathsf{tenc}(\mathcal{C}_{\mathrm{in},\mathfrak{n}}) + \ell_\mathfrak{n} + a_\mathfrak{n} + r_\mathfrak{n} \end{array}\right).$$

- *Verifier time (in ring and module operations):*

$$O\left(\begin{array}{l} t_{\odot_{K^\mathsf{sl}}} + t_{\mathsf{sl}_a} + \displaystyle\sum_{\mathsf{i} \in [\mathfrak{n}]} 2^{k_\mathsf{i}} a_\mathsf{i} \\ + \displaystyle\sum_{\mathsf{i} \in [\mathfrak{n}-1]} \left(\begin{array}{l} k_\mathsf{i} p_\mathsf{i} + \mathsf{teval}(\mathsf{ze}_\mathsf{i}) + t_{\mathsf{ood},\mathsf{i}} + t_\mathsf{i} 2^{k_\mathsf{i}} \\ + \mathsf{b}(\odot_{K-K_\mathsf{i}} \mathsf{sl}^{\#}_{\mathcal{C}_\mathsf{i}}, t_\mathsf{i}) + \mathsf{b}(\mathsf{sl}^{\$}_{\mathcal{C}_\mathsf{i}}, t_\mathsf{i}) \\ + \mathsf{b}(\odot_{K-K_\mathsf{i}} \mathsf{sl}_{\mathsf{ze}^{\leftarrow}_{\mathsf{ood},\mathsf{i}}}, t_{\mathsf{ood},\mathsf{i}}) + \mathsf{b}(\mathsf{sl}_{\mathsf{ze}^{\rightarrow}_{\mathsf{ood},\mathsf{i}}}, t_{\mathsf{ood},\mathsf{i}}) \end{array}\right) \\ + \mathsf{tenc}(\mathcal{C}_{\mathrm{in},\mathfrak{n}}) + \ell_\mathfrak{n} + a_\mathfrak{n} + t_\mathfrak{n} \end{array}\right).$$

- **Round-by-round security:** *For every* $\delta_\mathsf{i} \in (0, \delta(\mathcal{C}_\mathsf{i}))$, $\mathsf{i} \in [\mathfrak{n}]$, *the protocol has RBR knowledge soundness with relaxation* $\tilde{\mathcal{R}}^{\delta_1}_{\mathcal{C}_{\mathrm{in},1},\mathsf{sl}_0}$, *errors* $\boldsymbol{\epsilon}_1 \| \cdots \| \boldsymbol{\epsilon}_{\mathfrak{n}-1} \| \boldsymbol{\epsilon}_{\mathsf{base}}$ *defined below, and total extraction time*

$$O\left(\sum_{\mathsf{i} \in [\mathfrak{n}-1]} \left(m_\mathsf{i}\left(k_\mathsf{i} 2^{k_\mathsf{i}} + 1\right) + \mathsf{tenc}(\mathcal{C}_\mathsf{i}) + \sum_{j \in [k_\mathsf{i}]} \mathsf{tcor}(\mathcal{C}_\mathsf{i}^{\equiv 2^{k_\mathsf{i}-j}})\right) + m_\mathfrak{n} + \mathsf{tcor}(\mathcal{C}_\mathfrak{n})\right).$$

– *For every* $\mathfrak{i} \in [\mathfrak{n}-1]$*, define*

$$\boldsymbol{\epsilon}_{\mathfrak{i}} \coloneqq \left(\begin{array}{l} \Lambda(\mathcal{C}_{\mathfrak{i}}^{\equiv 2^{k_{\mathfrak{i}}}}, \delta_{\mathfrak{i}}) \cdot \epsilon_{\mathsf{pil}}^{\mathcal{M}}(\mathfrak{C}, 1, 2), \\ \left(\epsilon_{\mathsf{mca}}(\mathcal{C}_{\mathfrak{i}}^{\equiv 2^{k_{\mathfrak{i}}-j}}, \delta_{\mathfrak{i}}) + \Lambda(\mathcal{C}_{\mathfrak{i}}^{\equiv 2^{k_{\mathfrak{i}}-j+1}}, \delta_{\mathfrak{i}}) \cdot \epsilon_{\mathsf{pil}}^{\mathcal{M}}(\mathfrak{C}, 1, d_s) \right)_{j \in [k_{\mathfrak{i}}]}, \\ \dfrac{\Lambda(\mathcal{C}_{\mathrm{in},\mathfrak{i}+1}, \delta_{\mathfrak{i}+1})^2}{2} \cdot \varepsilon_{\mathsf{ood},\mathfrak{i}}, \\ (1-\delta_{\mathfrak{i}})^{t_{\mathfrak{i}}} + \Lambda(\mathcal{C}_{\mathrm{in},\mathfrak{i}+1}, \delta_{\mathfrak{i}+1}) \cdot \varepsilon_{\mathsf{zero},\mathfrak{i}} \end{array} \right) .$$

– *Define*

$$\boldsymbol{\epsilon}_{\mathsf{base}} \coloneqq \left(\epsilon_{\mathsf{mca}}(\mathcal{C}_{\mathfrak{n}}, \delta_{\mathfrak{n}}) + \Lambda(\mathcal{C}_{\mathfrak{n}}^{\equiv 2}, \delta_{\mathfrak{n}}) \cdot \epsilon_{\mathsf{pil}}^{\mathcal{M}}(\mathfrak{C}, 1, 2), \quad (1-\delta_{\mathfrak{n}})^{t_{\mathfrak{n}}} \right) .$$

- **Zero-knowledge:** *Suppose that:*
  – *for every* $\mathfrak{i} \in [\mathfrak{n}-1]$*,* $\mathsf{ze}_{\mathsf{ood},\mathfrak{i}}$ *is* $(r_{\mathsf{ood},\mathfrak{i}}, \zeta_{\mathsf{ze},\mathfrak{i}})$*-private.*
  – *for every* $\mathfrak{i} \in [\mathfrak{n}]$*,* $\mathsf{Enc}_{\mathcal{C}_{\mathfrak{i}}}$ *is a* $t_{\mathfrak{i}}$*-query zero-knowledge encoding with* $p_{\mathfrak{i}}$ *spare coordinates and error* $\zeta_{\mathcal{C}_{\mathfrak{i}}}$ *(Definition 3.25).*

  Then the protocol is honest-verifier zero-knowledge on fresh inputs (Definition 4.3) with error

$$\sum_{\mathfrak{i} \in [\mathfrak{n}-1]} \left(2^{k_{\mathfrak{i}}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}}} + 2^{k_{\mathfrak{i}+1}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}+1}} + \zeta_{\mathsf{ze},\mathfrak{i}} \right)$$

and simulator query complexity t_1*. The simulator is uniform-query.*

The relation of Section 1.1 is the case $a_1 = 0$. A reduction that outputs an instance of $\mathcal{R}_{\mathcal{C}_{\mathrm{in},1}, \mathbf{sl}_0}$ and has fresh output (Definition 4.5) can be composed with this IOPP by Theorem 4.7. If the reduction is HVZK on fresh inputs for $\mathfrak{D}^{\leq t_1}$, then so is the composition. When the witnesses of its input relation have no randomness, as for R1CS, every input is fresh, and the composition is HVZK.

Proof. Let $\mathbf{sl}_0$ be the initial pair from the theorem statement. For every $\mathfrak{i} \in [\mathfrak{n}-1]$, we apply Theorem 9.12 with

$$(\mathcal{C}, \mathcal{C}', k, w, t, t_{\mathsf{ood}}) = (\mathcal{C}_{\mathfrak{i}}, \mathcal{C}_{\mathrm{in},\mathfrak{i}+1}, k_{\mathfrak{i}}, 1, t_{\mathfrak{i}}, t_{\mathsf{ood},\mathfrak{i}}) .$$

Here $\mathcal{C}_{\mathfrak{i}}$ has message length $\ell_{\mathfrak{i}} + a_{\mathfrak{i}}$, and the linear form on the messages of its interleaving is the pair $\mathbf{sl}_{\mathfrak{i}-1}$. The switch encodes the folded message, mask, and seed of stage $\mathfrak{i}$ together with an OOD pad. We order the message coordinates of $\mathcal{C}_{\mathrm{in},\mathfrak{i}+1}$ so that the main block of the folded message is the component $\boldsymbol{f}$ of the next message and all remaining coordinates form its component $\boldsymbol{v}$. This is possible by the dimension identities. Let $\mathbf{sl}_{\mathfrak{i}}$ be the output linear form split accordingly. Thus the resulting IOR $\mathsf{IOR}_{\mathfrak{i}}$ maps $\mathcal{R}_{\mathcal{C}_{\mathrm{in},\mathfrak{i}}, \mathbf{sl}_{\mathfrak{i}-1}}$ to $\mathcal{R}_{\mathcal{C}_{\mathrm{in},\mathfrak{i}+1}, \mathbf{sl}_{\mathfrak{i}}}$. Let $\mathsf{IOR}_{\mathsf{base}}$ be the protocol of Theorem 7.1 for $\mathcal{R}_{\mathcal{C}_{\mathrm{in},\mathfrak{n}}, \mathbf{sl}_{\mathfrak{n}-1}}$ with query count $t_{\mathfrak{n}}$. The final IOPP is $\mathsf{IOR}_1 \circ \cdots \circ \mathsf{IOR}_{\mathfrak{n}-1} \circ \mathsf{IOR}_{\mathsf{base}}$.

The round, communication, and query bounds and the extraction time are the sums of the corresponding bounds in Theorems 7.1 and 9.12, and the RBR error vectors are concatenated, by Theorem 4.7.

Zero-knowledge. For $\mathfrak{i} \in [\mathfrak{n}-1]$, put

$$\epsilon_{\mathfrak{i}} \coloneqq 2^{k_{\mathfrak{i}}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}}} + 2^{k_{\mathfrak{i}+1}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}+1}} + \zeta_{\mathsf{ze},\mathfrak{i}}, \qquad E_s \coloneqq \sum_{\mathfrak{i}=s}^{\mathfrak{n}-1} \epsilon_{\mathfrak{i}} .$$

- *Base case.* By Theorem 7.1, there is a perfect uniform-query simulator $\mathbf{S}_{\mathfrak{n}}$ making $t_{\mathfrak{n}}$ queries to its input. Its error is $E_{\mathfrak{n}} = 0$, for every input and hence on fresh inputs.
- *Inductive step ($\mathfrak{i} = \mathfrak{n} - 1$ down to 1).* Suppose the tail beginning at $\mathfrak{i}+1$ is HVZK on fresh inputs via a uniform-query simulator $\mathbf{S}_{\mathfrak{i}+1}$ with error $E_{\mathfrak{i}+1}$ and query complexity $t_{\mathfrak{i}+1}$. For any distinguisher $\mathcal{D}$, Claim 4.11 implies

  $$\mathcal{D}_{\mathbf{S}_{\mathfrak{i}+1}} \in \mathfrak{D}^{\leq t_{\mathfrak{i}+1}} .$$

  By Claim 3.26, the encoding of the next code $\mathcal{C}_{\mathrm{in},\mathfrak{i}+1}$ is, for every choice of the masks of its rows, $t_{\mathfrak{i}+1}$-query zero-knowledge with error $2^{k_{\mathfrak{i}+1}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}+1}}$, and reordering its message coordinates does not affect this property. Hence, by Theorem 9.12, stage $\mathfrak{i}$ is HVZK on fresh inputs via a uniform-query simulator with error $\epsilon_{\mathfrak{i}}$ and query complexity $t_{\mathfrak{i}}$, and it has fresh output. Applying Theorem 4.7 gives a uniform-query simulator for the tail on fresh inputs with error $E_{\mathfrak{i}}$ and query complexity $t_{\mathfrak{i}}$.
- *Conclusion.* The simulator for the full protocol has error E_1 on fresh inputs and makes t_1 uniform queries to the original implicit instance.

Efficiency. The prover evaluates the two initial linear forms once and maintains the coefficient vectors of the current linear form on the main blocks and on the auxiliary blocks. In each stage it folds these vectors and adds the batched generator and OOD rows that the code switch introduces. The old vectors cost $O(2^{k_{\mathfrak{i}}}(\ell_{\mathfrak{i}} + a_{\mathfrak{i}}))$ to fold, and the sumcheck messages of the mask polynomial cost $O(k_{\mathfrak{i}} p_{\mathfrak{i}})$ operations (Section 6). Computing the masks of the rows of the oracle of stage $\mathfrak{i}$ from its mask costs $O(2^{k_{\mathfrak{i}}} p_{\mathfrak{i}})$ operations. This gives the prover bound.

The verifier maintains the coefficient vectors of the auxiliary blocks, and leaves the form on the main blocks unevaluated until the base case. Fixing acts on the main and the auxiliary blocks separately, so the main generator and OOD rows introduced at stage $\mathfrak{i}$ then have $K - K_{\mathfrak{i}}$ variables fixed. Maintaining the explicit vectors costs $O\left(\sum_{\mathfrak{i}\in[\mathfrak{n}]} 2^{k_{\mathfrak{i}}} a_{\mathfrak{i}}\right)$ in addition to the initial evaluation of sl_a and the batched terms for $\mathsf{sl}^{\$}_{\mathcal{C}_{\mathfrak{i}}}$ and $\mathsf{sl}^{\rightarrow}_{\mathsf{ze}_{\mathsf{ood},\mathfrak{i}}}$. Adding the verifier costs of the reductions and base case gives the stated bound. □

10.1 Reed–Solomon instantiation

We instantiate Theorem 10.2 with Reed–Solomon codes to obtain the parameters of Corollary 2, with parameter choices resembling those of [ACFY25]. Let ℓ be a sufficiently large power of two and m the initial evaluation-domain size. We set $k := \lfloor \log\log \ell \rfloor$ and $\mathfrak{n} := 1 + \lceil \log(\ell)/k \rceil$. In Theorem 10.2, we use $k_{\mathfrak{i}} = k$ for $\mathfrak{i} \in [\mathfrak{n}-2]$, $k_{\mathfrak{n}-1} = \log \ell - (\mathfrak{n}-2)k \in [1,k]$, and $k_{\mathfrak{n}} = 0$. We choose evaluation domains on which each base code $\mathcal{C}_{\mathfrak{i}}$ can be encoded by FFT in $O(m_{\mathfrak{i}} \log m_{\mathfrak{i}})$ field operations.

- We set $\mathcal{M} = R = \mathbb{F}$ and $d_s = 3$, so that a mask has $p_{\mathfrak{i}} = 2$ coordinates for $\mathfrak{i} < \mathfrak{n}$, and $p_{\mathfrak{n}} = 0$.
- We require $r_1 \geq t_1$ and $r_1 = \Theta(\lambda)$, and, for every $\mathfrak{i} \in [2,\mathfrak{n}]$, set $r_{\mathfrak{i}} := t_{\mathfrak{i}}$. We take $\mathcal{C}_{\mathfrak{i}}$ to be the Reed–Solomon evaluation code for polynomials of degree less than $\ell_{\mathfrak{i}} + a_{\mathfrak{i}} + p_{\mathfrak{i}} + r_{\mathfrak{i}}$, where $a_{\mathfrak{i}}$ is the length of its auxiliary block (chosen below, with $a_1 = 0$), with $m_1 = m \geq \ell_1 + p_1 + r_1$ and, for $\mathfrak{i} \in [2,\mathfrak{n}]$, evaluation domain size $m_{\mathfrak{i}} = \Theta(2^{-(k+\mathfrak{i}-1)} m + \ell_{\mathfrak{i}} + a_{\mathfrak{i}} + r_{\mathfrak{i}} + k)$, rounded to a supported evaluation-domain size, at least $\ell_{\mathfrak{i}} + a_{\mathfrak{i}} + p_{\mathfrak{i}} + r_{\mathfrak{i}}$. Its encoding $\mathsf{Enc}_{\mathcal{C}_{\mathfrak{i}}} \colon \mathbb{F}^{\ell_{\mathfrak{i}}+a_{\mathfrak{i}}} \times \mathbb{F}^{p_{\mathfrak{i}}+r_{\mathfrak{i}}} \to \mathbb{F}^{m_{\mathfrak{i}}}$ is

that of Lemma 3.29 with $p_{\mathfrak{i}}$ of its message coordinates regarded as randomness. It is an $r_{\mathfrak{i}}$-query perfect zero-knowledge encoding with $p_{\mathfrak{i}}$ spare coordinates.

- We take $\ell_{\mathfrak{n}} = 1$ and $\ell_{\mathfrak{i}} = 2^{k_{\mathfrak{i}+1}}\ell_{\mathfrak{i}+1}$ for every $\mathfrak{i} \in [\mathfrak{n}-1]$. Then $\ell = 2^k \ell_1$, since $\sum_{\mathfrak{i}\in[\mathfrak{n}-1]} k_{\mathfrak{i}} = \log \ell$. In particular, $\ell_{\mathfrak{n}-1} = \ell_{\mathfrak{n}} = 1$, consistently with $k_{\mathfrak{n}} = 0$.
- We choose the auxiliary block lengths $a_2, \ldots, a_{\mathfrak{n}}$ successively as

$$a_{\mathfrak{i}+1} = \left\lceil \frac{a_{\mathfrak{i}} + p_{\mathfrak{i}} + r_{\mathfrak{i}} + 1}{2^{k_{\mathfrak{i}+1}}} \right\rceil \qquad (\mathfrak{i} \in [\mathfrak{n}-1]) \,,$$

  and set $r_{\mathsf{ood},\mathfrak{i}} = 2^{k_{\mathfrak{i}+1}} a_{\mathfrak{i}+1} - a_{\mathfrak{i}} - p_{\mathfrak{i}} - r_{\mathfrak{i}} \geq 1$ for $\mathfrak{i} \in [\mathfrak{n}-1]$.

We set $t_{\mathsf{ood},\mathfrak{i}} = 1$ and take as $\mathsf{ze}_{\mathsf{ood},\mathfrak{i}}$ a Reed–Solomon generator row sampled over $\mathbb{F}$, extended by Lemma 9.4 with $r_{\mathsf{ood},\mathfrak{i}}$ padding coordinates. We generate and apply this zero-evader in $O(\ell_{\mathfrak{i}} + 2^{k_{\mathfrak{i}+1}} a_{\mathfrak{i}+1})$ operations. Assume that the proximity parameters can be chosen in the Reed–Solomon mutual correlated agreement range for every $\mathcal{C}_{\mathfrak{i}}$ so that $\sum_{\mathfrak{i}\in[\mathfrak{n}]} t_{\mathfrak{i}} = O(\lambda)$, and assume $\mathfrak{n} = O(\lambda)$, $\log m = O(\log \ell)$, and $\log \lambda = O(\log \ell)$. Here the rate of $\mathcal{C}_{\mathfrak{i}}$ is $(\ell_{\mathfrak{i}} + a_{\mathfrak{i}} + p_{\mathfrak{i}} + r_{\mathfrak{i}})/m_{\mathfrak{i}}$, including the auxiliary block and the encoding randomness. For example, taking $t_{\mathfrak{i}} = \lceil (\lambda + \log_2(2\mathfrak{n}))/(-\log_2(1-\delta_{\mathfrak{i}})) \rceil$ bounds the total spot-check error by $2^{-\lambda-1}$. We assume that $\mathbb{F}$ is large enough for the remaining error terms in Theorem 10.2 to sum to at most $2^{-\lambda-1}$. All encodings are perfect, so the protocol is perfect HVZK on fresh inputs.

For the final Reed–Solomon code, the base-case verifier does not compute its encoding on the entire evaluation domain. As Construction 7.2 permits, it evaluates the corresponding polynomial only at the points that it queries. This costs $\widetilde{O}(\ell_{\mathfrak{n}} + a_{\mathfrak{n}} + r_{\mathfrak{n}} + t_{\mathfrak{n}}) + O(t_{\mathfrak{n}} \log m_{\mathfrak{n}}) = O(\lambda \log \ell) + \widetilde{O}(\lambda)$, rather than $\mathsf{tenc}(\mathcal{C}_{\mathfrak{n}})$, and leaves the base-case checks and their security proof unchanged.

For Reed–Solomon codes, we have $\mathsf{t}_{\mathcal{C}}^{\mathbf{P}} = O(t(\ell + \log m)) + \widetilde{O}(r+t)$ and, after folding all message variables, $\mathsf{t}_{\mathcal{C}}^{\mathbf{V}} = O(t \log m) + \widetilde{O}(r+t)$ (see Lemma 9.7 and the transposition principle above). The message-size and repetition assumptions give

$$\sum_{\mathfrak{i}\in[\mathfrak{n}-1]} t_{\mathfrak{i}} \ell_{\mathfrak{i}} \leq \frac{\ell}{2^k} \sum_{\mathfrak{i}\in[\mathfrak{n}-1]} t_{\mathfrak{i}} = O\left(\frac{\lambda \ell}{2^k}\right) .$$

Every stage but the last divides the auxiliary block by $2^{k_{\mathfrak{i}+1}} \geq 2$ before adding $O(r_{\mathfrak{i}} + 2^k)$ coordinates, so the recurrence for $a_{\mathfrak{i}}$ gives

$$\sum_{\mathfrak{i}\in[2,\mathfrak{n}]} 2^{k_{\mathfrak{i}}} a_{\mathfrak{i}} + \sum_{\mathfrak{i}\in[\mathfrak{n}-1]} r_{\mathsf{ood},\mathfrak{i}} = O(\lambda + \mathfrak{n} \cdot 2^k) = O(\lambda \log \ell) \,.$$

Also, $\sum_{\mathfrak{i}\in[\mathfrak{n}]} m_{\mathfrak{i}} = O(m + \lambda + \mathfrak{n} \cdot 2^k) = O(m)$, since $m \geq r_1 = \Theta(\lambda)$ and $m \geq \ell/2^k$. Moreover, $\sum_{\mathfrak{i}\in[2,\mathfrak{n}]} 2^{k_{\mathfrak{i}}} m_{\mathfrak{i}} = O(m + 2^k(\lambda + \log \ell))$, since $\sum_{\mathfrak{i}\in[2,\mathfrak{n}]} 2^{k_{\mathfrak{i}}} \ell_{\mathfrak{i}} = O(\ell_1) = O(m)$. With the $O(\lambda + \log \ell)$ field elements that the prover sends directly, and since $2^k \leq \log \ell$ and $(\log \ell)^2 = O(m)$, the prover sends $O(m + \lambda \log \ell)$ field elements in total. It encodes these interleaved codewords in $O(m \log m + \lambda(\log \ell)^2 + (\log \ell)^3)$ field operations. The last two terms are absorbed by $O(\lambda \ell / \log \ell)$. The batched auxiliary-block and randomness rows can be computed in $\widetilde{O}(\lambda \log \ell) + O(\lambda \log m)$ operations in total. We obtain the claimed time bounds.

Since $2^k \leq \log \ell$, query answers can be padded to the alphabet $\mathbb{F}^{\log \ell}$.

11 Full construction of the zero-knowledge IOPP

We write out the protocol in the form that we implement, for a list of codes over a finite ring R and a right R-module $\mathcal{M}$, with masking degree three. It applies Theorem 9.12 in every stage but the last and the sublinear protocol of Theorem 8.1 in the last. Unlike in Theorem 10.2, the last oracle is therefore interleaved and has masks, and the last stage folds it before the base case. The section ends with the Reed–Solomon codes of our implementation. Figure 7 illustrates the construction, drawn as the figures of Section 2.7.

Layout and encodings. Let R be a finite ring, $\mathcal{M}$ a right R-module generated by $\boldsymbol{m} \in \mathcal{M}^{d_{\mathcal{M}}}$, and $\mathfrak{C} \subseteq \mathcal{Z}(R)$ a finite nonempty challenge set. A mask has $p \coloneqq 2d_{\mathcal{M}}$ coordinates, indexed by pairs $(u, e) \in [d_{\mathcal{M}}] \times \{0, 1\}$.

There are $\mathfrak{n} \geq 1$ stages. In stage $\mathfrak{i}$, let $k_{\mathfrak{i}} \geq 1$ be the folding parameter, let $\ell_{\mathfrak{i}}$ be the length of the main block of a row and $a_{\mathfrak{i}}$ the length of its auxiliary block, with $a_1 = 0$, and let $t_{\mathfrak{i}}$ be the query count. The code $\mathcal{C}_{\mathfrak{i}} \subseteq \mathcal{M}^{m_{\mathfrak{i}}}$ is $(R, \mathcal{M})$-additive and has an encoding

$$\mathsf{Enc}_{\mathcal{C}_{\mathfrak{i}}} \colon R^{\ell_{\mathfrak{i}}+a_{\mathfrak{i}}} \times R^{p+r_{\mathfrak{i}}} \to \mathcal{C}_{\mathfrak{i}} \,,$$

whose randomness consists of a mask of p coordinates and a seed of $r_{\mathfrak{i}}$ coordinates. A row message of $\mathcal{C}_{\mathfrak{i}}$ is a main block followed by an auxiliary block. We assume that $\mathsf{Enc}_{\mathcal{C}_{\mathfrak{i}}}$ is a $t_{\mathfrak{i}}$-query zero-knowledge encoding with p spare coordinates and error $\zeta_{\mathcal{C}_{\mathfrak{i}}}$ (Definition 3.25). For $\mathfrak{i} < \mathfrak{n}$, the prover adds a pad of $r_{\mathsf{ood},\mathfrak{i}}$ coordinates in the code switch of stage $\mathfrak{i}$, and the lengths satisfy

$$\ell_{\mathfrak{i}} = 2^{k_{\mathfrak{i}+1}} \ell_{\mathfrak{i}+1}, \qquad a_{\mathfrak{i}} + p + r_{\mathfrak{i}} + r_{\mathsf{ood},\mathfrak{i}} = 2^{k_{\mathfrak{i}+1}} a_{\mathfrak{i}+1} \,. \tag{10}$$

The $t_{\mathsf{ood},\mathfrak{i}} \geq 1$ out-of-domain samples of stage $\mathfrak{i} < \mathfrak{n}$ are given by a zero-evader $\mathsf{ze}_{\mathsf{ood},\mathfrak{i}} \colon D_{\mathsf{ood},\mathfrak{i}} \to \mathcal{M}^{t_{\mathsf{ood},\mathfrak{i}} \times (\ell_{\mathfrak{i}} + 2^{k_{\mathfrak{i}+1}} a_{\mathfrak{i}+1})}$ over R (Definition 3.5) that is $(r_{\mathsf{ood},\mathfrak{i}}, \zeta_{\mathsf{ze},\mathfrak{i}})$-private (Section 9.2), and its batching coefficients by a zero-evader $\mathsf{ze}_{\mathfrak{i}} \colon D_{\mathsf{ze},\mathfrak{i}} \to \mathcal{Z}(R)^{1+t_{\mathsf{ood},\mathfrak{i}}+t_{\mathfrak{i}}}$ over $\mathcal{M}$ (Definition 3.6).

The oracle of stage $\mathfrak{i}$ is

$$f_{\mathfrak{i}} = \mathsf{Enc}_{\mathcal{C}_{\mathfrak{i}}^{\equiv 2^{k_{\mathfrak{i}}}}} \left((\boldsymbol{f}_{\mathfrak{i}}, \boldsymbol{v}_{\mathfrak{i}}), (\boldsymbol{s}_{\mathfrak{i}}, \boldsymbol{r}_{\mathfrak{i}}) \right) \,,$$

where $\boldsymbol{f}_{\mathfrak{i}} \in R^{2^{k_{\mathfrak{i}}} \ell_{\mathfrak{i}}}$ is the main message, $\boldsymbol{v}_{\mathfrak{i}} \in R^{2^{k_{\mathfrak{i}}} a_{\mathfrak{i}}}$ the auxiliary message, $\boldsymbol{s}_{\mathfrak{i}} \in (R^p)^{2^{k_{\mathfrak{i}}}}$ the masks, and $\boldsymbol{r}_{\mathfrak{i}} \in R^{2^{k_{\mathfrak{i}}} r_{\mathfrak{i}}}$ the seeds of its rows. As in Construction 6.4, the masks $\boldsymbol{s}_{\mathfrak{i}}$ are determined by a tuple $\boldsymbol{\sigma}_{\mathfrak{i}} \in (R^p)^{k_{\mathfrak{i}}+1}$. The input oracle f_1 is given, with uniform $\boldsymbol{\sigma}_1$ and seeds, and the prover sends each later oracle in the preceding code switch. The stage's incoming claim is

$$\mu_{\mathfrak{i}} = \langle \boldsymbol{c}_{\mathfrak{i}}, \boldsymbol{f}_{\mathfrak{i}} \rangle + \langle \boldsymbol{d}_{\mathfrak{i}}, \boldsymbol{v}_{\mathfrak{i}} \rangle \in \mathcal{M} \,, \tag{11}$$

where the public coefficient vectors $\boldsymbol{c}_{\mathfrak{i}} \in \mathcal{M}^{2^{k_{\mathfrak{i}}} \ell_{\mathfrak{i}}}$ and $\boldsymbol{d}_{\mathfrak{i}} \in \mathcal{M}^{2^{k_{\mathfrak{i}}} a_{\mathfrak{i}}}$ are represented by succinct linear forms. Initially $\mu_1 = \mu$ and $\boldsymbol{c}_1 = \mathsf{sl}(\mathsf{st})$ for the input instance (μ, st) of $\mathcal{R}_{\mathcal{C}_1^{\equiv 2^{k_1}}, \mathsf{sl}}$, so the claim is the input linear constraint; there is no $\boldsymbol{d}_1$, since $a_1 = 0$.

The mask polynomial. Let $\hat{s}_{\mathfrak{i},u,e} = \sum_{j=0}^{k_{\mathfrak{i}}} \boldsymbol{\sigma}_{\mathfrak{i},j,u,e} \cdot \mathtt{X}_{j+1} \cdots \mathtt{X}_{k_{\mathfrak{i}}}$ be the polynomials of $\boldsymbol{\sigma}_{\mathfrak{i}}$ in Definition 6.1, whose values on $\{0,1\}^{k_{\mathfrak{i}}}$ are the masks $\boldsymbol{s}_{\mathfrak{i}}$. The mask polynomial of $\boldsymbol{\sigma}_{\mathfrak{i}}$ for masking degree three and, by Lemma 6.2, its value at a point $\boldsymbol{\gamma} \in \mathfrak{C}^{k_{\mathfrak{i}}}$ are

$$\mathcal{U}_{\mathfrak{i}}(\mathtt{X}_1, \ldots, \mathtt{X}_{k_{\mathfrak{i}}}) \coloneqq \sum_{u \in [d_{\mathcal{M}}]} \boldsymbol{m}_u \cdot (\hat{s}_{\mathfrak{i},u,0} + \hat{s}_{\mathfrak{i},u,1} \cdot (\mathtt{X}_1 + \cdots + \mathtt{X}_{k_{\mathfrak{i}}})) \,, \qquad \mathcal{U}_{\mathfrak{i}}(\boldsymbol{\gamma}) = \langle \mathsf{sl}_{\mathcal{U}}(\boldsymbol{\gamma}, \boldsymbol{m}), \mathsf{Fold}(\boldsymbol{s}_{\mathfrak{i}}, \boldsymbol{\gamma}) \rangle \,,$$

where $\mathsf{sl}_{\mathcal{U}}(\boldsymbol{\gamma}, \boldsymbol{m}) = (\boldsymbol{m}_u,\ \boldsymbol{m}_u \cdot (\gamma_1 + \cdots + \gamma_{k_{\mathfrak{i}}}))_{u \in [d_{\mathcal{M}}]} \in \mathcal{M}^p$.

Construction 11.1. The prover and verifier use the layout above and perform the following steps for $\mathfrak{i} = 1, \ldots, \mathfrak{n}$.

1. **Mask sum.** The prover sends $\tilde{\mu}_\mathfrak{i} = \sum_{\boldsymbol{b} \in \{0,1\}^{k_\mathfrak{i}}} \mathcal{U}_\mathfrak{i}(\boldsymbol{b}) \in \mathcal{M}$.

2. **Sumcheck batching randomness.** The verifier samples $\varepsilon_\mathfrak{i} \leftarrow \mathfrak{C}$.

3. **Sumcheck rounds.** For $\mathtt{X} = (\mathtt{X}_1, \ldots, \mathtt{X}_{k_\mathfrak{i}})$, the prover and verifier run sumcheck on

$$H_\mathfrak{i}(\mathtt{X}) = \mathcal{U}_\mathfrak{i}(\mathtt{X}) + (\langle \boldsymbol{c}_\mathfrak{i}(\mathtt{X}, \cdot), \boldsymbol{f}_\mathfrak{i}(\mathtt{X}, \cdot) \rangle + \langle \boldsymbol{d}_\mathfrak{i}(\mathtt{X}, \cdot), \boldsymbol{v}_\mathfrak{i}(\mathtt{X}, \cdot) \rangle) \cdot \varepsilon_\mathfrak{i}$$

   with target $\tilde{\mu}_\mathfrak{i} + \mu_\mathfrak{i} \cdot \varepsilon_\mathfrak{i}$, and the verifier samples every challenge uniformly from $\mathfrak{C}$. Each round polynomial has degree at most two. The verifier recovers its linear coefficient from the current target, which equals the sum of its values at 0 and 1, so the prover sends two coefficients per round. Let $\boldsymbol{\gamma}_\mathfrak{i}$ be the challenge vector, $\mu'_\mathfrak{i}$ the final target, and $\bar{f}_\mathfrak{i}, \bar{\boldsymbol{f}}_\mathfrak{i}, \bar{\boldsymbol{v}}_\mathfrak{i}, \bar{\boldsymbol{s}}_\mathfrak{i}, \bar{\boldsymbol{r}}_\mathfrak{i}$ the folds of $f_\mathfrak{i}, \boldsymbol{f}_\mathfrak{i}, \boldsymbol{v}_\mathfrak{i}, \boldsymbol{s}_\mathfrak{i}, \boldsymbol{r}_\mathfrak{i}$ through all $k_\mathfrak{i}$ challenges. The output claim is

$$\mu'_\mathfrak{i} = \langle \boldsymbol{c}'_\mathfrak{i}, \bar{\boldsymbol{f}}_\mathfrak{i} \rangle + \langle \boldsymbol{d}'_\mathfrak{i}, \bar{\boldsymbol{v}}_\mathfrak{i} \rangle + \langle \mathsf{sl}_\mathcal{U}(\boldsymbol{\gamma}_\mathfrak{i}, \boldsymbol{m}), \bar{\boldsymbol{s}}_\mathfrak{i} \rangle \ , \tag{12}$$

   where $\boldsymbol{c}'_\mathfrak{i} = \boldsymbol{c}_\mathfrak{i}(\boldsymbol{\gamma}_\mathfrak{i}, \cdot) \cdot \varepsilon_\mathfrak{i}$ and $\boldsymbol{d}'_\mathfrak{i} = \boldsymbol{d}_\mathfrak{i}(\boldsymbol{\gamma}_\mathfrak{i}, \cdot) \cdot \varepsilon_\mathfrak{i}$.

4. **Code switch, if $\mathfrak{i} < \mathfrak{n}$.** The prover samples a pad $\boldsymbol{r}_{\mathsf{ood},\mathfrak{i}} \leftarrow R^{r_{\mathsf{ood},\mathfrak{i}}}$ and forms the next messages

$$\boldsymbol{f}_{\mathfrak{i}+1} \coloneqq \bar{\boldsymbol{f}}_\mathfrak{i}, \qquad \boldsymbol{v}_{\mathfrak{i}+1} \coloneqq (\bar{\boldsymbol{v}}_\mathfrak{i}, \bar{\boldsymbol{s}}_\mathfrak{i}, \bar{\boldsymbol{r}}_\mathfrak{i}, \boldsymbol{r}_{\mathsf{ood},\mathfrak{i}}) \in R^{2^{k_{\mathfrak{i}+1}} a_{\mathfrak{i}+1}} \ , \tag{13}$$

   arranged into the main and auxiliary blocks of $2^{k_{\mathfrak{i}+1}}$ rows. It samples a uniform $\boldsymbol{\sigma}_{\mathfrak{i}+1} \in (R^p)^{k_{\mathfrak{i}+1}+1}$ and uniform seeds $\boldsymbol{r}_{\mathfrak{i}+1}$, lets $\boldsymbol{s}_{\mathfrak{i}+1}$ be the masks determined by $\boldsymbol{\sigma}_{\mathfrak{i}+1}$, and sends $f_{\mathfrak{i}+1}$. If $\mathfrak{i}+1 = \mathfrak{n}$, it also sends the mask g of the base case below.

   Out-of-domain samples and answers. The verifier samples $\rho_{\mathsf{ood},\mathfrak{i}} \leftarrow D_{\mathsf{ood},\mathfrak{i}}$. The prover sends

$$\boldsymbol{y}_\mathfrak{i} = \mathsf{ze}_{\mathsf{ood},\mathfrak{i}}(\rho_{\mathsf{ood},\mathfrak{i}}) \cdot (\boldsymbol{f}_{\mathfrak{i}+1}, \boldsymbol{v}_{\mathfrak{i}+1}) \in \mathcal{M}^{t_{\mathsf{ood},\mathfrak{i}}} \ . \tag{14}$$

   In-domain samples and batching. The verifier samples $t_\mathfrak{i}$ independent uniform points in $[m_\mathfrak{i}]$ and $\rho_\mathfrak{i} \leftarrow D_{\mathsf{ze},\mathfrak{i}}$. A query at x yields the column $f_\mathfrak{i}(x)$, which the verifier folds to obtain $\bar{f}_\mathfrak{i}(x)$. For each sampled point, the consistency equation is

$$\bar{f}_\mathfrak{i}(x) = \mathsf{Enc}_{\mathcal{C}_\mathfrak{i}}\left((\boldsymbol{f}_{\mathfrak{i}+1}, \bar{\boldsymbol{v}}_\mathfrak{i}), (\bar{\boldsymbol{s}}_\mathfrak{i}, \bar{\boldsymbol{r}}_\mathfrak{i})\right)(x) \, ,$$

   a linear constraint on the message $(\boldsymbol{f}_{\mathfrak{i}+1}, \boldsymbol{v}_{\mathfrak{i}+1})$ of $f_{\mathfrak{i}+1}$. The verifier batches the output claim of the sumcheck, the $t_{\mathsf{ood},\mathfrak{i}}$ equations of the OOD answers, and the $t_\mathfrak{i}$ consistency equations with the coefficients $\mathsf{ze}_\mathfrak{i}(\rho_\mathfrak{i})$. The batched target $\mu_{\mathfrak{i}+1}$ is the same combination of $\mu'_\mathfrak{i}$, the entries of $\boldsymbol{y}_\mathfrak{i}$, and the values $\bar{f}_\mathfrak{i}(x)$. Grouping the coefficients of $\boldsymbol{f}_{\mathfrak{i}+1}$ and of $\boldsymbol{v}_{\mathfrak{i}+1}$ gives $\boldsymbol{c}_{\mathfrak{i}+1}$ and $\boldsymbol{d}_{\mathfrak{i}+1}$ in Equation 11.

5. **Base case, if $\mathfrak{i} = \mathfrak{n}$.** The prover has sent a mask

$$g = \mathsf{Enc}_{\mathcal{C}_\mathfrak{n}}\left((\boldsymbol{f}_\star, \boldsymbol{v}_\star), (\boldsymbol{s}_\star, \boldsymbol{r}_\star)\right)$$

with all four vectors uniform, together with $f_{\mathfrak{n}}$. For $\mathfrak{n} = 1$, the prover sends g together with f_1 if it sends the input oracle itself, as in our evaluation (Section 1.3); if f_1 is given, it sends g at this point, as a separate oracle. The prover sends

$$\mu_\star = \langle \boldsymbol{c}'_{\mathfrak{n}}, \boldsymbol{f}_\star \rangle + \langle \boldsymbol{d}'_{\mathfrak{n}}, \boldsymbol{v}_\star \rangle + \langle \mathsf{sl}_{\mathcal{U}}(\boldsymbol{\gamma}_{\mathfrak{n}}, \boldsymbol{m}), \boldsymbol{s}_\star \rangle \ ,$$

the verifier samples $\gamma_\star \leftarrow \mathfrak{C}$, and the prover discloses

$$(\boldsymbol{f}^*, \boldsymbol{v}^*, \boldsymbol{s}^*, \boldsymbol{r}^*) = (\boldsymbol{f}_\star, \boldsymbol{v}_\star, \boldsymbol{s}_\star, \boldsymbol{r}_\star) + (\bar{\boldsymbol{f}}_{\mathfrak{n}}, \bar{\boldsymbol{v}}_{\mathfrak{n}}, \bar{\boldsymbol{s}}_{\mathfrak{n}}, \bar{\boldsymbol{r}}_{\mathfrak{n}}) \cdot \gamma_\star \, .$$

The verifier checks that

$$\langle \boldsymbol{c}'_{\mathfrak{n}}, \boldsymbol{f}^* \rangle + \langle \boldsymbol{d}'_{\mathfrak{n}}, \boldsymbol{v}^* \rangle + \langle \mathsf{sl}_{\mathcal{U}}(\boldsymbol{\gamma}_{\mathfrak{n}}, \boldsymbol{m}), \boldsymbol{s}^* \rangle = \mu_\star + \mu'_{\mathfrak{n}} \cdot \gamma_\star$$

and that

$$\mathsf{Enc}_{\mathcal{C}_{\mathfrak{n}}}\left((\boldsymbol{f}^*, \boldsymbol{v}^*), (\boldsymbol{s}^*, \boldsymbol{r}^*)\right)(x) = g(x) + \bar{f}_{\mathfrak{n}}(x) \cdot \gamma_\star$$

at $t_{\mathfrak{n}}$ uniform points x. In the compiled argument, the verifier computes each queried symbol $g(x)$ from this equation and authenticates it, so the prover does not transmit it.

Security. For $\mathfrak{i} < \mathfrak{n}$, the steps of stage $\mathfrak{i}$ are the IOR of Theorem 9.12, and the steps of stage $\mathfrak{n}$ are the IOPP of Theorem 8.1. By Theorem 4.7, the construction has the round-by-round errors of these theorems and is HVZK on fresh inputs with error

$$\sum_{\mathfrak{i} \in [\mathfrak{n}-1]} \left(2^{k_{\mathfrak{i}}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}}} + 2^{k_{\mathfrak{i}+1}} \cdot \zeta_{\mathcal{C}_{\mathfrak{i}+1}} + \zeta_{\mathsf{ze},\mathfrak{i}} \right) + 2^{k_{\mathfrak{n}}} \cdot \zeta_{\mathcal{C}_{\mathfrak{n}}} \, .$$

Sending g together with $f_{\mathfrak{n}}$ rather than in the base case leaves both analyses unchanged, because g is sampled independently of all earlier messages and is fixed before $\gamma_\star$ is sampled. Omitting the linear coefficient of a round polynomial also leaves them unchanged, because the verifier's check determines this coefficient from the other two.

Communication. The oracles, including the input oracle, have $\sum_{\mathfrak{i} \in [\mathfrak{n}]} 2^{k_{\mathfrak{i}}} m_{\mathfrak{i}} + m_{\mathfrak{n}}$ symbols of $\mathcal{M}$ in total. The prover sends directly

$$2 \sum_{\mathfrak{i} \in [\mathfrak{n}]} k_{\mathfrak{i}} + \mathfrak{n} + \sum_{\mathfrak{i} \in [\mathfrak{n}-1]} t_{\mathsf{ood},\mathfrak{i}} + 1 \quad \text{elements of } \mathcal{M} \text{ and} \quad \ell_{\mathfrak{n}} + a_{\mathfrak{n}} + p + r_{\mathfrak{n}} \quad \text{elements of } R \, .$$

There are $\sum_{\mathfrak{i} \in [\mathfrak{n}]} t_{\mathfrak{i}}$ column queries and $\sum_{\mathfrak{i} \in [\mathfrak{n}]} k_{\mathfrak{i}} + 3\mathfrak{n}$ rounds. Each column has $2^{k_{\mathfrak{i}}}$ entries, and the final oracle has one additional masking row. Compared with the same schedule without zero-knowledge, a query reads no additional symbol except in this masking row, and the terminal disclosure grows by $a_{\mathfrak{n}} + p + r_{\mathfrak{n}}$ elements of R.

Reed–Solomon codes. For Reed–Solomon codes we take a finite field $\mathbb{F}$ with $R = \mathcal{M} = \mathfrak{C} = \mathbb{F}$ and $\boldsymbol{m} = (1)$, so that $p = 2$, and seeds of length $r_{\mathfrak{i}} = t_{\mathfrak{i}}$. The code $\mathcal{C}_{\mathfrak{i}}$ is a Reed–Solomon code of length $m_{\mathfrak{i}} \geq \ell_{\mathfrak{i}} + a_{\mathfrak{i}} + t_{\mathfrak{i}} + 2$. For a row with message $\boldsymbol{w}$, mask (s_0, s_1), and seed $\boldsymbol{r}$, the codeword consists of the values at $m_{\mathfrak{i}}$ distinct points of the polynomial

$$\mathsf{ev}(\boldsymbol{r}, \mathtt{X}) + s_0 \cdot \mathtt{X}^{t_{\mathfrak{i}}} + s_1 \cdot \mathtt{X}^{t_{\mathfrak{i}}+1} + \mathtt{X}^{t_{\mathfrak{i}}+2} \cdot \mathsf{ev}(\boldsymbol{w}, \mathtt{X}), \qquad \text{where} \qquad \mathsf{ev}(\boldsymbol{w}, \mathtt{X}) \coloneqq \sum_{j=0}^{|\boldsymbol{w}|-1} \boldsymbol{w}_j \cdot \mathtt{X}^j \, .$$

By Lemma 3.29, this is a t_{i}-query perfect zero-knowledge encoding with two spare coordinates. The auxiliary blocks have lengths

$$a_{\mathsf{i}+1} \coloneqq \left\lceil \frac{a_{\mathsf{i}} + t_{\mathsf{i}} + 3}{2^{k_{\mathsf{i}+1}}} \right\rceil , \qquad \text{so that} \qquad r_{\mathsf{ood},\mathsf{i}} = 2^{k_{\mathsf{i}+1}} a_{\mathsf{i}+1} - (a_{\mathsf{i}} + t_{\mathsf{i}} + 2) \geq 1 .$$

Every stage but the last has $t_{\mathsf{ood},\mathsf{i}} = 1$ out-of-domain sample, whose seed $\rho_{\mathsf{ood},\mathsf{i}}$ consists of a point $z_{\mathsf{i}} \leftarrow \mathbb{F}$ and a uniform nonzero $\boldsymbol{\beta}_{\mathsf{i}} \in \mathbb{F}^{r_{\mathsf{ood},\mathsf{i}}}$, and its answer is

$$y_{\mathsf{i}} = \mathsf{ev}(\bar{\boldsymbol{f}}_{\mathsf{i}}, z_{\mathsf{i}}) + z_{\mathsf{i}}^{\ell_{\mathsf{i}}} \cdot \mathsf{ev}\left((\bar{\boldsymbol{v}}_{\mathsf{i}}, \bar{\boldsymbol{s}}_{\mathsf{i}}, \bar{\boldsymbol{r}}_{\mathsf{i}}), z_{\mathsf{i}}\right) + \langle \boldsymbol{\beta}_{\mathsf{i}}, \boldsymbol{r}_{\mathsf{ood},\mathsf{i}} \rangle .$$

By Lemma 9.4, this zero-evader is $(r_{\mathsf{ood},\mathsf{i}}, 0)$-private. The batching coefficients are the powers $1, \rho_{\mathsf{i}}, \ldots, \rho_{\mathsf{i}}^{t_{\mathsf{i}}+1}$ of a uniform $\rho_{\mathsf{i}} \in \mathbb{F}$. The construction is then perfect HVZK on fresh inputs. Its oracles have $\sum_{\mathsf{i}\in[\mathfrak{n}]} 2^{k_{\mathsf{i}}} m_{\mathsf{i}} + m_{\mathfrak{n}}$ field elements in total, and the prover sends $2\sum_{\mathsf{i}\in[\mathfrak{n}]} k_{\mathsf{i}} + 2\mathfrak{n} + \ell_{\mathfrak{n}} + a_{\mathfrak{n}} + t_{\mathfrak{n}} + 2$ field elements directly.

Merkle commitments. Our implementation encodes every row with $t_{\mathsf{i}} + 1$ seed coordinates instead of t_{i}. With these Reed–Solomon codes, the prover can then commit to the oracles with Merkle trees whose leaves are not salted (Appendix B). The extra coordinate adds one field element to every folded seed, which the next stage encodes, and removes one salt per query from the compiled argument.

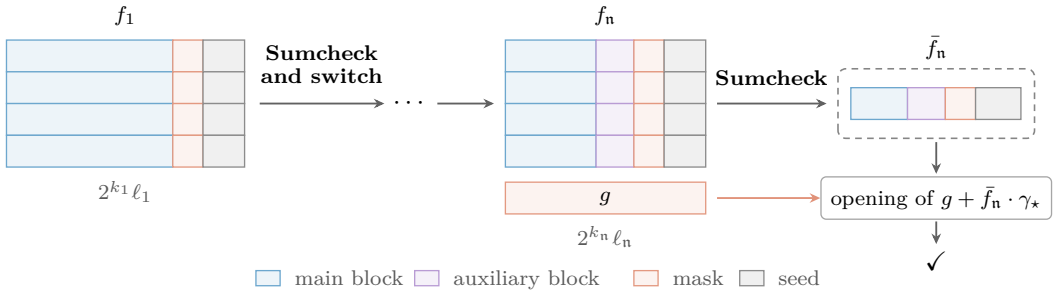

Figure 7: The protocol of Section 11. Stages $1, \ldots, \mathfrak{n}-1$ fold their oracle and re-encode the result in the rows of the next one. The last stage folds $f_{\mathfrak{n}}$ and applies the base case to the virtual fold $\bar{f}_{\mathfrak{n}}$, using the mask g that the prover sent together with $f_{\mathfrak{n}}$. The main-message lengths of f_1 and $f_{\mathfrak{n}}$ appear below them. Row counts and widths are schematic.

12 Zero-knowledge reduction from R1CS

We describe a reduction from $\mathcal{R}_{\mathsf{R1CS}}$ to testing proximity to the constrained code relation of Definition 5.7. The reduction is similar to the Σ-IOP from [ACFY25], with a sumcheck whose messages are hidden by the mask polynomial of Definition 6.1. The R1CS constraints are equations over the ring R, so the sumcheck at the core of the reduction produces *ring-valued* claims. At the end, we combine these claims into a single module-valued claim using a zero-evader over R (Definition 3.5).

Definition 12.1. *The relation $\mathcal{R}_{\mathsf{R1CS}}$ is the set of all pairs $((R, n_0, \ell, A, B, C, v), w)$ (triples of Section 3.1 with empty implicit instance) where R is a finite ring, $n_0, \ell \in \mathbb{N}$, A, B, C are $(\ell + n_0) \times (\ell + n_0)$ matrices over R, $v \in R^{n_0}$, and $w \in R^{\ell}$, such that for all $i \in [(\ell + n_0)]$:*

$$\left(\sum_{j=1}^{(\ell+n_0)} A_{i,j} \cdot z_j\right) \cdot \left(\sum_{j=1}^{(\ell+n_0)} B_{i,j} \cdot z_j\right) = \sum_{j=1}^{(\ell+n_0)} C_{i,j} \cdot z_j$$

for $z \coloneqq (v, w) \in R^{(\ell+n_0)}$.

Notation. Throughout this section we assume that $\ell = n_0$ (so that $(\ell + n_0) = 2\ell$) and that ℓ is a power of 2, and we fix bijections identifying $\{0,1\}^{\log \ell}$ with $[\ell]$ and $\{0,1\}^{\log \ell + 1}$ with $[(\ell + n_0)]$ (so that the last coordinate of $\{0,1\}^{\log \ell + 1}$ selects between the input part, at value 0, and the witness part, at value 1, of $z = (v, w)$), using them as appropriate to translate between the two domains. We use the following multilinear extensions:

- $\hat{f}_v \in R^{<2}[\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell}]$ is the unique multilinear polynomial with $\hat{f}_v(\boldsymbol{b}) = v(\boldsymbol{b})$ for every $\boldsymbol{b} \in \{0,1\}^{\log \ell}$, and $\hat{f}_w$ is defined analogously for the witness w.
- for every $M \in \{A, B, C\}$, $\hat{f}_M \in R^{<2}[\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell + 1}, \mathtt{Y}_1, \ldots, \mathtt{Y}_{\log \ell + 1}]$ is the unique multilinear polynomial with $\hat{f}_M(\boldsymbol{a}, \boldsymbol{b}) = M[\boldsymbol{a}, \boldsymbol{b}]$ for all $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^{\log \ell + 1}$.

We use the mask polynomial of Definition 6.1 with $k = \log \ell + 1$ variables, masking degree d_{R1CS}, $\mathcal{M} = R$, and $\boldsymbol{m} = (1)$. For a tuple $\boldsymbol{\xi} \in (R^{d_{\mathsf{R1CS}}-1})^{\log \ell + 2}$, we write $\mathcal{U}^{\boldsymbol{\xi}}$ for its mask polynomial. Finally, $\mathrm{nnz}(A, B, C)$ denotes the total number of nonzero entries of A, B, C.

Theorem 12.2. *Consider the following ingredients:*

- *a masking degree $d_{\mathsf{R1CS}} \geq 6$, the outer mask length $\ell^{\mathsf{zk}}_{\mathrm{out}} \coloneqq (\log \ell + 2)(d_{\mathsf{R1CS}} - 1)$, and an integer $a \geq 3 + \ell^{\mathsf{zk}}_{\mathrm{out}}$.*
- *$\mathcal{C} \subseteq \Sigma^m$ with a zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell+a} \times R^r \to \Sigma^m$, whose messages consist of a witness of length ℓ followed by an auxiliary part of length a.*
- *$\mathfrak{C} \subseteq \mathcal{Z}(R)$ a strong sampling set with $\{0,1\} \subseteq \mathfrak{C}$ and $|\mathfrak{C}| \geq 3$.*
- *$\mathsf{ze} \colon D_{\mathsf{ze}} \to \mathcal{M}^{1\times 4}$ a zero-evader over R with error $\varepsilon_{\mathsf{zero}}$, whose coordinates we index by A, B, C, s.*

Define the linear form on the messages of $\mathcal{C}$

$$\mathsf{sl}\,(\mathsf{st}, \mathsf{st}^{\mathsf{in}}, \mathsf{st}^{\mathsf{out}}) \coloneqq (\mathsf{sl}(\mathsf{st}), \mathsf{sl}^{\mathsf{in}}(\mathsf{st}^{\mathsf{in}}), \mathsf{sl}^{\mathsf{out}}(\mathsf{st}^{\mathsf{out}}), 0)\ ,$$

which applies sl to the witness and the remaining forms to the auxiliary part, padded with zeros,

where[6]

$$\mathsf{sl}\colon ((A,B,C),\boldsymbol{\alpha},\rho) \mapsto \left(\sum_{M\in\{A,B,C\}} \mathsf{ze}(\rho)_M \cdot \hat{f}_M(\boldsymbol{\alpha},\boldsymbol{b},1)\right)_{\boldsymbol{b}\in\{0,1\}^{\log \ell}} \in \mathcal{M}^{1\times \ell}\,,$$

$$\mathsf{sl}^{\mathsf{in}}\colon (\alpha,\rho) \mapsto \Big(\mathsf{ze}(\rho)_M \cdot (\alpha^2-\alpha)\Big)_{M\in\{A,B,C\}} \in \mathcal{M}^{1\times 3}\,,$$

$$\mathsf{sl}^{\mathsf{out}}\colon (\boldsymbol{\alpha},\rho) \mapsto \mathsf{ze}(\rho)_s \cdot (\mathsf{sl}_{\mathcal{U}}(\boldsymbol{\alpha},(1)) \cdot \alpha_{i+1}\cdots\alpha_{\log \ell+1})_{i\in\{0,\ldots,\log \ell+1\}} \in \mathcal{M}^{1\times \ell^{\mathsf{zk}}_{\mathrm{out}}}\,.$$

Construction 12.3 is an IOR from $\mathcal{R}_{\mathsf{R1CS}}$ to $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$ with the following properties.

- *Round complexity:* $O(\log \ell)$.
- *Prover communication: m symbols of Σ sent as one oracle message, and* $(\log \ell + 1)\cdot d_{\mathsf{R1CS}} + 4$ *elements of R sent directly.*
- *Verifier oracle queries: none. The output oracle is the oracle sent by the prover.*
- *Prover time (in ring and module operations):* $O(\mathsf{tenc}(\mathcal{C}) + d_{\mathsf{R1CS}}\cdot \ell + \mathrm{nnz}(A,B,C))$.
- *Verifier time (in ring and module operations):* $O(\log \ell \cdot d_{\mathsf{R1CS}} + \ell + \mathrm{nnz}(A,B,C))$.
- **Round-by-round security:** *For every $\delta \in (0,1)$, Construction 12.3 has RBR knowledge soundness with relaxation $(\mathcal{R}_{\mathsf{R1CS}}, \widetilde{\mathcal{R}}^{\delta}_{\mathcal{C},\mathsf{sl}})$ and errors*

$$\begin{pmatrix} \Lambda(\mathcal{C},\delta)\cdot \epsilon^R_{\mathsf{pil}}(\mathfrak{C},\log \ell+2,2),\ \Big(\Lambda(\mathcal{C},\delta)\cdot \epsilon^R_{\mathsf{pil}}(\mathfrak{C},1,d_{\mathsf{R1CS}})\Big)_{j\in[\log \ell]}\,, \\ \Lambda(\mathcal{C},\delta)\cdot \epsilon^R_{\mathsf{pil}}(\mathfrak{C}\setminus\{0,1\},1,d_{\mathsf{R1CS}}),\ \Lambda(\mathcal{C},\delta)\cdot \varepsilon_{\mathsf{zero}} \end{pmatrix}.$$

  The total extraction time is $O(\ell + \ell^{\mathsf{zk}}_{\mathrm{out}})$.
- **Zero-knowledge:** *If R and $\mathcal{M}$ are finite and $\mathsf{Enc}_{\mathcal{C}}$ is a t-query zero-knowledge encoding with error ζ, then the reduction is HVZK for $\mathfrak{D}^{\le t}$ with error ζ (and no query complexity as $\mathcal{R}_{\mathsf{R1CS}}$ has no implicit instance), and it has fresh output (Definition 4.5).*

Construction 12.3. Inputs. The honest prover receives $((R,n_0,\ell,A,B,C,v),w) \in \mathcal{R}_{\mathsf{R1CS}}$ and the verifier receives (R,n_0,ℓ,A,B,C,v).

Interaction phase.

1. **Witness-hiding masks.** For each $M \in \{A,B,C\}$, the honest prover samples an inner mask $\bar{\xi}_M \leftarrow R$. We write $\bar{\boldsymbol{\xi}} \coloneqq (\bar{\xi}_A,\bar{\xi}_B,\bar{\xi}_C)$ and
$$\hat{p}_M(\mathtt{X}) \coloneqq \bar{\xi}_M \cdot (\mathtt{X}^2 - \mathtt{X}) \in R^{<3}[\mathtt{X}]\,.$$
By construction, $\hat{p}_M(0) = \hat{p}_M(1) = 0$.

2. **Sumcheck mask.** The honest prover samples an outer mask $\boldsymbol{\xi} \leftarrow R^{\ell^{\mathsf{zk}}_{\mathrm{out}}}$. Let $\mathcal{U}^{\boldsymbol{\xi}}$ be the mask polynomial of $\boldsymbol{\xi}$.

3. **Witness encoding.** The prover sends $f \in \Sigma^m$ and a value $\tilde{\mu} \in R$. In the honest case, the prover sets $\boldsymbol{f} \coloneqq w$, samples $\boldsymbol{r} \leftarrow R^r$, and sends the encoding of the witness together with all masks,
$$f \coloneqq \mathsf{Enc}_{\mathcal{C}}\left(\left(\boldsymbol{f},\bar{\boldsymbol{\xi}},\boldsymbol{\xi},0\right),\boldsymbol{r}\right)\,, \qquad \tilde{\mu} \coloneqq \sum_{\boldsymbol{b}\in\{0,1\}^{\log \ell+1}} \mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{b})\,,$$
where the auxiliary part is padded with zeros to length a.

[6] We slightly abuse notation and treat the entire description of the matrices (A,B,C) as part of the state of sl, from which the multilinear extensions $\hat{f}_M$ are computed.

4. **Zerocheck randomness.** The verifier samples and sends $\varepsilon \leftarrow \mathfrak{C}$ and $\boldsymbol{\beta} \leftarrow \mathfrak{C}^{\log \ell+1}$.

   In the honest case, the prover defines the following polynomials:

   - *Boolean extension of the input/witness:*

   $$\hat{f}_z(\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell+1}) \coloneqq \hat{f}_v(\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell}) \cdot (1 - \mathtt{X}_{\log \ell+1}) + \hat{f}_w(\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell}) \cdot \mathtt{X}_{\log \ell+1}\,.$$

   - *Masked matrix-vector polynomials:* for every $M \in \{A, B, C\}$,

   $$\hat{F}_M(\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell+1}) \coloneqq \sum_{\boldsymbol{b} \in \{0,1\}^{\log \ell+1}} \hat{f}_M(\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell+1}, \boldsymbol{b}) \cdot \hat{f}_z(\boldsymbol{b}) + \hat{p}_M(\mathtt{X}_{\log \ell+1})\,.$$

   - *Constraint polynomial:* $\hat{g} \coloneqq \hat{F}_A \cdot \hat{F}_B - \hat{F}_C$. Note that $\hat{g}$ has individual degree at most 4 and that, since each $\hat{p}_M$ vanishes on $\{0,1\}$, if the R1CS instance is satisfied then $\hat{g}$ vanishes on $\{0,1\}^{\log \ell+1}$.

5. **Sumcheck.** Let $\boldsymbol{\alpha} \coloneqq \emptyset$. For each $j \in [\log \ell + 1]$:

   - The prover sends $\hat{h}_j \in R^{<d_{\mathsf{R1CS}}}[\mathtt{X}]$. In the honest case,

   $$\hat{h}_j(\mathtt{X}) \coloneqq \sum_{\boldsymbol{a} \in \{0,1\}^{\log \ell+1-j}} \left(\mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{\alpha}, \mathtt{X}, \boldsymbol{a}) + \varepsilon \cdot \hat{g}(\boldsymbol{\alpha}, \mathtt{X}, \boldsymbol{a}) \cdot \mathsf{eq}(\boldsymbol{\beta}, (\boldsymbol{\alpha}, \mathtt{X}, \boldsymbol{a}))\right)\,.$$

     (Note that the second summand has degree at most $5 < d_{\mathsf{R1CS}}$ in $\mathtt{X}$.)
   - The verifier checks that $\hat{h}_j(0) + \hat{h}_j(1) = \hat{h}_{j-1}(\alpha_{j-1})$ (for $j = 1$, the verifier compares to $\tilde{\mu}$).
   - The verifier samples and sends $\alpha_j \leftarrow \mathfrak{C}$ (except in the last round, where $\alpha_{\log \ell+1} \leftarrow \mathfrak{C} \setminus \{0,1\}$) and updates $\boldsymbol{\alpha} \coloneqq \boldsymbol{\alpha} \| \alpha_j$.

6. **Final sumcheck values.** The prover sends $v_M \in R$ for every $M \in \{A, B, C\}$. In the honest case,

   $$v_M \coloneqq \hat{F}_M(\boldsymbol{\alpha}) = \sum_{\boldsymbol{b} \in \{0,1\}^{\log \ell+1}} \hat{f}_M(\boldsymbol{\alpha}, \boldsymbol{b}) \cdot \hat{f}_z(\boldsymbol{b}) + \hat{p}_M(\alpha_{\log \ell+1})\,.$$

   The verifier computes the public values

   $$u_M \coloneqq \sum_{\boldsymbol{b} \in \{0,1\}^{\log \ell}} \hat{f}_M(\boldsymbol{\alpha}, \boldsymbol{b}, 0) \cdot \hat{f}_v(\boldsymbol{b})\,,$$

   so that, in the honest case,

   $$v_M - u_M = \sum_{\boldsymbol{b} \in \{0,1\}^{\log \ell}} \hat{f}_M(\boldsymbol{\alpha}, \boldsymbol{b}, 1) \cdot \hat{f}_w(\boldsymbol{b}) + \hat{p}_M(\alpha_{\log \ell+1})\,.$$

7. **Joint constraint.** The verifier samples and sends $\rho \leftarrow D_{\mathsf{ze}}$.

   Both parties set the states $\mathsf{st} \coloneqq ((A, B, C), \boldsymbol{\alpha}, \rho)$, $\mathsf{st}^{\mathsf{in}} \coloneqq (\alpha_{\log \ell+1}, \rho)$, and $\mathsf{st}^{\mathsf{out}} \coloneqq (\boldsymbol{\alpha}, \rho)$, and the target

   $$\mu \coloneqq \sum_{M \in \{A,B,C\}} \mathsf{ze}(\rho)_M \cdot (v_M - u_M) + \mathsf{ze}(\rho)_s \cdot \left(\hat{h}_{\log \ell+1}(\alpha_{\log \ell+1}) - \varepsilon \cdot (v_A \cdot v_B - v_C) \cdot \mathsf{eq}(\boldsymbol{\beta}, \boldsymbol{\alpha})\right)\,.$$

8. **Output claim:** The prover and the verifier both implicitly define the output instance for $\mathcal{R}_{\mathcal{C},\mathsf{sl}}$:

$$\mathbb{x} = (\mu, (\mathsf{st}, \mathsf{st}^{\mathsf{in}}, \mathsf{st}^{\mathsf{out}}))\ ,$$
$$\mathbb{y} = f\ .$$

In the honest case, the prover also defines the witness:

$$\mathbb{w} = \left(\left(\boldsymbol{f}, \bar{\boldsymbol{\xi}}, \boldsymbol{\xi}, 0\right), \boldsymbol{r}\right)\ .$$

Proof sketch of Theorem 12.2. We sketch completeness, round-by-round knowledge soundness, and honest-verifier zero-knowledge.

Completeness. In the honest case, for $\mathtt{X} = (\mathtt{X}_1, \ldots, \mathtt{X}_{\log \ell+1})$, the prover sumchecks the polynomial

$$H(\mathtt{X}) \coloneqq \mathcal{U}^{\boldsymbol{\xi}}(\mathtt{X}) + \varepsilon \cdot \hat{g}(\mathtt{X}) \cdot \mathsf{eq}(\boldsymbol{\beta}, \mathtt{X})\,,$$

whose individual degrees are at most $5 < d_{\mathsf{R1CS}}$. Since the R1CS instance is satisfied and each $\hat{p}_M$ vanishes on $\{0,1\}$, the polynomial $\hat{g}$ vanishes on $\{0,1\}^{\log \ell+1}$, so $\sum_{\boldsymbol{b}\in\{0,1\}^{\log \ell+1}} H(\boldsymbol{b}) = \tilde{\mu}$ and all sumcheck checks pass. For the output claim: since the challenges are central, $\hat{h}_{\log \ell+1}(\alpha_{\log \ell+1}) = H(\boldsymbol{\alpha})$ and $\hat{g}(\boldsymbol{\alpha}) = \hat{F}_A(\boldsymbol{\alpha}) \cdot \hat{F}_B(\boldsymbol{\alpha}) - \hat{F}_C(\boldsymbol{\alpha}) = v_A \cdot v_B - v_C$, and hence, by Item 1 of Lemma 6.2,

$$\langle \mathsf{sl}^{\mathsf{out}}(\mathsf{st}^{\mathsf{out}}), \boldsymbol{\xi}\rangle = \mathsf{ze}(\rho)_s \cdot \mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{\alpha}) = \mathsf{ze}(\rho)_s \cdot \left(\hat{h}_{\log \ell+1}(\alpha_{\log \ell+1}) - \varepsilon \cdot (v_A \cdot v_B - v_C) \cdot \mathsf{eq}(\boldsymbol{\beta}, \boldsymbol{\alpha})\right)\ .$$

Also, $\langle \mathsf{sl}^{\mathsf{in}}(\mathsf{st}^{\mathsf{in}}), \bar{\boldsymbol{\xi}}\rangle = \sum_M \mathsf{ze}(\rho)_M \cdot \hat{p}_M(\alpha_{\log \ell+1})$ and $\langle \mathsf{sl}(\mathsf{st}), \boldsymbol{f}\rangle = \sum_M \mathsf{ze}(\rho)_M \cdot \sum_{\boldsymbol{b}\in\{0,1\}^{\log \ell}} \hat{f}_M(\boldsymbol{\alpha}, \boldsymbol{b}, 1) \cdot \hat{f}_w(\boldsymbol{b})$ (both use the module axiom $x\cdot(r\cdot s) = (x\cdot r)\cdot s$ and the centrality of the challenges). Summing all contributions and using the decomposition of $v_M - u_M$ in Item 6 yields μ, so $(\mathbb{x}, \mathbb{y}, \mathbb{w}) \in \mathcal{R}_{\mathcal{C},\mathsf{sl}}$.

Round-by-round knowledge soundness. Ignoring masks, we follow the Σ-IOP analysis for $\mathcal{R}_{\mathsf{R1CS}}$ in [ACFY25]. The masks are part of the message of f, so a codeword close to f determines them, as in Lemma 6.6, and the zero-evader combines the final ring-valued claims into a module-valued claim. The state function is analogous to the one in Lemma 6.6: at each partial transcript, a valid round witness consists of a codeword of $\mathcal{C}$ within distance δ of $\mathbb{y}$ whose message satisfies the current sumcheck claim. The oracle f is never folded (unlike in Lemma 6.6), so no mutual-correlated-agreement term appears in the errors. By taking a union bound over the at most $\Lambda(\mathcal{C}, \delta)$ such witnesses, we obtain the leading factor in each error coordinate. The error coordinates are bounded as follows.

- *Zerocheck randomness.* Fix extracted messages and let $\hat{g}$ be the associated constraint polynomial. The extracted inner-mask polynomials $\hat{p}_M$ vanish on $\{0,1\}$ by construction, so $\hat{g}$ vanishes on $\{0,1\}^{\log \ell+1}$ if and only if the extracted witness $\boldsymbol{f}$ satisfies the R1CS instance. Consider

$$P(\mathtt{E}, \mathtt{R}_1, \ldots, \mathtt{R}_{\log \ell+1}) \coloneqq \sum_{\boldsymbol{b}\in\{0,1\}^{\log \ell+1}} \hat{g}(\boldsymbol{b}) \cdot \mathtt{E} \cdot \mathsf{eq}(\mathtt{R}_1, \ldots, \mathtt{R}_{\log \ell+1}, \boldsymbol{b}) + \sum_{\boldsymbol{b}\in\{0,1\}^{\log \ell+1}} \mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{b}) - \tilde{\mu}\,,$$

  a polynomial in $\log \ell + 2$ variables of individual degree at most 1. If the extracted witness does not satisfy the R1CS instance then the coefficient polynomial of $\mathtt{E}$ is nonzero (it is the multilinear extension of $(\hat{g}(\boldsymbol{b}))_{\boldsymbol{b}\in\{0,1\}^{\log \ell+1}}$). If it does, but $\tilde{\mu} \neq \sum_{\boldsymbol{b}} \mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{b})$, then the constant term is nonzero. In either case $P \neq 0$, so the sumcheck claim becomes true with probability at most $\epsilon^R_{\mathsf{pil}}(\mathfrak{C}, \log \ell + 2, 2)$ over $(\varepsilon, \boldsymbol{\beta}) \leftarrow \mathfrak{C} \times \mathfrak{C}^{\log \ell+1}$.

- *Sumcheck rounds.* As in Lemma 6.6: if the claim at round j is false, then $\hat{h}_j$ differs, as a polynomial in $R^{<d_{\mathsf{R1CS}}}[\mathtt{X}]$, from the partial-sum polynomial determined by the extracted messages, so the two agree at α_j with probability at most $\epsilon^R_{\mathsf{pil}}(\mathfrak{C}, 1, d_{\mathsf{R1CS}})$ (in the last round, $\epsilon^R_{\mathsf{pil}}(\mathfrak{C}\setminus\{0,1\}, 1, d_{\mathsf{R1CS}})$). A subset of a strong sampling set is again a strong sampling set.
- *Zero-evader randomness.* Define the discrepancy vector $\boldsymbol{\delta} \in R^4$ with coordinates

$$\delta_M \coloneqq \sum_{\boldsymbol{b}\in\{0,1\}^{\log \ell}} \hat{f}_M(\boldsymbol{\alpha}, \boldsymbol{b}, 1)\cdot \boldsymbol{f}(\boldsymbol{b}) + \hat{p}_M(\alpha_{\log \ell+1}) - (v_M - u_M) \qquad (M \in \{A, B, C\})\,,$$

$$\delta_s \coloneqq \mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{\alpha}) - \left(\hat{h}_{\log \ell+1}(\alpha_{\log \ell+1}) - \varepsilon\cdot(v_A\cdot v_B - v_C)\cdot \mathsf{eq}(\boldsymbol{\beta}, \boldsymbol{\alpha})\right)\,.$$

  If $\boldsymbol{\delta} = 0$ then the final sumcheck claim holds for the extracted messages: $\delta_A = \delta_B = \delta_C = 0$ give $\hat{F}_M(\boldsymbol{\alpha}) = v_M$, and then $\delta_s = 0$ gives $\hat{h}_{\log \ell+1}(\alpha_{\log \ell+1}) = \mathcal{U}^{\boldsymbol{\xi}}(\boldsymbol{\alpha}) + \varepsilon\cdot\hat{g}(\boldsymbol{\alpha})\cdot\mathsf{eq}(\boldsymbol{\beta}, \boldsymbol{\alpha})$. Hence, if the final sumcheck claim is false then $\boldsymbol{\delta} \neq 0$, and the output claim, which reads $\mathsf{ze}(\rho)\cdot\boldsymbol{\delta} = 0$, holds with probability at most $\varepsilon_{\mathsf{zero}}$ over $\rho \leftarrow D_{\mathsf{ze}}$.

The extractor, given a witness for the relaxed output relation, outputs the R1CS witness $w \coloneqq \boldsymbol{f}$. The total extraction time is $O(\ell + \ell^{\mathsf{zk}}_{\mathrm{out}})$.

Honest-verifier zero-knowledge. Fix any $\mathcal{D} \in \mathfrak{D}^{\leq t}$ and define a simulator for its view. Let $\mathbf{S}_{\mathcal{C}}$ denote the simulator for the t-query zero-knowledge encoding of $\mathcal{C}$.

$\mathbf{S}^{\mathcal{D}}(R, n_0, \ell, A, B, C, v)$:

1. Sample verifier randomness as in the protocol: $\varepsilon \leftarrow \mathfrak{C}$, $\boldsymbol{\beta} \leftarrow \mathfrak{C}^{\log \ell+1}$, $(\alpha_1, \ldots, \alpha_{\log \ell}) \leftarrow \mathfrak{C}^{\log \ell}$, $\alpha_{\log \ell+1} \leftarrow \mathfrak{C}\setminus\{0,1\}$, $\rho \leftarrow D_{\mathsf{ze}}$.
2. Sample $(v_A, v_B, v_C) \leftarrow R^3$.
3. Sample $(\tilde{\mu}, (\hat{h}_j)_{j\in[\log \ell+1]})$ uniformly from the set $\mathcal{T}_{\boldsymbol{\alpha}}$ of consistent transcripts of Lemma 6.2 (instantiated with $\mathcal{M} \coloneqq R$, and $k = \log \ell + 1$), i.e., uniformly conditioned on the verifier's sumcheck checks. (Sample $\tilde{\mu} \leftarrow R$ and all coefficients of each $\hat{h}_j$ except the linear one, which is then determined by the corresponding check, as in the proof of Lemma 6.2.)
4. Compute the explicit output instance $\mathbb{x}$ as in Items 7 and 8, using the sampled values (v_A, v_B, v_C) and the public values $(u_M)_{M\in\{A,B,C\}}$ computed from (A, B, C, v) and $\boldsymbol{\alpha}$.
5. Run $\mathcal{D}$ on $\mathbb{x}$ and answer its (non-adaptive) queries to $\mathbb{y} = f$ using $\mathbf{S}_{\mathcal{C}}$. Retain the queries and answers in the view.
6. Output the simulated view.

We compare distributions in two claims, conditioning on the verifier randomness.

Transcript claim. Fix the prover's witness and inner masks (thus fixing $\hat{g}$ and the values (v_A, v_B, v_C) from Item 6). The honest transcript decomposes as

$$(\tilde{\mu}, \hat{h}_1, \ldots, \hat{h}_{\log \ell+1}) = \pi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) + \boldsymbol{\tau}\,,$$

where $\pi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ is the transcript of Lemma 6.2 for $\boldsymbol{\xi}$ (with $\mathcal{M} \coloneqq R$, $\boldsymbol{m} = (1)$, and $k = \log \ell + 1$) and $\boldsymbol{\tau}$ is the transcript produced by the honest prover with zero outer mask. The round polynomials of $\boldsymbol{\tau}$ are then the honest sumcheck of $\varepsilon\cdot\hat{g}\cdot\mathsf{eq}(\boldsymbol{\beta}, \cdot)$. Because $\hat{g}$ vanishes on $\{0,1\}^{\log \ell+1}$ in the honest case, they satisfy the sumcheck consistency checks, and each has degree at most $5 < d_{\mathsf{R1CS}}$, so $\boldsymbol{\tau} \in \mathcal{T}_{\boldsymbol{\alpha}}$. Since $\boldsymbol{\xi}$ is uniform over $R^{\ell^{\mathsf{zk}}_{\mathrm{out}}}$, $\pi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ is uniform over the subgroup $\mathcal{T}_{\boldsymbol{\alpha}}$ by Item 2 of Lemma 6.2 (here we use that R is finite). The honest transcript $\pi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) + \boldsymbol{\tau}$ is a translate of $\pi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ by the subgroup element $\boldsymbol{\tau}$, hence uniform over $\mathcal{T}_{\boldsymbol{\alpha}}$ as well. Because the transcript is uniform over $\mathcal{T}_{\boldsymbol{\alpha}}$ for every fixing

of the witness and inner masks, it is independent of (v_A, v_B, v_C) and of the witness. This matches the simulator's choice in Item 3.

Value claim. By the decomposition in Item 6, in the honest execution, for each M,

$$v_M = \mathrm{const}_M + \bar{\xi}_M \cdot \left(\alpha^2_{\log \ell+1} - \alpha_{\log \ell+1}\right) ,$$

where const_M depends only on the witness and the verifier randomness. Since $0, 1, \alpha_{\log \ell+1}$ are distinct elements of the strong sampling set $\mathfrak{C}$, the central element $\alpha^2_{\log \ell+1} - \alpha_{\log \ell+1} = (\alpha_{\log \ell+1} - 0) \cdot (\alpha_{\log \ell+1} - 1)$ is invertible, so multiplication by it is a bijection on R. The mask $\bar{\xi}_M$ is uniform, hence v_M is uniform over R conditioned on the verifier randomness and the witness. The three masks are independent, so the triple (v_A, v_B, v_C) is uniform over R^3 and independent of the witness. This matches the simulator's choice in Item 2.

We now compare the full views. First fix the message encoded in f and all direct prover messages, and condition on the verifier randomness and on the randomness of $\mathcal{D}_0$ from which its query count is computed. The query list remains uniform, and the encoding randomness of f is uniform and independent of this list and of the conditioning. We replace the answers of f by those of $\mathbf{S}_{\mathcal{C}}$, which changes the view by at most ζ by Definition 3.21 and Claim 3.22.

After this step, no oracle answer depends on the message encoded in f. By the transcript claim, for every fixing of the witness and inner masks, the honest transcript has the same distribution as the independent uniform transcript sampled in Item 3, and every other component is obtained from the transcript by the same randomized map in both distributions. Next, condition on the witness. By the value claim, (v_A, v_B, v_C) has the same distribution as the independent uniform triple sampled in Item 2, and every other component is again obtained from this triple by the same randomized map in both distributions.

The resulting distribution is the output of the simulator, so the simulation error is at most ζ.

Finally, the reduction has fresh output. Its output implicit instance is the proof string f, which the verifier does not query. The direct messages of the prover depend only on the message of f, and the prover samples $\boldsymbol{r}$ uniformly and uses it only to compute f. Conditioned on the view of the verifier and on this message, the randomness $\boldsymbol{r}$ is therefore uniform. □

13 High-distance codes from dispersers

We discuss distance amplification via disperser graphs [ABNNR92; GRS12] and the efficiency gains it gives.

13.1 Preliminaries

The following two definitions and subsequent proposition are from [GRS12].

Definition 13.1. *Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R| = m$. We say that G is a (ε, γ)-**disperser** if for every $S \subseteq L$ with $|S| \geq (1-\varepsilon)m$ the neighborhood $N(S) \coloneqq \{j \in R \mid \exists\, i \in S : (i,j) \in E\}$ satisfies $|N(S)| \geq (1-\gamma)|R| = (1-\gamma)m$.*

Definition 13.2. *Let $\mathcal{C} \subseteq \mathbb{F}^m$ be a code with minimum distance $\delta(\mathcal{C}) \geq 1-\varepsilon$. Let $G = (L \cup R, E)$ be a (ε, γ)-disperser with $|L| = |R| = m$, and assume G is right-regular of degree d. For every $j \in R$, fix an ordered list $(i_{j,1}, i_{j,2}, \ldots, i_{j,d}) \in L^d$ of its d left incidences (allowing repetitions when G has multi-edges). Define the* **amplification map** *$A_G \colon \mathbb{F}^m \to (\mathbb{F}^d)^m$ by $(A_G(c))_j \coloneqq \left(c_{i_{j,1}}, \ldots, c_{i_{j,d}}\right)$ for each $j \in R$.*

Proposition 13.3 (Code-amplification lemma)**.** *Let $\mathcal{C}$, G, and A_G be as in Definition 13.2. Then the amplified code $G(\mathcal{C}) \coloneqq \{A_G(c) \mid c \in \mathcal{C}\} \subseteq (\mathbb{F}^d)^m$ has minimum distance $\delta(G(\mathcal{C})) \geq 1-\gamma$.*

Proof. Take distinct $c, c' \in \mathcal{C}$ and let $S \coloneqq \{i \in L \mid c_i \neq c'_i\}$. Since $\delta(\mathcal{C}) \geq 1-\varepsilon$, we have $|S| \geq (1-\varepsilon)m$ and hence $|N(S)| \geq (1-\gamma)m$ by Definition 13.1. For any $j \in N(S)$ there exists $k \in \{1, \ldots, d\}$ with $i_{j,k} \in S$, so the j-th symbols of $A_G(c)$ and $A_G(c')$ differ in coordinate k of the d-tuple. Therefore the number of right indices where the amplified codewords differ is at least $|N(S)| \geq (1-\gamma)m$. □

Below we show via a standard probabilistic method argument that random graphs are good dispersers (similar lemmas can be found in [Coh91; TUZ07]). The edges of the graph are sampled at random such that each right vertex has d left-neighbors. We now give a degree that suffices to guarantee the parameters of the disperser.

Proposition 13.4 (Probabilistic sampling of dispersers)**.** *Let $\varepsilon, \gamma, \alpha \in (0,1)$, let $m, d \in \mathbb{N}$ with $m \geq 1$, and let L, R be vertex sets with $|L| = |R| = m$. The random bipartite graph $G = (L \cup R, E)$ (with multi-edges) is obtained by choosing, for each $j \in R$, a multiset of d neighbors in L uniformly at random with replacement (right-degree d). If*

$$d > \frac{\lceil (1-\varepsilon)m \rceil \ln \frac{me}{\lceil (1-\varepsilon)m \rceil} + \lceil \gamma m \rceil \ln \frac{me}{\lceil \gamma m \rceil} - \ln \alpha}{\lceil \gamma m \rceil \cdot (-\ln \varepsilon)},$$

then, with probability at least $1-\alpha$, the graph G is a (ε, γ)-disperser.

Proof. If G fails to be a (ε, γ)-disperser, then there exist $S \subseteq L$ and $T \subseteq R$ with $|S| \geq \lceil (1-\varepsilon)m \rceil$ and $|T| \geq \lceil \gamma m \rceil$ such that every neighbor of each $j \in T$ lies in $L \setminus S$, i.e., $N(S) \subseteq R \setminus T$. We may restrict to $|S| = \lceil (1-\varepsilon)m \rceil$ and $|T| = \lceil \gamma m \rceil$ for the union bound below.

Fix S, T of sizes $|S| = \lceil (1-\varepsilon)m \rceil$ and $|T| = \lceil \gamma m \rceil$. Let $p \coloneqq \frac{m-|S|}{m} \leq \varepsilon$ be the probability that a single random neighbor lies in $L \setminus S$. For a fixed $j \in T$, the probability that all d independently sampled neighbors (with replacement) lie in $L \setminus S$ is at most $p^d \leq \varepsilon^d$. Independence across $j \in$

T yields $\Pr[\text{all neighbors of all } j \in T \text{ avoid } S] \le \left(\varepsilon^d\right)^{|T|} = \varepsilon^{d\lceil \gamma m \rceil}$. By a union bound over all choices of S and T of these sizes, $\Pr[\text{failure}] \le \binom{m}{\lceil (1-\varepsilon)m \rceil}\binom{m}{\lceil \gamma m \rceil}\varepsilon^{d\lceil \gamma m \rceil}$. Using $\binom{N}{k} \le \left(\frac{Ne}{k}\right)^k$ gives $\Pr[\text{failure}] \le \left(\frac{me}{\lceil (1-\varepsilon)m \rceil}\right)^{\lceil (1-\varepsilon)m \rceil}\left(\frac{me}{\lceil \gamma m \rceil}\right)^{\lceil \gamma m \rceil}\varepsilon^{d\lceil \gamma m \rceil}$. We conclude by making the right-hand side at most α. □

Use in our protocols. Our protocols can be instantiated with any $\mathbb{F}$-additive code with a zero-knowledge encoding, including distance-amplified codes. Although amplified codewords are over a larger alphabet, they remain $\mathbb{F}$-additive: a vector in $(\mathbb{F}^d)^m$ is equivalently a vector in $\mathbb{F}^{d \cdot m}$. There are generic results on mutual correlated agreement for every $\mathbb{F}$-additive code [GKL24; GCXK26; BCGM26]. Moreover, if $\mathcal{C} \subseteq \mathbb{F}^m$ has a strong t-query zero-knowledge encoding $\mathsf{Enc}_{\mathcal{C}} \colon \mathbb{F}^\ell \times \mathbb{F}^r \to \mathbb{F}^m$, then the amplified code $G(\mathcal{C}) = \{A_G(c) : c \in \mathcal{C}\}$ has a strong $\lfloor t/d \rfloor$-query zero-knowledge encoding: each query to one amplified symbol reveals d base-code symbols. (Strong zero knowledge is needed here since a uniform amplified location induces a non-uniform set of base-code locations.) In sum, distance-amplified codes can be used in our constructions as the code $\mathcal{C}$ (for example in Theorem 8.1).

13.2 Amplified Reed–Solomon code

We describe how distance amplification can be used to improve the encoding time of codes, in particular Reed–Solomon codes, given any target distance. The distance of the codes used in the protocols determines the number of queries (and hence the argument size after compilation via Merkle commitments), so it is beneficial to have a code with distance $1 - \gamma$ for small γ.

In general, distance amplification turns any code into a code of any distance, using graphs with sufficiently large degrees (which correspond to a sufficiently large alphabet for the output code). In the following, we describe the code obtained by distance amplification of the Reed–Solomon code, which allows for faster encoding for a target distance.

Definition 13.5 (Amplified Reed–Solomon code)**.** *Fix $c > 1$ and a message length ℓ. Let $\mathcal{L} \subseteq \mathbb{F}$ be an evaluation domain of size $m := \lceil c\ell \rceil$ in a field $\mathbb{F}$ with $|\mathbb{F}| \ge m$, let $L := \mathcal{L}$, and let $G = (L \cup R, E)$ be a right-regular bipartite graph of degree d that is a (ε, γ)-disperser with $\varepsilon := \frac{\ell - 1}{m}$. The code $\mathsf{ARS}[\mathbb{F}, m, \ell, G]$ is defined via the encoding that maps a message $\mathsf{msg} \in \mathbb{F}^\ell$ to $A_G(u) \in (\mathbb{F}^d)^{|R|}$, where $u := \mathsf{Enc}_{\mathsf{RS}}(\mathsf{msg}) \in \mathsf{RS}[\mathbb{F}, m, \ell]$ (following Definition 13.2).*

The following claim follows by Definition 13.2 and Proposition 13.3.

Claim 13.6 (Parameters of the amplified RS code)**.** *The code in Definition 13.5 has alphabet $\mathbb{F}^d$, block length $|R|$ (in alphabet symbols), and minimum distance at least $1 - \gamma$.*

Encoding time for two ways to reach relative distance $1 - \gamma$. We want to encode a message $\mathsf{msg} \in \mathbb{F}^\ell$ into a codeword with minimum distance at least $1 - \gamma$. We compare two methods.

1. Plain Reed–Solomon: Here the evaluation domain has size $\left\lceil \frac{\ell - 1}{\gamma} \right\rceil$, in a large enough field $\mathbb{F}$, so that $\delta(\mathsf{RS}[\mathbb{F}, \left\lceil \frac{\ell - 1}{\gamma} \right\rceil, \ell]) \ge 1 - \gamma$. Encoding msg consists of (i) interpolating the degree-$(\ell - 1)$ polynomial from the ℓ message symbols, and (ii) evaluating it at all points of the chosen domain. Assuming FFT-style algorithms on $\mathcal{L}$, this requires at most

$$c_{\mathrm{FFT}} \cdot \ell \cdot \log_2 \ell + c_{\mathrm{FFT}} \cdot \left\lceil \frac{\ell - 1}{\gamma} \right\rceil \cdot \log_2 \left\lceil \frac{\ell - 1}{\gamma} \right\rceil$$

field operations. (If $\mathsf{Enc}_{\mathsf{RS}}$ treats the message as polynomial coefficients, only the evaluation term remains.)

2. Amplified Reed–Solomon (with factor $c := m/\ell$ and a right-regular disperser of degree d from the main bound): The message $\mathsf{msg} \in \mathbb{F}^{\ell}$ is first encoded as a Reed–Solomon codeword on an evaluation domain of size $c \cdot \ell$, in at most $c_{\mathrm{FFT}} \cdot \ell \cdot \log_2 \ell + c_{\mathrm{FFT}} \cdot (c \cdot \ell) \cdot \log_2 (c \cdot \ell)$ field operations (again without the interpolation term if the message is given as coefficients). The amplification map A_G is then applied using the disperser graph, with $d \cdot |R|$ copy operations. If, as in Definition 13.5, one sets $|R| = c \cdot \ell$, this amplification step costs $d \cdot c \cdot \ell$ symbols. Overall, the encoding time amounts to:

$$c_{\mathrm{FFT}} \cdot \ell \cdot \log_2 \ell + c_{\mathrm{FFT}} \cdot (c \cdot \ell) \cdot \log_2 (c \cdot \ell) + d \cdot c \cdot \ell \,.$$

For the amplified option, we apply Proposition 13.4 with $n := c \cdot \ell$, $\varepsilon := \frac{\ell-1}{c \cdot \ell}$, γ as in the target distance, and any fixed $\alpha \in (0,1)$ for the construction's failure probability. Concretely, let d be the least integer larger than

$$\frac{(1-\varepsilon)n \cdot \ln \frac{ne}{(1-\varepsilon)n} + \gamma\, n \cdot \ln \frac{ne}{\gamma n} - \ln \alpha}{\gamma n \cdot (-\ln \varepsilon)} \quad \text{with} \quad n = c \cdot \ell, \quad \varepsilon = \frac{\ell - 1}{c \cdot \ell} \,.$$

With this choice, the amplified variant is faster than plain RS when

$$\underbrace{c_{\mathrm{FFT}} \cdot (c \cdot \ell) \cdot \log_2 (c \cdot \ell)}_{\text{RS on size } c \cdot \ell} + \underbrace{d \cdot c \cdot \ell}_{\text{amplification copies}} \leq \underbrace{c_{\mathrm{FFT}} \cdot \left(\frac{\ell - 1}{\gamma}\right) \cdot \log_2 \left(\frac{\ell - 1}{\gamma}\right)}_{\text{RS on size } (\ell-1)/\gamma} .$$

Intuitively, when γ is small (demanding very large plain-RS block length), the right-hand side grows like $\frac{\ell}{\gamma} \log_2 \frac{\ell}{\gamma}$, while the extra cost of amplification grows only linearly in $\frac{\ell}{\gamma}$ through $d \cdot c \cdot \ell$ (since d from the lemma scales like $\frac{(1-\frac{1}{c}) \ln \frac{e}{1-\frac{1}{c}} + \gamma \ln \frac{e}{\gamma}}{\gamma \ln c}$ with $n = c \cdot \ell$). Thus, for small γ and fixed c, the amplified RS method is cheaper.

Benefits of amplified RS. Amplified Reed–Solomon codes offer benefits in time and space.

- *No large FFTs.* The FFT is for size $\Theta(\ell)$ rather than $\Theta(\ell/\gamma)$. This improves encoding time, by trading a $\Theta(\ell/\gamma)$-size FFT for a $\Theta(c\,\ell)$-size FFT plus a $\Theta\,(\ell/\gamma)$-time pass to gather neighbors.

- *Space complexity.* The *encoding space* required for ARS is also decreased, if one has efficient access to the graph G. Even if one were to use in-place FFTs, the encoding space of the Reed–Solomon code at that distance is ℓ/γ, while we can afford smaller FFTs requiring only $O(\ell)$ size. Storing G in memory takes $O(\frac{\ell}{\gamma} \log \ell)$, making the space complexity slightly *worse* than in-place FFT implementations. Since strongly-explicit dispersers (where the neighborhood of a vertex could be computed in polylogarithmic time) with $d = O(1/\gamma)$ are not known, one would have to store the sampled G in memory. An alternative approach would be to *stream* G, performing the alphabet aggregation for the neighbors of each vertex when their neighborhoods are streamed. This streaming option is much less clear for FFTs. Furthermore, in practice, one could use a PRG that, in principle, replaces the notion of a strongly explicit construction, as the neighbor set of each vertex could be sampled "on the fly" in a reproducible way.

13.3 Distance amplification and Merkle-compiled argument size

We discuss potential efficiency gains from distance amplification for argument size. While the proposed application here increases the encoding time (as opposed to the decrease in encoding time mentioned in the previous section), we show that it can decrease the part of the argument size due to Merkle openings.

A case study. To investigate this point, we consider an example, which is based on the codeword batching protocol from [BCFW25]. In this protocol, the prover sends as an oracle a purported codeword, and the verifier makes t independent spot-checks to an oracle. When the protocol's soundness relies on the fact that two distinct codewords disagree on a $1-\varepsilon$ fraction of coordinates (i.e., the underlying code has relative distance at least $1-\varepsilon$), then each independent check catches a cheating strategy with probability at least $1-\varepsilon$, and hence the overall soundness error is at most ε^t. Thus, to drive the soundness error below $2^{-\lambda}$, it suffices to take $t \geq \frac{\lambda}{-\log_2 \varepsilon}$.

The cost of a Merkle commitment. Consider compiling such a protocol into an argument by committing to the codeword via a Merkle commitment whose leaves are the codeword symbols. Each verifier query is then answered by the leaf value and the authentication path, i.e., the sibling hashes along the root-to-leaf path. For a codeword of block length m (in symbols), the path has length $\lceil \log_2 m \rceil$. Hence, measured as "one field symbol plus one hash digest" units, the total proof communication contributed by these t oracle queries is $t \cdot (1 + \lceil \log_2 m \rceil)$, up to constant factors that depend only on the fixed hash function.

Merkle compilation cost for an amplified code. Now replace the underlying code $\mathcal{C} \subseteq \mathbb{F}^m$ by its amplified version $G(\mathcal{C}) \subseteq (\mathbb{F}^d)^m$ from Definition 13.2, where G is a right-regular (ε, γ)-disperser of degree d and $|R| = m$. By Proposition 13.3, the amplified code has relative distance at least $1-\gamma$, so the same "independent spot-check" reasoning yields soundness error at most γ^t, and to achieve soundness error below $2^{-\lambda}$ it suffices to take $t \geq \frac{\lambda}{-\log_2 \gamma}$. After compilation, each query now opens a *single amplified symbol*, i.e., an element of $\mathbb{F}^d$, together with an authentication path of length $\lceil \log_2 m \rceil$. Hence the argument size contributed by the t oracle queries becomes $t \cdot (d + \lceil \log_2 m \rceil)$.

In the disperser-based setting of Definitions 13.1 and 13.2, the Merkle depth terms coincide and amplification yields an improvement for this part of the argument when

$$d < (1 + \lceil \log_2 m \rceil) \cdot \frac{-\log_2 \gamma}{-\log_2 \varepsilon} - \lceil \log_2 m \rceil \,. \tag{15}$$

Instantiating d via Proposition 13.4. Fix any target $\gamma \in (0,1)$ and any $\alpha \in (0,1)$. To assess whether Equation 15 can hold for the *best* (i.e., smallest) degree guaranteed by Proposition 13.4, let d be the least integer larger than its bound. Ignoring ceiling effects and the additive $-\ln \alpha$ term (which is divided by $\gamma\, m$ and is negligible once m is large), the bound has the following dominant form:

$$d \;\approx\; \frac{(1-\varepsilon)\,\ln \frac{e}{1-\varepsilon}}{\gamma \cdot (-\ln \varepsilon)} \;+\; \frac{\ln \frac{e}{\gamma}}{-\ln \varepsilon} \,.$$

In particular, for fixed ε this degree grows proportionally to $1/\gamma$ as $\gamma \to 0$. Since the required number of openings decreases only as $1/(-\log_2 \gamma)$, very small γ forces d so large that Equation 15 eventually fails: in this regime, amplifying to extremely high relative distance makes each Merkle leaf so large that the reduction in the number of openings cannot compensate.

Concrete parameters. The condition in Equation 15 *can* hold for moderate target distances. For instance, take $m = 2^{20}, \alpha = 2^{-40}, \lambda = 128, \varepsilon = \frac{1}{3}$. Then the plain compilation needs $t \approx 81$

openings and the query/opening contribution is approximately $81 \cdot (1+20) \approx 1.70 \cdot 10^3$ units. If we target $\gamma = 0.1$ (relative distance 0.9), the smallest d guaranteed by Proposition 13.4 is $d = 12$, and the amplified compilation needs $t \approx 39$ openings, contributing about $39 \cdot (12+20) \approx 1.25 \cdot 10^3$ units, a reduction of roughly 27%. In contrast, if we target $\gamma = 0.01$ (relative distance 0.99), the same calculation yields $d = 91$ and $t \approx 20$, giving about $20 \cdot (91 + 20) \approx 2.22 \cdot 10^3$ units, which is *larger* than in the plain case. These examples illustrate that, under Proposition 13.4, the Merkle-query part improves only when the requested distance increase is substantial but not so extreme that the required d becomes the dominant term in each opening.

Distance amplification slows down the prover. To compare *prover* runtime, we separate (a) committing to the oracle (encoding, plus building the Merkle commitment) from (b) answering the t openings. In the comparison we ignore the base encoding time of $\mathcal{C}$ (it is the same in both variants) and we count the total number of field-symbol copies/hashes.

As before, we assume that $m = 2^{20}$ so the Merkle commitment has 2^{20} leaves and $2^{20}-1$ internal nodes. In the *plain* case, committing to $\mathcal{C} \subseteq \mathbb{F}^m$ requires hashing 2^{20} leaves (each containing one field symbol) and $2^{20}-1$ internal nodes, i.e., about $(2^{20})+(2^{20}-1) = 2{,}097{,}151$ hash computations. Answering $t \approx 81$ openings transmits 81 field symbols and $81 \cdot 20$ sibling hashes, i.e., $81 \cdot (1+20) = 1{,}701$ units in the same model. Overall, the plain prover's Merkle-related work is $\approx 2{,}098{,}852$ units.

For $\gamma = 0.1$, the probabilistic-method bound yields $d = 12$ and $t \approx 39$. In the *amplified* case, beyond the same base encoding, the prover must (i) compute A_G (about $dm = 12 \cdot 2^{20} = 12{,}582{,}912$ symbol copies), and (ii) hash 2^{20} leaves each containing $d = 12$ field symbols (about another 12,582,912 symbol-hash units), plus the same $2^{20}-1$ internal-node hashes. Answering the openings transmits 39 leaves of size 12 plus $39 \cdot 20$ sibling hashes, i.e., $39 \cdot (12+20) = 1{,}248$ units. Altogether, this totals roughly $12{,}582{,}912 + 12{,}582{,}912 + (2^{20}-1) + 1{,}248 = 26{,}215{,}647$ units, i.e., about a 12.5× increase over the plain prover time in this model. In other words, even though the *communication* in openings improves (roughly 27% above), the *prover* becomes slower because it must compute and commit to d-times larger leaves.

A A generic MCA bound for $(R, \mathcal{M})$-additive codes

We prove the elementary geometric MCA bound for $(R, \mathcal{M})$-additive codes. Over rings, the only extra requirement is that challenges come from a strong sampling set, so that the difference of two distinct challenges is invertible.

Lemma A.1. *Let $\mathcal{C} \subseteq \mathcal{M}^m$ be an $(R, \mathcal{M})$-additive code, let $\mathfrak{C} \subseteq R$ be a strong sampling set, and let $\mathsf{PG}\colon \mathfrak{C} \to R^2$ be $\gamma \mapsto (1, \gamma)$ with γ uniform in $\mathfrak{C}$. Let $D \coloneqq \lfloor \delta \cdot m \rfloor$. If*

$$3D < \delta(\mathcal{C}) \cdot m$$

(in particular, if $\delta < \delta(\mathcal{C})/3$), then the MCA error of PG for $\mathcal{C}$ at distance δ satisfies

$$\epsilon_{\mathsf{mca}}(\mathcal{C}, \delta) \leq \frac{D+1}{|\mathfrak{C}|}\,.$$

The same bound holds for $\gamma \mapsto (1-\gamma, \gamma)$.

Proof. Fix $f_1, f_2 \in \mathcal{M}^m$. Let $B \subseteq \mathfrak{C}$ be the bad challenges. For each $\gamma \in B$, fix a witness $S_\gamma \subseteq [m]$ and $u_\gamma \in \mathcal{C}$ with $|[m] \setminus S_\gamma| \leq D$ and

$$(f_1 + f_2 \cdot \gamma)(S_\gamma) = u_\gamma(S_\gamma)\,.$$

We prove $|B| \leq D + 1$.

If $D = 0$, from two bad challenges we could interpolate codewords $u_1, u_2 \in \mathcal{C}$ with $f_i = u_i$ on all coordinates, a contradiction. Hence $|B| \leq 1$ in this case. Assume now $D \geq 1$ and, towards contradiction, $|B| \geq D + 2$. In particular $|B| \geq 3$. Fix distinct $a, b \in B$ and set

$$u_2 \coloneqq (u_a - u_b) \cdot (a - b)^{-1}, \qquad u_1 \coloneqq u_a - u_2 \cdot a\,.$$

These are codewords because $\mathcal{C}$ is an R-submodule and $a - b$ is invertible. For every $\gamma \in B$,

$$u_\gamma = u_1 + u_2 \cdot \gamma\,. \tag{16}$$

Indeed this is immediate for a, b. For any third challenge, the three agreement sets intersect in more than $(1 - \delta(\mathcal{C}))m$ coordinates. On this intersection, by interpolating from a, b we obtain $f_i = u_i$, so the two codewords in Equation 16 agree there and hence are equal.

Let $e_i \coloneqq f_i - u_i$, let

$$E \coloneqq \{x \in [m] : (e_1(x), e_2(x)) \neq (0, 0)\}\,,$$

and, for $\gamma \in B$, let

$$Z_\gamma \coloneqq \{x \in E : e_1(x) + e_2(x) \cdot \gamma = 0\}\,.$$

By Equation 16, $|E \setminus Z_\gamma| \leq D$. Also $Z_\gamma \neq \emptyset$: otherwise f_1, f_2 would agree with u_1, u_2 on S_γ, contradicting badness. Finally, the sets Z_γ are pairwise disjoint: if $x \in Z_\gamma \cap Z_{\gamma'}$ and $\gamma \neq \gamma'$, then

$$e_2(x) \cdot (\gamma - \gamma') = 0\,,$$

so $e_2(x) = 0$ and then $e_1(x) = 0$, contradicting $x \in E$.

Writing $t \coloneqq |E|$, disjointness gives $|B| \le t$ when $t \le D$, and otherwise

$$|B| \cdot (t - D) \le \sum_{\gamma \in B} |Z_\gamma| \le t \,,$$

so $|B| \le t/(t-D) \le D+1$. In both cases $|B| \le D+1$, contradicting $|B| \ge D+2$. Thus $|B| \le D+1$, and a uniform challenge is bad with probability at most $(D+1)/|\mathfrak{C}|$.

For $\gamma \mapsto (1-\gamma, \gamma)$, this case reduces to the previous one for f_1 and $f_2 - f_1$, since $f_1 \cdot (1-\gamma) + f_2 \cdot \gamma = f_1 + (f_2 - f_1) \cdot \gamma$. If f_1 and $f_2 - f_1$ agree with codewords on a set, then so do f_1 and f_2. $\square$

B Hiding Merkle commitments without salts

With the Reed–Solomon codes of Section 11, the oracles of Construction 11.1 can be committed without salts when every row is encoded with one more seed coordinate than the number of queries. Recall that the transformation of [BCS16] compiles an HVZK IOP into a zero-knowledge argument by committing to every oracle with a Merkle tree whose leaves are *salted*: the leaf at position x is the hash of the symbol at x together with a fresh random string, which the prover reveals when x is queried. Every query thus adds one salt to the argument. With the additional coordinate, an unopened symbol retains enough entropy to hide itself, and a single ring element per encoding seed replaces the salts.

Residual entropy. We show that the unopened columns of an interleaved Reed–Solomon codeword remain unpredictable given the opened columns and a linear combination of the rows. The linear combination accounts for the folded codeword, which determines everything that later messages reveal about the oracle.

Lemma B.1. *Let R be finite, let $\mathcal{L} \subseteq \mathcal{Z}(R)$ be a strong sampling set, let $t < \ell \leq |\mathcal{L}|$, and let $\mathsf{Enc}_{\mathcal{C}} \colon R^{\ell-t-1} \times R^{t+1} \to R^{\mathcal{L}}$ be the encoding of Lemma 3.29 with randomness length $t+1$. Fix $K \geq 1$, messages $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K \in R^{\ell-t-1}$, coefficients $\boldsymbol{c} \in \mathcal{Z}(R)^K$, a set $S \subseteq \mathcal{L}$ with $|S| \leq t$, and $x \in \mathcal{L} \setminus S$. Sample $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_K \leftarrow R^{t+1}$ independently and let $f_i \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{v}_i, \boldsymbol{r}_i)$. Conditioned on the opened columns $(f_i[S])_{i \in [K]}$ and on the combination $\sum_i \boldsymbol{r}_i \cdot \boldsymbol{c}_i$ of the seeds, the column $(f_i(x))_{i \in [K]}$ is uniform over a coset of the subgroup*

$$N_{\boldsymbol{c}} \coloneqq \left\{ \boldsymbol{y} \in R^K \;:\; \sum_{i \in [K]} \boldsymbol{y}_i \cdot \boldsymbol{c}_i = 0 \right\} \subseteq R^K .$$

In particular, it takes every value with conditional probability at most $|N_{\boldsymbol{c}}|^{-1} \leq |R|^{-(K-1)}$.

Proof. Let $s \coloneqq |S|$ and $Z(\mathtt{X}) \coloneqq \prod_{a \in S} (\mathtt{X} - a)$, a monic polynomial of degree s with central coefficients. We view each seed as a polynomial $\hat{r}_i \in R^{<t+1}[\mathtt{X}]$. Division with remainder by Z is a bijection $\hat{r}_i \mapsto (\hat{q}_i, \hat{\rho}_i)$ onto $R^{<t+1-s}[\mathtt{X}] \times R^{<s}[\mathtt{X}]$, where $\hat{r}_i = \hat{q}_i \cdot Z + \hat{\rho}_i$, so the pairs $(\hat{q}_i, \hat{\rho}_i)$ are independent and uniform.

For $a \in S$, the symbol $f_i(a)$ depends on $\boldsymbol{r}_i$ only through $\hat{\rho}_i(a)$, and by Lemma 3.9 the polynomial $\hat{\rho}_i$ is determined by its values on S. Hence conditioning on $f_i[S]$ fixes $\hat{\rho}_i$ and leaves $\hat{q}_i$ uniform and independent. Since $\boldsymbol{c}$ is central, $\sum_i \hat{r}_i \cdot \boldsymbol{c}_i = (\sum_i \hat{q}_i \cdot \boldsymbol{c}_i) \cdot Z + \sum_i \hat{\rho}_i \cdot \boldsymbol{c}_i$, so further conditioning on the combination of the seeds amounts to conditioning on $\sum_i \hat{q}_i \cdot \boldsymbol{c}_i$. The tuple $(\hat{q}_i)_i$ is then uniform over a coset of the subgroup of tuples with $\sum_i \hat{q}_i \cdot \boldsymbol{c}_i = 0$. Evaluation at x is a group homomorphism that maps this subgroup onto $N_{\boldsymbol{c}}$: its image is contained in $N_{\boldsymbol{c}}$, and constant polynomials show surjectivity since $t + 1 - s \geq 1$. Thus $(\hat{q}_i(x))_i$ is uniform over a coset of $N_{\boldsymbol{c}}$.

Finally, $f_i(x) = \hat{q}_i(x) \cdot Z(x) + w_i$, where w_i is fixed by the conditioning and $Z(x) = \prod_{a \in S} (x - a)$ is central and invertible because $\mathcal{L}$ is a strong sampling set. Multiplication by $Z(x)$ maps cosets of $N_{\boldsymbol{c}}$ bijectively to cosets of $N_{\boldsymbol{c}}$, which proves the first claim. The bound on $|N_{\boldsymbol{c}}|$ holds because $N_{\boldsymbol{c}}$ is the kernel of a homomorphism from R^K to R. □

Hiding. Unpredictable leaves can be hashed without salts. Let H be a random oracle, and let the digest of the leaf at position x of an oracle f be $H(x, f(x))$.

Lemma B.2. *Let $f \in \Sigma^m$ and z be jointly distributed and independent of H, where z determines a set $S \subseteq [m]$, and let $\kappa \geq 0$ be such that $\Pr[f(x) = \sigma \mid z] \leq 2^{-\kappa}$ for every $x \notin S$, every $\sigma \in \Sigma$, and every value of z. Let $h_x \coloneqq H(x, f(x))$, and let $(u_x)_{x \notin S}$ be independent uniform strings of the same length. For every algorithm A that makes at most T queries to H, the outputs of $A^H(z, (h_x)_{x \notin S})$ and $A^H(z, (u_x)_{x \notin S})$ have statistical distance at most $T \cdot 2^{-\kappa}$.*

Proof. The two experiments are identical unless A queries H at $(x, f(x))$ for some $x \notin S$: until then, the digests h_x are uniform strings independent of the answers of H. In the second experiment the view of A is independent of $(f(x))_{x \notin S}$ given z, so each query is of this form with probability at most $2^{-\kappa}$. A union bound over the queries gives the claim. □

The digests of the internal nodes of a Merkle tree, and hence its root and all authentication paths, are computed from the leaf digests. By Lemma B.2, the commitment to f and the openings of the positions in S can therefore be simulated from z and uniform strings, provided that z contains the opened symbols $f[S]$.

Application to our construction. Consider Construction 11.1 with the Reed–Solomon codes of Section 11 and with $t_{\mathfrak{i}} + 1$ seed coordinates per row instead of $t_{\mathfrak{i}}$ in every encoding. Fix a stage $\mathfrak{i}$ and the messages and masks of the rows of $f_{\mathfrak{i}}$. The view of the verifier depends on the seeds $\boldsymbol{r}_{\mathfrak{i}}$ only through the at most $t_{\mathfrak{i}}$ opened columns of $f_{\mathfrak{i}}$ and through the folded seed $\bar{\boldsymbol{r}}_{\mathfrak{i}}$, a linear combination of the seeds of the rows with coefficients $\mathsf{eq}(\boldsymbol{b}, \boldsymbol{\gamma}_{\mathfrak{i}})$. By Lemma B.1 with $K = 2^{k_{\mathfrak{i}}} \geq 2$, applied to the encoding whose message includes the mask, every unopened column of $f_{\mathfrak{i}}$ takes each value with probability at most $|\mathbb{F}|^{-1}$ conditioned on this view. For the last oracle, which has the base-case mask g as an additional row as in our evaluation, we apply the lemma to its $2^{k_{\mathfrak{n}}} + 1$ rows, since the base case discloses a combination of $\bar{\boldsymbol{r}}_{\mathfrak{n}}$ and $\boldsymbol{r}_\star$. By Lemma B.2 with $\kappa = \log_2 |\mathbb{F}|$, the commitment to each $f_{\mathfrak{i}}$ is hiding without salts.

The cost of the additional randomness is one more coordinate in every folded seed and in the disclosed seed $\boldsymbol{r}^*$. In contrast, salting adds one string per opened leaf, that is, $\sum_{\mathfrak{i} \in [\mathfrak{n}]} t_{\mathfrak{i}}$ strings to the argument.

Acknowledgments

The authors are supported in part by the Ethereum Foundation, the Sui Foundation, and Polygon Labs.

References

[ABF26] Gal Arnon, Dan Boneh, and Giacomo Fenzi. *Open Problems in List Decoding and Correlated Agreement*. Cryptology ePrint Archive, Paper 2026/680. 2026.

[ABNNR92] N. Alon, J. Bruck, J. Naor, M. Naor, and R.M. Roth. "Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs". In: *IEEE Transactions on Information Theory* 38.2 (1992), pp. 509–516. DOI: 10.1109/18.119713.

[ACFY24] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "STIR: Reed–Solomon Proximity Testing with Fewer Queries". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 380–413.

[ACFY25] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. "WHIR: Reed–Solomon Proximity Testing with Super-Fast Verification". In: *Proceedings of the 44th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '25. 2025, pp. 214–243.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *Proceedings of the 24th ACM Conference on Computer and Communications Security*. CCS '17. 2017, pp. 2087–2104.

[ARR25] Noor Athamnah, Noga Ron-Zewi, and Ron D. Rothblum. "Linear Prover IOPs in Log Star Rounds". In: *Proceedings of the 23rd Theory of Cryptography Conference*. TCC '25. 2025, pp. 335–368.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 701–732.

[BBHV22] Laasya Bangalore, Rishabh Bhadauria, Carmit Hazay, and Muthuramakrishnan Venkitasubramaniam. "On Black-Box Constructions of Time and Space Efficient Sublinear Arguments from Symmetric-Key Primitives". In: *Proceedings of the 20th Theory of Cryptography Conference*. TCC '22. 2022, pp. 417–446.

[BBMHRS21] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoit Razet, and Peter Scholl. "Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and $\mathbb{Z}_{2^k}$". In: *Proceedings of the 28th ACM Conference on Computer and Communications Security*. CCS '21. 2021, pp. 192–211. DOI: 10.1145/3460120.3484812.

[BCFFMMZ25] Anubhav Baweja, Alessandro Chiesa, Elisabetta Fedele, Giacomo Fenzi, Pratyush Mishra, Tushar Mopuri, and Andrew Zitek-Estrada. "Time-Space Tradeoffs for Sumcheck". In: *Proceedings of the 23rd Theory of Cryptography Conference*. TCC '25. 2025, pp. 37–70.

[BCFGRS17] Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. "Zero Knowledge Protocols from Succinct Constraint Detection". In: *Proceedings of the 15th Theory of Cryptography Conference*. TCC '17. 2017, pp. 172–206.

[BCFRRZ25] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. "Blaze: Fast SNARKs from Interleaved RAA Codes". In: *Proceedings of the 44th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '25. 2025, pp. 123–152.

[BCFW25] Benedikt Bünz, Alessandro Chiesa, Giacomo Fenzi, and William Wang. "Linear-Time Accumulation Schemes". In: *Proceedings of the 23rd Theory of Cryptography Conference*. TCC '25. 2025, pp. 369–399.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, pp. 19–46.

[BCGGHJ17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. ASIACRYPT '17. 2017, pp. 336–365.

[BCGGRS19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. "Linear-Size Constant-Query IOPs for Delegating Computation". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 494–521.

[BCGM26] Sarah Bordage, Alessandro Chiesa, Ziyi Guan, and Ignacio Manzur. "All Polynomial Generators Preserve Distance with Mutual Correlated Agreement". In: *Proceedings of the 41st Computational Complexity Conference*. CCC '26. 2026, 24:1–24:18.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. "Quasilinear-Size Zero Knowledge from Linear-Algebraic PCPs". In: *Proceedings of the 13th Theory of Cryptography Conference*. TCC '16-A. 2016, pp. 33–64.

[BCHKS25] Eli Ben-Sasson, Dan Carmon, Ulrich Haböck, Swastik Kopparty, and Shubhangi Saraf. *On Proximity Gaps for Reed–Solomon Codes*. Cryptology ePrint Archive, Report 2025/2055. 2025.

[BCL22] Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. "Zero-Knowledge IOPs with Linear-Time Prover and Polylogarithmic-Time Verifier". In: *Proceedings of the 41st Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '22. 2022, pp. 275–304.

[BCRSVW19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. 2019, pp. 103–128.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC '16-B. 2016, pp. 31–60.

[BFRW26] Benedikt Bünz, Giacomo Fenzi, Ron D. Rothblum, and William Wang. "TensorSwitch: Nearly Optimal Polynomial Commitments from Tensor Codes". In: *Proceedings of the 46th Annual International Cryptology Conference*. CRYPTO '26. 2026, pp. 297–329.

[BGKS20] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. "DEEP-FRI: Sampling Outside the Box Improves Soundness". In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*. ITCS '20. 2020, 5:1–5:32.

[BMMS26a] Anubhav Baweja, Pratyush Mishra, Tushar Mopuri, and Matan Shtepel. "FICS and FACS: Fast IOPPs and Accumulation via Code-Switching". In: *Proceedings of the 46th Annual International Cryptology Conference*. CRYPTO '26. 2026, pp. 557–591.

[BMMS26b] Anubhav Baweja, Pratyush Mishra, Tushar Mopuri, and Matan Shtepel. "Query-Optimal IOPPs for Linear-Time Encodable Codes". In: *Proceedings of the 45th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '26. 2026, pp. 95–120.

[BMNW25] Benedikt Bünz, Pratyush Mishra, Wilson Nguyen, and William Wang. "Arc: Accumulation for Reed–Solomon Codes". In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO '25. 2025, pp. 128–160.

[BRW26] Benedikt Bünz, Ron Rothblum, and William Wang. *Flock: Fast Proving for Batch Boolean Computations*. Cryptology ePrint Archive, Paper 2026/1329. 2026. URL: https://eprint.iacr.org/2026/1329.

[Ber] Daniel J. Bernstein. *The transposition principle*. URL: https://cr.yp.to/transposition.html.

[Blo26] Remco Bloemen. *ProveKit: Privacy for the Real World*. https://world.org/blog/engineering/provekit-privacy-for-the-real-world. 2026.

[Bor57] Jan L. Bordewijk. "Inter-reciprocity applied to electrical networks". In: *Applied Scientific Research, Section B* 6.1 (1957), pp. 1–74.

[CCHLRR18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. *Fiat–Shamir From Simpler Assumptions*. Cryptology ePrint Archive, Report 2018/1004. 2018.

[CFS17] Alessandro Chiesa, Michael A. Forbes, and Nicholas Spooner. *A Zero Knowledge Sumcheck and its Applications*. https://arxiv.org/abs/1704.02086. arXiv:1704.02086 [cs.CC]. 2017. DOI: 10.48550/arXiv.1704.02086.

[CH26] Matteo Campanelli and Mathias Hall-Andersen. "General Techniques for Building SNARKs over the Integers". In: *Proceedings of the 29th International Conference on Practice and Theory of Public-Key Cryptography*. PKC '26. 2026, pp. 197–230.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. "Succinct Arguments in the Quantum Random Oracle Model". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019, pp. 1–29.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 769–793.

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. "Verifying computations with streaming interactive proofs". In: *Proceedings of the VLDB Endowment* 5.1 (2011), pp. 25–36.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024. URL: https://github.com/hash-based-snargs-book.

[Coh91] Aviad Cohen. "Disperser Graphs, Deterministic Amplification and Imperfect Random Sources". PhD Thesis, PDF available online at https://www.math.ias.edu/~avi/STUDENTS/acthesis.pdf. PhD thesis. Hebrew University of Jerusalem, 1991.

[DHRR26] Rahul Dalal, Tamir Hemo, Eugene Rabinovich, and Ron Rothblum. *VEIL: Lightweight Zero-Knowledge for Hash-Based Multilinear Proof Systems*. Cryptology ePrint Archive, Report 2026/683. 2026.

[DL25] Hugo Delavenne and Louise Lallemand. "Codes on Any Cayley Graph Have an Interactive Oracle Proof of Proximity". In: *Proceedings of the 24th International Conference on Cryptology and Network Security*. CANS '25. 2025, pp. 335–357.

[DM11] Irit Dinur and Or Meir. "Derandomized Parallel Repetition via Structured PCPs". In: *Computational Complexity* 20.2 (2011), pp. 207–327. DOI: 10.1007/s00037-011-0013-5. URL: https://link.springer.com/article/10.1007/s00037-011-0013-5.

[DMR25] Hugo Delavenne, Tanguy Medevielle, and Élina Roussel. "Interactive Oracle Proofs of Proximity to Codes on Graphs". In: *2025 IEEE International Symposium on Information Theory*. ISIT '25. 2025, pp. 1–6.

[DP25] Benjamin Diamond and Jim Posen. "Succinct Arguments over Towers of Binary Fields". In: *Proceedings of the 44th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '25. 2025.

[DR04] Irit Dinur and Omer Reingold. "Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem". In: *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '04. 2004, pp. 155–164.

[Dia25] Benjamin Diamond. *Zero-Knowledge Polynomial Commitment in Binary Fields*. Cryptology ePrint Archive, Report 2025/1015. 2025.

[Din07] Irit Dinur. "The PCP theorem by gap amplification". In: *Journal of the ACM* 54.3 (2007), p. 12.

[Fs26] Matteo Frigo and abhi shelat. "Anonymous Credentials from ECDSA". In: *IACR Communications in Cryptology* 3.1 (2026). DOI: 10.62056/a3qjmpgxq.

[GCXK26] Yiwen Gao, Dongliang Cai, Yang Xu, and Haibin Kan. "From List-Decodability to Proximity Gaps". In: *IEEE Transactions on Information Theory* 72.6 (2026), pp. 3809–3819.

[GGR11] Parikshit Gopalan, Venkatesan Guruswami, and Prasad Raghavendra. "List Decoding Tensor Products and Interleaved Codes". In: *SIAM Journal on Computing* 40.5 (2011), pp. 1432–1462.

[GKL24] Yiwen Gao, Haibin Kan, and Yuan Li. *Linear Proximity Gap for Linear Codes within the 1.5 Johnson Bound*. Cryptology ePrint Archive, Report 2024/1810. 2024.

[GLSTW23] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. "Brakedown: Linear-time and field-agnostic SNARKs for R1CS". In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO '23. 2023, pp. 193–226.

[GNR26] Kobi Gurkan, Andrija Novakovic, and Ron D. Rothblum. *Bolt: Faster SNARKs from Sketched Codes*. Cryptology ePrint Archive, Paper 2026/310. 2026. URL: https://eprint.iacr.org/2026/310.

[GNSV23] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. "Rinocchio: SNARKs for Ring Arithmetic". In: *Journal of Cryptology* 36.4 (2023), p. 41. DOI: 10.1007/s00145-023-09481-3.

[GNW04] Marcus Greferath, Alexander Nechaev, and Robert Wisbauer. "Finite Quasi-Frobenius Modules and Linear Codes". In: *Journal of Algebra and its Applications* 3.3 (2004), pp. 247–272. DOI: 10.1142/S0219498804000873.

[GOS24] Tom Gur, Jack O'Connor, and Nicholas Spooner. "Perfect Zero-Knowledge PCPs for #P". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. STOC '24. 2024, pp. 1724–1730.

[GOS25] Tom Gur, Jack O'Connor, and Nicholas Spooner. "A Zero-Knowledge PCP Theorem". In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*. STOC '25. 2025, pp. 986–994.

[GRS12] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential coding theory*. Vol. 2. 1. 2012.

[GS00] Oded Goldreich and Shmuel Safra. "A Combinatorial Consistency Lemma with Application to Proving the PCP Theorem". In: *SIAM Journal on Computing* 29.4 (2000). Earlier version: FOCS 1997, pp. 1132–1154. DOI: 10.1137/S0097539797315744.

[GWVHDČK25] Albert Garreta, Hendrik Waldner, Ilia Vlasov, Katerina Hristova, Luca Dall'Ava, Marko Čupić, and Matthew Klein. "Zinc: Succinct Arguments with Small Arithmetization Overheads from IOPs of Proximity to the Integers". In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO '25. 2025, pp. 259–291. DOI: 10.1007/978-3-032-01907-3_9.

[Goo] *Opening up 'Zero-Knowledge Proof' technology to promote privacy in age assurance*. https://blog.google/technology/safety-security/opening-up-zero-knowledge-proof-technology-to-promote-privacy-in-age-assurance/. 2025.

[HK24] Ulrich Haböck and Al Kindi. *A note on adding zero-knowledge to STARK*. Cryptology ePrint Archive, Report 2024/1037. 2024.

[IKW09] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. "New Direct-Product Testers and 2-Query PCPs". In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*. Preliminary version appeared as ECCC TR08-016. 2009, pp. 131–140. DOI: 10.1145/1536414.1536436.

[ISVW13] Yuval Ishai, Amit Sahai, Michael Viderman, and Mor Weiss. "Zero Knowledge LTCs and Their Applications". In: *Proceedings of the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, and of the 17th International Workshop on Randomization and Computation*. APPROX-RANDOM '13. 2013, pp. 607–622.

[JLXYY25] Yuhao Jia, Songsong Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. "Polynomial Commitments for Galois Rings and Applications to SNARKs over $\mathbb{Z}_{2^k}$". In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO '25. 2025, pp. 515–548.

[Jo26] Sunghyeon Jo. *Interleaving Stability for Mutual Correlated Agreement and Curve Decodability*. Cryptology ePrint Archive, Paper 2026/891. 2026. URL: https://eprint.iacr.org/2026/891.

[LSS24] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. "The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy". In: *Proceedings of the 31st ACM Conference on Computer and Communications Security*. CCS '24. 2024, pp. 3125–3137. DOI: 10.1145/3658644.3690330.

[Lig] *Reshaping KYC/AML in Web3*. https://ligero-inc.com/kyc-aml. 2025.

[MZ25] Dor Minzer and Kai Zhe Zheng. "Improved Round-by-round Soundness IOPs via Reed–Muller Codes". In: *Proceedings of the 66th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '25. 2025, pp. 1286–1294.

[Mei12] Or Meir. "Combinatorial PCPs with Short Proofs". In: *Proceedings of the 27th Annual IEEE Conference on Computational Complexity*. CCC '12. 2012, pp. 345–355.

[Mei13] Or Meir. "IP = PSPACE Using Error-Correcting Codes". In: *SIAM Journal on Computing* 42.1 (2013), pp. 380–403.

[Mid] *Miden*. https://github.com/0xPolygonMiden.

[NA25] Andrija Novakovic and Guillermo Angeris. *Ligerito: A Small and Concretely Fast Polynomial Commitment Scheme*. Cryptology ePrint Archive, Report 2025/1187. 2025.

[NKM97] Alexander A. Nechaev, Alexander S. Kuz'min, and Vladimir T. Markov. "Linear Codes over Finite Rings and Modules". In: *Fundamentalnaya i Prikladnaya Matematika* 3.1 (1997), pp. 195–254. URL: https://www.mathnet.ru/eng/fpm211.

[NTZ25] Vineet Nair, Justin Thaler, and Michael Zhu. *Proving CPU Executions in Small Space*. Cryptology ePrint Archive, Report 2025/611. 2025.

[Ope26] OpenVM Contributors. *Announcing OpenVM 2.0, Powered by SWIRL*. https://blog.openvm.dev/2.0. 2026.

[Plo] *Plonky3: WHIR and its zero-knowledge variant*. https://github.com/Plonky3/Plonky3/tree/main/whir. 2026.

[Pol] *Polygon*. https://polygon.technology.

[Pro] *ProveKit: ZO0K implementation in the WHIR backend*. https://github.com/worldfnd/whir/tree/main/src/protocols/zook. 2026.

[RACCDFMW26] Roy Rinberg, Alex Allain, Alessandro Chiesa, Miranda Christ, Rahul Dalal, Giacomo Fenzi, Ben Mayer, and Guy Weissenberg. *FlashProve: an open-source library for provable inference*. Technical report. Code: https://github.com/anthropics/FlashProve. 2026. URL: https://www.anthropic.com/document/flashprove-technical-report.

[RR20] Noga Ron-Zewi and Ron Rothblum. "Local Proofs Approaching the Witness Length". In: *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. FOCS '20. 2020, pp. 846–857.

[RR22] Noga Ron-Zewi and Ron D. Rothblum. "Proving as Fast as Computing: Succinct Arguments with Constant Prover Overhead". In: *Proceedings of the 54th ACM Symposium on the Theory of Computing*. STOC '22. 2022, pp. 1353–1363.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. "Constant-Round Interactive Proofs for Delegating Computation". In: *Proceedings of the 48th ACM Symposium on the Theory of Computing*. STOC '16. 2016, pp. 49–62.

[RS60] I. S. Reed and G. Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304.

[RW24] Noga Ron-Zewi and Mor Weiss. "Zero-knowledge IOPs Approaching Witness Length". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 105–137.

[Ris] *Risc0*. https://risc0.com.

[SV22] Eduardo Soria-Vazquez. "Doubly Efficient Interactive Proofs over Infinite and Non-Commutative Rings". In: *Proceedings of the 20th Theory of Cryptography Conference*. TCC '22. 2022, pp. 497–525. DOI: 10.1007/978-3-031-22318-1_18.

[Spa] *Spartan-WHIR: A Spartan-based SNARK built on Plonky3*. https://github.com/ethereum/spartan-whir. 2026.

[Sta] *StarkNet*. https://www.starknet.io/.

[Suc] *Succinct*. https://succinct.xyz.

[Suc26] Succinct Labs. *SLOP: WHIR Polynomial Commitment Scheme*. https://github.com/succinctlabs/sp1/tree/main/slop/crates/whir. 2026.

[TUZ07] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. "Lossless Condensers, Unbalanced Expanders, and Extractors". In: *Combinatorica* 27.2 (2007), pp. 213–240. DOI: 10.1007/s00493-007-0053-2. URL: https://www.cs.utexas.edu/~diz/pubs/condenser.pdf.

[VSBW13] Victor Vu, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. "A hybrid architecture for interactive verifiable computation". In: *Proceedings of the 34th IEEE Symposium on Security and Privacy*. Oakland '13. 2013, pp. 223–237.

[VW26] Himanshu Vashishth and Mor Weiss. *Composition Theorems for Zero-Knowledge IOPs*. Cryptology ePrint Archive, Paper 2026/307. 2026. URL: https://eprint.iacr.org/2026/307.

[Woo09] Jay A. Wood. "Foundations of Linear Codes Defined over Finite Modules: The Extension Theorem and the MacWilliams Identities". In: *Codes over Rings*. Ed. by Patrick Solé. Vol. 6. Series on Coding Theory and Cryptology. World Scientific, 2009, pp. 124–190. DOI: 10.1142/9789812837691_0004.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Orion: Zero Knowledge Proof with Linear Prover Time". In: *Proceedings of the 42nd Annual International Cryptology Conference*. CRYPTO '22. 2022, pp. 299–328.

[XZZPS19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[ZCF24] Hadas Zeilberger, Binyi Chen, and Ben Fisch. "BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes". In: *Proceedings of the 44th Annual International Cryptology Conference*. Vol. 14929. CRYPTO '24. 2024, pp. 138–169.

[dS25] Thomas den Hollander and Daniel Slamanig. "A Crack in the Firmament: Restoring Soundness of the Orion Proof System and More". In: *Proceedings of the 31st International Conference on the Theory and Application of Cryptology and Information Security*. ASIACRYPT '25. 2025, pp. 396–426. DOI: 10.1007/978-981-95-5116-3_13.

[lea26] leanEthereum. *leanVM: Minimal Hash-Based zkVM for a Post-Quantum Ethereum*. https://github.com/leanEthereum/leanVM/tree/koalabear. KoalaBear implementation. 2026.