Query-Optimal IOPPs for Linear-Time Encodable Codes

Anubhav Baweja
abaweja@upenn.edu
UPenn

Pratyush Mishra
prat@upenn.edu
UPenn

Tushar Mopuri
tmopuri@upenn.edu
UPenn

Matan Shtepel
mshtepel@andrew.cmu.edu
CMU

October 6, 2025

Abstract

We present the first Interactive Oracle Proofs of Proximity (IOPPs) for linear-time encodable codes that achieve λ-bit security with *linear* prover time and *optimal* $O(\lambda)$ query complexity. This implies (via standard techniques) the first IOP for NP with $O(n)$ prover time and $O(\lambda)$ query complexity, and hence also the first SNARK for NP in the random oracle model with linear prover time and $O(\lambda^2 \log n)$ proof size.

The technical core of our result is a novel IOPP for tensor codes. Our tensor IOPP leverages error correction in a novel way to reduce checking proximity of a purported codeword to the tensor code to checking the proximity of $\Theta(\lambda)$-many of its columns to the column code. Our key insight is that it in fact suffices to just prove that a *constant fraction* of these new proximity claims hold (as opposed to all of them). We devise a new *lossy batching* protocol that provides the foregoing guarantee with just $O(\lambda)$ query complexity. By combining this tensor IOPP with prior "codeswitching" reductions, we obtain IOPPs for a large class of linear-time encodable codes.

We complement our IOPP construction with a lower bound that shows that, when proving proximity to constant-rate codes, one cannot construct IOPPs with query complexity better than $O(\lambda)$. This establishes the optimality of our IOPP's query complexity.

Contents

1 Introduction **1**
- 1.1 Contributions . . . 2
- 1.2 Related work . . . 3

2 Technical overview **6**
- 2.1 Interactive oracle reductions . . . 6
- 2.2 IORs for common relations . . . 7
- 2.3 IOPP for codeswitchable codes . . . 8
- 2.4 Warmup: InterleavedReduce IOR . . . 9
- 2.5 Using tensor codes . . . 12
- 2.6 Lossy batching . . . 15
- 2.7 Final IOPP construction . . . 18

3 Preliminaries **20**
- 3.1 Notation . . . 20
- 3.2 Coding theory . . . 21
- 3.3 Mutual correlated agreement for multilinear proximity generators . . . 22
- 3.4 Codeswitchable codes . . . 23

4 Interactive oracle reductions **25**
- 4.1 Relation prerequisites . . . 25
- 4.2 Definition . . . 25
- 4.3 Important indexed oracle pair relations . . . 27
- 4.4 Round-by-round tree-extractability . . . 29

5 Lossy batching **32**
- 5.1 The LossyBatch IOR . . . 32

6 The TensorReduce IOR **37**
- 6.1 The SpotCheckReduce IOR . . . 40

7 Prover and query optimal IOPP for tensor codes **45**
- 7.1 Preliminaries . . . 45
- 7.2 Selecting the codes . . . 46
- 7.3 The FinishingIOPP . . . 46
- 7.4 The funneling lemma . . . 47
- 7.5 Construction of TensorIOPP using TensorReduce and FinishingIOPP . . . 48
- 7.6 IOPPs for codeswitchable codes . . . 48

8 Query complexity lower bound for IOPPs **49**

Acknowledgements **51**

References **51**

A Mutual correlated agreement for multilinear proximity generators **53**
- A.1 Extending to more than two input words . . . 54
- A.2 Mutual correlated agreement preserves list decoding . . . 55

B Deferred proofs **57**
B.1 Proof of Lemma 5.3 57
B.2 Proof of Lemma 6.2 59
B.3 Proof of Lemma 6.4 63
B.4 Proof of Lemma 7.5 66

1 Introduction

Succinct Non-interactive ARguments of Knowledge (SNARKs) are short cryptographic proofs that can be verified quickly. They are a key tool in many applications, including the design of concretely efficient decentralized and distributed systems. Successful deployment of SNARKs in these applications necessitates constructions which are efficient in all metrics: prover time, verifier time, and proof size.

SNARKs from IOPs. A standard and popular approach for constructing SNARKs has been to apply (a variant of) the BCS transformation [BCS16] to information-theoretic *Interactive Oracle Proofs* (IOPs). SNARKs constructed in this manner enjoy numerous attractive benefits, including plausible post-quantum security, transparent setup, and avoidance of public-key cryptography, which has made them an attractive target for deployment in applications. It is thus an important research direction to improve the efficiency of these SNARKs on all the aforementioned metrics.

Let us examine the factors that determine the efficiency of IOP-based SNARKs. Let the maximum size of oracles sent by the IOP prover be n, the prover time of the IOP be $T_{\mathcal{P}} = \Omega(n)$, the verifier time be $T_{\mathcal{V}}$, and the query complexity be q. When aiming for λ-bit security, the prover time of the SNARK is $T_{\mathcal{P}}$ plus $O(n)$ hashes, the verifier time is $T_{\mathcal{V}}$ plus $O(q \log n)$ hashes,[1] and the proof size is $O(\lambda \cdot q \cdot \log n)$ bits. Improving these SNARK efficiency parameters thus boils down to improving IOP prover time and query complexity.

IOPPs. In almost all known IOP constructions, these parameters are determined by the efficiency of a particular underlying component called an Interactive Oracle Proof *of Proximity* (IOPP) for a linear code $\mathcal{C}$. IOPs use IOPPs to prove that the prover's oracles are close to a codeword in $\mathcal{C}$, (and sometimes that the underlying message satisfies some additional condition). The prover time of such IOPs is determined by the encoding time of $\mathcal{C}$ and the IOPP's prover time, while query complexity is determined almost entirely by that of the IOPP. Thus, to improve IOP efficiency (and hence SNARK efficiency), it is crucial to construct IOPPs that (a) support linear-time encodable codes; and (b) have efficient prover time and low query complexity. We provide a high-level recap of IOPP constructions, and identify a key open question.

- **IOPPs for polynomial codes.** FRI [BBHR18] is an IOPP for Reed–Solomon codes with $O(\lambda \log n)$ query complexity. Subsequent works [BGKS20; BCIKS23; ACFY24; ACFY25] have improved upon FRI by reducing query complexity, culminating in the recent work of Minzer and Zheng [MZ25] that constructs an IOPP for trivariate Reed–Muller codes with $O(\lambda + \lambda^2 \log\log n / \log^2 n)$ query complexity (which is $O(\lambda)$ for a range of reasonable security parameters).

- **IOPPs for linear-time encodable codes.** Unfortunately, both Reed–Solomon and Reed–Muller codes have quasilinear encoding time, and hence the IOP provers which use these require $O(n \log n)$ time. Note that this is the case even when IOPP prover time is linear, since the prover must at least write down the codeword. To overcome this, recent work [BCG20; BCL22; GLSTW23; BCFRRZ25; NST24; BMMS25] has constructed IOPPs for *linear-time* encodable codes with *linear-time* provers. However, these IOPPs have worse query complexity: the state-of-the-art [BMMS25] requires $\Omega(\lambda \log\log n + \log n)$ queries.

This gap suggests the following open question:

Are there IOPPs for linear-time encodable codes that achieve linear prover time and $O(\lambda)$ *query complexity?*

[1] With the exception of STIR [ACFY24], the verifier time of most IOPs is bounded by $O(q \log n)$, and so the corresponding SNARK's verifier time is also upper-bounded by $O(q \log n)$.

1.1 Contributions

In this paper, we answer this question affirmatively by constructing an IOPP for all *codeswitchable* codes [RR24; BMMS25] with $O(n)$ prover time and $O(\lambda)$ query complexity, when the security parameter λ is in the range $[\log n, n^{1/3})$. This class of codes includes several linear-time encodable codes (e.g., RAA codes [DJM98; BCFRRZ25] and the Brakedown code [GLSTW23]) that have been used in the SNARK literature.[2] When used to construct an IOP, our IOPP yields the first (to the best of our knowledge) IOP for NP with $O(n)$ prover time and $O(\lambda)$ query complexity.

Furthermore, we show that this query complexity is optimal for IOPPs for codes with constant rate, thus answering a question raised by Minzer and Zheng [MZ25]. We detail our contributions below.

(1) IOPP for codeswitchable codes. We construct an IOPP for *multilinear evaluation* [BCFRRZ25]. In such IOPPs, the prover convinces a verifier with oracle access to a purported codeword $\boldsymbol{w} \in \mathbb{F}^n$ that it knows a message $\boldsymbol{m} \in \mathbb{F}^k$ such that for an evaluation point $\boldsymbol{z} \in \mathbb{F}^{\log k}$ and claimed evaluation $y \in \mathbb{F}$ (both known to the verifier), the following claims hold:

$$\delta(\boldsymbol{w}, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})) < \delta_{\mathcal{C}}/2 \quad \text{and} \quad \hat{m}(\boldsymbol{z}) = y \ ,$$

where $\hat{m}(\boldsymbol{X})$ is the multilinear extension of $\boldsymbol{m}$ and $\delta_{\mathcal{C}}$ is the relative distance of the code. For brevity, in this work we refer to IOPPs for multilinear evaluation as IOPPs, and defer to Section 1.2 a discussion of other notions of IOPPs used in the literature.

Our main contribution is the following IOPP for any codeswitchable code.

Theorem 1.1 (informal, Theorem 7.7)**.** *Let λ be a security parameter, and let $\mathcal{C}$ be a codeswitchable code over a field $\mathbb{F}$ of size $\Omega(2^{\lambda})$ with block length n. Then there exists an IOPP for $\mathcal{C}$ with $O(n)$ prover time, $O(\lambda + \log n)$ query complexity, $O(\lambda \log n)$ verifier time, alphabet $\mathbb{F}$, and soundness error* $\mathrm{negl}(\lambda)$.

Our IOPP, and hence IOPs constructed from it, enjoy the following additional desirable properties. First, our IOPP does not impose any special conditions on the field $\mathbb{F}$ beyond its size. That is, it is *field-agnostic*, which is desirable in applications which impose their own restrictions on the field.[3] Next, our IOPP is *round-by-round tree-extractable* [BMMS25], which means that it can be used to construct SNARKs in the ROM with rewinding knowledge soundness.[4]

The IOPP in Theorem 1.1 uses a prior codeswitching reduction [BMMS25] to reduce the claim about proximity to the codeswitchable code $\mathcal{C}$ to a claim about proximity to linear-time encodable *tensor* code $\mathcal{C}'$ without adding any asymptotic overheads. To complete the construction, we need an IOPP for $\mathcal{C}'$ with the same efficiency parameters as in Theorem 1.1. We detail this IOPP next.

(2) IOPP for tensor codes. Our main technical contribution is an IOPP for a linear-time encodable tensor code $\mathcal{C}'$ that achieves the same efficiency parameters as those mentioned in Theorem 1.1. We defer a detailed technical discussion of our IOPP to Theorem 7.6, but present below a novel underlying technique that we call *lossy batching*.

[2] In more detail, it has been shown that all tensor codes and systematic LDPC codes are codeswitchable. We expect that other codes of interest such as Reed–Solomon, Reed–Muller, Basefold's code [ZCF24] and Expand-Accumulate codes [BFKTWZ24] can also be shown to be codeswitchable.

[3] For example, Block et al. [BFKTWZ24] note that the circuit that verifies an ECDSA signature on the `secp256k1` curve (popular in blockchains) is $\approx 25\times$ smaller when expressed over the native base field of the curve than over a FFT-friendly field.

[4] Achieving a stronger notion of *straightline* knowledge soundness for the SNARK would require showing that our IOPP satisfies a similar straightline notion of round-by-round extractability [BCFW25]. However, our efforts in this direction did not succeed, and it seems likely that such a result would require new ideas.

(3) Lossy batching. Many IOPPs [AHIV17; GLSTW23; BCFRRZ25; BMMS25], including ours, contain a step where the prover demonstrates the proximity of ℓ input codewords to a code by reducing these claims to a proximity claim about a single *output* codeword. Unfortunately, existing batching techniques require $O(\lambda)$ queries to each of the ℓ input codewords, which is too expensive in our construction where $\ell = \Theta(\lambda)$.

We overcome this limitation via the following observation: in our construction, it actually suffices to show proximity for a large (constant) fraction of the ℓ input codewords, as opposed to showing proximity for *all* of them. We leverage this observation by developing an interactive reduction that shows how to prove this relaxed claim with just $O(1)$ queries to each of the $\ell = \Theta(\lambda)$ input codewords. To the best of our knowledge this observation and reduction are novel and may be of independent interest. We present an overview of this reduction in Section 2.6.

(4) Query complexity lower bound. Minzer and Zheng [MZ25] conjectured that any linear-length IOP based on error correcting codes must have $O(\lambda)$ query complexity, and this could go down to $O(\lambda/\log n)$ for polynomial-length IOPs. Since all known constructions of IOPs are obtained via IOPPs, we confirm this by proving the following lower bound regarding IOPPs for error correcting codes.

Theorem 1.2 (informal, Theorem 8.3)**.** *Let* $\mathsf{IOPP} = (\mathcal{P}, \mathcal{V})$ *be a perfectly complete IOPP for a linear code* $\mathcal{C}$ *with block length* n *and message length* k*, that has soundness error at most* $2^{-\lambda}$*. Then,* $\mathcal{V}$ *must make at least* $\Omega(\lambda/\log(n/k))$ *queries to the claimed codeword* $[\![\boldsymbol{w}]\!]$*.*

Our lower bound indicates that to obtain IOPs with $O(n)$ prover time and $o(\lambda)$ query complexity, we will have to depart from the approach of proving proximity to error correcting codes.

1.2 Related work

We provide a brief overview of prior work that improves the prover or query complexity of IOPPs, and compare the efficiency parameters of their IOPPs with ours in Table 1.

There are three main notions of IOPPs, all of which can be used to construct IOPs for NP. In the first notion considered in the literature, referred to simply as IOPPs, a prover simply convinces a verifier with oracle access to some $\boldsymbol{w} \in \mathbb{F}^n$ that it knows $\boldsymbol{m} \in \mathbb{F}^k$ such that the proximity claim $\delta(\boldsymbol{w}, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})) < \delta_{\mathcal{C}}/2$ holds. Such IOPPs are exclusively considered for polynomial codes, with the polynomial structure of the code allowing the seminal sumcheck protocol [LFKN92] to be used to reduce checks about membership in an NP language to checking proximity of a purported encoding of the witness to the polynomial code [ACFY24; MZ25].

Since none of the known polynomial codes are linear-time encodable, subsequent work has shown that IOPPs for non-polynomial codes can also be used to construct IOPs for NP [BMMS25, Lemma 9.4]. However, because such codes do not intrinsically have any polynomial structure, a polynomial-based claim must be enforced over $[\![\boldsymbol{w}]\!]$ in addition to the proximity claim to the code. One instantiation of this are IOPPs for multilinear evaluation, which are the focus of this work.

A third line of work, [ACFY25], presents IOPPs for constrained Reed–Solomon codes, where the prover convinces the verifier with oracle access to $[\![\boldsymbol{w}]\!]$ that it knows $\boldsymbol{m} \in \mathbb{F}^k$ such that both the proximity claim $\delta(\boldsymbol{w}, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})) < \delta_{\mathcal{C}}/2$ and the following sumcheck-based claim holds:

$$\sum_{\boldsymbol{b} \in \{0,1\}^{\log k}} \hat{f}(\hat{m}(\boldsymbol{b}), \boldsymbol{b}) = 0 \ ,$$

for some weight polynomial $\hat{f}(\boldsymbol{X}) \in \mathbb{F}[Z, X_1, X_2, \ldots, X_{\log k}]$. Such claims capture multilinear evaluation, but can also express a rich class of other constraints including inner product claims.

FRI and BaseFold. Ben-Sasson et al. [BBHR18] initiated the study of concretely-efficient IOPPs by designing FRI, an IOPP for Reed–Solomon (RS) codes that achieves linear prover time, $O(\lambda \log n)$ verifier time, and $O(\lambda \log n)$ query complexity. BaseFold [ZCF24] generalized FRI to work with a larger class of codes called *foldable* codes, and showed how to construct an IOPP for multilinear evaluation on top of the base protocol.

STIR and WHIR. STIR [ACFY24] is an IOPP for Reed–Solomon codes that improves upon FRI by reducing the verifier's query complexity to $O(\lambda \log \log n + \log n)$ query complexity, but compromises on IOPP prover time, which is now $O(n \log n)$. WHIR [ACFY25] improves upon STIR in two ways: it incorporates support for sumcheck-like constraints on the encoded polynomial, which includes multilinear evaluation queries, and it reduces the IOPP verifier time by eliminating an additive $O(\lambda^2)$ term.

IOPPs for Reed–Muller codes. The recent work of [MZ25] constructs an IOPP for Reed–Muller codes with $O(n)$ prover time, $O\left(\lambda + \frac{\lambda^2 \log \log n}{\log^2 n}\right)$ query complexity and $O(\lambda \log n)$ verifier time. This IOPP is based on line-versus-point tests in the low-soundness regime and only requires $O(\log \log n)$ rounds of interaction. We note that this IOPP is query-optimal only in the regime where $\lambda = O(\log^2 n / \log \log n)$, as opposed to our IOPP which is query-optimal as long as $\lambda = O(n^{1-\gamma})$, for some constant $\gamma > 2/3$. The latter is (much) more easily satisfied by standard settings of λ and n.

Furthermore, their work focuses on proving that their IOPP is round-by-round *sound*, as opposed to *knowledge-sound*, since for efficiently decodable codes like Reed–Muller and Reed–Solomon, knowledge-soundness follows almost trivially. However, this is not true for several linear-time encodable codes, which are not known to decodable in polynomial time (such as RAA and Brakedown). Thus, our work takes care in showing explicitly that our IOPP is *round-by-round tree-extractable* (RBRTE, see Section 4.4 for details).

Brakedown. For all $t \geq 1$, Brakedown [GLSTW23] construct an IOPP for a linear time encodable code with a linear-time prover and $O(n^{1/t})$ verifier time and query complexity. This IOPP makes use of interleaved codes, and requires t rounds of interaction.

Blaze and BrakingBase. Blaze [BCFRRZ25] and BrakingBase [NST24] constructed the first IOPPs for a linear-time encodable code that achieve polylogarithmic verifier complexity. Unlike prior works [BCG20; BCL22; RR24; RR24], they do so without PCP composition. At a high level, both IOPPs reduce the size of the claim to $n/\log n$ using $O(\lambda \log n)$ queries, and then use codeswitching techniques to switch to using existing IOPPs for quasi-linear-time encodable codes.

FICS. [BMMS25] construct an IOPP, FICS, for all codeswitchable codes that follows the same template as [BCFRRZ25; NST24], but improves the query complexity to $O(\lambda \log \log n + \log n)$ by using codeswitching techniques to reduce the query complexity of reducing the size of the claim from n to $n/\log n$.

Tensor IOPPs. In contrast to our work which is in the negligible soundness regime, previous work that has constructed IOPs for NP in the constant soundness regime has also (implicitly) constructed IOPPs for tensor codes with constant soundness error. In particular, [BCG20] constructs a linear-time prover t-round IOPP for t-wise tensor codes such that the verifier time and query complexity are $\Omega(n^{1/t})$ (see [BCG20, Lemma 10.1]). The subsequent work of [BCL22] improved the query complexity to $O(\log n)$ (using proof composition), but the verifier time remains $\Omega(n^{1/t})$.

Another line of work by Ron-Zewi and Rothblum [RR24] also (implicitly) constructs constant query IOPPs for tensor codes, but they achieve neither polylogarithmic verifier time nor linear prover time. However, the focus of this work was minimizing the length of the oracles sent, and not the foregoing efficiency parameters. In subsequent work, [RR25] focus on prover time, and present an IOPP for tensor codes with linear prover time and $\mathrm{polylog}(n)$ query complexity.

IOPP	code	prover time	oracle queries	verifier time
FRI [BBHR18]	RS	$O(n)$	$O(\lambda \log n)$	$O(\lambda \log n)$
BaseFold [ZCF24]	Foldable	$O(n)$	$O(\lambda \log n)$	$O(\lambda \log n)$
Brakedown [GLSTW23]	Brakedown	$O(n)$	$O(\lambda\sqrt{n})$	$O(\lambda\sqrt{n})$
Blaze [BCFRRZ25]	RAA	$O(n)$	$O(\lambda \log n)$	$O(\lambda \log n)$
BrakingBase [NST24]	Brakedown	$O(n)$	$O(\lambda \log n)$	$O(\lambda \log n)$
STIR [ACFY24]	RS	$O(n \log n)$	$O(\lambda \log \log n + \log n)$	$O(\lambda \log n + \lambda^2)$
WHIR [ACFY25]	RS	$O(n \log n)$	$O(\lambda \log \log n + \log n)$	$O(\lambda \log n)$
FICS [BMMS25]	CS	$O(n)$	$O(\lambda \log \log n + \log n)$	$O(\lambda \log n \log \log n)$
[MZ25]	RM	$O(n)$	$O\left(\lambda + \frac{\lambda^2 \log \log n}{\log^2 n}\right)$	$O(\lambda \log n)$
Our IOPP	CS	$O(n)$	$O(\lambda + \log n)$	$O(\lambda \log n)$

Table 1: Comparison with prior IOPPs. CS denotes the class of efficiently codeswitchable codes.

The techniques used in the constant soundness regime differ significantly from ours; they do not make use of the generator matrix of their codes, and often employ proof composition.

2 Technical overview

We begin by recapping the necessary background on interactive oracle reductions (IORs) in Section 2.1 and some useful tools in Section 2.2. Next, in Section 2.3, we describe the high-level architecture of our IOPP for codeswitchable codes. We then describe the two key components of this architecture: the TensorReduce IOR (Sections 2.4 to 2.6), and how it is used to obtain our IOPP (Section 2.7). We refer the reader to Section 3 for an overview of our notation and the properties of linear codes that we use in this work.

2.1 Interactive oracle reductions

We construct our IOPP by composing a sequence of *interactive oracle reductions (IORs)* [BCGGRS19; BMNW25; BMMS25; BCFW25]. The latter are interactive protocols that allow a prover to reduce the task of convincing a verifier that an instance is in a language $\mathcal{L}(\mathcal{R})$ to convincing the verifier that a related instance is in $\mathcal{L}(\mathcal{R}')$ for some other relation $\mathcal{R}'$. In this paper, we will specifically work with IORs for *oracle relations*, which we define next.

An **oracle relation** is a set of triples of the form $(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w})$, where $\mathbb{x}$ is the explicit instance, $\vec{\mathbb{y}}$ is an implicit instance specified as a set of oracles, and $\mathbb{w}$ is the witness. [5] For brevity, in the rest of the technical overview we refer to oracle relations simply as relations.

An **interactive oracle reduction** from a relation $\mathcal{R} \coloneqq \{(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w})\}$ to another relation $\mathcal{R}' \coloneqq \{(\mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}')\}$ is a k-round interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$. Upon input $(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}$, in each round $i \in [\mathsf{k}]$, $\mathcal{P}$ sends a plain-text message msg_i and an oracle $\vec{\Pi}_i$ to $\mathcal{V}$, and $\mathcal{V}$ replies with public-coin randomness $\vec{r}_i$. In the last round, $\mathcal{V}$ queries the oracles $\vec{\mathbb{y}}$ and $[\vec{\Pi}_i]_{i\in[\mathsf{k}]}$ at certain points, and chooses to either accept or reject. If $\mathcal{V}$ accepts, $\mathcal{P}$ outputs a new witness $\mathbb{w}'$ and $\mathcal{V}$ outputs a new instance $(\mathbb{x}', \mathbb{y}')$, where the contents of $\mathbb{y}'$ are selected from the input oracle strings $\vec{\mathbb{y}}$ and the oracles $[\vec{\Pi}_i]_{i\in[\mathsf{k}]}$ sent by $\mathcal{P}$ during the interaction. We say $(\mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}')$ is the output of the IOR. See Section 4.2 for a formal definition of IORs.

Completeness for IORs is defined naturally: if $\mathcal{P}$ is an honest prover with a valid witness $\mathbb{w}$, the output of the IOR's is in $\mathcal{R}'$. Knowledge soundness for IORs is more involved. In this work, we prove that our IORs satisfy *round-by-round tree-extractability* (RBRTE) [BMMS25], a strong notion that enables proving knowledge-soundness guarantees for the non-interactive reduction obtained by applying (a variant of) the BCS transformation [BCS16; BMNW25] to the IOR.

For the purposes of the technical overview, we will only discuss *soundness* of our IORs, since RBRTE is a delicate notion that requires a lot of detail to show rigorously. Soundness for IORs is defined naturally: if the input tuple $(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \notin \mathcal{R}$ then the probability that the output tuple $(\mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathcal{R}'$, and $\mathcal{V}$ does not reject, is at most $\epsilon > 0$, where ϵ is the *soundness error* of the IOR.

Remark 2.1 (IOPPs)**.** An interactive oracle proof of proximity (IOPP) for a relation $\mathcal{R}$ can be seen as an IOR from $\mathcal{R}$ to the trivial relation $\mathcal{R}_{\mathsf{true}} \coloneqq \{(\mathbb{x} \coloneqq \mathsf{accept}, \vec{\mathbb{y}} \coloneqq \bot, \mathbb{w} \coloneqq \bot)\}$, consisting of only one tuple with an empty witness.

Composition of IORs. Given IORs IOR_1 from $\mathcal{R}_1$ to $\mathcal{R}_2$ and IOR_2 from $\mathcal{R}_2$ to $\mathcal{R}_3$, one can obtain an IOR from $\mathcal{R}_1$ to $\mathcal{R}_3$ by composing these IORs as follows: first, use IOR_1 to reduce the claim "$(\mathbb{x}_1, \vec{\mathbb{y}}_1) \in \mathcal{L}(\mathcal{R}_1)$" to the claim "$(\mathbb{x}_2, \vec{\mathbb{y}}_2) \in \mathcal{L}(\mathcal{R}_2)$", and then use IOR_2 to reduce this latter claim to the final claim "$(\mathbb{x}_3, \vec{\mathbb{y}}_3) \in \mathcal{L}(\mathcal{R}_3)$". [BMMS25] show that RBRTE IORs are well-behaved under composition.

[5] In the technical sections we use *indexed oracle relations*, whose members additionally contain an *index* $\mathbb{i}$. For simplicity, in the technical overview we will deal only with 'plain' oracle relations, since for any indexed oracle relation $\mathcal{R}$, a fixed index $\mathbb{i}$ defines a plain oracle relation $\mathcal{R}^{\mathbb{i}} = \{(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \mid (\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}\}$.

2.2 IORs for common relations

State-of-the-art work on IOPPs [ZCF24; ACFY25; BCFRRZ25; NST24; BMMS25] leverages error-correcting codes and the sumcheck protocol [LFKN92] to obtain efficient constructions. FICS [BMMS25] distills a framework that formalizes this methodology by casting these IOPPs as a composition of interactive oracle reductions (IORs) between certain 'code-based' oracle relations. Specifically, we use three oracle relations based on codes introduced in FICS, as well as IORs between these relations.

In the rest of this section, we will use $\mathcal{C}$ to denote a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, i.e. $\mathcal{C}$ is a linear code over $\mathbb{F}$ with block length n, message length k, and (constant) relative distance $\delta_{\mathcal{C}}$. Additionally, we use $\boldsymbol{a} \sim_{\varepsilon} \boldsymbol{b}$ to denote that $\delta(\boldsymbol{a}, \boldsymbol{b}) < \varepsilon$, where $\delta(\cdot, \cdot)$ denotes the fractional Hamming distance between the two input vectors.

The SEP relation. The first relation, $\mathsf{SEP}^{(\mathcal{C},\varepsilon,\ell)}$, asserts that the instance oracles are ε-close to codewords of $\mathcal{C}$, and that the underlying messages satisfy a sumcheck-like constraint.

Definition 2.2 (informal)**.** The *sumprod-evaluation proximity (SEP)* relation $\mathsf{SEP}^{(\mathcal{C},\varepsilon,\ell)}$ consists of tuples of the form

- $\mathbb{x} = (f(\boldsymbol{X}, \boldsymbol{Z}), \sigma)$, where $f(X_1, \ldots, X_{\log k}, Z_1, \ldots, Z_\ell) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_{\log k + \ell}]$ and $\sigma \in \mathbb{F}$,
- $\vec{\mathbb{y}} = [\boldsymbol{w}_i]_{i \in [\ell]}$, where, for each $i \in [\ell]$, $\boldsymbol{w}_i \in \mathbb{F}^n$, and
- $\mathbb{w} = [\boldsymbol{m}_i]_{i \in [\ell]}$, where, for each $i \in [\ell]$, $\boldsymbol{m}_i \in \mathbb{F}^k$,

such that for each $i \in [\ell]$, the word $\boldsymbol{w}_i$ is ε-close to the encoding of $\boldsymbol{m}_i$ (i.e., $\boldsymbol{w}_i \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i)$), and the multilinear extensions[6] $\hat{m}_1, \ldots, \hat{m}_\ell$ of the messages $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$ satisfy the following summation claim:

$$\sum_{\boldsymbol{b} \in \{0,1\}^{\log k}} f(\boldsymbol{b}, \hat{m}_1(\boldsymbol{b}), \ldots, \hat{m}_\ell(\boldsymbol{b})) = \sigma \ .$$

SEP constraints are expressive; for example, they can be used to encode inner product claims over vectors. We illustrate this with an example: consider an $\mathsf{SEP}^{(\mathcal{C},\varepsilon,2)}$ claim with $\mathbb{x} = (f(\boldsymbol{X}, \boldsymbol{Z}_1, \boldsymbol{Z}_2), y), \vec{\mathbb{y}} = (\boldsymbol{w}_1, \boldsymbol{w}_2), \mathbb{w} = (\boldsymbol{m}_1, \boldsymbol{m}_2)$ such that $f(\boldsymbol{X}, Z_1, Z_2) \coloneqq Z_1 \cdot Z_2$. Such an SEP instance enforces that

$$\sum_{\boldsymbol{b} \in \{0,1\}^{\log k}} \hat{m}_1(\boldsymbol{b}) \cdot \hat{m}_2(\boldsymbol{b}) = y \iff \langle \boldsymbol{m}_1, \boldsymbol{m}_2 \rangle = y \ .$$

The MEP relation. The next relation, $\mathsf{MEP}^{(\mathcal{C},\varepsilon,\ell)}$, is similar to $\mathsf{SEP}^{(\mathcal{C},\varepsilon,\ell)}$, except that it enforces a different constraint, namely that the (multilinear extensions of the) underlying messages satisfy certain evaluation claims specified in the MEP instance.

Definition 2.3 (informal)**.** The relation $\mathsf{MEP}^{(\mathcal{C},\varepsilon,\ell)}$ consists of tuples of the form

- $\mathbb{x} = [(\boldsymbol{z}_i, y_i)]_{i \in [\ell]}$, where for each $i \in [\ell]$, $\boldsymbol{z}_i \in \mathbb{F}^{\log k}$ and $y_i \in \mathbb{F}$,
- $\vec{\mathbb{y}} = [\boldsymbol{w}_i]_{i \in [\ell]}$, where $\boldsymbol{w}_i \in \mathbb{F}^n$ for each $i \in [\ell]$,
- $\mathbb{w} = [\boldsymbol{m}_i]_{i \in [\ell]}$, where $\boldsymbol{m}_i \in \mathbb{F}^k$ for each $i \in [\ell]$,

such that for each $i \in [\ell]$, the word $\boldsymbol{w}_i$ is ε-close to the encoding of $\boldsymbol{m}_i$ (i.e., $\boldsymbol{w}_i \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i)$), and the multilinear extension $\hat{m}_i$ of $\boldsymbol{m}_i$ satisfies $\hat{m}_i(\boldsymbol{z}_i) = y_i$.

Our goal of constructing an IOPP for multilinear evaluation (Theorem 1.1) corresponds exactly to the goal of constructing an IOPP for $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$, for some $\varepsilon \in (0, \delta_{\mathcal{C}}/2)$.

[6] The multilinear extension $\hat{v} : \mathbb{F}^n \to \mathbb{F}$ of a vector $\boldsymbol{v} \in \mathbb{F}^{2^n}$ is the unique n-variate multilinear polynomial such that $\hat{v}(\langle i - 1 \rangle) = \boldsymbol{v}[i]$ for all $i \in [2^n]$, where $\langle x \rangle$ is the binary representation of x in the most-significant-bit-first order.

The HEP relation. The ℓ evaluation claims in $\mathsf{MEP}^{(\mathcal{C},\varepsilon,\ell)}$ involve distinct evaluation points $[\boldsymbol{z}_i]_{i\in[\ell]}$. When all evaluation claims involve the same point $\boldsymbol{z}$, we instead use the *homogeneous multilinear-evaluation proximity relation*, $\mathsf{HEP}^{(\mathcal{C},\varepsilon,\ell)}$, which is otherwise identical to $\mathsf{MEP}^{(\mathcal{C},\varepsilon,\ell)}$.

Remark 2.4 (notation for relations)**.** Often we will consider $\mathsf{MEP}^{(\mathcal{C},\varepsilon,\ell)}$ with $\ell = 1$, in which case we will abuse notation and just write $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$ for convenience. We will also just use MEP when talking about $\mathsf{MEP}^{(\mathcal{C},\varepsilon,\ell)}$ inline, if $(\mathcal{C},\varepsilon,\ell)$ is clear from context. We use analogous notation for HEP and SEP.

Sumcheck IOR. Our most important building block is the Sumcheck IOR from $\mathsf{SEP}^{(\mathcal{C},\varepsilon,\ell)}$ to $\mathsf{HEP}^{(\mathcal{C},\varepsilon,\ell)}$. This IOR just uses the sumcheck protocol to reduce the sumcheck claim to (multiple) evaluation claims at a uniformly random evaluation point. Sumcheck has $O(\ell n)$ prover time, $O(\ell \log k)$ verifier time, and requires no queries to the input oracles.

Codeswitch IOR. Codeswitching is a powerful tool that has been used in past constructions of efficient IOPPs [RR24; RR25; BCFRRZ25; NST24; BMMS25]. Informally, codeswitching allows one to convert proximity claims with respect to one code $\mathcal{C}$ to proximity claims with respect to a (potentially) different code $\mathcal{C}'$, while also reducing any additional constraints on the input codeword to related constraints on the output codeword. [BMMS25] focus on the case where these additional constraints are multilinear evaluation constraints, and hence formalize codeswitching as an IOR from $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{C}',\varepsilon')}$. They also show that there exist efficient codeswitching IORs for a large class of codes:

Lemma 2.5 ([BMMS25, Lemma 2.5])**.** *Let $\mathcal{C}$ be either a tensor code or a low-density parity check code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, and let $\mathcal{C}'$ be an arbitrary code with parameters $(\mathbb{F}, n', k, \delta_{\mathcal{C}'})$ and encoding time $T_{\mathcal{C}'}$. Assume that both codes have constant rate and distance.*

Then, for $\varepsilon \in (\delta_{\mathcal{C}}/8, \delta_{\mathcal{C}}/4)$ and $\varepsilon' \in (\delta_{\mathcal{C}'}/8, \delta_{\mathcal{C}'}/4)$, there exists an IOR from $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{C}',\varepsilon')}$ with the following efficiency properties:

prover time	*verifier time*	*verifier queries*
$O(n + T_{\mathcal{C}'})$	$O(\lambda \log n)$	$O(\lambda)$

Definition 2.6 (informal)**.** We say a code $\mathcal{C}$ is *codeswitchable* if there exists an IOR from $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{C}',\varepsilon')}$ with the same efficiency properties and range of ε and ε' as in Lemma 2.5.

2.3 IOPP for codeswitchable codes

We detail the high-level architecture of the IOPP for multilinear-evaluation underlying Theorem 1.1. Our construction proceeds in two steps.

First, we observe that many linear-time encodable codes of interest like the RAA code [DJM98; BCFRRZ25] and the Brakedown code [GLSTW23] are LDPC [NST24; BMMS25] and thus, by Lemma 2.5, have efficient codeswitching IORs. Hence, we employ such an IOR to reduce an $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$ claim to an $\mathsf{MEP}^{(\mathcal{T},\varepsilon_{\mathcal{T}})}$ claim, where $\mathcal{T}$ is a linear-time encodable *tensor* code. This IOR requires $O(n)$ prover time and query complexity. Then, we construct an IOPP for MEP claims about $\mathcal{T}$ that requires $O(n)$ prover time and $O(\lambda + \log n)$ query complexity. Theorem 1.1 is then obtained by composition of these two protocols.

Thus, our main contribution is constructing the foregoing IOPP for tensor codes. We focus on tensor codes $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$, where $\mathcal{C}_2$ is also a tensor code, i.e., $\mathcal{C}_2 = \mathcal{B} \otimes \mathcal{B}$ for some code $\mathcal{B}$. The core building block of our IOPP, which we call $\mathsf{TensorIOPP}$, is the IOR underlying the following lemma. We call this IOR $\mathsf{TensorReduce}'$, and it reduces an MEP claim over $\mathcal{T}$ to an MEP claim over a code $\mathcal{C}'$ with a smaller message length.[7]

[7] We name it $\mathsf{TensorReduce}'$ to distinguish it from its main component $\mathsf{TensorReduce}$, which is introduced in Section 2.5.

Lemma 2.7 (informal)**.** *Fix a field $\mathbb{F}$, a message length k, and a security parameter λ. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be linear-time encodable codes having parameters $(\mathbb{F}, n_1, k_1 = O(\lambda^{-1/3}k^{1/3}), \delta_{\mathcal{C}_1})$ and $(\mathbb{F}, n_2, k_2 = O(\lambda^{1/3}k^{2/3}), \delta_{\mathcal{C}_2})$ respectively, such that both $\mathcal{C}_1$ and $\mathcal{C}_2$ have constant rate and relative distance. Additionally, let $\mathcal{C}_2$ be the two-dimensional tensor of a linear code $\mathcal{B}$ (i.e., $\mathcal{C}_2 = \mathcal{B} \otimes \mathcal{B}$).*

Let $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ be a tensor code having parameters $(\mathbb{F}, n = n_1 n_2, k = k_1 k_2, \delta_{\mathcal{T}} = \delta_{\mathcal{C}_1}\delta_{\mathcal{C}_2})$, and let $\mathcal{C}'$ be an arbitrary linear-time encodable code with parameters $(\mathbb{F}, n', k' = k_2, \delta_{\mathcal{C}'})$. For reasonable constants $\varepsilon \in (0, \delta_{\mathcal{T}}/2), \varepsilon' \in (0, \delta_{\mathcal{C}'}/2)$, there exists an IOR $\mathsf{TensorReduce}'$ *from* $\mathsf{MEP}^{(\mathcal{T},\varepsilon)}$ *to* $\mathsf{MEP}^{(\mathcal{C}',\varepsilon')}$ *with the following efficiency parameters:*

prover time	*verifier time*	*verifier queries*
$O(n)$	$O(\lambda \log n)$	$O(\lambda)$

Thus, $\mathsf{TensorReduce}'$ reduces message length from k to $O(\lambda^{1/3}k^{2/3})$. This is a polynomial reduction in message length if $\lambda = O(n^{1-\gamma})$ for some $\gamma > 0$ (see Section 1.2). If $\mathcal{C}'$ is a tensor code itself, $\mathsf{TensorReduce}'$ can be composed with itself. Performing this composition $O(1)$ times therefore yields an IOR where the message length of the final output is $O(n^{1/3})$. At this point, we can perform a final codeswitch to a code with an efficient IOPP (e.g., WHIR [ACFY25] for Reed–Solomon codes), and can then invoke this IOPP.[8]

Overview of following sections. In the rest of this section, we first detail the main ideas underlying $\mathsf{TensorReduce}$ (Sections 2.4 to 2.6), and then, in Section 2.7, discuss how to use $\mathsf{TensorReduce}$ to obtain $\mathsf{TensorIOPP}$. For simplicity of exposition, we provide proof sketches of the *soundness* of our IORs as opposed to the stronger notion of RBRTE.

2.4 Warmup: $\mathsf{InterleavedReduce}$ IOR

Like $\mathsf{TensorIOPP}$, many prior IOPPs for MEP can be cast as compositions of length-reducing IORs from $\mathsf{MEP}^{(\mathcal{C},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{C}',\varepsilon')}$, where the message length of $\mathcal{C}'$ is shorter than that of $\mathcal{C}$. In particular, prior work has constructed such IORs for many different input codes $\mathcal{C}$, including Reed–Solomon codes [ACFY25], foldable codes [ZCF24], interleaved codes [GLSTW23; BFKTWZ24; BCFRRZ25; NST24; BMMS25], and tensor codes [BCG20; BCL22; RR24; RR25; BMMS25].

Our $\mathsf{TensorReduce}$ IOR builds on the ideas underlying the approach for interleaved codes, which we review below. Interleaved codes can be viewed as a special case of tensor codes where one of the codes is the identity code.

Definition 2.8 (interleaved codes)**.** Let $\mathcal{B}$ be a linear code with parameters $(\mathbb{F}, n_2, k_2, \delta_{\mathcal{B}})$. Given an *interleaving parameter* $k_1 \in \mathbb{N}$, the k_1-interleaved code $\mathcal{B}^{k_1}$ is a linear code with parameters $(\mathbb{F}^{k_1}, n_2, k_2, \delta_{\mathcal{B}})$ such that for every $\boldsymbol{C} \in \mathcal{B}^{k_1} \subseteq \mathbb{F}^{k_1 \times n_2}$, every row of $\boldsymbol{C}$ is a codeword of $\mathcal{B}$. Given a message $\boldsymbol{M} \in \mathbb{F}^{k_1 \times k_2}$, $\mathsf{Enc}_{\mathcal{B}^{k_1}}(\boldsymbol{M})$ is obtained by encoding each row of $\boldsymbol{M}$ under $\mathcal{B}$. Importantly, the distance of $\mathcal{B}^{k_1}$ is the same as that of $\mathcal{B}$ because we define the interleaved code over the larger alphabet $\mathbb{F}^{k_1}$.

MEP for interleaved codes. Consider an $\mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$ claim, where $\mathcal{B}$ is a linear code as defined in Definition 2.8. This claim consists of a tuple $(\mathbb{x} = (\boldsymbol{z}, y), \vec{\mathbb{y}} = \boldsymbol{W}, \mathbb{w} = \boldsymbol{M})$, where $\boldsymbol{M} \in \mathbb{F}^{k_1 \times k_2}$. However, instead of $\boldsymbol{z} \in \mathbb{F}^{\log k_2}$ as the definition of MEP would suggest, for the purposes of interleaved codes, we let $\boldsymbol{z} \in \mathbb{F}^{\log k_1 + \log k_2}$, and impose the evaluation constraint $\hat{M}(\boldsymbol{z}) = y$, where $\hat{M}$ is the multilinear extension of the matrix $\boldsymbol{M} \in \mathbb{F}^{k_1 \times k_2}$.

[8] Note that with such small message and codeword lengths, one can even invoke quadratic-time encoding procedures for RS codes without violating the linear-time requirement. This additionally means that our overall IOPP can be field-agnostic even when invoking WHIR, as the quadratic-time algorithms do not rely on the field structure (i.e., high 2-adicity) required for efficient FFTs.

The Interleaved Reduce IOR. At a high level, InterleavedReduce follows the blueprint of Ligero [AHIV17] and Brakedown [GLSTW23]: reduce a proximity claim about the interleaved code $\mathcal{B}^{k_1}$ to a proximity claim about the base code $\mathcal{B}$, by "folding" the rows of $\boldsymbol{W}$ with respect to random coefficients. If the prover $\mathcal{P}$ proves to the verifier $\mathcal{V}$ that the folding is done correctly, then $\mathcal{V}$ is convinced that the output proximity claim about $\mathcal{B}$ implies the input proximity claim about $\mathcal{B}^{k_1}$.

More precisely, InterleavedReduce is an IOR from $\mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{B},\varepsilon')}$ where $\varepsilon, \varepsilon'$ are constants that we will fix later. An $\mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$ claim consists of tuples of the form $(\mathbb{x} = (\boldsymbol{z}, y), \vec{\mathbb{y}} = \boldsymbol{W}, \mathbb{w} = \boldsymbol{M})$, where $\boldsymbol{M} \in \mathbb{F}^{k_1 \times k_2}$ and $\boldsymbol{z} \in \mathbb{F}^{\log k_1 + \log k_2}$. $\mathcal{P}$ wants to convince $\mathcal{V}$ that:

$$\boldsymbol{W} \sim_\varepsilon \mathsf{Enc}_{\mathcal{B}^{k_1}}(\boldsymbol{M}) \quad \text{and} \quad \hat{M}(\boldsymbol{z}) = y \ .$$

Parse $\boldsymbol{z}$ as $(\boldsymbol{z}_1 \,\|\, \boldsymbol{z}_2)$, where $\boldsymbol{z}_1 \in \mathbb{F}^{\log k_1}$ and $\boldsymbol{z}_2 \in \mathbb{F}^{\log k_2}$. Define $\hat{m}'$ as the multilinear extension of $\boldsymbol{m}'$, which is the linear combination of the rows of $\boldsymbol{M}$ with respect to $[\mathrm{eq}(\langle i-1\rangle, \boldsymbol{z}_1)]_{i\in[k_1]}$. We observe that $\hat{M}(\boldsymbol{z}) = \hat{m}'(\boldsymbol{z}_2)$ by properties of multilinear extensions. Thus, if $\mathcal{P}$ convinces $\mathcal{V}$ that the $\boldsymbol{w}'$ is (close to) the correct folding of $\boldsymbol{W}$ with respect to these coefficients, then the evaluation claim of the output $\mathsf{MEP}^{(\mathcal{B},\varepsilon')}$ claim implies the evaluation claim of the input $\mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$ claim.

It remains to show how a similar implication holds for the proximity claims, for which we require a property of linear codes called *mutual correlated agreement* [Zei24; GKL24; ACFY25]. Roughly, mutual correlated agreement guarantees that if the random linear combination of some words is close to the code, then not only are the original words close to the code, but they also satisfy a strong agreement property that we specify below (see Section 3.3 for details).

Lemma 2.9 (informal, see Theorem 3.11)**.** *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ and $m \in \mathbb{N}$, with $M := 2^m$. Let $\beta > (1-\delta_{\mathcal{C}})^{1/3}$ be an* agreement parameter*. For every $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \in \mathbb{F}^n$, the following is true with overwhelming probability over $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$: if there exists a set $S \subseteq [n]$ and a codeword $\boldsymbol{c} \in \mathcal{C}$ such that $|S| \geq \beta n$ and*

$$\sum_{i\in[M]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}) \cdot \boldsymbol{w}_i[S] = \boldsymbol{c}[S] \ ,$$

then there exist $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M \in \mathcal{C}$ such that $\boldsymbol{w}_i[S] = \boldsymbol{c}_i[S]$ for all $i \in [M]$ and

$$\sum_{i\in[M]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}) \cdot \boldsymbol{c}_i = \boldsymbol{c} \ .$$

This lemma thus guarantees that if $\boldsymbol{z}_1$ is chosen uniformly at random, and if $\mathcal{P}$ proves that $\boldsymbol{w}'$ is close to the correct folding of $\boldsymbol{W}$, then $\boldsymbol{w}$ being close to $\mathsf{Enc}_{\mathcal{B}}(\boldsymbol{m}')$ implies that $\boldsymbol{W}$ is close to $\mathsf{Enc}_{\mathcal{B}^{k_1}}(\boldsymbol{M})$ with overwhelming probability. In order to use this lemma, we assume for simplicity that the evaluation point $\boldsymbol{z}$ in the InterleavedReduce IOR is equal to a value $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\log k}$ sampled uniformly at random by $\mathcal{V}$, as this can easily be realized using the sumcheck protocol. We demonstrate the high-level idea of this reduction in Fig. 1, and provide an informal description of the IOR in Fig. 2.

Input: $\mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$ claim: $(\mathbb{x} = (\boldsymbol{r}, y), \vec{\mathbb{y}} = \boldsymbol{W}, \mathbb{w} = \boldsymbol{M})$ such that $\boldsymbol{W} \sim_\varepsilon \mathsf{Enc}_{\mathcal{B}^{k_1}}(\boldsymbol{M})$ and $\hat{M}(\boldsymbol{r}) = y$.

1. $\mathcal{P}$ and $\mathcal{V}$ parse $\boldsymbol{r}$ as $(\boldsymbol{r}_1 \,\|\, \boldsymbol{r}_2) \in \mathbb{F}^{\log k_1} \times \mathbb{F}^{\log k_2}$.
2. $\mathcal{P}$ computes a linear combination of the rows of $\boldsymbol{M}$ with respect to $\mathrm{eq}(\cdot, \boldsymbol{r}_1)$:

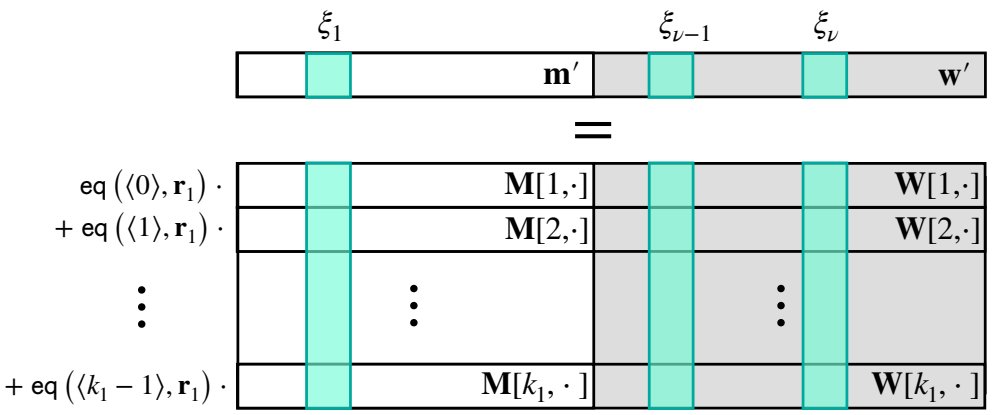

Figure 1: An illustration of the spot checks in the InterleavedReduce IOR. For clarity, assume the codes are systematic, with the white and gray indices corresponding to the systematic and non-systematic parts of the codewords respectively. The turquoise indices correspond to the indices that the verifier queries. The verifier checks that the weighted sum (corresponding to multilinear evaluation) of every turquoise column of the interleaved word $\boldsymbol{W}$ is equal to the corresponding turquoise index of the word $\boldsymbol{w}'$.

$\boldsymbol{m}' \leftarrow \sum_{i\in[k_1]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}_1) \cdot \boldsymbol{M}[i,\cdot]$.

3. $\mathcal{P}$ encodes $\boldsymbol{m}'$ under $\mathcal{B}$: $\boldsymbol{w}' \leftarrow \mathsf{Enc}_{\mathcal{B}}(\boldsymbol{m}')$, and sends $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$.
4. $\mathcal{V}$ samples $\nu := \lceil -\lambda / \log(1-\varepsilon/2) \rceil$ indices $\xi_1, \ldots, \xi_\nu \xleftarrow{\$} [n_2]$ and sends them to $\mathcal{P}$.
5. For each $j \in [\nu]$:
6. $\quad \mathcal{V}$ queries all elements of the column $\boldsymbol{W}[\cdot, \xi_j]$.
7. $\quad \mathcal{V}$ queries $\boldsymbol{w}'[\xi_j]$ and checks that

$$\boldsymbol{w}'[\xi_j] = \sum_{i\in[k_1]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}_1) \cdot \boldsymbol{W}[i, \xi_j] \ . \tag{1}$$

8. If all checks pass, output the following $\mathsf{MEP}^{(\mathcal{B},\varepsilon')}$ claim that asserts $\boldsymbol{w}' \sim_{\varepsilon'} \mathsf{Enc}_{\mathcal{B}}(\boldsymbol{m}')$ and $\hat{m}'(\boldsymbol{r}_2) = y$:

$$\mathbb{x}_{\mathsf{MEP}} := (\boldsymbol{r}_2, y), \quad \vec{\mathbb{y}}_{\mathsf{MEP}} := \boldsymbol{w}', \quad \mathbb{w}_{\mathsf{MEP}} := \boldsymbol{m}' \ .$$

Figure 2: InterleavedReduce IOR from $\mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{B},\varepsilon')}$ (informal description).

Completeness of InterleavedReduce is easy to verify, and we describe a high-level proof of soundness.

Proof of soundness. We show that if $\mathcal{V}$ accepts with probability at least $1 - 2^{-\lambda}$ over the sampled indices in Item 4, then $(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathsf{MEP}^{(\mathcal{B}^{k_1},\varepsilon)}$. Define $\boldsymbol{w}_{\mathsf{fold}} := \sum_{i\in[k_1]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}_1) \cdot \boldsymbol{W}[i,\cdot]$ as the linear combination of the rows of $\boldsymbol{W}$ with respect to $\mathrm{eq}(\cdot, \boldsymbol{r}_1)$.

1. Using spot checks: Given the checks in Eq. (1), $\mathcal{V}$ only accepts if all indices $\{\xi_j\}_{j\in[\nu]}$ sampled in Item 4 are such that $\boldsymbol{w}'[\xi_j] = \boldsymbol{w}_{\mathsf{fold}}[\xi_j]$. If $\delta(\boldsymbol{w}', \boldsymbol{w}_{\mathsf{fold}}) > \varepsilon/2$, then the probability of this event occurring (and thus $\mathcal{V}$ accepting) is at most $(1-\varepsilon/2)^\nu \le 2^{-\lambda}$. The contrapositive of this statement is that if $\mathcal{V}$

accepts with probability at least $1 - 2^{-\lambda}$, then there exists a set $S \subseteq [n_2]$ such that $\boldsymbol{w}'[S] = \boldsymbol{w}_{\mathsf{fold}}[S]$ and $|S| \geq (1 - \varepsilon/2)n_2$.

2. Using guarantee of $\mathsf{MEP}^{(\mathcal{B},\varepsilon')}$ claim: Since $(\mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathsf{MEP}^{(\mathcal{B},\varepsilon')}$, there exists a codeword $\boldsymbol{c}' \in \mathcal{B}$ such that $\boldsymbol{c}' \sim_{\varepsilon'} \boldsymbol{w}'$. By setting $\varepsilon' = \varepsilon/2$, and using that $\boldsymbol{w}'[S] = \boldsymbol{w}_{\mathsf{fold}}[S]$ and $|S| \geq (1 - \varepsilon/2)n_2$, we get that there exists an *agreement set* $A \subseteq S$ such that $\boldsymbol{c}'[A] = \boldsymbol{w}_{\mathsf{fold}}[A]$ and $|A| \geq (1 - \varepsilon)n_2$.

3. Using mutual correlated agreement: By Lemma 2.9, existence of the agreement set A, and definition of $\boldsymbol{w}_{\mathsf{fold}}$, with overwhelming probability there exist codewords $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k_1} \in \mathcal{B}$ such that $\boldsymbol{W}[i, A] = \boldsymbol{c}_i[A]$ for all $i \in [k_1]$ and $\sum_{i \in [k_1]} \mathrm{eq}(\langle i - 1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{c}_i = \boldsymbol{c}'$. Consider the message matrix $\boldsymbol{M}$ whose rows $\boldsymbol{M}[1, \cdot], \ldots, \boldsymbol{M}[k_1, \cdot]$ are the messages underlying $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k_1}$ respectively.

   (a) Proving proximity claim: Since $\boldsymbol{m}' = \sum_{i \in [k_1]} \mathrm{eq}(\langle i - 1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{M}[i, \cdot]$, and $|A| \geq (1 - \varepsilon)n_2$, we get that $\boldsymbol{W}[i, \cdot] \sim_{\varepsilon} \boldsymbol{c}_i$ for all $i \in [k_1]$, and thus $\boldsymbol{W} \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{B}^{k_1}}(\boldsymbol{M})$, proving the desired proximity claim.

   (b) Proving evaluation claim: Since $\hat{m}'(\boldsymbol{r}_2) = y$, we get that $\hat{M}(\boldsymbol{r}_1, \boldsymbol{r}_2) = y$, proving the desired evaluation claim. □

Challenge 1: InterleavedReduce is insufficient for achieving IOPPs with $O(\lambda)$ queries. One might hope that for a clever choice of k_1, by iteratively composing InterleavedReduce (and perhaps moving to the "list decoding regime") we could construct an IOPP with $O(\lambda)$ overall query complexity. However, the query complexity of the aforementioned reduction is $\Theta(\lambda k_1)$, since the verifier must query all k_1 elements of $\nu = \Theta(\lambda)$ many columns of the codeword. Thus, we must set $k_1 = O(1)$, and we have not made much progress, since the message length remains $\Theta(k)$ after applying the InterleavedReduce IOR.

To overcome this challenge, we need to avoid querying entire columns of the interleaved codeword $\boldsymbol{W}$. We do so by using tensor codes, as opposed to interleaved codes.

2.5 Using tensor codes

We now show how to use tensor codes (Definition 3.6) to overcome the foregoing challenge. The ν column checks performed by $\mathcal{V}$ in InterleavedReduce are of the following form: for each chosen column index $j \in [n_2]$, $\mathcal{V}$ queries the entire column $\boldsymbol{W}_{\mathsf{int}}[\cdot, j]$ of the interleaved codeword $\boldsymbol{W}_{\mathsf{int}}$, queries $\boldsymbol{w}'[j]$, and checks that

$$\boldsymbol{w}'[j] = \sum_{i \in [k_1]} \mathrm{eq}(\langle i - 1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{W}_{\mathsf{int}}[i, j] \ .$$

This is precisely the evaluation claim portion of an MEP claim having as witness $\boldsymbol{W}_{\mathsf{int}}[\cdot, j]$. In fact, since all ν evaluation claims have the same evaluation point $\boldsymbol{r}_1$, they can be viewed as the ν evaluation claims of a single HEP claim.

Thus, if the columns of $\boldsymbol{W}_{\mathsf{int}}$ were encoded with a linear code (as required by HEP), then instead of querying entire columns, $\mathcal{V}$ could output an HEP claim about these columns. To complete the IOPP, $\mathcal{P}$ and $\mathcal{V}$ could then run an IOPP for HEP to check this latter claim.

Switching to tensor codes. One can satisfy this requirement of encoding the columns under some code $\mathcal{C}_1$ by switching to encoding the original message $\boldsymbol{m}$ under a *tensor code*. Concretely, by encoding the columns of the interleaved codeword $\boldsymbol{W}_{\mathsf{int}}$ with some linear code $\mathcal{C}_1$, one obtains precisely an encoding of message $\boldsymbol{m}$ under the tensor code $\mathcal{C}_1 \otimes \mathcal{C}_2$. We now describe how to extend the ideas of Section 2.4 to work with tensor codes while avoiding high query complexity.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be linear-time encodable codes with constant relative distance and parameters $(\mathbb{F}, n_1, k_1, \delta_{\mathcal{C}_1})$ and $(\mathbb{F}, n_2, k_2, \delta_{\mathcal{C}_2})$ respectively. Additionally, assume that $\mathcal{C}_2$ is codeswitchable (see Definition 2.6). Let $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ be a tensor code with parameters $(\mathbb{F}, n := n_1 n_2, k := k_1 k_2, \delta_{\mathcal{T}} := \delta_{\mathcal{C}_1} \cdot \delta_{\mathcal{C}_2})$.

Consider an $\mathsf{MEP}^{(\mathcal{T},\varepsilon)}$ claim consisting of:

$$\mathbb{x} = (\boldsymbol{r}, y), \quad \vec{\mathbb{y}} = \boldsymbol{W} \in \mathbb{F}^{n_1 \times n_2}, \quad \mathbb{w} = \boldsymbol{M} \in \mathbb{F}^{k_1 \times k_2} \ ,$$

with the promise that $\boldsymbol{W} \sim_\varepsilon \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})$ and $\hat{M}(\boldsymbol{r}) = y$. Let $\boldsymbol{C}_{\mathsf{int}} \in \mathbb{F}^{k_1 \times n_2}$ be the encoding of $\boldsymbol{M}$ under the interleaved code $\mathcal{C}_2^{k_1}$.

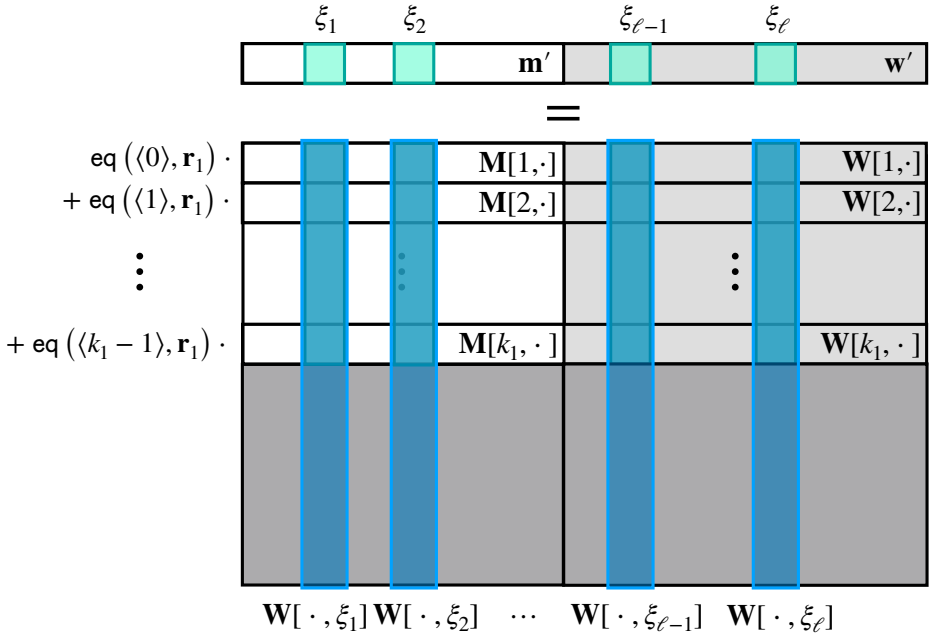

Figure 3: An illustration of the HEP claim used to implement the spot checks in TensorReduce. The depicted codes are systematic, with the white indices corresponding to the systematic part, and the gray and dark gray indices corresponding to the non-systematic parts of codewords of $\mathcal{C}_2$ and $\mathcal{C}_1$ respectively. The turquoise indices correspond to the indices that the verifier queries and the blue indices correspond to the indices that are involved in the HEP claim.

We describe an approach to constructing the TensorReduce IOR that reduces $\mathsf{MEP}^{(\mathcal{T},\varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{C}_2,\varepsilon'/2)} \times \mathsf{HEP}^{(\mathcal{C}_1,\varepsilon',\nu)}$ where $\varepsilon' = 1 - \sqrt{1-\varepsilon}$. We demonstrate the high-level idea of the reduction in Fig. 3, and provide an informal description of the IOR in Fig. 4. The differences between TensorReduce and InterleavedReduce are highlighted in blue in Fig. 4.

Input: $\mathsf{MEP}^{(\mathcal{T},\varepsilon)}$ claim: $(\mathbb{x} = (\boldsymbol{r}, y), \vec{\mathbb{y}} = \boldsymbol{W}, \mathbb{w} = \boldsymbol{M})$ such that $\boldsymbol{W} \sim_\varepsilon \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})$ and $\hat{M}(\boldsymbol{r}) = y$.

1. $\mathcal{P}$ and $\mathcal{V}$ parse $\boldsymbol{r}$ as $(\boldsymbol{r}_1 \,\|\, \boldsymbol{r}_2) \in \mathbb{F}^{\log k_1} \times \mathbb{F}^{\log k_2}$.
2. $\mathcal{P}$ computes a linear combination of the rows of $\boldsymbol{M}$ with respect to $\mathrm{eq}(\cdot, \boldsymbol{r}_1)$:
   $\boldsymbol{m}' \leftarrow \sum_{i \in [k_1]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}_1) \cdot \boldsymbol{M}[i,\cdot]$.
3. $\mathcal{P}$ encodes $\boldsymbol{m}'$ under $\mathcal{C}_2$: $\boldsymbol{w}' \leftarrow \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}')$, and sends $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$.

4. $\mathcal{V}$ samples $\nu := \lceil -\lambda / \log(1-\varepsilon'/2) \rceil$ indices $\xi_1, \ldots, \xi_\nu \xleftarrow{\$} [n_2]$ and sends them to $\mathcal{P}$.
5. For each $j \in [\nu]$: $\mathcal{V}$ queries $\boldsymbol{w}'[\xi_j]$.
6. Output the following $\mathsf{HEP}^{(\mathcal{C}_1, \varepsilon', \nu)}$ claim that asserts that for each $i \in [\nu]$, $\sum_{j \in [k_1]} \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{W}[j, \xi_i] = \boldsymbol{w}'[\xi_i]$:

$$\mathbb{x}_{\mathsf{HEP}} = (\boldsymbol{r}_1, [\boldsymbol{w}'[\xi_i]]_{i \in [\nu]}) \ , \ \vec{\mathbb{y}}_{\mathsf{HEP}} = [\boldsymbol{W}[\cdot, \xi_i]]_{i \in [\nu]}, \ , \ \mathbb{w}_{\mathsf{HEP}} = [\boldsymbol{C}_{\mathsf{int}}[\cdot, \xi_i]]_{i \in [\nu]} \ .$$

7. Output the following $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$ claim that asserts $\boldsymbol{w}' \sim_{\varepsilon'/2} \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}')$ and $\hat{m}'(\boldsymbol{r}_2) = y$:

$$\mathbb{x}_{\mathsf{MEP}} := (\boldsymbol{r}_2, y), \quad \vec{\mathbb{y}}_{\mathsf{MEP}} := \boldsymbol{w}', \quad \mathbb{w}_{\mathsf{MEP}} := \boldsymbol{m}' \ .$$

Figure 4: TensorReduce IOR from $\mathsf{MEP}^{(\mathcal{T}, \varepsilon)}$ to $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)} \times \mathsf{HEP}^{(\mathcal{C}_1, \varepsilon', \nu)}$, where $\varepsilon' = 1 - \sqrt{1-\varepsilon}$ (informal description).

Completeness of TensorReduce is easy to verify, and we describe a high-level proof of soundness. The proof sketch is similar to that of Fig. 2, since the HEP claim provides the same guarantee as the column checks from InterleavedReduce.

Proof of soundness. We show that if both the $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$ and $\mathsf{HEP}^{(\mathcal{C}_1, \varepsilon', \nu)}$ claims are true with probability at least $1 - 2^{-\lambda}$ over the sampled indices in Item 5, then the original $\mathsf{MEP}^{(\mathcal{T}, \varepsilon)}$ claim is true. Firstly, we define $\boldsymbol{W}_{\mathsf{int}} \in \mathbb{F}^{k_1 \times n_2}$ such that $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j])$ is the codeword closest to $\boldsymbol{W}[\cdot, j]$ for each $j \in [n_2]$. Additionally, define $\boldsymbol{w}_{\mathsf{fold}} := \sum_{i \in [k_1]} \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{W}_{\mathsf{int}}[i, \cdot]$.

1. Using guarantee of $\mathsf{HEP}^{(\mathcal{C}_1, \varepsilon', \nu)}$ claim: This is the same as the guarantee provided by the spot checks in InterleavedReduce. If the claim holds, then for all sampled indices $\{\xi_j\}_{j \in [\nu]}$, we have

$$\boldsymbol{w}'[\xi_j] = \boldsymbol{w}_{\mathsf{fold}}[\xi_j] \quad \text{and} \quad \mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, \xi_j]) \sim_{\varepsilon'} \boldsymbol{W}[\cdot, \xi_j] \ .$$

   If $\delta(\boldsymbol{w}', \boldsymbol{w}_{\mathsf{fold}}) \geq \varepsilon'/2$, then the probability of this event occurring is at most $(1 - \varepsilon'/2)^\nu \leq 2^{-\lambda}$. Similarly, if there are at least $\varepsilon' n_2/2$ column indices $j \in [n_2]$ such that $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j]) \nsim_{\varepsilon'} \boldsymbol{W}[\cdot, j]$, then the probability of the event occurring is at most $(1 - \varepsilon')^\nu \leq 2^{-\lambda}$. The contrapositive of these statements is that if the HEP claim is true with probability at least $1 - 2^{-\lambda}$ over the sampled indices, then there exists a set $S \subseteq [n_2]$ such that $|S| \geq (1 - \varepsilon'/2) n_2$, and for all $j \in S$, we have the following two guarantees:

   (a) $\boldsymbol{w}_{\mathsf{fold}}[j] = \boldsymbol{w}'[j]$, and
   (b) $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j]) \sim_{\varepsilon'} \boldsymbol{W}[\cdot, j]$.

2. Using guarantee of $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$ claim: Since $(\mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$, there exists a codeword $\boldsymbol{c}' \in \mathcal{C}_2$ such that $\boldsymbol{c}' \sim_{\varepsilon'/2} \boldsymbol{w}'$. Using the existence of such a $\boldsymbol{c}'$ and the guarantees from the HEP claim, we get that there exists an agreement set $A \subseteq S$ such that $|A| \geq (1 - \varepsilon') n_2$, and for all $j \in A$, we have the following two guarantees:

   (a) $\boldsymbol{w}_{\mathsf{fold}}[j] = \boldsymbol{c}'[j]$, and
   (b) $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j]) \sim_{\varepsilon'} \boldsymbol{W}[\cdot, j]$.

3. Using mutual correlated agreement: By Lemma 2.9, Item 2a, and the definition of $\boldsymbol{w}_{\mathsf{fold}}$, with overwhelming probability there exist codewords $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k_1} \in \mathcal{C}_2$ such that $\boldsymbol{W}_{\mathsf{int}}[i, A] = \boldsymbol{c}_i[A]$ for all $i \in [k_1]$, and furthermore $\sum_{i \in [k_1]} \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{c}_i = \boldsymbol{c}'$. Consider the message matrix $\boldsymbol{M}$ whose rows $\boldsymbol{M}[1, \cdot], \ldots, \boldsymbol{M}[k_1, \cdot]$ are the messages underlying $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k_1}$ respectively.

   (a) Proving proximity claim: We claim that $\boldsymbol{W} \sim_\varepsilon \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})$, and show this by establishing a lower bound on the number of indices $(i, j) \in [n_1] \times [n_2]$ such that $\boldsymbol{W}[i, j] = \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})[i, j]$. Firstly, for any $i \in [k_1]$ and $j \in A$, we have $\boldsymbol{W}_{\mathsf{int}}[i, j] = \boldsymbol{c}_i[j] = \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{M}[i, \cdot])[j]$ by definition of the $\boldsymbol{c}_i$'s. Furthermore, by Item 2b we get that $\boldsymbol{W}[\cdot, j] \sim_{\varepsilon'} \mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j])$ for all $j \in A$. Thus, for any $j \in A$, there exist at least $(1-\varepsilon')n_1$ indices $i \in [n_1]$ such that

$$\begin{aligned} \boldsymbol{W}[i, j] &= \mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j])[i] \\ &= \mathsf{Enc}_{\mathcal{C}_1}([\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{M}[i', \cdot])[j]]_{i' \in [k_1]})[i] \\ &= \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})[i, j] \ . \end{aligned}$$

   Thus, the number of indices where $\boldsymbol{W}$ and $\mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})$ are equal is at least $(1-\varepsilon')n_1|A| \geq (1-\varepsilon')^2 n_1 n_2 \geq (1-\varepsilon)n_1 n_2$, where the last inequality follows from plugging in $\varepsilon' = 1 - \sqrt{1-\varepsilon}$.

   (b) Proving evaluation claim: Since $\boldsymbol{c}' = \sum_{i \in [k_1]} \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{c}_i$, and $\mathcal{C}_2$ is a linear code, we have $\boldsymbol{m}' = \sum_{i \in [k_1]} \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{r}_1) \cdot \boldsymbol{M}[i, \cdot]$. Thus, $\hat{m}'(\boldsymbol{r}_2) = y$ implies that $\hat{M}(\boldsymbol{r}_1, \boldsymbol{r}_2) = y$, proving the evaluation claim as desired. □

Challenge 2: current techniques for proving $\mathsf{HEP}^{(\mathcal{C}_1, \varepsilon, \Theta(\lambda))}$ claims are too inefficient. Since $\mathcal{C}' \coloneqq \mathcal{C}_1 \otimes \mathcal{C}_2$ must be linear-time encodable, it must have constant rate, implying that $\mathcal{C}_1$ must also have constant rate. Thus, our query lower bound for MEP (see Section 8) implies that any IOPP for each $\mathsf{MEP}^{(\mathcal{C}_1, \varepsilon)}$ claim requires $\Omega(\lambda)$ queries. Even if we had a query optimal IOPP for $\mathsf{MEP}^{(\mathcal{C}_1, \varepsilon)}$, this implies that naively using such an IOPP for proving all $\Theta(\lambda)$ $\mathsf{MEP}^{(\mathcal{C}_1, \varepsilon)}$ claims will require $\Omega(\lambda^2)$ queries. However, we have made progress: we can now reduce the message length from $k = k_1 k_2$ to k_2 with potentially only $O(\lambda^2)$ queries, whereas the interleaved approach would require $O(\lambda k_1)$ queries. Thus we have 'decoupled' k_1 from the number of queries, and can now set k_1 to be super-constant.

2.6 Lossy batching

We overcome this final barrier of reducing the number of queries from $O(\lambda^2)$ to $O(\lambda)$ as follows: instead of convincing $\mathcal{V}$ that *all* $\nu = \Theta(\lambda)$ evaluation and proximity claims underlying $\mathsf{HEP}^{(\mathcal{C}_1, \varepsilon, \nu)}$ claims are true, $\mathcal{P}$ convinces $\mathcal{V}$ that a *large fraction* of these claims are true. By increasing the number of columns chosen by a constant factor, such a promise suffices for the soundness proof from Section 2.5 to still go through. In particular, we can draw the same conclusion as in Item 1 of the proof of soundness of Fig. 4 with this weaker promise. We refer to this approach as *lossy batching*, since we can afford to "lose" the guarantee of some of the $\mathsf{MEP}^{(\mathcal{C}_1, \varepsilon)}$ claims, instead of requiring all of them to hold.

Firstly, we define the lossy HEP relation, which we call LHEP. Then, we show how to reduce a large LHEP claim to a small SEP claim using the lossy batching IOR, which only requires $O(\lambda)$ queries as opposed to $\Omega(\lambda^2)$ queries required to prove the HEP claim. Finally, we show that if the TensorReduce IOR is changed to reduce to an LHEP claim instead of an HEP claim, the soundness proof for Fig. 4 can be adapted to still work.

Definition 2.10 (informal)**.** The relation $\mathsf{LHEP}^{(\mathcal{C}, \varepsilon, \ell, \gamma_{\mathsf{lost}})}$ consists of tuples of the form

- $\mathbb{x} = (\boldsymbol{z}, [y_i]_{i\in[\ell]})$, where $\boldsymbol{z} \in \mathbb{F}^{\log k}$ and $y_i \in \mathbb{F}$ for each $i \in [\ell]$,
- $\vec{\mathbb{y}} = [\boldsymbol{w}_i]_{i\in[\ell]}$, where $\boldsymbol{w}_i \in \mathbb{F}^n$ for each $i \in [\ell]$,
- $\mathbb{w} = [\boldsymbol{m}_i]_{i\in[\ell]}$, where $\boldsymbol{m}_i \in \mathbb{F}^k$ for each $i \in [\ell]$,

such that there exists a set $I \subseteq [\ell]$ such that $|I| \geq (1-\gamma_{\mathsf{lost}})\ell$, and for all $i \in I$, we have $\boldsymbol{w}_i \sim_\varepsilon \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i)$, and $\hat{m}_i(\boldsymbol{z}) = y_i$.

Note that the LHEP relation is identical to the HEP relation, except that instead of requiring all ℓ proximity and evaluation claims to hold, it only requires that at least $(1-\gamma_{\mathsf{lost}})$-fraction of them hold.

Firstly, the evaluation claims can be proven with ℓ-many inner product claims about $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$. However, $\mathcal{P}$ can instead initialize a single message $\boldsymbol{m}' \in \mathbb{F}^{k\ell}$ as the concatenation of $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$, and prove a single inner product claim about $\boldsymbol{m}'$ that captures all ℓ evaluation claims.

Secondly, the proximity claims can be proven via spot checks: $\mathcal{V}$ samples $\mu := \lceil -1/(\gamma_{\mathsf{lost}} \log(1-\varepsilon))\rceil$ random indices $\xi_{i,1}, \ldots, \xi_{i,\mu} \in [n]$ for each $i \in [\ell]$, and queries $\boldsymbol{w}_i[\xi_{i,j}]$ for each $j \in [\mu]$. For each $i \in [\ell]$ and $j \in [\mu]$, $\mathcal{P}$ then proves that $\boldsymbol{w}_i[\xi_{i,j}] = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i)[\xi_{i,j}]$. This can be proven via an inner product claim between $\boldsymbol{m}_i$ and $\boldsymbol{G}_{\mathcal{C}}[\xi_{i,j}, \cdot]$, the $\xi_{i,j}$-th row of the generator matrix of $\mathcal{C}$. The resulting $\ell\mu$ inner product claims can then be batched into a single inner product claim about $\boldsymbol{m}'$.

We encapsulate these ideas in the following LossyBatch IOR, and refer the reader to Section 5 for more details.

Input: $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ claim: $\mathbb{x} = (\boldsymbol{z}, [y_i]_{i\in[\ell]}), \vec{\mathbb{y}} = [\boldsymbol{w}_i]_{i\in[\ell]}, \mathbb{w} = [\boldsymbol{m}_i]_{i\in[\ell]}$ such that there exists a set $I \subseteq [\ell]$ where $|I| \geq (1-\gamma_{\mathsf{lost}})\ell$, and for all $i \in I$, we have $\boldsymbol{w}_i \sim_\varepsilon \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i)$, and $\hat{m}_i(\boldsymbol{z}) = y_i$.

1. $\mathcal{P}$ initializes the new message for the output SEP instance: $\boldsymbol{m}' \leftarrow (\boldsymbol{m}_1 \,\|\, \ldots \,\|\, \boldsymbol{m}_\ell) \in \mathbb{F}^{k\ell}$.
2. $\mathcal{P}$ encodes $\boldsymbol{m}'$ under $\mathcal{C}'$: $\boldsymbol{w}' := \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}')$ and sends $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$.
3. For each $i \in [\ell]$, $\mathcal{V}$ samples $\mu := \lceil -1/(\gamma_{\mathsf{lost}} \log(1-\varepsilon))\rceil$ random indices $\xi_{i,1}, \ldots, \xi_{i,\mu} \in [n]$ and sends them to $\mathcal{P}$.
4. For each $i \in [\ell]$ and $j \in [\mu]$: $\mathcal{V}$ queries $\boldsymbol{w}_i[\xi_{i,j}]$.
5. $\mathcal{P}$ and $\mathcal{V}$ output an $\mathsf{SEP}^{(\mathcal{C}',\varepsilon')}$ claim $(\mathbb{x} = (f, \sigma), \vec{\mathbb{y}} = \boldsymbol{w}', \mathbb{w} = \boldsymbol{m}')$ which enforces that there exist $\boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_\ell^* \in \mathbb{F}^k$ such that:
   (a) **Evaluation claims:** For each $i \in [\ell]$, $\hat{m}_i^*(\boldsymbol{z}) = y_i$. These ℓ evaluation claims can be batched into one inner product claim over $\boldsymbol{m}'$, and therefore captured by SEP.
   (b) **Spot check claims:** for each $i \in [\ell], j \in [\mu]$, $\boldsymbol{w}_i[\xi_{i,j}] = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i^*)[\xi_{i,j}]$. For a fixed i and j, this is an inner product claim between $\boldsymbol{m}_i^*$ and $\boldsymbol{G}_{\mathcal{C}}[\xi_{i,j}, \cdot]$. These $\ell\mu$ inner product claims can be batched into one inner product claim over $\boldsymbol{m}'$, and therefore captured by SEP.

   We omit the details of how this SEP claim is constructed (and in particular what f and σ are precisely). The full construction is very involved, and is described in detail in Section 5.

Figure 5: LossyBatch IOR from $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon')}$ (informal description).

Remark 2.11 (Repetition of message)**.** In the full construction of LossyBatch in Section 5, we require that $\boldsymbol{m}'$ is of the form $(\boldsymbol{m}_1 \,\|\, \ldots \,\|\, \boldsymbol{m}_\ell)^n$, i.e., the concatenation of $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$ repeated n times. Thus, $\mathcal{C}'$ must have message length $nk\ell$ instead of $k\ell$ as stated in the informal description above. We omit the details of why this is necessary, and refer the reader to Section 5 for more details.

Lemma 2.12 (informal)**.** *Let $\ell \in \mathbb{N}$, $\mathcal{C}$ be a codeswitchable code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, and $\mathcal{C}'$ be a linear-time encodable code with parameters $(\mathbb{F}, n', nk\ell, \delta_{\mathcal{C}'})$. There exists an IOR,* LossyBatch*, from* $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ *to* $\mathsf{SEP}^{(\mathcal{C}',\varepsilon')}$ *where $\varepsilon' \in (0, \delta_{\mathcal{C}'}/2)$, such that the IOR prover runs in linear-time and the verifier only makes $\lceil -1/(\gamma_{\mathsf{lost}} \log(1-\varepsilon)) \rceil$ queries to each of the ℓ oracles (assuming ε is a constant), thus making $O(\ell)$ queries in total.*

Proof of soundness. We prove the following statement: if the output SEP claim is satisfied, then the messages $\boldsymbol{m}_1^*, \ldots, \boldsymbol{m}_\ell^*$ defined in Item 5 fail to satisfy the LHEP claim with probability at most $2^{-\ell}$.

1. Using Claim 5a: If the LHEP claim is false, then there exists a set $I \subseteq [\ell]$ such that $|I| \geq \gamma_{\mathsf{lost}}\ell$ and for all $i \in I$, we have either $\hat{m}_i^*(\boldsymbol{z}) \neq y_i$, or $\delta(\boldsymbol{w}_i, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i^*)) \geq \varepsilon$. Claim 5a guarantees that $\hat{m}_i^*(\boldsymbol{z}) = y_i$ for all $i \in [\ell]$, and thus we must have $\delta(\boldsymbol{w}_i, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i^*)) \geq \varepsilon$ for all $i \in I$.

2. Using Claim 5b: For all $i \in [\ell]$ and $j \in [\mu]$, we have $\boldsymbol{w}_i[\xi_{i,j}] = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i^*)[\xi_{i,j}]$. Consider a fixed $i \in I$. The probability that $\boldsymbol{w}_i[\xi_{i,j}] = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i^*)[\xi_{i,j}]$ for all $j \in [\mu]$ despite the fact that $\delta(\boldsymbol{w}_i, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i^*)) \geq \varepsilon$ is at most $(1-\varepsilon)^\mu$. Thus, the probability that this occurs for all $i \in I$ is at most $(1-\varepsilon)^{\mu|I|} \leq (1-\varepsilon)^{\mu\gamma_{\mathsf{lost}}\ell} \leq 2^{-\ell}$, where the second inequality holds since $\mu = \lceil -1/(\gamma_{\mathsf{lost}} \log(1-\varepsilon)) \rceil$. □

Reduction to LHEP in TensorReduce. Recall that the motivation for introducing the LHEP oracle relation and the LossyBatch IOR is to replace the output HEP claim of TensorReduce. Thus, we make two changes to Fig. 4:

- Sample $\nu' = \lceil 8\lambda \ln 2/\varepsilon' \rceil$ columns instead of $\nu = \lceil -\lambda/\log(1-\varepsilon'/2) \rceil$ columns in Item 5.
- Output an $\mathsf{LHEP}^{(\mathcal{C}_1,\varepsilon',\nu',\gamma_{\mathsf{lost}})}$ claim instead of an $\mathsf{HEP}^{(\mathcal{C}_1,\varepsilon',\nu)}$ claim in Item 6 where $\gamma_{\mathsf{lost}} = \varepsilon'/4$.

The LHEP claim guarantees that there exists a set $I \subseteq [\nu']$ such that $|I| \geq (1-\varepsilon'/4)\nu'$ and

$$\forall i \in [I]: \quad \boldsymbol{W}[\cdot, \xi_i] \sim_{\varepsilon'} \mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, \xi_i]) \text{ and } \hat{W}_{\mathsf{int}}[\cdot, \xi_i](\boldsymbol{r}_1) = \boldsymbol{w}'[\xi_i] \ . \tag{2}$$

The proof of soundness of Fig. 4 remains mostly unchanged, except for Item 1. In particular, we need to show that there exists a set $S \subseteq [n_2]$ such that $|S| \geq (1-\varepsilon'/2)n_2$, and for all $j \in S$, we have

1. $\boldsymbol{w}_{\mathsf{fold}}[j] = \boldsymbol{w}'[j]$, and
2. $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j]) \sim_{\varepsilon'} \boldsymbol{W}[\cdot, j]$.

For clarity, we only discuss how to prove Item 2, since Item 1 can be proved in the same way. Say there exists a disagreement set $D \subseteq [n_2]$ such that $|D| \geq \varepsilon' n_2/2$ and $\delta(\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j]), \boldsymbol{W}[\cdot, j]) \geq \varepsilon'$ for all $j \in D$. Define an indicator random variable X_i for each $i \in D$ such that $X_i = 1$ if i is among the column indices sampled by $\mathcal{V}$ and $X_i = 0$ otherwise. Note that the X_i's are negatively correlated random variables since ν' columns are sampled without replacement. Thus, the probability that any $X_i = 1$ is ν'/n_2. Therefore, we have $\mathbb{E}[X] = \Pr[X_1 = 1] \cdot |D| \geq (\nu'/n_2) \cdot (\varepsilon' n_2/2) = \varepsilon'\nu'/2$, where $X \coloneqq \sum_{j \in D} X_j$ is the number of columns sampled by $\mathcal{V}$ that are in the disagreement set D. The LHEP claim can only be satisfied if $X < \varepsilon'\nu'/4$. Using the multiplicative Chernoff bound (which also applies to negatively correlated random variables), we get

$$\Pr[X < \varepsilon'\nu'/4] \leq \Pr[X < \mathbb{E}[X]/2] \leq \exp(-\mathbb{E}[X]/8) \leq \exp(-\varepsilon'\nu'/8) \leq 2^{-\lambda} \ ,$$

where the last inequality holds due to our new choice of ν'. Therefore, if the LHEP claim is true, then we have $|D| < \varepsilon' n_2/2$ with overwhelming probability, which is the same as the guarantee provided by the column checks from Section 2.4.

Challenge 3: obtaining a non-interactive argument of knowledge. In this technical overview, we have proven soundness of the IORs we constructed, which is not sufficient to obtain a non-interactive argument of knowledge using the BCS transformation [BCS16]. Ideally, we would like to prove a stronger notion of soundness such as round-by-round extractability [BCFW25] for these IORs, which does imply such a non-interactive argument of knowledge. However, proving such a *straight-line* property for the TensorReduce IOR poses a challenge. In particular, consider the TensorReduce IOR first presented in Section 2.5 (which outputs an HEP claim). A straight-line extractor can only see the witness of a single HEP output claim and a single MEP output claim, which consist of $O(\lambda)$ columns of $\boldsymbol{W}_{\mathsf{int}}$, and the folded witness $\boldsymbol{m}'$ respectively. However, to extract a valid witness $\boldsymbol{M}$ for the input MEP claim, these are not sufficient since there are multiple possible $\boldsymbol{M}$ that are consistent with these output witnesses and the input instance. Thus, proving a straight-line notion of extractability for TensorReduce is a bottleneck.

Therefore, we have to settle for rewinding extraction, that is, allowing the extractor to rewind the prover multiple times in order to output $\boldsymbol{M}$. Fortunately, the recent notion of round-by-round tree-extractability (RBRTE), introduced by [BMMS25], is helpful in this regard. They show that if an IOR is RBRTE, then the non-interactive argument obtained by applying the BCS transformation is (adaptive) rewinding knowledge sound. In the main technical sections, we prove that our IORs are RBRTE, and then use the composition theorem from [BMMS25] to prove that the IOPP obtained by composing our IORs is also RBRTE.

2.7 Final IOPP construction

An MEP to MEP reduction. The TensorReduce described in Sections 2.5 and 2.6 takes as input an $\mathsf{MEP}^{(\mathcal{C}_1 \otimes \mathcal{C}_2, \varepsilon)}$ claim with message length k and block length n and outputs the following:

- an $\mathsf{LHEP}^{(\mathcal{C}_1, \varepsilon', \nu')}$ claim with message length k_1, and
- an $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$ claim with message length k_2.

However, in order to complete the proof of Lemma 2.7, and apply TensorReduce recursively, we need to output a single MEP claim over a tensor code with a message length of $O(\lambda^{1/3}k^{2/3})$. Recall that k_1 and k_2 are chosen such that $k_2 = n_1 k_1 \nu' = O(\lambda^{1/3}k^{2/3})$. In the following construction of TensorReduce$'$, $\mathcal{C}' = \mathcal{C}'_1 \otimes \mathcal{C}'_2$ is a tensor code with message length k_2 such that $\mathcal{C}'_2$ is a codeswitchable code (see Definition 2.6).

1. Run the TensorReduce IOR from Sections 2.5 and 2.6 to obtain an $\mathsf{LHEP}^{(\mathcal{C}_1, \varepsilon', \nu')}$ claim and an $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$ claim.
2. Run the LossyBatch IOR from Section 2.6 to reduce the $\mathsf{LHEP}^{(\mathcal{C}_1, \varepsilon', \nu')}$ claim to an $\mathsf{SEP}^{(\mathcal{C}', \varepsilon')}$ claim.
3. Run the Sumcheck IOR to reduce the $\mathsf{SEP}^{(\mathcal{C}', \varepsilon')}$ to a $\mathsf{MEP}^{(\mathcal{C}', \varepsilon')}$ claim.
4. Run the CodeSwitch IOR from Lemma 2.5 to reduce the $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon'/2)}$ claim to an $\mathsf{MEP}^{(\mathcal{C}', \varepsilon')}$ claim.

The resulting two $\mathsf{MEP}^{(\mathcal{C}', \varepsilon')}$ claims can then be combined into a single $\mathsf{MEP}^{(\mathcal{C}', \varepsilon')}$ claim using standard techniques. By the choice of k_1 and k_2 in Lemma 2.7, the message length of $\mathcal{C}'$ is $k_2 = n_1 k_1 \nu' = O(\lambda^{1/3}k^{2/3})$, completing the proof of Lemma 2.7.

Constructing TensorIOPP from TensorReduce$'$. Finally, we discuss how TensorReduce$'$ is used to construct TensorIOPP. Let $\lambda = O(k^{1-\gamma})$, for some $\gamma > 2/3$. One application of TensorReduce$'$ reduces the message length from $\kappa_1 := k$ to $\kappa_2 := O(\lambda^{1/3}k^{2/3})$. If κ_i is the message length after $i-1$ applications of TensorReduce$'$, then we have

$$\kappa_i = O\left(k^{(1-\gamma+\gamma\cdot(2/3)^{i-1})}\right) .$$

This is where the requirement that $\gamma > 2/3$ comes in, since this guarantees that there exists a constant c such that $1 - \gamma + \gamma \cdot (2/3)^{c-1} \leq 1/3$, implying that $\kappa_c = O(k^{1/3})$.

Using codes with inverse polynomial rate. Once the message length is $O(k^{1/3})$, the output code $\mathcal{C}'$ of TensorReduce is set as a Reed–Solomon code RS with message length $\kappa_c = O(k^{1/3})$ and rate $1/n^{1/3}$.

Notice that this code has block length $O(n^{2/3})$ and is thus encodable in $O(n)$ time, using the naive encoding algorithm that uses the generator matrix of a code and takes $O(n^{2/3}) \cdot O(k^{1/3}) = O(n)$ time.

Utilizing the fact that this code has inverse polynomial rate $1/n^{1/3}$ (and thus distance $1 - 1/n^{1/3}$), we instantiate the WHIR IOPP [ACFY25] for constrained Reed–Solomon codes (which captures MEP) with the appropriate parameters to obtain an IOPP for $\mathsf{MEP}^{(\mathsf{RS},\varepsilon)}$ with $O(n)$ prover time and $O(\lambda)$ queries. At a high level, this IOPP iteratively reduces the message length by a constant factor. Typically, when the code has constant rate, this requires $O(\lambda)$ queries per reduction (at least for a non-constant number of rounds), leading to $\omega(\lambda)$ queries overall. However, since the rate of RS is $O(1/n^{1/3})$, only $O(\lambda/\log n)$ queries are required per reduction and $O(\lambda)$ queries are required overall. The full construction of $\mathsf{TensorIOPP}$ can be found in Section 7.

3 Preliminaries

3.1 Notation

For a finite field $\mathbb{F}$ and $n \in \mathbb{N}$, bold lowercase letters $\boldsymbol{x} \in \mathbb{F}^n$ denote vectors of field elements. Unless otherwise specified, we assume that vectors over $\mathbb{F}$ are zero-indexed. $\boldsymbol{x}[i]$ denotes the i-th element of $\boldsymbol{x}$.[9] Given $\boldsymbol{x} \in \mathbb{F}^n$ and $y \in \mathbb{F}$, we use $y \cdot \boldsymbol{x}$ to denote $[y \cdot \boldsymbol{x}[i]]_{i \in [n]}$. Given $\ell \in \mathbb{N}$, we use $\boldsymbol{x}^\ell$ to denote the vector $(\boldsymbol{x}, \ldots, \boldsymbol{x})$, where $\boldsymbol{x}$ is repeated ℓ times. Given a set $S \subseteq [n]$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^n$, we say $\boldsymbol{x}[S] = \boldsymbol{y}[S]$ if $\boldsymbol{x}[i] = \boldsymbol{y}[i]$ for all $i \in S$.

We use bold uppercase letters $\boldsymbol{M} \in \mathbb{F}^{n \times m}$ to denote matrices over $\mathbb{F}$. $\boldsymbol{M}[i, j]$, $\boldsymbol{M}[i, \cdot]$, and $\boldsymbol{M}[\cdot, j]$ denote the element in the i-th row and j-th column, the i-th row, and the j-th column of $\boldsymbol{M}$ respectively.

Interplay between vectors and matrices. Looking ahead, we will need to consider fields of the form $(\mathbb{F}^m)^n$. We identify the elements of such a field with matrices $\boldsymbol{V} \in \mathbb{F}^{m \times n}$, where each column of $\boldsymbol{V}$ is thought of as an element of $\mathbb{F}^m$. Given an matrix $\boldsymbol{V} \in \mathbb{F}^{m \times n}$ written in uppercase, we use the corresponding lowercase $\boldsymbol{v} \in \mathbb{F}^{mn}$ to denote the vector obtained by concatenating the rows of $\boldsymbol{V}$. Similarly, when m and n are clear from context, given a vector $\boldsymbol{v} \in \mathbb{F}^{mn}$, we use $\boldsymbol{V} \in \mathbb{F}^{m \times n}$ to denote the matrix $\boldsymbol{V} \in \mathbb{F}^{m \times n}$ such that $\boldsymbol{v}$ is obtained by concatenating the rows of $\boldsymbol{V}$.

Binary representation. Given $i, n \in \mathbb{N}$, $\langle i \rangle_n$ represents the n-bit binary representation of i in most-significant-bit-first order. We drop the subscript n when it is clear from context.

Multilinear polynomials. We use $\mathbb{F}^{\leq d}[X_1, \ldots, X_\mu]$ to denote the set of all μ-variate polynomials over $\mathbb{F}$ such that the degree of each variable in every monomial is at most d. We denote a multivariate polynomial by $p(X_1, \ldots, X_\mu)$ or by $p(\boldsymbol{X})$ when μ is clear from context. $\mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ is the set of all μ-variate *multilinear* polynomials. $p(\boldsymbol{X}) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ is uniquely defined by its evaluations $\boldsymbol{p} := [p(\langle i \rangle)]_{i=0}^{2^\mu - 1}$ over the μ-dimensional boolean hypercube $\{0,1\}^\mu \subset \mathbb{F}^\mu$.

Multilinear extension. The *multilinear extension* of a vector $\boldsymbol{m} \in \mathbb{F}^{2^\mu}$ is the unique multilinear polynomial $\hat{m}(\boldsymbol{X}) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ such that $\hat{m}(\langle i - 1 \rangle) = \boldsymbol{m}[i]$ for all $i \in [2^\mu]$.

Lagrange basis polynomial. The *Lagrange basis polynomial* $\mathrm{eq}_\mu(\boldsymbol{X}, \boldsymbol{Y}) := \prod_{i=1}^{\mu} (X_i Y_i + (1 - X_i)(1 - Y_i))$ is a multilinear polynomial that evaluates to 1 if and only if $\boldsymbol{X} = \boldsymbol{Y}$ for any $\boldsymbol{X}, \boldsymbol{Y} \in \{0,1\}^\mu$. We omit μ when it is clear from context. For any $p(\boldsymbol{X}) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$ and $\boldsymbol{z} \in \mathbb{F}^\mu$, we have

$$p(\boldsymbol{z}) = \sum_{\boldsymbol{i} \in \{0,1\}^\mu} p(\boldsymbol{i}) \cdot \mathrm{eq}(\boldsymbol{i}, \boldsymbol{z}) \ .$$

We use $\mathbf{eq}(\boldsymbol{z})$ to denote the vector of evaluations of $\mathrm{eq}(\boldsymbol{X}, \boldsymbol{z})$ over $\{0,1\}^\mu$.

Hamming distance. We use Δ to denote Hamming distance, i.e., for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^n$, $\Delta(\boldsymbol{x}, \boldsymbol{y}) := |\{i \in [n] : \boldsymbol{x}[i] \neq \boldsymbol{y}[i]\}|$, and δ to denote fractional Hamming distance, i.e., $\delta(\boldsymbol{x}, \boldsymbol{y}) := \Delta(\boldsymbol{x}, \boldsymbol{y})/n$. For a set $S \subset \mathbb{F}^n$ and an element $\boldsymbol{x} \in \mathbb{F}^n$, we define $\Delta(\boldsymbol{x}, S) := \min_{\boldsymbol{s} \in S} \Delta(\boldsymbol{x}, \boldsymbol{s})$, and $\delta(\boldsymbol{x}, S) := \Delta(\boldsymbol{x}, S)/n$. We also use $\boldsymbol{x} \sim_\varepsilon \boldsymbol{y}$ to denote $\delta(\boldsymbol{x}, \boldsymbol{y}) < \varepsilon$.

Lemma 3.1 (Ore-DeMillo-Lipton-Schwartz-Zippel lemma)**.** *Let $f \in \mathbb{F}^{\leq d}[X_1, \ldots, X_\mu]$ be a non-zero polynomial. Then*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^\mu} [f(\boldsymbol{r}) = 0] \leq \tfrac{d \cdot \mu}{|\mathbb{F}|} \ .$$

[9] When it helps to make notation more clear and does not lead to ambiguity, we sometimes use x_i to denote the same.

Lemma 3.2. *Let $f \in \mathbb{F}^{\leq d}[X_1, \ldots, X_{\mu_1+\mu_2}]$ be a polynomial that is not constant in the first μ_1 variables. Then,*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\mu_2}} [f(\boldsymbol{X}, \boldsymbol{r}) \text{ is a constant polynomial}] \leq \tfrac{d \cdot \mu_2}{|\mathbb{F}|} \quad .$$

Proof. If f is not constant in the first μ_1 variables, then there exists a term $c \cdot g(X_{\mu_1+1}, \ldots, X_{\mu_1+\mu_2}) \cdot \prod_{i=1}^{\mu_1} X_i^{a_i}$ for some $g \in \mathbb{F}^{\leq d}[X_{\mu_1+1}, \ldots, X_{\mu_1+\mu_2}]$ and $a_i \in \{0, 1\}$ where $c \neq 0$ and there exists at least one $i \in [\mu_1]$ such that $a_i = 1$. If $f(\boldsymbol{X}, \boldsymbol{r})$ is a constant polynomial, this term must be a constant, which can only occur if $g(\boldsymbol{r}) = 0$, and this occurs with probability at most $(d \cdot \mu_2)/|\mathbb{F}|$ by Lemma 3.1. □

3.2 Coding theory

Definition 3.3 (Linear codes)**.** A linear code $\mathcal{C}$ of message length k, block length n, over an alphabet $\mathbb{F}$ is a k-dimensional subspace of $\mathbb{F}^n$. The *rate* of the code is defined to be k/n and the *distance* of a code $\mathcal{C}$ is defined to be

$$\delta_{\mathcal{C}} \coloneqq \min_{\boldsymbol{x} \neq \boldsymbol{y} \in \mathcal{C}} \delta(\boldsymbol{x}, \boldsymbol{y})$$

We abbreviate the parameters of a code as $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. We use $\boldsymbol{c} \in \mathcal{C}$ to denote codewords and $\boldsymbol{w} \in \mathbb{F}^n$ to denote arbitrary words. We say a word $\boldsymbol{w} \in \mathbb{F}^n$ is δ-far from (or δ-close to) a code $\mathcal{C}$ if $\delta(\boldsymbol{w}, \mathcal{C}) > \delta$ (or $\delta(\boldsymbol{w}, \mathcal{C}) \leq \delta$).

Encoding. An *encoding function* of a code $\mathcal{C}$ is a bijection $\mathsf{Enc}_{\mathcal{C}} : \mathbb{F}^k \to \mathcal{C}$. The message $\boldsymbol{m} \in \mathbb{F}^k$ of a codeword $\boldsymbol{c} \in \mathcal{C}$ relative to an encoding function $\mathsf{Enc}_{\mathcal{C}}$ is the unique vector such that $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}) = \boldsymbol{c}$. If $\mathcal{C}$ is linear, there exists a $k \times n$ matrix $\boldsymbol{G}_{\mathcal{C}}$ known as the *generator matrix* such that $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}) = \boldsymbol{G}_{\mathcal{C}}\boldsymbol{m}$. We also use $T_{\mathcal{C}}$ to denote the time complexity of encoding a message under $\mathcal{C}$.

Definition 3.4 (List decoding)**.** Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, $\boldsymbol{w} \in \mathbb{F}^n$ be a word, and δ be a proximity parameter, we define

$$\Lambda(\mathcal{C}, \boldsymbol{w}, \delta) \coloneqq \{\boldsymbol{c} \in \mathcal{C} : \delta(\boldsymbol{c}, \boldsymbol{w}) \leq \delta\} \quad .$$

We also use $\Lambda(\mathcal{C}, \delta)$ to denote $\max_{\boldsymbol{w}} \Lambda(\mathcal{C}, \boldsymbol{w}, \delta_{\mathcal{C}})$. We say that $\mathcal{C}$ is (δ, ℓ)-*list decodable* if $|\Lambda(\mathcal{C}, \boldsymbol{w}, \delta)| \leq \ell$ for every $\boldsymbol{w} \in \mathbb{F}^n$. We say that δ is in the 'unique decoding regime' if $\mathcal{C}$ is $(\delta, 1)$-list decodable, and in the 'list decoding regime' if $\mathcal{C}$ is (δ, ℓ)-list decodable for some $\ell = \mathrm{poly}(n)$.

Lemma 3.5 (Out-of-domain evaluation)**.** *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, let $\boldsymbol{w} \in \mathbb{F}^n$ be an arbitrary word, and let $\delta \in (0, \delta_{\mathcal{C}})$. Then, we have*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\log k}} \left[\begin{array}{r} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{u}) \in \Lambda(\mathcal{C}, \boldsymbol{w}, \delta) \\ \exists \text{ distinct } \boldsymbol{u}, \boldsymbol{v} : \mathbb{F}^k \text{ s.t. } \text{ and } \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{v}) \in \Lambda(\mathcal{C}, \boldsymbol{w}, \delta) \\ \text{and } \hat{u}(\boldsymbol{r}) = \hat{v}(\boldsymbol{r}) \end{array} \right] \leq \frac{|\Lambda(\mathcal{C}, \delta)|^2}{2} \cdot \frac{\log k}{|\mathbb{F}|} \quad .$$

Proof. If $\hat{u}(\boldsymbol{X}) \neq \hat{v}(\boldsymbol{X})$, then $\hat{u}(\boldsymbol{r}) = \hat{v}(\boldsymbol{r})$ with probability at most $\log k/|\mathbb{F}|$ by Lemma 3.1. Applying the union bound over all pairs $(\boldsymbol{u}, \boldsymbol{v})$ such that $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{u}), \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{v}) \in \Lambda(\mathcal{C}, \boldsymbol{w}, \delta)$, we get the desired bound. □

Definition 3.6 (Tensor codes)**.** Let $\mathbb{F}$ be a finite field, $\mathcal{C}_1$ be a linear code with parameters $(\mathbb{F}, n_1, k_1, \delta_{\mathcal{C}_1})$ and generator matrix $\boldsymbol{G}_{\mathcal{C}_1} \in \mathbb{F}^{n_1 \times k_1}$, and $\mathcal{C}_2$ be a linear code with parameters $(\mathbb{F}, n_2, k_2, \delta_{\mathcal{C}_2})$ and generator matrix $\boldsymbol{G}_{\mathcal{C}_2} \in \mathbb{F}^{n_2 \times k_2}$. The *tensor code* $\mathcal{C}_1 \otimes \mathcal{C}_2$ is the linear code with the generator matrix $\boldsymbol{G}_{\mathcal{C}_1} \otimes \boldsymbol{G}_{\mathcal{C}_2}$,

where $\otimes$ denotes the Kronecker product. It is well known that $\mathcal{C}_1 \otimes \mathcal{C}_2$ has parameters $(\mathbb{F}, n_1 n_2, k_1 k_2, \delta_{\mathcal{C}_1} \delta_{\mathcal{C}_2})$. The encoding time of $\mathcal{C}_1 \otimes \mathcal{C}_2$ is $O(n_2 \cdot T_{\mathcal{C}_1} + k_1 \cdot T_{\mathcal{C}_2})$ (first encoding the rows under $\mathcal{C}_2$, then encoding the columns under $\mathcal{C}_1$).

Definition 3.7 (Interleaved codes)**.** Let $m \in \mathbb{N}$ be the interleaving parameter and $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. The m-interleaved code $\mathcal{C}^m \subseteq (\mathbb{F}^m)^n$ is a linear code with parameters $(\mathbb{F}^m, n, k, \delta_{\mathcal{C}})$ such that for every $\boldsymbol{C} \in \mathcal{C}^m$, every row of $\boldsymbol{C}$ is a codeword of $\mathcal{C}$. Given a message $\boldsymbol{M} \in (\mathbb{F}^m)^k$, $\mathsf{Enc}_{\mathcal{C}^m}(\boldsymbol{M})$ is obtained by encoding each row of $\boldsymbol{M}$ under $\mathcal{C}$.

Erasure correction.

Definition 3.8. A code $\mathcal{C}$ with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ supports erasure correction with correction time $\mathsf{ecor}_{\mathcal{C}}$ if there exists a deterministic algorithm $\mathcal{E}_{\mathcal{C}}$ (the erasure corrector) that given as input some $\boldsymbol{w} \in \mathbb{F}^n$ and a set $S \subseteq [n]$, behaves as follows:

- If $|S| \geq (1 - \delta_{\mathcal{C}})n$ and there exists a codeword $\boldsymbol{c} \in \mathcal{C}$ such that $\boldsymbol{c}[S] = \boldsymbol{w}[S]$ (note that such a codeword is unique), then $\mathcal{E}_{\mathcal{C}}(\boldsymbol{w})$ outputs $\boldsymbol{c}$.
- Otherwise, $\mathcal{E}_{\mathcal{C}}(\boldsymbol{w})$ outputs $\perp$.

Moreover, $\mathcal{E}_{\mathcal{C}}$ performs at most $\mathsf{ecor}_{\mathcal{C}}$ field operations.

Lemma 3.9 ([GRS12])**.** *Every linear code $\mathcal{C}$ with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ supports erasure correction with correction time $O(n^3)$.*

3.3 Mutual correlated agreement for multilinear proximity generators

Linear codes exhibit a property called *correlated agreement* used extensively in previous work on IOPPs and accumulation schemes [AHIV17; BBHR18; ACFY24]. A stronger notion known as *mutual correlated agreement* was introduced in [ACFY25], but was explored only for Reed–Solomon codes. Related but slightly 'weaker' notions that apply to arbitrary linear codes in the list decoding regime were introduced in [GKL24; Zei24]. Before we can explain how they differ, we first recall the definition of mutual correlated agreement from [ACFY25].

Definition 3.10 (Mutual correlated agreement for proximity generators)**.** Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. We say that Gen is a proximity generator for $\mathcal{C}$ with mutual correlated agreement with proximity bound B^* and error err^* if for every $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_\ell \in \mathbb{F}^n$ and $\delta \in (0, 1 - \mathsf{B}^*(\mathcal{C}, \ell))$ the following holds:

$$\Pr_{(r_1,\ldots,r_\ell) \leftarrow \mathsf{Gen}(\ell)} \left[\exists S \subset [n] \quad \text{s.t.} \quad \begin{array}{l} |S| \geq (1-\delta)n \ , \\ \exists \boldsymbol{c} \in \mathcal{C} : \sum_{j \in [\ell]} r_j \cdot \boldsymbol{w}_j[S] = \boldsymbol{c}[S] \ , \\ \exists i \in [\ell], \forall \boldsymbol{c}' \in \mathcal{C} : \boldsymbol{w}_i[S] \neq \boldsymbol{c}'[S] \end{array} \right] \leq \mathsf{err}^*(\mathcal{C}, \ell, \delta) \ .$$

The results in [GKL24; Zei24] can be shown to imply results similar to the above definition, but only upon imposing certain additional conditions on δ. However both of these results, as well as most other prior work on mutual correlated agreement only consider proximity generators where:

1. $\mathsf{Gen}(\ell)$ samples $r_i \xleftarrow{\$} \mathbb{F}$, for each $i \in [\ell]$, and outputs $[r_i]_{i \in [\ell]}$.
2. $\mathsf{Gen}(\ell)$ samples $r \xleftarrow{\$} \mathbb{F}$, defines $r_i := r^{i-1}$ for each $i \in [\ell]$, and outputs $[r_i]_{i \in [\ell]}$.

Our setting requires a different kind of proximity generator, which we describe next.

Mutual correlated agreement for multilinear proximity generators. We define a *multilinear proximity generator* to be the generator $\mathsf{Gen}(\ell)$ that on input $\ell = 2^m$ samples $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$ and outputs $[\mathrm{eq}(\langle j - 1 \rangle, \boldsymbol{r})]_{j \in [\ell]}$.

Previous work [DP24; AER24; DG25] has also (implicitly) analyzed multilinear proximity generators, but only proves that the weaker notion of correlated agreement holds in the unique decoding regime. In Appendix A, we prove the following theorem regarding mutual correlated agreement for multilinear proximity generators in the list decoding regime.

Theorem 3.11. *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ and $m \in \mathbb{N}$, with $M := 2^m$. Additionally, let $\epsilon \in (0, 1/2n]$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$ be such that $\{\beta_\epsilon n\} > 1/2$.*[10] *For every $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \in \mathbb{F}^n$, the following holds:*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m} \left[\exists S \subset [n] \quad s.t. \quad \begin{array}{c} |S| = \lceil \beta_\epsilon n \rceil \ , \\ \exists \boldsymbol{c} \in \mathcal{C} : \sum_{j \in [M]} \boldsymbol{w}_j[S] \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}) = \boldsymbol{c}[S] \ , \\ \forall [\boldsymbol{c}_i]_{i \in [M]} \in \mathcal{C}^M : \exists j \in [M] \ s.t. \ \boldsymbol{w}_j[S] \neq \boldsymbol{c}_j[S] \end{array} \right] \leq \tfrac{3^m}{\epsilon^2 |\mathbb{F}|} \ .$$

The proof of this theorem uses the result from [Zei24], and thus inherits the need to impose an additional condition on the 'agreement parameter' β_ϵ, which is analogous to the expression $(1 - \delta)$ from Definition 3.10.

Remark 3.12. In the interest of brevity, we refer to the result in Theorem 3.11 as mutual correlated agreement for multilinear proximity generators, although it does not exactly match Definition 3.10. We think this is reasonable as the additional conditions we impose on the proximity parameter are quite weak and do not impose any limitations on the applicability of the result.

Mutual correlated agreement preserves list decoding. We also prove the following lemma in Appendix A, which is useful in proving that our IORs are RBRTE.

Lemma 3.13. *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ and $m \in \mathbb{N}$, with $M := 2^m$. Additionally, let $\epsilon \in (0, 1/2n]$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$ be such that $\{\beta_\epsilon n\} > 1/2$. Define $\delta := 1 - \lceil \beta_\epsilon n \rceil / n$. For every $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \in \mathbb{F}^n$, the following holds:*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m} \left[\begin{array}{c} \Lambda\left(\mathcal{C}, \sum_{j \in [M]} \boldsymbol{w}_j \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}), \delta\right) \\ \neq \\ \left\{ \sum_{j \in [M]} \boldsymbol{c}_j \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}) : [\boldsymbol{c}_j]_{j \in [M]} \in \Lambda\left(\mathcal{C}^M, [\boldsymbol{w}_j]_{j \in [M]}, \delta\right) \right\} \end{array} \right] \leq \tfrac{3^m}{\epsilon^2 |\mathbb{F}|} \ .$$

In words, this lemma states that states that mutual correlated agreement for multilinear proximity generators preserves list decoding, i.e. applying a multilinear proximity generator and then list decoding gives the same result as first list decoding, and then applying the proximity generator to all possible lists of results. The proof requires the fact that multilinear proximity generators exhibit mutual correlated agreement (Theorem 3.11).

3.4 Codeswitchable codes

Codeswitching is a powerful tool that has been used in past influential works [RR24; RR25; BCFRRZ25] which reduces checking a claim about the proximity of $\boldsymbol{w}$ to $\mathcal{C}$ to checking a claim about the proximity of $\boldsymbol{w}'$ to a (possibly different) code $\mathcal{C}'$. [BMMS25] formalized this as an IOR.

Definition 3.14 (CodeSwitch IOR)**.** CodeSwitch is an IOR from $\mathsf{MEP}^{(\mathcal{C}, \varepsilon, 1)}$ to $\mathsf{MEP}^{(\mathcal{C}', \varepsilon', 1)}$ such that $\mathcal{C}$ and $\mathcal{C}'$ are linear codes with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ and $(\mathbb{F}, n', k, \delta_{\mathcal{C}'})$ respectively, and $\varepsilon \in (\delta_{\mathcal{C}}/8, \delta_{\mathcal{C}}/4)$, and $\varepsilon' \in (\delta_{\mathcal{C}'}/8, \delta_{\mathcal{C}'}/4)$.

[10] We use $\{x\}$ denotes the fractional part of x.

This definition is then used to define *efficiently codeswitchable codes*.

Definition 3.15 (Efficiently codeswitchable codes)**.** If there exists an RBRTE IOR CodeSwitch from $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ to $\mathsf{MEP}^{(\mathcal{C}',\varepsilon',1)}$, for all $\varepsilon \in (\delta_{\mathcal{C}}/8, \delta_{\mathcal{C}}/4)$, such that the prover time is $O(n + T_{\mathcal{C}'})$, the number of oracle queries is $O(\lambda)$, and the verifier time is $O(\lambda \log n)$, then $\mathcal{C}$ is said to be efficiently codeswitchable.

For brevity, in this paper we refer to efficiently codeswitchable codes as codeswitchable codes. [BMMS25] show that tensor codes and LDPC codes are efficiently codeswitchable.

4 Interactive oracle reductions

4.1 Relation prerequisites

Indexed relations. An *indexed relation* $\mathcal{R}$ is a set of triples $\{(\mathbb{i}, \mathbb{x}, \mathbb{w})\}$ where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance, and $\mathbb{w}$ is the witness; the corresponding *indexed language* $\mathcal{L}(\mathcal{R})$ is the set of pairs $(\mathbb{i}, \mathbb{x})$ for which there exists a witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$. We use $\mathcal{R}^{\mathbb{i}}$ to denote the set of all tuples $(\mathbb{x}, \mathbb{w})$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in \mathcal{R}$.

Indexed pair relations. Some proof systems exhibit a gap between completeness and soundness. We formalize this using indexed pair relations of the form $\mathcal{R}_{\mathsf{pair}} = (\mathcal{R}, \tilde{\mathcal{R}})$, where completeness holds for $\mathcal{R}$, but soundness guarantees membership in a *relaxed* indexed relation $\tilde{\mathcal{R}} \supseteq \mathcal{R}$.

Indexed oracle pair relations. Indexed oracle pair relations are indexed pair relations where the instance is split into a *short instance* $\mathbb{x}$ and a vector of *instance oracles* $\vec{\mathbb{y}} = (\mathbb{y}_1, \ldots, \mathbb{y}_{|\vec{\mathbb{y}}|})$. Each instance oracle $\mathbb{y}_i$ is an oracle to a vector over some alphabet Σ. In this work, we only consider indexed oracle pair relations where Σ is a finite field $\mathbb{F}$. We write $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}_{\mathsf{pair}}$ to denote membership in an indexed oracle pair relation $\mathcal{R}_{\mathsf{pair}}$. When $\mathcal{R}_{\mathsf{pair}} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed pair relation, this notation is overloaded and can be used to denote membership in either $\mathcal{R}$ or $\tilde{\mathcal{R}}$ depending on the context.

Indexed polynomial-oracle relations. An indexed polynomial-oracle relation $\mathcal{R}$ is an indexed relation where the instance is split into a *short instance* $\mathbb{x}$ and a vector of *instance polynomials* $\vec{\mathbb{y}} = \{f_1, \ldots, f_{|\vec{\mathbb{y}}|}\}$, where each f_i is a polynomial over $\mathbb{F}$. Unlike indexed oracle pair relations, note that these are *not* pair relations.

4.2 Definition

We recall the definition of *interactive oracle reductions* (IORs) [BCGGRS19; BMNW25; BMMS25; BCFW25]. Intuitively, an IOR from indexed oracle pair relation $\mathcal{R}$ to $\mathcal{R}'$ is an interactive protocol between a prover and a verifier that reduces the verifier's task of deciding whether there exists witness $\mathbb{w}$ s.t. $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}$ to the task of deciding whether there exists witness $\mathbb{w}'$ s.t. $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathcal{R}'$ for some $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}')$ which are determined through the (randomized) execution of the protocol. We provide a formal definition below.

Formal definition. Let $\mathcal{R}$ and $\mathcal{R}'$ be indexed oracle pair relations. A (public-coin, holographic) interactive oracle reduction from $\mathcal{R}$ to $\mathcal{R}'$ is a tuple of polynomial-time algorithms $\mathsf{IOR} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ with the following behavior.

- The indexer $\mathbf{I}$ is a deterministic algorithm which receives as input an index $\mathbb{i}$ (for $\mathcal{R}$). It outputs a *short index* ι, *index oracle* $\vec{\mathbb{I}}$, and a new index $\mathbb{i}'$ (for $\mathcal{R}'$).
- The prover $\mathbf{P}$ is a multi-round algorithm which receives as input the index $\mathbb{i}$, short instance $\mathbb{x}$, instance oracles $\vec{\mathbb{y}}$, and witness $\mathbb{w}$. It engages in k rounds of interaction with the verifier $\mathbf{V}$. In the i-th round, $\mathbf{P}$ sends a proof oracle $\vec{\Pi}_i$ and a non-oracle message msg_i, and then receives a challenge $\vec{r}_i$ from $\mathbf{V}$.
- The verifier $\mathbf{V}$ is a multi-round algorithm which receives as input the short index ι, short instance $\mathbb{x}$, oracle access to index oracle $\vec{\mathbb{I}}$, and oracle access to instance oracles $\vec{\mathbb{y}}$. It engages in k rounds of interaction with $\mathbf{P}$. In the i-th round, $\mathbf{V}$ receives oracle access to a proof oracle $\vec{\Pi}_i$ and a non-oracle message msg_i, and then sends a uniformly random challenge $\vec{r}_i$.

At the end of the interaction, $\mathbf{V}$ outputs a decision $b \in \{0, 1\}$ and a new instance $(\mathbb{x}', \vec{\mathbb{y}}')$. $\vec{\mathbb{y}}'$ is returned implicitly as a function of the old instance oracles $\vec{\mathbb{y}}$, the index oracles $\vec{\mathbb{I}}$, and the prover message oracles $\vec{\Pi}$; this allows the $\mathbf{V}$ to still be succinct. Additionally, $\mathbf{P}$ outputs a new witness $\mathbb{w}'$. We denote the IOR

interaction by

$$(b, \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \leftarrow \left\langle \mathbf{P}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}), \mathbf{V}^{\vec{\mathbb{I}}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}) \right\rangle \quad .$$

Without loss of generality, $\mathbf{P}$ always moves first and $\mathbf{V}$ is split into two phases. In the *interaction phase*, it samples challenges and sends them to $\mathbf{P}$. In the *query phase*, it queries the index, instance, and proof oracles.

Completeness. An IOR is said to be complete if for any $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{R}$,

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

Soundness. An IOR is said to have soundness error ε if for any $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}) \notin \mathcal{L}(\mathcal{R})$ and adversary $\widetilde{\mathbf{P}}$,

$$\Pr\left[b = 1 \wedge (\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \mathcal{R}' \;\middle|\; \begin{array}{l} (\iota, \vec{\mathbb{I}}, \mathbb{i}') \leftarrow \mathbf{I}(\mathbb{i}) \\ (b, \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \leftarrow \left\langle \widetilde{\mathbf{P}}, \mathbf{V}^{\vec{\mathbb{I}}, \vec{\mathbb{y}}, \vec{\Pi}}(\iota, \mathbb{x}) \right\rangle \end{array} \right] \leq \varepsilon(\mathbb{i}, \mathbb{x}).$$

Knowledge soundness. Given that IOPPs are used to construct SNARKs (via IOPs) [ACFY24; ACFY25; BMMS25], it is important to ensure that the non-interactive argument obtained via (a variant of) the BCS transform [BCS16] is knowledge sound. For this, prior work [BCS16; BMNW25] has shown that it is sufficient to prove that an IOP/ IOR satisfies state-restoration knowledge-soundness (SRKS). However, SRKS is an unwieldy definition: it is often difficult to directly prove that a protocol satisfies it.

As a result, prior work has presented easier-to-prove notions of knowledge soundness, including round-by-round extractability (RBRE) [BCFW25] and round-by-round tree-extractability (RBRTE) [BMMS25]. Both of these notions have proven useful in the context of IORs that rely on linear-time encodable codes, which are particularly challenging to prove knowledge soundness for as they are not known to be decodable in polynomial time. Ideally, we would prove our IORs RBRE, as this implies that the non-interactive arguments obtained after applying the (variant of) Fiat–Shamir transform are straight-line extractable. Unfortunately, the existing definitions for round-by-round extractability do not seem to apply to some techniques used in the TensorReduce IOR (see Section 6). As such, we prove that all of our constructions satisfy RBRTE, thereby implying non-interactive arguments that satisfy rewinding knowledge soundness. We recall the definition of RBRTE in Section 4.4.

Efficiency measures. We consider the following efficiency measures for an IOR.

- *Alphabet:* Σ is the set of symbols used to write the index, instance, and proof strings.
- *Round complexity:* k is the number of rounds of interaction in the protocol.
- *Plaintext length:* PLen is the sum of the lengths of the prover messages, $\sum_{i=1}^{\mathsf{k}} |\mathsf{msg}_i|$.
- *Query complexity:* $\mathsf{Queries}$ is the number of locations that $\mathbf{V}$ queries from the index oracles $\vec{\mathbb{I}}$, instance oracles $\vec{\mathbb{y}}$, and proof oracles $\vec{\Pi}$.
- *Oracle length:* OLen is the sum of the lengths of the index oracles $\sum_{j=1}^{|\vec{\mathbb{I}}|} |\mathbb{I}_j|$, instance oracles $\sum_{j=1}^{|\vec{\mathbb{y}}|} |\mathbb{y}_j|$, and proof oracles $\sum_{i=1}^{\mathsf{k}} |\vec{\Pi}_i|$.
- *Indexer and prover time:* $\mathsf{T}_{\mathcal{I}}$ and $\mathsf{T}_{\mathcal{P}}$ is the number of Σ operations performed by $\mathbf{I}$ and $\mathbf{P}$ respectively in the protocol.
- *Verifier time:* $\mathsf{T}_{\mathcal{V}}$ is the number of Σ operations performed by $\mathbf{V}$, *not including the operations required to query the instance oracles* (which is handled by the query complexity).
- *Randomness:* R is the total number of random bits used by $\mathbf{V}$.

When we discuss our IORs in detail, we omit the discussion of the alphabet in our efficiency measures, since we only consider a given finite field $\mathbb{F}$ as the alphabet. Furthermore, we omit the discussion of randomness, since that is already captured by the query complexity and verifier time.

Remark 4.1 (Fixed indices)**.** We sometimes write that an IOR is from $\mathcal{R}_1^{\mathbb{i}_1}$ to $\mathcal{R}_2^{\mathbb{i}_2}$, if we are only concerned with the case when the index for the input from $\mathcal{R}_1$ is always $\mathbb{i}_1$, which implies that the index for $\mathcal{R}_2$ is also fixed to $\mathbb{i}_2$ (which must be the output of the indexer on input $\mathbb{i}_1$).

Definition 4.2 (Interactive Oracle Proofs of Proximity (IOPPs))**.** An IOPP for an indexed oracle pair relation $\mathcal{R}$ is an IOR from $\mathcal{R}$ to the "trivial" relation $\mathcal{R}_{\mathsf{true}} \coloneqq \{(\mathbb{i} \coloneqq \bot, \mathbb{x} \coloneqq \mathsf{accept}, \vec{\mathbb{y}} \coloneqq \bot, \mathbb{w} \coloneqq \bot)\}$, consisting of only one element with an empty witness.

Remark 4.3 (Interactive Oracle Proofs (IOPs))**.** An IOP is simply a special case of an IOPP, where the indexed oracle pair relation $\mathcal{R}$ has an empty instance oracle, i.e., $\mathcal{R}$ consists of tuples of the form $(\mathbb{i}, \mathbb{x}, \bot, \mathbb{w})$.

4.3 Important indexed oracle pair relations

SEP. The *sumprod-evaluation proximity* (SEP) relation consists of ℓ claimed codewords $[\boldsymbol{w}_i]_{i\in[\ell]}$, and enforces a proximity claim about each codeword as well as a sumcheck claim over the multilinear extensions of the messages underlying the ℓ codewords.

Indexed oracle relation 4.4. The *sumprod-evaluation proximity relation* $\mathsf{SEP} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed oracle pair relation with $\mathcal{R}$ and $\tilde{\mathcal{R}}$ both consisting of tuples of the following form:
- index: $(\mathcal{C}, \varepsilon, \ell)$, where $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, $\varepsilon \in (0, 1)$ is the proximity parameter, and $\ell \in \mathbb{N}$,
- instance: $(f(\boldsymbol{X}, \boldsymbol{Z}), \sigma)$, where $f(X_1, \ldots, X_{\log k}, Z_1, \ldots, Z_\ell) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_{\log k+\ell}]$ and $\sigma \in \mathbb{F}$,
- instance oracles: $[\boldsymbol{w}_i]_{i\in[\ell]}$, where $\boldsymbol{w}_i \in \mathbb{F}^n$ for each $i \in [\ell]$,
- witness: $[\boldsymbol{m}_i]_{i\in[\ell]}$, where $\boldsymbol{m}_i \in \mathbb{F}^k$ for each $i \in [\ell]$,

such that

$$\sum_{\boldsymbol{b}\in\{0,1\}^{\log k}} f(\boldsymbol{b}, \hat{m}_1(\boldsymbol{b}), \ldots, \hat{m}_\ell(\boldsymbol{b})) = \sigma \ ,$$

as well as the following distinct conditions for $\mathcal{R}$ and $\tilde{\mathcal{R}}$ respectively

$$\begin{aligned} \mathcal{R}: &\qquad \forall i \in [\ell]: \quad \boldsymbol{w}_i = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ , \\ \tilde{\mathcal{R}}: &\qquad \forall i \in [\ell]: \quad \boldsymbol{w}_i \sim_\varepsilon \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ . \end{aligned}$$

MEP. The *multilinear-evaluation proximity* (MEP) relation consists of ℓ claimed codewords $[\boldsymbol{w}_i]_{i\in[\ell]}$, and enforces both a proximity claim as well as an evaluation claim over the multilinear extension of the message underlying each codeword.

Indexed oracle relation 4.5. The *multilinear-evaluation proximity relation* $\mathsf{MEP} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed oracle pair relation with $\mathcal{R}$ and $\tilde{\mathcal{R}}$ both consisting of tuples of the following form:
- index: $(\mathcal{C}, \varepsilon, \ell)$, where $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, $\varepsilon \in (0, 1)$ is the proximity parameter, and $\ell \in \mathbb{N}$ is the number of polynomials,
- instance: $[(\boldsymbol{z}_i, y_i)]_{i\in[\ell]}$, where for each $i \in [\ell]$: $\boldsymbol{z}_i \in \mathbb{F}^{\log k}$ and $y_i \in \mathbb{F}$,
- instance oracles: $[\boldsymbol{w}_i]_{i\in[\ell]}$, where for each $i \in [\ell]$: $\boldsymbol{w}_i \in \mathbb{F}^n$,

- witness: $[\boldsymbol{m}_i]_{i\in[\ell]}$, where for each $i \in [\ell] : \boldsymbol{m}_i \in \mathbb{F}^k$.

such that $\hat{m}_i(\boldsymbol{z}_i) = y_i$ for all $i \in [\ell]$, and satisfy the following distinct conditions for $\mathcal{R}$ and $\tilde{\mathcal{R}}$ respectively:

$$\begin{aligned} \mathcal{R} : &\quad \forall i \in [\ell] : \quad \boldsymbol{w}_i = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ , \\ \tilde{\mathcal{R}} : &\quad \forall i \in [\ell] : \quad \boldsymbol{w}_i \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ . \end{aligned}$$

HEP. Finally, the *homogeneous multilinear-evaluation proximity relation* (HEP) is a special case of the MEP relation, where the evaluation claim over all ℓ messages is with respect to the same evaluation point $\boldsymbol{z}$.

Indexed oracle relation 4.6. The *homogeneous multilinear-evaluation proximity relation* $\mathsf{HEP} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed oracle pair relation with $\mathcal{R}$ and $\tilde{\mathcal{R}}$ both consisting of tuples of the following form:

- index: $(\mathcal{C}, \varepsilon, \ell)$, where $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, $\varepsilon \in (0, 1)$ is the proximity parameter, and $\ell \in \mathbb{N}$,
- instance: $(\boldsymbol{z}, [y_i]_{i\in[\ell]})$, where $\boldsymbol{z} \in \mathbb{F}^{\log k}$ and for each $i \in [\ell] : y_i \in \mathbb{F}$,
- instance oracles: $[\boldsymbol{w}_i]_{i\in[\ell]}$, where for each $i \in [\ell] : \boldsymbol{w}_i \in \mathbb{F}^n$,
- witness: $[\boldsymbol{m}_i]_{i\in[\ell]}$, where for each $i \in [\ell] : \boldsymbol{m}_i \in \mathbb{F}^k$,

such that $\hat{m}_i(\boldsymbol{z}) = y_i$ for all $i \in [\ell]$ and satisfy the following distinct conditions for $\mathcal{R}$ and $\tilde{\mathcal{R}}$ respectively

$$\begin{aligned} \mathcal{R} : &\quad \forall i \in [\ell] : \quad \boldsymbol{w}_i = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ , \\ \tilde{\mathcal{R}} : &\quad \forall i \in [\ell] : \quad \boldsymbol{w}_i \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ . \end{aligned}$$

The proximity threshold of a code. While in the foregoing definitions we allow the proximity parameter to be any value in the range $(0, 1)$, in most of our IORs we must restrict the proximity parameter to be at most a certain threshold, which we call the *proximity threshold* of the code denoted by $\varepsilon(\mathcal{C})$. The necessity of this threshold stems from the fact that the soundness of our IORs relies on mutual correlated agreement for multilinear proximity generators (see Section 3.3), which is only proven to hold up to this threshold.

Definition 4.7 (Proximity threshold)**.** Let $\epsilon = 1/2n$. We define the *proximity threshold* of a code $\mathcal{C}$ with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ to be the largest value $\varepsilon(\mathcal{C}) \in (0, \delta_{\mathcal{C}})$ such that there exists an integer $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$ that satisfies the following conditions:

- $\{\beta_\epsilon n\} > 1/2$, where $\{x\}$ denotes the fractional part of x, and
- $\lceil (1 - \varepsilon(\mathcal{C}))n \rceil - 1 < \beta_\epsilon n \leq \lceil (1 - \varepsilon(\mathcal{C}))n \rceil$.

The foregoing conditions imply that the proximity threshold $\varepsilon(\mathcal{C})$ must be in the 'list decoding regime', i.e. $\mathcal{C}$ is $(\varepsilon(\mathcal{C}), \ell)$-list decodable for some $\ell = \mathrm{poly}(n)$.

The Sumcheck IOR. We recall the following IOR from $\mathsf{SEP}^{(\mathcal{C},\varepsilon,\ell)}$ to $\mathsf{HEP}^{(\mathcal{C},\varepsilon,\ell)}$ [BMMS25].

Lemma 4.8 ([BMMS25, Lemma 5.7])**.** *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. For all $\varepsilon \in (0, \delta_{\mathcal{C}}/2)$, there exists an RBRTE IOR,* Sumcheck*, from $\mathsf{SEP}^{(\mathcal{C},\varepsilon,\ell)}$ to $\mathsf{HEP}^{(\mathcal{C},\varepsilon,\ell)}$, with the following efficiency measures:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$\log k$	$O(\ell \log k)$	0	0	$O(1)$	$O(t_f \cdot (\log k + \ell) \cdot n)$	$O(t_f(\log k + \ell) + \ell \log k)$

where t_f is the number of terms in the combination polynomial f.

4.4 Round-by-round tree-extractability

In this section we recall the definition of round-by-round tree-extractability (RBRTE) from [BMMS25].

Constrained transcript trees. Before we define RBRTE, we must define constrained transcript trees. At a high level, a constrained transcript tree T of an IOR with respect to the input $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$ is a 'tree of accepting transcripts' for the IOR run on the given input. In more detail, the nodes of the tree are labelled with prover messages (with the nodes in layer i containing prover messages for round i) and the edges of the tree are labelled with verifier challenges (with outgoing edges from layer i containing verifier challenges for round i), such that every path from the root to a leaf corresponds to an accepting view for the IOR verifier $\mathcal{V}$ with input corresponding to $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$. Additionally, we require that the tree satisfies certain constraints, which we capture as 'challenge' and 'vertical' predicates. We present a formal definition below.

Definition 4.9 (Constrained transcript tree)**.** Let $\mathsf{IOR} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ be a k-round IOR from an indexed oracle relation $\mathcal{R}$ to an index oracle relation $\mathcal{R}'$, and let

- $\vec{a} := a_1, \ldots, a_{\mathsf{k}} \in \mathbb{N}$ be arity parameters,
- for all $i \in [\mathsf{k}]$, VD_i and PD_i be the domain of possible verifier and prover messages in the i-th round of the interaction respectively.
- φ be a 'vertical' boolean-output predicate which takes as input an index-instance pair, an IOR transcript of the form $(\alpha_1, \rho_1, \ldots, \alpha_{\mathsf{k}}, \rho_{\mathsf{k}})$ where $\alpha_i := (\Pi_i, \mathsf{msg}_i)$ is the i-th prover message and ρ_i is the i-th verifier challenge, and
- θ be a 'challenge' boolean-output predicate which takes as input an ordered set of verifier challenges from $\mathsf{VD}_1^{a_1} \times \mathsf{VD}_2^{a_1 a_2} \times \ldots \times \mathsf{VD}_{\mathsf{k}}^{\prod_{i \in [\mathsf{k}]} a_i}$.

We say that a $(\vec{a}, \varphi, \theta, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$-constrained transcript tree for IOR is a depth-k tree T that satisfies the following properties:

1. Each node in the i-th layer has a_i-many children.[11]
2. For every layer $i \in [\mathsf{k}]$: every node in layer i is labeled with an IOR prover message for round i.
3. For every layer $i \in [\mathsf{k}]$: the a_i outgoing edges are labeled with distinct verifier challenges.
4. Every path from the root to a leaf corresponds to an accepting view for the IOR verifier $\mathcal{V}$ with inputs corresponding to $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$, such that the verifier outputs $(\mathbb{x}', \vec{\mathbb{y}}')$ consistent with the value in the leaf.
5. The leaf nodes are labeled with tuples $(\mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}')$ such that $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}') \in \vec{\mathcal{R}}'$, where $\mathbb{i}'$ is the output of $\mathbf{I}$ on input $\mathbb{i}$.
6. The vertical predicate is satisfied: for every path p down the tree, the transcript $(\alpha_1, \rho_1, \ldots, \alpha_{\mathsf{k}}, \rho_{\mathsf{k}})$ corresponding to p satisfies
$$\varphi(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_{\mathsf{k}}, \rho_{\mathsf{k}}) = 1 \ .$$
7. The challenge predicate is satisfied:
$$\theta(\rho(\mathsf{T})) = 1 \ ,$$
where we use $\rho(\mathsf{T})$ to denote the ordered set of all verifier challenges on the edges of T.

For brevity, we will sometimes use $(\vec{a}, \varphi, \theta, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$-tree to denote $(\vec{a}, \varphi, \theta, \mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$-constrained transcript tree.

Round-by-round tree-extractability definition. We start by defining vertical and challenge predicates, where we use VD_i and PD_i to denote the domain of possible verifier and prover messages in the i-th round of the interaction respectively.

[11] Such a tree has $\mathsf{k}+1$ layers, with layer 1 being the root layer and layer $\mathsf{k}+1$ being the leaf layer, and the tree has $\prod_{i \in [\mathsf{k}]} a_i$ leaves.

Definition 4.10 ($(\overrightarrow{\mathsf{VD}}, \vec{a})$-challenge-predicate)**.** Consider a k-round IOR, IOR, from $\mathcal{R}$ to $\mathcal{R}'$. We say that θ is a $(\overrightarrow{\mathsf{VD}}, \vec{a})$-challenge-predicate if for $\rho(\mathsf{T}) \xleftarrow{\$} \mathsf{VD}_1^{a_1} \times \mathsf{VD}_2^{a_1 a_2} \times \ldots \times \mathsf{VD}_{\mathsf{k}}^{\prod_{i\in[\mathsf{k}]} a_i}$, i.e. each challenge in $\rho(\mathsf{T})$ is independently sampled from the corresponding VD_i, we have

$$\Pr_{\rho(\mathsf{T}) \xleftarrow{\$} \mathsf{VD}_1^{a_1} \times \mathsf{VD}_2^{a_1 a_2} \times \ldots \times \mathsf{VD}_{\mathsf{k}}^{\prod_{i\in[\mathsf{k}]} a_i}} [\theta(\rho(\mathsf{T})) = 0] < \mathrm{negl}(\lambda) \ .$$

Definition 4.11 ($(\overrightarrow{\mathsf{PD}}, \overrightarrow{\mathsf{VD}}, \mathsf{k})$-vertical-predicate)**.** Consider a k-round IOR, IOR, from $\mathcal{R}$ to $\mathcal{R}'$. We say that $\varphi : (\mathsf{PD}_1 \times \mathsf{VD}_1) \times \ldots \times (\mathsf{PD}_{\mathsf{k}} \times \mathsf{VD}_{\mathsf{k}}) \to \{0,1\}$[12] is a $(\overrightarrow{\mathsf{PD}}, \overrightarrow{\mathsf{VD}}, \mathsf{k})$-vertical-predicate for IOR if the following properties hold:

- The function returns 1 on an empty transcript. That is,

$$\varphi(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \varnothing) = 1 \ .$$

- The output value is unchanged by prover messages. That is, for all $i \in [\mathsf{k}]$:

$$\varphi(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \alpha_1, \rho_1, \ldots, \alpha_{i-1}, \rho_{i-1}, \alpha_i) = \varphi(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \alpha_1, \rho_1, \ldots, \alpha_{i-1}, \rho_{i-1}) \ ,$$

  for any $\alpha_i \in \mathsf{PD}_i$.
- The probability of the output value going from 1 to 0 by a verifier challenge is $\mathrm{negl}(\lambda)$. That is, for all $i \in [\mathsf{k}]$ and $(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \alpha_1, \rho_1, \ldots, \alpha_i)$ such that $\varphi(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \alpha_1, \rho_1, \ldots, \alpha_i) = 1$, the following holds:

$$\Pr_{\rho_i \xleftarrow{\$} \mathsf{VD}_i} [\varphi(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \alpha_1, \rho_1, \ldots, \alpha_i, \rho_i) = 0] \leq \mathrm{negl}(\lambda) \ .$$

Using these definitions, we can now define round-by-round tree-extractability.

Definition 4.12 (Round-by-round tree-extractable IORs)**.** Consider IOR, an IOR from $\mathcal{R}$ to $\mathcal{R}'$, and $\mathtt{i}$, an index of $\tilde{\mathcal{R}}$. IOR is said to be $(\vec{a}, \varphi, \theta)$-*round-by-round tree-extractable (RBRTE)* with respect to index $\mathtt{i}$ if the following conditions hold:

- for any instance $(\mathtt{x}, \vec{\mathtt{y}})$, we have $\prod_{i=1}^{\mathsf{k}} a_i = \mathrm{poly}(|(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}})|)$,
- φ is a $(\overrightarrow{\mathsf{PD}}, \overrightarrow{\mathsf{VD}}, \mathsf{k})$-vertical-predicate,
- θ is a $(\overrightarrow{\mathsf{VD}}, \vec{a})$-challenge-predicate, and
- there exists a deterministic polynomial-time extractor $\mathcal{E}^{\mathrm{Tree}}$ such that for any instance $(\mathtt{x}, \vec{\mathtt{y}})$, $\mathcal{E}^{\mathrm{Tree}}$ takes as input a $(\vec{a}, \varphi, \theta, \mathtt{i}, \mathtt{x}, \vec{\mathtt{y}})$-constrained transcript tree T for IOR, and outputs a witness $\mathtt{w}$ such that $(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}}, \mathtt{w}) \in \tilde{\mathcal{R}}$.

Furthermore, we say IOR is RBRTE if for each index $\mathtt{i}$, there exists a tuple $(\vec{a}, \varphi, \theta)$ such that IOR is $(\vec{a}, \varphi, \theta)$-RBRTE with respect to $\mathtt{i}$.

Main results about RBRTE IORs. [BMMS25] prove the following useful results pertaining to RBRTE IORs.

Lemma 4.13 (Compilation of RBRTE IORs to NIRK [BMMS25, Lemma 4.5])**.** *Let* IOR *be an RBRTE IOR from* $\mathcal{R}$ *to* $\mathcal{R}'$*. Then the non-interactive reduction obtained by applying (a variant of) the BCS transform (from [BMNW25, Appendix B]) to* IOR *is a NIRK with rewinding knowledge soundness.*

[12] For brevity, we abuse notation and avoid mentioning the domain for $(\mathtt{i}, \mathtt{x}, \vec{\mathtt{y}})$.

Lemma 4.14 (RBRTE IOR composition [BMMS25, Lemma 4.7])**.** *Let* IOR *be an* $(\vec{a}, \varphi, \theta)$*-RBRTE* IOR *from* $\mathcal{R}$ *to* $\mathcal{R}'$ *and* IOR' *be an* $(\vec{a}', \varphi', \theta')$*-RBRTE* IOR *from* $\mathcal{R}'$ *to* $\mathcal{R}''$*. Then the sequential composition* $\mathsf{IOR}'' \coloneqq \mathsf{IOR}' \circ \mathsf{IOR}$ *is a* $(\vec{a} \,\|\, \vec{a}', \varphi \,\|\, \varphi', \theta \wedge \theta')$*-RBRTE IOR from* $\mathcal{R}$ *to* $\mathcal{R}''$*.*[13]

Furthermore, the efficiency metrics of $\mathsf{IOR}' \circ \mathsf{IOR}$ *are the sum of the round complexities, oracle lengths, message lengths, query complexities, prover times, and verifier times.*

[13] At a high level, the "concatenation" $\varphi \,\|\, \varphi'$ of the vertical predicates of a k-round and a k'-round IOR respectively is defined in the natural way: on transcripts of length less than k it behaves like φ, and on transcripts of length $> \mathsf{k}$, it outputs 1 if and only if φ outputs 1 on the first k prover and verifier messages and φ outputs 1 on the remaining prover and verifier messages.

5 Lossy batching

At a high level, the lossy homogenous multilinear-evaluation proximity (LHEP) relation is a relaxation of the HEP relation that asks that out of the ℓ-many proximity and evaluation claims that the LHEP instance comprises of, at least a $(1-\gamma_{\mathsf{lost}})$ fraction are true (whereas in the HEP relation all of them must are true).

Indexed oracle relation 5.1. The *lossy homogenous multilinear-evaluation proximity relation* $\mathsf{LHEP} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed oracle pair relation with $\mathcal{R}$ and $\tilde{\mathcal{R}}$ both consisting of tuples of the following form:

- index: $(\mathcal{C}, \varepsilon, \ell, \gamma_{\mathsf{lost}})$ where
  - $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ that is efficiently codeswitchable,
  - $\varepsilon \in \varepsilon(\mathcal{C})$ is the proximity parameter,
  - $\ell \in \mathbb{N}$ is the number of polynomials,
  - $\gamma_{\mathsf{lost}} \in (0,1)$ is the loss parameter,
- instance: $(\boldsymbol{z}, [y_i]_{i\in[\ell]})$, where $\boldsymbol{z} \in \mathbb{F}^{\log k}$ and for each $i \in [\ell] : y_i \in \mathbb{F}$,
- instance oracle: $[\boldsymbol{w}_i]_{i\in[\ell]}$, where for each $i \in [\ell] : \boldsymbol{w}_i \in \mathbb{F}^n$,
- witness: $[\boldsymbol{m}_i]_{i\in[\ell]}$, where for each $i \in [\ell] : \boldsymbol{m}_i \in \mathbb{F}^k$.

such that there exists a set $I \subseteq [\ell]$ that satisfies the conditions $|I| \geq (1-\gamma_{\mathsf{lost}}) \cdot \ell$ and

$$\forall i \in I : \quad \hat{m}_i(\boldsymbol{z}) = y_i \ ,$$

as well as following distinct conditions for $\mathcal{R}$ and $\tilde{\mathcal{R}}$ respectively

$$\begin{aligned} \mathcal{R} : &\qquad \forall i \in [\ell] : \quad \boldsymbol{w}_i = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i), \\ \tilde{\mathcal{R}} : &\qquad \forall i \in I : \quad \boldsymbol{w}_i \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \ . \end{aligned}$$

Below, we present LossyBatch, an IOR from $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',2)}$, where $\mathcal{C}'$ has parameters $(\mathbb{F}, n', k', \delta_{\mathcal{C}'})$ such that $k' = nk\ell$, and $\varepsilon' \in \varepsilon(\mathcal{C}')$ is new the proximity parameter. The verifier of this IOR makes only $O(\lambda + \ell/\gamma_{\mathsf{lost}})$ oracle queries in total, which is simply $O(\lambda)$ in the setting where this is IOR is invoked, since γ_{lost} is at least a constant and ℓ is always $O(\lambda)$.

5.1 The LossyBatch IOR

An $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ claim consists of tuples of the form

$$\mathbb{i} = (\mathcal{C}, \varepsilon, \ell, \gamma_{\mathsf{lost}}) \qquad \mathbb{x} = (\boldsymbol{z}, [y_i]_{i\in[\ell]}), \qquad \vec{\mathbb{y}} = [\boldsymbol{w}_i]_{i\in[\ell]}, \qquad \mathbb{w} = [\boldsymbol{m}_i]_{i\in[\ell]} \ ,$$

and the claim that there exists $I \subseteq [\ell]$ of size at least $(1-\gamma_{\mathsf{lost}}) \cdot \ell$ such that

$$\forall i \in I : \quad \boldsymbol{w}_i \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \quad \text{and} \quad \hat{m}_i(\boldsymbol{z}) = y_i \ . \tag{3}$$

The LossyBatch IOR proceeds as follows. $\mathcal{P}$ sends an oracle $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$, where $\boldsymbol{w}' = \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}')$ and $\boldsymbol{m}' \coloneqq \boldsymbol{m}^n_{\mathsf{conc}} \coloneqq (\boldsymbol{m}_1 \,\|\, \ldots \,\|\, \boldsymbol{m}_\ell)^n \in \mathbb{F}^{k'}$. $\mathcal{P}$ then proves the following claims about $\boldsymbol{w}'$:

1. **New proximity claim:** $\boldsymbol{w}' \sim_{\varepsilon'} \mathcal{C}'$.
2. **Out-of-domain evaluation claim:** $\hat{m}'(\boldsymbol{z}_{\mathsf{ood}}) = \sigma_{\mathsf{ood}}$, where $\boldsymbol{z}_{\mathsf{ood}} \in \mathbb{F}^{\log k'}$ is an out-of-domain evaluation point sent by $\mathcal{V}$.
3. **Repetition claim:** The underlying message $\boldsymbol{m}'$ is of the form $\boldsymbol{m}' = \boldsymbol{m}^n_{\mathsf{conc}}$ for some $\boldsymbol{m}_{\mathsf{conc}} \in \mathbb{F}^{k\ell}$.
4. **Multi-evaluation claim:** Let $\boldsymbol{m}_{\mathsf{first}} \in \mathbb{F}^{k\ell}$ be the first $k\ell$ elements of $\boldsymbol{m}'$. Then, for all $i \in [\ell]$, we have $\hat{m}_{\mathsf{first}}(\langle i-1 \rangle, \boldsymbol{z}) = y_i$.

5. **Multi-proximity claim:** There exists some $I \subseteq [\ell]$ such that $|I| \geq (1-\gamma_{\mathsf{lost}})\ell$ and for all $i \in I$ we have $\boldsymbol{w}_i \sim_\varepsilon \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik])$.

Intuitively, proving Claim 1 and Claim 2 binds $\mathcal{P}$ to a single underlying message $\boldsymbol{m}' \in \mathbb{F}^{k'}$. Proving Claim 3 restricts $\boldsymbol{m}'$ to be of the form $\boldsymbol{m}' = \boldsymbol{m}_{\mathsf{conc}}^n$ for some $\boldsymbol{m}_{\mathsf{conc}} \in \mathbb{F}^{k\ell}$. Finally, proving Claim 4 and Claim 5 proves Eq. (3). In the rest of this section, we show how to use a single $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',2)}$ instance to prove all of these claims simultaneously. Since Claim 1 is enforced by any SEP instance involving $\boldsymbol{w}'$, we focus on describing the sumcheck instance (the function f and target sum σ) for proving the remaining claims, which we break down into 4 sub-instances that we tie together in the end.

Out-of-domain evaluation claim. This is simply an evaluation claim over the multilinear polynomial $\hat{m}'$, which can be expressed as $\langle \boldsymbol{m}', \mathbf{eq}(\boldsymbol{z}_{\mathsf{ood}})\rangle = \sigma_{\mathsf{ood}}$. The corresponding SEP sub-instance is described in Eq. (5).

Repetition claim. Claim 3 is akin to proving that $\hat{m}'(\boldsymbol{x}, \boldsymbol{y}) = \hat{m}'(\boldsymbol{x}', \boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{F}^{\log n}$ and $\boldsymbol{y} \in \{0,1\}^{\log k + \log \ell}$. $\mathcal{V}$ samples $\boldsymbol{r}_R \xleftarrow{\$} \mathbb{F}^{\log k + \log \ell}$ and sends it to $\mathcal{P}$, who replies with a claimed value σ_{rep}. $\mathcal{V}$ now needs to check that $\hat{m}'(\boldsymbol{x}, \boldsymbol{r}_R) = \sigma_{\mathsf{rep}}$ for all $\boldsymbol{x} \in \mathbb{F}^{\log n}$. Therefore, $\mathcal{V}$ samples $\boldsymbol{r}_L \xleftarrow{\$} \mathbb{F}^{\log n}$ and sends it to $\mathcal{P}$, who then proves to $\mathcal{V}$ that $\hat{m}'(\boldsymbol{r}_L, \boldsymbol{r}_R) = \sigma_{\mathsf{rep}}$. This claim can be expressed as an inner product claim as follows. Define $\boldsymbol{q}_{\mathsf{rep}} \coloneqq \mathbf{eq}(\boldsymbol{r}_L) \otimes \mathbf{eq}(\boldsymbol{r}_R) \in \mathbb{F}^{k'}$, and check that $\langle \boldsymbol{m}', \boldsymbol{q}_{\mathsf{rep}}\rangle \ = \sigma_{\mathsf{rep}}$. If this check passes, then $\hat{m}'(\boldsymbol{X}, \boldsymbol{y})$ is a constant polynomial in $\boldsymbol{X}$ for all $\boldsymbol{y} \in \{0,1\}^{\log k + \log \ell}$ by Lemma 3.2. The corresponding SEP sub-instance is described in Eq. (6).

Multi-evaluation claim. At this point, $\mathcal{V}$ is convinced that $\boldsymbol{m}'$ is of the form $\boldsymbol{m}' = \boldsymbol{m}_{\mathsf{conc}}^n$ for some $\boldsymbol{m}_{\mathsf{conc}} \in \mathbb{F}^{k\ell}$. Let $\boldsymbol{m}_{\mathsf{conc}} = (\boldsymbol{m}'_1, \ldots, \boldsymbol{m}'_\ell)$, where $\boldsymbol{m}'_i \in \mathbb{F}^k$ for each $i \in [\ell]$. $\mathcal{P}$ can prove the ℓ evaluation claims, that $\hat{m}'_i(\boldsymbol{z}) = y_i$ for all $i \in [\ell]$, via a single SEP instance over the multilinear polynomial $\hat{m}'$. To do this, $\mathcal{V}$ samples random challenges $\beta_{\mathsf{eval}} \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$, and then $\mathcal{P}$ proves to $\mathcal{V}$ that

$$\sigma_{\mathsf{eval}} \coloneqq \sum_{i=1}^{\ell} \beta_{\mathsf{eval}}^i \cdot y_i = \sum_{i=1}^{\ell} \beta_{\mathsf{eval}}^i \cdot \hat{m}'(0^{\log n}, \langle i-1\rangle, \boldsymbol{z}) \ .$$

This is because $\hat{m}'(0^{\log n}, \langle i-1\rangle, \boldsymbol{z}) = \hat{m}'_i(\boldsymbol{z})$, for each $i \in [\ell]$. Thus the right-hand side can be reinterpreted as $\langle \boldsymbol{m}', \boldsymbol{q}_{\mathsf{eval}}\rangle$ where $\boldsymbol{q}_{\mathsf{eval}}$ is defined as

$$\boldsymbol{q}_{\mathsf{eval}} \coloneqq \left(\left(\beta_{\mathsf{eval}} \cdot \mathbf{eq}(\boldsymbol{z}) \,\|\, \beta_{\mathsf{eval}}^2 \cdot \mathbf{eq}(\boldsymbol{z}) \,\|\, \ldots \,\|\, \beta_{\mathsf{eval}}^\ell \cdot \mathbf{eq}(\boldsymbol{z})\right) \,\|\, 0^{(k'-k\ell)}\right) \ .$$

This SEP sub-instance is described in more detail in Eq. (7).

Multi-proximity claim. By proving Claims 1 to 4, $\mathcal{P}$ has proven several claims about the message $\boldsymbol{m}'$ underlying $\boldsymbol{w}'$, but it has not yet connected $\boldsymbol{m}'$ back to the input codewords $[\boldsymbol{w}_i]_{i\in[\ell]}$. $\mathcal{P}$ wants to prove that there exists $I \subseteq [\ell]$ of size at least $(1-\gamma_{\mathsf{lost}}) \cdot \ell$, that satisfies the conditions in Eq. (3) with respect to $[\boldsymbol{m}'_i]_{i\in[\ell]}$. This can be done as follows:

1. For each $i \in [\ell]$, $\mathcal{V}$ queries $[\![\boldsymbol{w}_i]\!]$ at $\nu \coloneqq \left\lceil \frac{1}{\gamma_{\mathsf{lost}}} \right\rceil$ random indices $[\xi_{i,i'}]_{i\in[\ell],i'\in[\nu]}$ to obtain values $[v_{i,i'}]_{i\in[\ell],i'\in[\nu]}$ respectively.
2. For each $i \in [n]$, let $\boldsymbol{g}_i$ be the i-th row of $\boldsymbol{G}_{\mathcal{C}} \in \mathbb{F}^{n\times k}$, the generator matrix of $\mathcal{C}$. $\mathcal{P}$ wants to prove to $\mathcal{V}$ that the following claim:

$$\forall i \in [\ell], i' \in [\nu] : \ v_{i,i'} = \langle \boldsymbol{g}_{\xi_{i,i'}}, \boldsymbol{m}'_i\rangle \tag{4}$$

We will show shortly how to set up an SEP sub-instance that enforces Eq. (4), but first we justify why it is sufficient for $\mathcal{V}$ to check that Eq. (4) is true. Say there exists a set $B \subseteq [\ell]$ such that $|B| > \gamma_{\mathsf{lost}} \cdot \ell$ and

$\delta(\boldsymbol{w}_i, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}'_i)) > \varepsilon$ for all $i \in B$. For each $i \in B$, let bad_i be the set of "bad" indices of $\boldsymbol{w}_i$ (i.e. for each $j \in \mathsf{bad}_i$, $\boldsymbol{w}_i[j] \neq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}'_i)[j]$). Then the probability that none of the indices in $[\xi_{i,i'}]_{i' \in [\nu]}$ are in bad_i is at most $(1-\varepsilon)^{\nu}$, for each $i \in B$. Thus, the probability that this is true of all $i \in B$ is at most

$$((1-\varepsilon)^{\nu})^{|B|} \;\leq\; (1-\varepsilon)^{\nu\ell\gamma_{\mathsf{lost}}} \;\leq\; 2^{\log(1-\varepsilon)\cdot\ell} \quad .$$

If $\ell \geq \lceil -\lambda / \log(1-\varepsilon) \rceil$ (which is the setting in which the IOPP for tensor codes invokes the LossyBatch IOR in Section 7), then this soundness error is negligible in λ. Therefore, if Eq. (3) is not true, then Eq. (4) is true only with negligible probability.

Finishing touches. We mention a few details of the final IOR that we have omitted in our overview thus far.

1. We describe how the checks in Eq. (4) can be captured using a single SEP sub-instance. In the indexing phase, $\mathcal{I}$ outputs $[\![\boldsymbol{w}_G]\!]$, where $\boldsymbol{w}_G \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}_G)$ with

$$\boldsymbol{m}_G \coloneqq (\boldsymbol{g}_1^{\ell}, \boldsymbol{g}_2^{\ell}, \ldots, \boldsymbol{g}_n^{\ell}) \in \mathbb{F}^{k'} \quad .$$

   In the interactive phase, $\mathcal{V}$ sends $\beta_{\mathsf{prox}} \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$, which will be used to batch together the $\ell\nu$ many inner product claims in Eq. (4). Define $\boldsymbol{q}_{\mathsf{prox}} \coloneqq (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n)$, where for each $s \in [n]$, $\boldsymbol{q}_s \coloneqq (\boldsymbol{q}_{s,1}, \boldsymbol{q}_{s,2}, \ldots, \boldsymbol{q}_{s,\ell})$ such that for each $s \in [n], i \in [\ell]$,

$$\boldsymbol{q}_{s,i} \coloneqq \begin{cases} \beta_{\mathsf{prox}}^{i+(i'-1)\ell} \cdot \mathbf{1}^k & \text{if there exists } i' \in [\nu] \text{ such that } \xi_{i,i'} = s \\ \mathbf{0}^k & \text{otherwise} \end{cases}$$

   We use $\boldsymbol{q}_{\mathsf{prox}}$ to define an SEP instance that enforces the checks in Eq. (4) by checking that

$$\sum_{i \in [\ell], i' \in [\nu]} \beta_{\mathsf{prox}}^{i+(i'-1)\ell} \cdot v_{i,i'} = \left\langle \begin{matrix} \boldsymbol{m}', \\ \boldsymbol{m}_G, \\ \boldsymbol{q}_{\mathsf{prox}} \end{matrix} \right\rangle$$

$$\implies \sum_{i \in [\ell], i' \in [\nu]} \beta_{\mathsf{prox}}^{i+(i'-1)\ell} \cdot v_{i,i'} = \left\langle \begin{matrix} (\boldsymbol{m}_{\mathsf{conc}}, \boldsymbol{m}_{\mathsf{conc}}, \ldots, \boldsymbol{m}_{\mathsf{conc}}), \\ (\boldsymbol{g}_1^{\ell}, \boldsymbol{g}_2^{\ell}, \ldots, \boldsymbol{g}_n^{\ell}), \\ (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n) \end{matrix} \right\rangle \quad .$$

2. To simplify the overall workflow, we batch all four SEP sub-instances corresponding to Claims 2 to 5 together with respect to a random challenge $\rho \xleftarrow{\$} \mathbb{F}$, and output a single SEP instance. We provide more details on this in towards the end of Fig. 6.

Remark 5.2 (verifier efficiency)**.** We note that in all the aforementioned SEP sub-instances, we have been careful in ensuring that $\mathcal{V}$ can efficiently evaluate $\hat{q}_{\mathsf{ood}}(\boldsymbol{X})$, $\hat{q}_{\mathsf{rep}}(\boldsymbol{X})$, $\hat{q}_{\mathsf{eval}}(\boldsymbol{X})$ and $\hat{q}_{\mathsf{prox}}(\boldsymbol{X})$ on its own.

Full construction. We present the full construction of the LossyBatch IOR in Fig. 6.

Indexer input: $\mathbb{i} = (\mathcal{C}, \varepsilon, \ell, \gamma_{\mathsf{lost}})$.
Indexer output: The indexer defines $k' \coloneqq n \cdot k \cdot \ell$ and outputs

- $\mathbb{i}' \coloneqq (\mathcal{C}', \varepsilon', 2)$, where $\mathcal{C}'$ is a linear code with parameters $(\mathbb{F}, n', k', \delta_{\mathcal{C}'})$ such that $k' = n \cdot k \cdot \ell$.
- $\iota \coloneqq \perp$, and
- $\vec{\mathbb{I}} \coloneqq \boldsymbol{w}_G$, where $\boldsymbol{w}_G$ is defined as follows:
  1. Let $\boldsymbol{G}_{\mathcal{C}} \in \mathbb{F}^{n \times k}$ be the generator matrix of $\mathcal{C}$, and let $\boldsymbol{g}_i \in \mathbb{F}^k$ be the i-th row of $\boldsymbol{G}_{\mathcal{C}}$.

2. Define $\boldsymbol{m}_G \coloneqq (\boldsymbol{g}_1^\ell, \boldsymbol{g}_2^\ell, \ldots, \boldsymbol{g}_n^\ell) \in \mathbb{F}^{k'}$ and $\boldsymbol{w}_G \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}_G)$.

Input instance: $\mathbb{x} = (\boldsymbol{z}, [y_i]_{i\in[\ell]})$.
Input instance oracle: $\vec{\mathbb{y}} = [\boldsymbol{w}_i]_{i\in[\ell]}$.
Input witness: $\mathbb{w} = [\boldsymbol{m}_i]_{i\in[\ell]}$.

$\left\langle \mathcal{P}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}), \mathcal{V}^{\vec{\mathbb{I}},\vec{\mathbb{y}}}(\iota, \mathbb{x}) \right\rangle$:

1. Define $\boldsymbol{m}_{\mathsf{conc}} \coloneqq (\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_\ell)$ and $\boldsymbol{m}' \coloneqq \boldsymbol{m}_{\mathsf{conc}}^n$.
2. $\mathcal{P}$ computes $\boldsymbol{w}' \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}')$ and sends $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$.

Setting up the SEP statement that proves Claim 2.

3. $\mathcal{V}$ samples an out-of-domain evaluation point $\boldsymbol{z}_{\mathsf{ood}} \xleftarrow{\$} \mathbb{F}^{\log k'}$ and sends it to $\mathcal{P}$.
4. $\mathcal{P}$ computes $\sigma_{\mathsf{ood}} \coloneqq \hat{m}'(\boldsymbol{z}_{\mathsf{ood}})$ and sends it to $\mathcal{V}$.
5. Define

$$\begin{aligned} &f_{\mathsf{ood}}(X_1, \ldots, X_{\log k'}, Z_1, Z_2) \coloneqq \hat{q}_{\mathsf{ood}}(\boldsymbol{X}) \cdot Z_1, \text{ where} \\ &\boldsymbol{q}_{\mathsf{ood}} \coloneqq \mathbf{eq}(\boldsymbol{z}_{\mathsf{ood}}) \ . \end{aligned} \tag{5}$$

Setting up the SEP statement that proves Claim 3.

6. $\mathcal{V}$ samples a random evaluation point $\boldsymbol{r}_R \xleftarrow{\$} \mathbb{F}^{\log k + \log \ell}$ and sends it to $\mathcal{P}$.
7. $\mathcal{P}$ computes $\sigma_{\mathsf{rep}} \coloneqq \hat{m}_{\mathsf{conc}}(\boldsymbol{r}_R)$ and sends it to $\mathcal{V}$.
8. $\mathcal{V}$ samples another random evaluation point $\boldsymbol{r}_L \xleftarrow{\$} \mathbb{F}^{\log n}$ and sends it to $\mathcal{P}$.
9. Define

$$\begin{aligned} &f_{\mathsf{rep}}(X_1, \ldots, X_{\log k'}, Z_1, Z_2) \coloneqq \hat{q}_{\mathsf{rep}}(\boldsymbol{X}) \cdot Z_1, \text{ where} \\ &\boldsymbol{q}_{\mathsf{rep}} \coloneqq \mathbf{eq}(\boldsymbol{r}_L) \otimes \mathbf{eq}(\boldsymbol{r}_R) \ . \end{aligned} \tag{6}$$

Setting up the SEP statement that proves Claim 4.

10. $\mathcal{V}$ samples random challenges $\beta_{\mathsf{eval}} \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
11. Define $\sigma_{\mathsf{eval}} \coloneqq \sum_{i\in[\ell]} \beta_{\mathsf{eval}}^i \cdot y_i$ and

$$\begin{aligned} &f_{\mathsf{eval}}(X_1, \ldots, X_{\log k'}, Z_1, Z_2) \coloneqq \hat{q}_{\mathsf{eval}}(\boldsymbol{X}) \cdot Z_1, \text{ where} \\ &\boldsymbol{q}_{\mathsf{eval}} \coloneqq \left(\beta_{\mathsf{eval}} \cdot \mathbf{eq}(\boldsymbol{z}) \parallel \beta_{\mathsf{eval}}^2 \cdot \mathbf{eq}(\boldsymbol{z}) \parallel \ldots \parallel \beta_{\mathsf{eval}}^\ell \cdot \mathbf{eq}(\boldsymbol{z})\right) \parallel 0^{(k'-k\ell)} \ . \end{aligned} \tag{7}$$

Setting up the SEP statement that proves Claim 5.

12. $\mathcal{V}$ samples indices $\xi_{i,i'} \xleftarrow{\$} [n]$ for each $i \in [\ell]$ and $i' \in [\nu]$, and $\beta_{\mathsf{prox}} \xleftarrow{\$} \mathbb{F}$ and sends them to $\mathcal{P}$.
13. For each $i \in [\ell]$ and $i' \in [\nu]$, $\mathcal{V}$ queries $[\![\boldsymbol{w}_i]\!]$ at index $\xi_{i,i'}$ to obtain $v_{i,i'} \coloneqq \boldsymbol{w}_i[\xi_{i,i'}]$.
14. Define $\sigma_{\mathsf{prox}} \coloneqq \sum_{i\in[\ell],i'\in[\nu]} \beta_{\mathsf{prox}}^{i+(i'-1)\ell} \cdot v_{i,i'}$ and

$$\begin{aligned} &f_{\mathsf{prox}}(X_1, \ldots, X_{\log k'}, Z_1, Z_2) \coloneqq \hat{q}_{\mathsf{prox}}(\boldsymbol{X}) \cdot Z_1 \cdot Z_2, \text{ where} \\ &\boldsymbol{q}_{\mathsf{prox}} \coloneqq (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n) \ , \end{aligned} \tag{8}$$

where for each $i'' \in [n]$, $\boldsymbol{q}_{i''} \coloneqq (\boldsymbol{q}_{i'',1}, \boldsymbol{q}_{i'',2}, \ldots, \boldsymbol{q}_{i'',\ell})$, where for each $i \in [\ell]$, $\boldsymbol{q}_{i'',i} \coloneqq \beta_{\mathsf{prox}}^{i+(i'-1)\ell} \cdot \mathbf{1}^k$ if there exists $i' \in [\nu]$ such that $\xi_{i,i'} = i''$, and $\boldsymbol{q}_{i'',i} \coloneqq \mathbf{0}^k$ otherwise.

Batching the SEP statements.

15. $\mathcal{V}$ samples a random challenge $\rho \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
16. Define $f \coloneqq f_{\mathsf{ood}} + \rho \cdot f_{\mathsf{rep}} + \rho^2 \cdot f_{\mathsf{eval}} + \rho^3 \cdot f_{\mathsf{prox}}$ and $\sigma \coloneqq \sigma_{\mathsf{ood}} + \rho \cdot \sigma_{\mathsf{rep}} + \rho^2 \cdot \sigma_{\mathsf{eval}} + \rho^3 \cdot \sigma_{\mathsf{prox}}$.
17. Define $\mathbb{x}' \coloneqq (f(\boldsymbol{X}, \boldsymbol{Z}), \sigma)$, $\vec{\mathbb{y}}' \coloneqq (\boldsymbol{w}', \boldsymbol{w}_G)$, $\mathbb{w}' \coloneqq (\boldsymbol{m}', \boldsymbol{m}_G)$.
18. $\mathcal{P}$ receives as output $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}')$ and $\mathcal{V}$ receives as output $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}')$.

Figure 6: The LossyBatch IOR from $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',2)}$.

Lemma 5.3. *Let $\varepsilon \in (0, \varepsilon(\mathcal{C})]$ and $\varepsilon' \in (0, \varepsilon(\mathcal{C}')]$, where $\varepsilon(\mathcal{C})$ is the proximity threshold of $\mathcal{C}$ (see Definition 4.7).* LossyBatch *is an RBRTE IOR from* $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ *to* $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',2)}$ *with the following parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
3	$O(1)$	$O(\ell/\gamma_{\mathsf{lost}})$	$O(nk\ell)$	$O(nk\ell + T_{\mathcal{C}'})$	$O(nk\ell + T_{\mathcal{C}'})$	$O(\ell/\gamma_{\mathsf{lost}} + \log n)$

We defer the proof of this lemma to Appendix B.1.

Remark 5.4 (Out-of-domain sample is extraneous)**.** We note that the out-of-domain sample $\boldsymbol{z}_{\mathsf{ood}}$ is not strictly necessary for the soundness of the IOR, since $(\boldsymbol{r}_L, \boldsymbol{r}_R)$ together can also act as an out-of-domain sample, in addition to enforcing the repetition claim. For clarity of exposition and the RBRTE proof, we have chosen to keep the two checks separate. The only advantage of bundling the checks together is that the number rounds of interaction is reduced to 2 (and some minor savings in soundness error and communication).

6 The TensorReduce IOR

Let $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ be a tensor code where $\mathcal{C}_1$ and $\mathcal{C}_2$ are linear codes with parameters $(\mathbb{F}, n_1, k_1, \delta_1)$ and $(\mathbb{F}, n_2, k_2, \delta_2)$ respectively. Also define $n \coloneqq n_1 n_2$ and $k \coloneqq k_1 k_2$. Looking ahead, we need to enforce that $n_2 = \lceil \lambda \ln 2 \rceil n_1 k_1$. Recall that an $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ claim consists of tuples of the form $(\mathbb{x} = (\boldsymbol{z}, y), \vec{\mathbb{y}} = \boldsymbol{w}, \mathbb{w} = \boldsymbol{m})$, such that the following claim holds:

$$\boldsymbol{w} \sim_\varepsilon \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{m}) \quad \text{and} \quad \hat{m}(\boldsymbol{z}) = y \ . \tag{9}$$

Randomizing the evaluation point. Looking ahead, we will use mutual correlated agreement (see Section 3.3) to reduce the aforementioned claims to smaller claims. Thus, the first step of the IOR is to reduce the claims in Eq. (9) to an evaluation claim at a random evaluation point $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\log k}$, as opposed to an arbitrary evaluation point $\boldsymbol{z}$. If $\boldsymbol{r}_1 \coloneqq \boldsymbol{r}[1 : \log k_1]$ and $\boldsymbol{r}_2 \coloneqq \boldsymbol{r}[\log k_1 + 1 : \log k]$, then we can use $\boldsymbol{r}_1$ as the 'folding' challenge for applying mutual correlated agreement.

This reduction is done via sumcheck and it is a standard technique used in previous IOPPs [ZCF24]. In more detail, $\hat{m}(\boldsymbol{z})$ can expressed as

$$\hat{m}(\boldsymbol{z}) = \sum_{i \in [k_1 k_2]} \boldsymbol{m}[i] \cdot \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{z}) \ ,$$

Consider the polynomial $\hat{v} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_{\log k}]$ defined as $\hat{v}(\boldsymbol{X}) \coloneqq \hat{m}(\boldsymbol{X}) \cdot \mathrm{eq}(\boldsymbol{X}, \boldsymbol{z})$. Then, we can rewrite the evaluation claim as

$$\hat{m}(\boldsymbol{z}) = \sum_{\boldsymbol{b} \in \{0,1\}^{\log k}} \hat{v}(\boldsymbol{b}) = y \ .$$

The sumcheck protocol reduces this claim to checking that $\hat{v}(\boldsymbol{r}) = \sigma_{\boldsymbol{r}}$, where $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\log k}$ and $\sigma_{\boldsymbol{r}}$ is a value obtained from the sumcheck protocol. Since $\hat{v}(\boldsymbol{r}) = \hat{m}(\boldsymbol{r}) \cdot \mathrm{eq}(\boldsymbol{r}, \boldsymbol{z})$, we have thus reduced the claim in Eq. (9) to the following claim:

$$\boldsymbol{w} \sim_\varepsilon \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{m}) \quad \text{and} \quad \hat{m}(\boldsymbol{r}) = y_{\boldsymbol{r}} \ , \tag{10}$$

where $y_{\boldsymbol{r}} \coloneqq \sigma_{\boldsymbol{r}} \cdot \mathrm{eq}(\boldsymbol{r}, \boldsymbol{z})^{-1}$. We formalize this with the following RandomizeEval protocol.

$\mathsf{RandomizeEval}\langle \mathcal{P}(\boldsymbol{z}, y, \hat{m}(\boldsymbol{X})), \mathcal{V}(\boldsymbol{z}, y) \rangle \to (\boldsymbol{r}, y_{\boldsymbol{r}})$

1. Given $\boldsymbol{z} \in \mathbb{F}^\mu$, $y \in \mathbb{F}$ and $\hat{m}(\boldsymbol{X}) \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\mu]$, define $v(\boldsymbol{X}) \coloneqq \hat{m}(\boldsymbol{X}) \cdot \mathrm{eq}(\boldsymbol{X}, \boldsymbol{z})$.
2. Define $\sigma_1 \coloneqq y$.
3. For $i \in [\mu]$:
4. $\quad \mathcal{P}$ sends $v_i(X) \coloneqq \sum_{\boldsymbol{b} \in \{0,1\}^{\mu - i}} v(r_1, \ldots, r_{i-1}, X, \boldsymbol{b})$ to $\mathcal{V}$.
5. $\quad \mathcal{V}$ checks that $v_i(0) + v_i(1) = \sigma_i$.
6. $\quad \mathcal{V}$ samples $r_i \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
7. $\quad$ Define $\sigma_{i+1} \coloneqq v_i(r_i)$.
8. Define $y_{\boldsymbol{r}} \coloneqq \sigma_{\mu+1} \cdot \mathrm{eq}(\boldsymbol{r}, \boldsymbol{z})^{-1}$.
9. $\mathcal{P}$ and $\mathcal{V}$ receive as output $(\boldsymbol{r}, y_{\boldsymbol{r}})$.

Sending the new oracle. Similar to the intuition provided in Section 2.5, $\mathcal{P}$ initializes the following polynomial:

$$\boldsymbol{m}_{\boldsymbol{r}_1} \coloneqq \sum_{i \in [k_1]} \boldsymbol{M}[i, \cdot] \cdot \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{r}_1) \ ,$$

where $\boldsymbol{M}$ is the matrix form of $\boldsymbol{m}$. We additionally define $\boldsymbol{r}_2 \coloneqq \boldsymbol{r}[\log k_1 + 1 : \log k]$.

In Section 2.5, we said that $\mathcal{P}$ sends $[\![\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})]\!]$ to $\mathcal{V}$. However, due to some technicalities that arise due to the invocation of SpotCheckReduce later on, $\mathcal{P}$ initializes $\boldsymbol{m}' \coloneqq \boldsymbol{m}_{\boldsymbol{r}_1}^{n_2/k_2}$ ($\boldsymbol{m}_{\boldsymbol{r}_1}$ repeated n_2/k_2 times), and sends $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$ where $\boldsymbol{w}' \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}')$.

No need to query new oracle. In Section 2.5, we said that $\mathcal{V}$ queries the new oracle $\boldsymbol{w}'$ at $\ell \coloneqq \lceil \lambda \ln 2 \rceil$ many random indices. However, this causes a *loss in proximity parameter*. That is, if $\mathcal{V}$ performs the column spot checks as described in Section 2.5 in order to prove Eq. (10), then the new claim about $\boldsymbol{w}'$ is a $\mathsf{MEP}^{(\mathcal{C}_2, \varepsilon - \varepsilon_b, 1)}$ claim where ε_b is a constant. This is not necessarily a problem for our full construction of TensorIOPP, since we only need to apply this reduction a constant number of times, before switching over to WHIR. However, this does increase the practical query complexity of the IOPP, since the proximity parameter of the output SEP claim is now $\varepsilon - \varepsilon_b$, which requires more queries to achieve the same soundness error. However, by implementing these column spot checks using codeswitching, we can avoid this loss in proximity parameter, and thus minimize the practical query overhead of these checks. This reduction proceeds as follows. Say $\mathcal{V}$ samples ℓ many random indices $[\xi_i]_{i \in [\ell]}$ from $[n_2]$ and sends them to $\mathcal{P}$. Instead of $\mathcal{V}$ querying $\boldsymbol{w}'$ at $[\xi_i]_{i \in [\ell]}$, $\mathcal{P}$ convinces $\mathcal{V}$ that

$$\forall i \in [\ell] : \quad \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}^*)[\xi_i] = y_i \ , \tag{11}$$

where y_i is the claimed value of $\boldsymbol{w}'[\xi_i]$. We capture this claim in the following relation.

Indexed oracle relation 6.1 (Spot check relation)**.** The *spot check relation* $\mathsf{Spot} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed oracle pair relation with $\mathcal{R}$ and $\tilde{\mathcal{R}}$ both consisting of tuples of the following form:

- index: $(\mathcal{C}, \mathcal{C}', \varepsilon, \ell)$ where $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, $\mathcal{C}'$ is a linear code with parameters $(\mathbb{F}, n', n, \delta_{\mathcal{C}'})$, $\varepsilon \in \varepsilon(\mathcal{C}')$ is the proximity parameter, and $\ell \in \mathbb{N}$ is the number of spot checks,
- instance: $([\xi_i]_{i \in [\ell]}, [y_i]_{i \in [\ell]})$, where $(\xi_1, \ldots, \xi_\ell) \in [n]^\ell$ are the spot check indices and $(y_1, \ldots, y_\ell) \in \mathbb{F}^\ell$ are the claimed values,
- instance oracles: $\boldsymbol{w}' \in \mathbb{F}^{n'}$, a purported codeword in $\mathcal{C}'$,
- witness: $\boldsymbol{m}' \in \mathbb{F}^n$,

where $\boldsymbol{m}' = \boldsymbol{m}^{(n/k)}$ for some $\boldsymbol{m} \in \mathbb{F}^k$, such that $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})[\xi_i] = y_i$ for all $i \in [\ell]$. Additionally, the following conditions are satisfied for $\mathcal{R}$ and $\tilde{\mathcal{R}}$ respectively:

$$\begin{aligned} \mathcal{R} : &\quad \boldsymbol{w}' = \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}') \ , \\ \tilde{\mathcal{R}} : &\quad \boldsymbol{w}' \sim_\varepsilon \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}') \ . \end{aligned}$$

In Section 6.1, we construct the SpotCheckReduce IOR from $\mathsf{Spot}^{(\mathcal{C}, \mathcal{C}', \varepsilon, \ell)}$ to $\mathsf{SEP}^{(\mathcal{C}', \varepsilon, 3)}$, which can be batched into the output SEP claim of TensorReduce.

Reduction to LHEP claim. To perform the column checks as described in Section 2.5, $\mathcal{P}$ and $\mathcal{V}$ set up a new LHEP claim that there exists $I \subseteq [\ell]$ such that $|I| > (1 - \gamma_{\mathsf{lost}})\ell$ and

$$((\boldsymbol{r}_1, [y_i]_{i \in I}), [\boldsymbol{W}[\cdot, \xi_i]]_{i \in I}) \in \mathcal{L}(\mathsf{HEP}^{(\mathcal{C}_1, \varepsilon_{\mathsf{int}}, \ell)}) \ ,$$

where $\varepsilon_{\mathsf{int}} = 1 - \sqrt{1 - \varepsilon}$ and $\gamma_{\mathsf{lost}} = \varepsilon_{\mathsf{int}} - \sqrt{2\varepsilon_{\mathsf{int}}}$. $\mathcal{P}$ and $\mathcal{V}$ can then run LossyBatch to reduce the LHEP claim to an SEP claim, which is then batched into the output SEP claim.

The full construction. We present the full construction of TensorReduce in Fig. 7.

Indexer input: $\mathbb{i} = (\mathcal{T}, \varepsilon, 1)$.
Indexer output: Parse $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ where $\mathcal{C}_1$ and $\mathcal{C}_2$ are linear codes with parameters $(\mathbb{F}, n_1, k_1, \delta_1)$ and $(\mathbb{F}, n_2, k_2, \delta_2)$ respectively, such that $n_2 = \lceil \lambda \ln 2 \rceil n_1 k_1$. Define $n \coloneqq n_1 n_2$ and $k \coloneqq k_1 k_2$. Define the following parameters:

- $\varepsilon_{\mathsf{int}} \coloneqq 1 - \sqrt{1-\varepsilon}$,
- $\ell \coloneqq \lceil \lambda \ln 2 \rceil$, and
- $\gamma_{\mathsf{lost}} \coloneqq \varepsilon_{\mathsf{int}} - \sqrt{2\varepsilon_{\mathsf{int}}}$.

Let $\mathcal{C}'$ be a linear code with parameters $(\mathbb{F}, n', k', \delta_{\mathcal{C}'})$ such that $k' = n_1 k_1 \ell$, and let $\varepsilon' \in \varepsilon(\mathcal{C}')$ be a new proximity parameter. Define the input indices for the invocations of LossyBatch and SpotCheckReduce as follows:

- $\mathbb{i}_{\mathsf{Spot}} \coloneqq (\mathcal{C}_2, \mathcal{C}', \varepsilon', \ell)$, with target output index $(\mathcal{C}', \varepsilon', 3)$, and
- $\mathbb{i}_{\mathsf{LHEP}} \coloneqq (\mathcal{C}_1, \varepsilon_{\mathsf{int}}, \ell, \gamma_{\mathsf{lost}})$, with target output index $(\mathcal{C}', \varepsilon', 2)$.

The overall output of the indexer is

$$\mathbb{i}' \coloneqq (\mathcal{C}', \varepsilon', 5) \ , \ \iota \coloneqq (\iota_{\mathsf{Spot}}, \iota_{\mathsf{LHEP}}) \ , \ \vec{\mathbb{I}} \coloneqq (\vec{\mathbb{I}}_{\mathsf{Spot}}, \vec{\mathbb{I}}_{\mathsf{LHEP}}) \ ,$$

where ι_{Spot}, $\vec{\mathbb{I}}_{\mathsf{Spot}}$, ι_{LHEP}, and $\vec{\mathbb{I}}_{\mathsf{LHEP}}$ are the short index and index oracles obtained by running the indexers of SpotCheckReduce and LossyBatch respectively.

Input instance: $\mathbb{x} = (\boldsymbol{z}, y)$ where $\boldsymbol{z} \in \mathbb{F}^{\log k_1 + \log k_2}$ and $y \in \mathbb{F}$.
Input instance oracle: $\vec{\mathbb{y}} = \boldsymbol{w} \in \mathbb{F}^{n_1 n_2}$.
Input witness: $\mathbb{w} = \boldsymbol{m} \in \mathbb{F}^{k_1 k_2}$.

$\left\langle \mathcal{P}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}), \mathcal{V}^{\vec{\mathbb{I}}, \vec{\mathbb{y}}}(\iota, \mathbb{x}) \right\rangle$:

1. $\mathcal{P}$ and $\mathcal{V}$ run RandomizeEval $\langle \mathcal{P}(\boldsymbol{z}, y, \hat{m}(\boldsymbol{X})), \mathcal{V}(\boldsymbol{z}, y) \rangle \to (\boldsymbol{r}, y_{\boldsymbol{r}})$.
2. Define $\boldsymbol{r}_1 \coloneqq \boldsymbol{r}[1 : \log k_1]$ and $\boldsymbol{r}_2 \coloneqq \boldsymbol{r}[\log k_1 + 1 : \log k]$.
3. $\mathcal{P}$ computes $\boldsymbol{m}_{\boldsymbol{r}_1} \coloneqq \sum_{i \in [k_1]} \boldsymbol{M}[i, \cdot] \cdot \mathrm{eq}(\langle i-1 \rangle, \boldsymbol{r}_1)$, and $\boldsymbol{w}_{\boldsymbol{r}_1} \coloneqq \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})$.
4. $\mathcal{P}$ computes $\boldsymbol{m}' \coloneqq \boldsymbol{m}_{\boldsymbol{r}_1}^{(n_2/k_2)} \in \mathbb{F}^{n_2}$, and $\boldsymbol{w}' \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}')$, and sends $[\![\boldsymbol{w}']\!]$ to $\mathcal{V}$.
5. $\mathcal{V}$ samples $\boldsymbol{z}_{\mathsf{ood}} \xleftarrow{\$} \mathbb{F}^{\log n_2}$ and sends it to $\mathcal{P}$.
6. $\mathcal{P}$ computes $\sigma_{\mathsf{ood}} \coloneqq \hat{m}'(\boldsymbol{z}_{\mathsf{ood}})$ and sends it to $\mathcal{V}$.
7. $\mathcal{V}$ samples indices $\xi_i \xleftarrow{\$} [n_2]$ for each $i \in [\ell]$ and sends $[\xi_i]_{i \in [\ell]}$ to $\mathcal{P}$.

Running SpotCheckReduce**.**

8. $\mathcal{P}$ defines $y_i \coloneqq \boldsymbol{w}_{\boldsymbol{r}_1}[\xi_i]$, for each $i \in [\ell]$, and sends $[y_i]_{i \in [\ell]}$ to $\mathcal{V}$.
9. Define the following $\mathsf{Spot}^{(\mathcal{C}_2, \mathcal{C}', \varepsilon', \ell)}$ claim:
$$\mathbb{x}_{\mathsf{Spot}} \coloneqq ([\xi_i]_{i \in [\ell]}, [y_i]_{i \in [\ell]}), \ \ \vec{\mathbb{y}}_{\mathsf{Spot}} \coloneqq \boldsymbol{w}', \ \ \mathbb{w}_{\mathsf{Spot}} \coloneqq \boldsymbol{m}' \ .$$
10. $\mathcal{P}$ and $\mathcal{V}$ run SpotCheckReduce$\langle \mathcal{P}(\mathbb{i}_{\mathsf{Spot}}, \mathbb{x}_{\mathsf{Spot}}, \vec{\mathbb{y}}_{\mathsf{Spot}}, \mathbb{w}_{\mathsf{Spot}}), \mathcal{V}^{\vec{\mathbb{I}}_{\mathsf{Spot}}, \vec{\mathbb{y}}_{\mathsf{Spot}}}(\iota_{\mathsf{Spot}}, \mathbb{x}_{\mathsf{Spot}}) \rangle$ and output the SEP claim $(\mathbb{x}_2, \vec{\mathbb{y}}_2, \mathbb{w}_2)$.
11. Parse $\mathbb{x}_2$ as $(f_{\mathsf{Spot}}(\boldsymbol{X}, Z_1, Z_2, Z_3), \sigma_{\mathsf{Spot}})$, $\vec{\mathbb{y}}_2$ as $(\boldsymbol{w}', \boldsymbol{w}_{\mathsf{Spot}}, \boldsymbol{w}'_{\mathsf{Spot}})$, and $\mathbb{w}_2$ as $(\boldsymbol{m}', \boldsymbol{m}_{\mathsf{Spot}}, \boldsymbol{m}'_{\mathsf{Spot}})$.

Running LossyBatch**.**

12. Define $\boldsymbol{W}_{\mathsf{int}} \in \mathbb{F}^{k_1 \times n_2}$ such that $\boldsymbol{W}_{\mathsf{int}}[i,\cdot] \leftarrow \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{M}[i,\cdot])$ for all $i \in [k_1]$.
13. Define the following $\mathsf{LHEP}^{(\mathcal{C}_1,\varepsilon_{\mathsf{int}},\ell,\gamma_{\mathsf{lost}})}$ claim:

$$\mathbb{x}_{\mathsf{LHEP}} := (\boldsymbol{r}_1, [y_i]_{i\in[\ell]}), \quad \vec{\mathbb{y}}_{\mathsf{LHEP}} := [\boldsymbol{W}[\cdot,\xi_i]]_{i\in[\ell]}, \quad \mathbb{w}_{\mathsf{LHEP}} := [\boldsymbol{W}_{\mathsf{int}}[\cdot,\xi_i]]_{i\in[\ell]} \ .$$

14. $\mathcal{P}$ and $\mathcal{V}$ run $\mathsf{LossyBatch}\left\langle \mathcal{P}(\mathbb{i}_{\mathsf{LHEP}}, \mathbb{x}_{\mathsf{LHEP}}, \vec{\mathbb{y}}_{\mathsf{LHEP}}, \mathbb{w}_{\mathsf{LHEP}}), \mathcal{V}^{\vec{\mathbb{I}}_{\mathsf{LHEP}},\vec{\mathbb{y}}_{\mathsf{LHEP}}}(\iota_{\mathsf{LHEP}}, \mathbb{x}_{\mathsf{LHEP}}) \right\rangle$ and output the SEP claim $(\mathbb{x}_3, \vec{\mathbb{y}}_3, \mathbb{w}_3)$.
15. Parse $\mathbb{x}_3$ as $(f_{\mathsf{LHEP}}(\boldsymbol{X}, Z_4, Z_5), \sigma_{\mathsf{LHEP}})$, $\vec{\mathbb{y}}_3$ as $(\boldsymbol{w}_{\mathsf{LHEP}}, \boldsymbol{w}'_{\mathsf{LHEP}})$, and $\mathbb{w}_3$ as $(\boldsymbol{m}_{\mathsf{LHEP}}, \boldsymbol{m}'_{\mathsf{LHEP}})$.

Batching the SEP claims.

16. $\mathcal{V}$ samples a random challenge $\rho \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
17. Define $\sigma' := y_{\boldsymbol{r}} + \rho \cdot \sigma_{\mathsf{ood}} + \rho^2 \cdot \sigma_{\mathsf{Spot}} + \rho^3 \cdot \sigma_{\mathsf{LHEP}}$ and

$$\begin{aligned} f'(\boldsymbol{X}, \boldsymbol{Z}) :=& \mathrm{eq}(\boldsymbol{X}, (0^{\log n - \log k} \,\|\, \boldsymbol{r}_2)) \cdot Z_1 + \rho \cdot \mathrm{eq}(\boldsymbol{X}, \boldsymbol{z}_{\mathsf{ood}}) \cdot Z_1 \\ &+ \rho^2 \cdot f_{\mathsf{Spot}}(\boldsymbol{X}, Z_1, Z_2, Z_3) + \rho^3 \cdot f_{\mathsf{LHEP}}(\boldsymbol{X}, Z_4, Z_5) \ . \end{aligned}$$

18. Define

$$\begin{aligned} &\mathbb{x}' := (f'(\boldsymbol{X}, \boldsymbol{Z}), \sigma') \ , \quad \vec{\mathbb{y}}' := (\boldsymbol{w}', \boldsymbol{w}_{\mathsf{Spot}}, \boldsymbol{w}'_{\mathsf{Spot}}, \boldsymbol{w}_{\mathsf{LHEP}}, \boldsymbol{w}'_{\mathsf{LHEP}}) \ , \\ &\mathbb{w}' := (\boldsymbol{m}', \boldsymbol{m}_{\mathsf{Spot}}, \boldsymbol{m}'_{\mathsf{Spot}}, \boldsymbol{m}_{\mathsf{LHEP}}, \boldsymbol{m}'_{\mathsf{LHEP}}) \ . \end{aligned}$$

19. $\mathcal{P}$ receives as output $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}')$ and $\mathcal{V}$ receives as output $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}')$.

Figure 7: TensorReduce, an IOR from $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',5)}$.

Lemma 6.2. *Let $\mathcal{C}$ and $\mathcal{C}'$ be as defined in Fig. 7. Additionally, let $\varepsilon \in (0, \min(\delta_{\mathcal{C}}/5, \varepsilon(\mathcal{C}))]$ and $\varepsilon' \in (0, \varepsilon(\mathcal{C}')]$, where $\varepsilon(\mathcal{C})$ is the proximity threshold of $\mathcal{C}$ (see Definition 4.7).* TensorReduce *is an RBRTE IOR from $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',5)}$ with the following efficiency parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$\log k_1 + 2\log k_2 + 8$	$O(\log k)$	$O(\lambda/\varepsilon)$	$O(n')$	$O(n + T_{\mathcal{C}'})$	$O(n + T_{\mathcal{C}'})$	$O(\lambda/\varepsilon + \log n)$

where $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ is a tensor code such that $\mathcal{C}_1$ has parameters $(\mathbb{F}, n_1, k_1, \delta_1)$ and $\mathcal{C}_2$ has parameters $(\mathbb{F}, n_2, k_2, \delta_2)$, and $k = k_1 k_2$ and $n = n_1 n_2$.

We defer the proof of this lemma to Appendix B.2.

6.1 The SpotCheckReduce IOR

In this section we present the SpotCheckReduce IOR from $\mathsf{Spot}^{(\mathcal{C},\mathcal{C}',\varepsilon,\ell)}$ (see Indexed oracle relation 6.1) to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon,3)}$, where $\mathcal{C}$ is a tensor code, and $\mathcal{C}'$ is linear-time encodable. Before we present the full construction, we first give a high level overview of SpotCheckReduce. Firstly, we recall the following lemma from [BMMS25].

Lemma 6.3 ([BMMS25], Lemma 6.4)**.** *Consider a tensor code $\mathcal{C} = \mathcal{B} \otimes \mathcal{B}$ with generator matrix $\boldsymbol{G}_{\mathcal{C}} = \boldsymbol{G}_{\mathcal{B}} \otimes \boldsymbol{G}_{\mathcal{B}}$, where $\mathcal{B}$ is a linear code with parameters $(\mathbb{F}, \sqrt{n}, \sqrt{k}, \delta_{\mathcal{B}})$ and generator matrix $\boldsymbol{G}_{\mathcal{B}} \in \mathbb{F}^{\sqrt{n} \times \sqrt{k}}$. Say $\hat{C} : \mathbb{F}^{\log n + \log k} \to \mathbb{F}$ is the unique multilinear polynomial such that $\hat{C}(\langle x-1\rangle, \langle y-1\rangle) = \boldsymbol{G}_{\mathcal{C}}[x][y]$ for all $x \in [n]$ and $y \in [k]$. Also $\hat{G}_{\mathcal{B}} : \mathbb{F}^{(\log n + \log k)/2} \to \mathbb{F}$ is the unique multilinear polynomial such that $\hat{G}_{\mathcal{B}}(\langle x-1\rangle, \langle y-1\rangle) = \boldsymbol{G}_{\mathcal{B}}[x][y]$ for all $x \in [\sqrt{n}]$ and $y \in [\sqrt{k}]$. Then we have that*

$$\hat{C}(\boldsymbol{x}, \boldsymbol{y}) = \hat{G}_{\mathcal{B}}(\boldsymbol{x}_L, \boldsymbol{y}_L) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{x}_R, \boldsymbol{y}_R) \ ,$$

where $\boldsymbol{x}_L, \boldsymbol{y}_L$ are the first halves of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively, and $\boldsymbol{x}_R, \boldsymbol{y}_R$ are the second halves of $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively.

Recall that to prove a Spot claim, $\mathcal{P}$ needs to prove that the message $\boldsymbol{m}'$ underlying $[\![\boldsymbol{w}']\!]$ is of the form $\boldsymbol{m}^{(n/k)}$ such that

$$\forall i \in [\ell] : \sum_{\boldsymbol{y} \in \{0,1\}^{\log k}} \hat{C}(\langle \xi_i - 1\rangle, \boldsymbol{y}) \cdot \hat{m}(\boldsymbol{y}) = y_i \ . \tag{12}$$

Checking the repetition claim can be done as in Section 5.1. That is, $\mathcal{V}$ sends $\boldsymbol{u}_R \in \mathbb{F}^{\log k}$ to $\mathcal{P}$, and $\mathcal{P}$ replies with σ_{rep}, the claimed value of $\hat{m}(\boldsymbol{u}_R)$. $\mathcal{V}$ then samples $\boldsymbol{u}_L \in \mathbb{F}^{\log n - \log k}$ and $\mathcal{P}$ proves to $\mathcal{V}$ that $\hat{m}'(\boldsymbol{u}_L, \boldsymbol{u}_R) = \sigma_{\mathsf{rep}}$ via sumcheck.

The setup. At a high level, the goal is to give $\mathcal{V}$ polynomial oracle access to $\hat{m}$, $\hat{G}_{\mathcal{B}}$ and $\hat{w}$, where $\hat{w}$ is the multilinear extension of $\boldsymbol{w} \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})$. However, because these polynomial oracles must be realized via codewords of $\mathcal{C}'$, we must ensure that the underlying messages have length n (the message length of $\mathcal{C}'$).

- Firstly, $\mathcal{V}$ already has access to $[\![\boldsymbol{w}]\!]$ as part of the instance oracle of Spot, which gives it polynomial oracle access to $\hat{m}$.
- Secondly, $\mathcal{V}$ gets polynomial oracle access to $\hat{G}_{\mathcal{B}}$ as follows. Define $\boldsymbol{b} \in \mathbb{F}^{\sqrt{nk}}$ as the vector obtained by concatenating the rows of $\boldsymbol{G}_{\mathcal{B}}$. Also define $\boldsymbol{m}_{\mathcal{B}} \coloneqq \boldsymbol{b}^{\sqrt{n/k}}$ and $\boldsymbol{w}_{\mathcal{B}} \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}_{\mathcal{B}})$. The indexer $\mathcal{I}$ computes $\boldsymbol{w}_{\mathcal{B}}$ and outputs $[\![\boldsymbol{w}_{\mathcal{B}}]\!]$. This gives $\mathcal{V}$ polynomial oracle access to $\hat{G}_{\mathcal{B}}$.
- Finally, in the beginning of the interactive phase, $\mathcal{P}$ computes $\boldsymbol{w} \coloneqq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})$ and sends $[\![\boldsymbol{w}_c]\!]$ to $\mathcal{V}$, where $\boldsymbol{w}_c \coloneqq \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{w})$, and gives $\mathcal{V}$ polynomial oracle access to $\boldsymbol{w}$.

Note that the following IOR we are going to describe *requires no oracle queries*, and therefore the following exposition exclusively deals with $\hat{m}, \hat{G}_{\mathcal{B}}, \hat{w}$ as opposed to any encodings.

Proving the spot checks. To begin, $\mathcal{V}$ samples $\beta \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$. Define the selector polynomial $\hat{s} : \mathbb{F}^{\log n} \to \mathbb{F}$ as follows:

$$\hat{s}(\boldsymbol{x}) \coloneqq \begin{cases} \beta^{i-1} & \text{if } \boldsymbol{x} = \langle \xi_i - 1\rangle \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{13}$$

Additionally, we define $\sigma \coloneqq \sum_{i=1}^{\ell} \beta^{i-1} \cdot \boldsymbol{w}[\xi_i]$. At the cost of incurring a soundness loss of $\mathrm{negl}(\lambda)$, due to Lemma 3.1, Eq. (12) can be rewritten as

$$\sum_{\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^{\log k + \log n}} \hat{C}(\boldsymbol{x}, \boldsymbol{y}) \cdot \hat{s}(\boldsymbol{x}) \cdot \hat{m}(\boldsymbol{y}) = \sigma \ . \tag{14}$$

Using Lemma 6.3, we can rewrite Eq. (14) as follows:

$$\sum_{\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^{\log k + \log n}} \hat{G}_{\mathcal{B}}(\boldsymbol{x}_L, \boldsymbol{y}_L) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{x}_R, \boldsymbol{y}_R) \cdot \hat{s}(\boldsymbol{x}) \cdot \hat{m}(\boldsymbol{y}) = \sigma \ . \tag{15}$$

While this is a single sumcheck claim, it cannot be proven efficiently via a naive application of sumcheck as the sum is over a domain of size $k \cdot n$. Therefore, we need to decompose this claim into sumcheck claims over smaller domains.

Reducing size of Eq. (15). Proving Eq. (15) is equivalent to proving the following claims:

$$\forall \boldsymbol{x} \in \{0,1\}^{\log n} : \quad \hat{c}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \{0,1\}^{\log k}} \hat{G}_{\mathcal{B}}(\boldsymbol{x}_L, \boldsymbol{y}_L) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{x}_R, \boldsymbol{y}_R) \cdot \hat{m}(\boldsymbol{y}) \ , \tag{16}$$

$$\sum_{\boldsymbol{x} \in \{0,1\}^{\log n}} \hat{c}(\boldsymbol{x}) \cdot \hat{s}(\boldsymbol{x}) = \sigma \ . \tag{17}$$

Eq. (17) is already a sumcheck claim over a domain of size n. Eq. (16) is true if and only if both sides are equal as polynomials. Therefore, in order to prove Eq. (16), $\mathcal{V}$ can send a random challenge $\boldsymbol{t} \in \mathbb{F}^n$ to $\mathcal{P}$, and $\mathcal{P}$ can return a value $\tau \in \mathbb{F}$ and convince $\mathcal{V}$ of the following claims:

$$\hat{c}(\boldsymbol{t}) = \tau \ , \tag{18}$$

$$\sum_{\boldsymbol{y} \in \{0,1\}^{\log k}} \hat{G}_{\mathcal{B}}(\boldsymbol{t}_L, \boldsymbol{y}_L) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{t}_R, \boldsymbol{y}_R) \cdot \hat{m}(\boldsymbol{y}) = \tau \ . \tag{19}$$

Eq. (18) is an evaluation claim over $\hat{c}$, and can thus be batched into the sumcheck claim in Eq. (17). Eq. (19) is a sumcheck claim over a domain of size k, but unfortunately cannot be written cleanly as an SEP claim, so in Fig. 8 we explicitly reduce Eq. (19) to evaluation claims, which can then be expressed in the language of SEP.

Note that this reduction is sound even in the list decoding regime, since $\boldsymbol{t}$ also acts as an out-of-domain evaluation point for $\boldsymbol{c}$, and thus binds $\boldsymbol{w}_c$ to a unique underlying polynomial $\hat{c}$ with overwhelming probability.

Full construction. In summary, SpotCheckReduce outputs an SEP claim that batches Eqs. (17) and (18) together, along with the evaluation claims derived from Eq. (19). We present the full construction in Fig. 8.

Indexer input: Index of Spot, $(\mathcal{C}, \mathcal{C}', \varepsilon, \ell)$ such that $\mathcal{C} = \mathcal{B} \otimes \mathcal{B}$, where $\mathcal{B}$ is a linear code with parameters $(\mathbb{F}, \sqrt{n}, \sqrt{k}, \delta_{\mathcal{B}})$, and $\mathcal{C}'$ is a linear code with parameters $(\mathbb{F}, n', n, \delta_{\mathcal{C}'})$.
Indexer output: Output $\mathbb{i}' := (\mathcal{C}', \varepsilon, 3)$.
Additionally, output $\iota := \bot$ and $\vec{\mathbb{I}} := \boldsymbol{w}_{\mathcal{B}}$ where $\boldsymbol{w}_{\mathcal{B}} := \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}_{\mathcal{B}})$ and $\boldsymbol{m}_{\mathcal{B}} := \boldsymbol{b}^{\sqrt{n/k}}$, and $\boldsymbol{b} \in \mathbb{F}^{\sqrt{nk}}$ is the concatenation of the rows of $\boldsymbol{G}_{\mathcal{B}}$.

Input instance: $\mathbb{x} = ([\xi_i]_{i \in [\ell]}, [y_i]_{i \in [\ell]})$, where each $\xi_i \in [n]$ and $y_i \in \mathbb{F}$,
Input instance oracles: $\vec{\mathbb{y}} = \boldsymbol{w}' \in \mathbb{F}^{n'}$,
Input witness: $\mathbb{w} = \boldsymbol{m}' \in \mathbb{F}^n$.

$\left\langle \mathcal{P}(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}), \mathcal{V}^{\vec{\mathbb{I}}, \vec{\mathbb{y}}}(\iota, \mathbb{x}) \right\rangle$:

1. Set $b \leftarrow 1$ and let $\boldsymbol{m} = \boldsymbol{m}'[1:k]$.
2. $\mathcal{P}$ defines $\boldsymbol{c} := \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})$, computes $\boldsymbol{w}_c := \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{c})$, and sends $[\![\boldsymbol{w}_c]\!]$ to $\mathcal{V}$.
3. $\mathcal{V}$ samples $\boldsymbol{t} \xleftarrow{\$} \mathbb{F}^{\log n}$ and sends them to $\mathcal{P}$.
4. $\mathcal{P}$ computes $\tau \leftarrow \hat{c}(\boldsymbol{t})$ and sends τ to $\mathcal{V}$.

Setting up the repetition check over m'.

5. $\mathcal{V}$ samples $\boldsymbol{u}_R \xleftarrow{\$} \mathbb{F}^{\log k}$ and send it to $\mathcal{P}$.
6. $\mathcal{P}$ computes $y_{\mathsf{rep}} \coloneqq \hat{m}(\boldsymbol{u}_R)$ and sends it to $\mathcal{V}$.

Reducing Eq. (19) to evaluation claims.

7. For $i \in [\log k]$:
8. If $i \leq \log k/2$:
9. Define $\boldsymbol{r}_i \coloneqq (r_1, \ldots, r_{i-1})$.
10. $\mathcal{P}$ computes the univariate polynomial

$$h_i(X) \coloneqq \sum_{\boldsymbol{x}_R \in \{0,1\}^{\log k/2}} \hat{G}_{\mathcal{B}}(\boldsymbol{x}_R, \boldsymbol{t}_R) \cdot \sum_{\boldsymbol{x}_L \in \{0,1\}^{\log k/2-i}} \hat{m}(\boldsymbol{r}_i, X, \boldsymbol{x}_L, \boldsymbol{x}_R) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{r}_i, X, \boldsymbol{x}_L, \boldsymbol{t}_L) \ .$$

11. Else:
12. Define $\boldsymbol{r}_L \coloneqq (r_1, \ldots, r_{\log k/2})$ and $\boldsymbol{r}_i \coloneqq (r_{\log k/2+1}, \ldots, r_{i-1})$.
13. $\mathcal{P}$ computes the univariate polynomial

$$h_i(X) \coloneqq \hat{G}_{\mathcal{B}}(\boldsymbol{r}_L, \boldsymbol{t}_L) \cdot \sum_{\boldsymbol{x}_R \in \{0,1\}^{\log k-i}} \hat{m}(\boldsymbol{r}_L, \boldsymbol{r}_i, X, \boldsymbol{x}_R) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{r}_i, X, \boldsymbol{x}_R, \boldsymbol{t}_R) \ .$$

14. $\mathcal{P}$ sends $h_i(X)$ to $\mathcal{V}$.
15. Define $\tau_i \coloneqq \tau$ if $i = 1$, and $\tau_i \coloneqq h_{i-1}(r_{i-1})$ otherwise.
16. $\mathcal{V}$ checks if $h_i(0) + h_i(1) = \tau_i$.
17. $\mathcal{V}$ samples $r_i \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$. If not, set $b \leftarrow 0$.
18. Define $\boldsymbol{r}_L \coloneqq (r_1, \ldots, r_{\log k/2})$ and $\boldsymbol{r}_R \coloneqq (r_{\log k/2+1}, \ldots, r_{\log k})$.
19. $\mathcal{P}$ sends $\tau_L \leftarrow \hat{G}_{\mathcal{B}}(\boldsymbol{r}_L, \boldsymbol{t}_L)$ and $\tau_R \leftarrow \hat{G}_{\mathcal{B}}(\boldsymbol{r}_R, \boldsymbol{t}_R)$ and $\tau_m \leftarrow \hat{m}(\boldsymbol{r})$ to $\mathcal{V}$.
20. $\mathcal{V}$ checks if $h_{\log k}(r_{\log k}) = \tau_L \cdot \tau_R \cdot \tau_m$. If not, set $b \leftarrow 0$.

Setting up the sumcheck claims.

21. $\mathcal{V}$ samples $\boldsymbol{u}_L \xleftarrow{\$} \mathbb{F}^{\log n - \log k}$, $\beta \xleftarrow{\$} \mathbb{F}$, and $\rho \xleftarrow{\$} \mathbb{F}$ and sends them to $\mathcal{P}$.
22. Define the selector polynomial $\hat{s} : \mathbb{F}^{\log n} \to \mathbb{F}$ as in Eq. (13).
23. Define $\sigma = y_{\mathsf{rep}} + \rho \cdot \left(\sum_{i=1}^{\ell} \beta^{i-1} \cdot y_i\right) + \rho^2 \cdot \tau + \rho^3 \cdot \tau_L + \rho^4 \cdot \tau_R + \rho^5 \cdot \tau_m$.
24. Define $\boldsymbol{q}_4, \boldsymbol{q}_5 \in \mathbb{F}^n$ such that the first $\sqrt{nk}$ indices $\boldsymbol{q}_4$ and $\boldsymbol{q}_5$ agree with $\mathbf{eq}(\boldsymbol{r}_L, \boldsymbol{t}_L)$ and $\mathbf{eq}(\boldsymbol{r}_R, \boldsymbol{t}_R)$ respectively, and the rest of the indices are 0.
25. Similarly, define $\boldsymbol{q}_6 \in \mathbb{F}^n$ such that $\boldsymbol{q}_6$ agrees with $\mathbf{eq}(\boldsymbol{r})$ on the first k indices and is 0 elsewhere.
26. Define the sumcheck polynomial $f : \mathbb{F}^{\log n+3} \to \mathbb{F}$ as follows:

$$\begin{aligned} f(\boldsymbol{X}, Z_1, Z_2, Z_3) \coloneqq\ & \mathrm{eq}(\boldsymbol{X}, (\boldsymbol{u}_L \,\|\, \boldsymbol{u}_R)) \cdot Z_1 + \rho \cdot (\hat{s}(\boldsymbol{X}) \cdot Z_2) + \rho^2 \cdot (\mathrm{eq}(\boldsymbol{X}, \boldsymbol{t}) \cdot Z_2) \\ & + \rho^3 \cdot (\hat{q}_4(\boldsymbol{X}) \cdot Z_3) + \rho^4 \cdot (\hat{q}_5(\boldsymbol{X}) \cdot Z_3) + \rho^5 \cdot (\hat{q}_6(\boldsymbol{X}) \cdot Z_1) \ . \end{aligned}$$

27. Define

$$\mathbb{x}' \coloneqq (f, \sigma), \quad \vec{\mathbb{y}}' \coloneqq (\boldsymbol{w}', \boldsymbol{w}_c, \boldsymbol{w}_{\mathcal{B}}), \quad \mathbb{w}' \coloneqq (\boldsymbol{m}', \boldsymbol{c}, \boldsymbol{m}_{\mathcal{B}}) \ .$$

28. $\mathcal{P}$ receives as output $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}', \mathbb{w}')$ and $\mathcal{V}$ receives as output $(\mathbb{i}', \mathbb{x}', \vec{\mathbb{y}}')$.

Figure 8: SpotCheckReduce from $\mathsf{Spot}^{(\mathcal{C},\mathcal{C}',\varepsilon,\ell)}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon,3)}$.

Lemma 6.4. *Let $\varepsilon \in [0, \varepsilon(\mathcal{C}'))$, where $\varepsilon(\mathcal{C})$ is the proximity threshold of $\mathcal{C}$ (see Definition 4.7). Fig. 8 is an IOR from $\mathsf{Spot}^{(\mathcal{C},\mathcal{C}',\varepsilon,\ell)}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon,3)}$ with the following efficiency parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$\log k + 3$	$O(\log k)$	0	$O(n')$	$O(T_{\mathcal{C}'})$	$O(T_{\mathcal{C}'})$	$O(\log n)$

where $\mathcal{C} = \mathcal{B} \otimes \mathcal{B}$ is a tensor code such that $\mathcal{B}$ has parameters $(\mathbb{F}, \sqrt{n}, \sqrt{k}, \delta_{\mathcal{B}})$ and $\mathcal{C}'$ is a linear code with parameters $(\mathbb{F}, n', n, \delta_{\mathcal{C}'})$.

We defer the proof of this lemma to Appendix B.3.

7 Prover and query optimal IOPP for tensor codes

We describe the full construction of TensorIOPP, an $O(\lambda + \log n)$-query IOPP for tensor codes. In particular, TensorIOPP is an IOPP for $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$, where $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ is a linear-time encodable code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{T}})$, where we will describe $\mathcal{C}_1$ and $\mathcal{C}_2$ shortly. Throughout this section, we assume that $\mathcal{T}$ has constant rate, i.e. $n = O(k)$.

7.1 Preliminaries

As outlined in Section 2.3, TensorIOPP is constructed by defining a sequence of codes with decreasing message length, and iteratively using TensorReduce over them. As mentioned in Section 1, we are aiming for a field-agnostic IOPP, so we recall the field-agnostic code family from Brakedown [GLSTW23].

Fact 7.1 (Brakedown code family [GLSTW23])**.** Let $\mathbb{F}$ be an arbitrary field of size $\Omega(2^{\lambda})$. There exists a code family over $\mathbb{F}$, i.e., a set of codes $\{\mathcal{C}_k\}_{k\in\mathbb{N}}$, with corresponding parameter functions $n(\cdot)$ and $\delta(\cdot)$, such that:
- For each k, $\mathcal{C}_k$ is a linear-time encodable code with parameters $(\mathbb{F}, n(k), k, \delta(k))$.
- There exist constants δ_0 and ρ_0, and some $k_0 \in \mathbb{N}$ such that for all $k \geq k_0$, we have $\delta(k) \geq \delta_0$ and $k/n(k) \geq \rho_0$.

We assume for simplicity that for all $k \geq k_0$, we have $\delta(k) \geq \delta_0$ and $k/n(k) = \rho_0$. This can be done by appending arbitrary elements to all codewords of any codes with rate larger than ρ_0. We call this the Brakedown code family.

Fact 7.2. Let $\mathbb{F}$ be an arbitrary field of size $\Omega(2^{\lambda})$, and let $k, n \in \mathbb{N}$ such that $k < n \leq |\mathbb{F}|$. There exists a Reed–Solomon code RS with parameters $(\mathbb{F}, n, k, 1 - (k-1)/n)$. Additionally, being a linear code, the encoding time of RS is $T_{\mathsf{RS}} = O(nk)$, independent of the field $\mathbb{F}$.

Our setup. We will describe a sequence of c codes $\mathcal{T}_1, \ldots, \mathcal{T}_{c-1}, \mathsf{RS}_c$, where the value of c will depend on λ and will be fixed at the end of the analysis. Let ρ_0 and δ_0 be as defined in Fact 7.1, and let $\lambda = O(n^{1-\gamma})$ for some $\gamma > 2/3$. For clarity of exposition, we only consider tensors of two codes of equal distance and rate (but potentially different message lengths). Thus, for each $i \in [c-1]$, define the tensor code $\mathcal{T}_i := \mathcal{C}_{i,1} \otimes \mathcal{C}_{i,2}$, such that codes $\mathcal{C}_{i,1}, \mathcal{C}_{i,2}$ have message lengths $k_{i,1}, k_{i,2}$ respectively, and both have rate ρ_i and distance δ_i. Thus for each $i \in [c-1]$, $\mathcal{T}_i$ has message length $k_i := k_{i,1}k_{i,2}$, rate ρ_i^2 and distance δ_i^2.

Simplifying the parameters of TensorReduce. The following lemma follows directly from Lemma 6.2 and the fact that we only consider tensors of codes of constant rate and distance.

Lemma 7.3. *Let $k \in \mathbb{N}$ be a message length, $\delta \in (0,1)$ be a base-code distance value, and $\rho \in (0,1)$ be a base-code rate value. There exists a function* $\mathsf{ComputeLength}(k, \rho, \delta) \to (k_1, k_2)$ *such that $k = k_1 k_2$ and there exists*
1. a code $\mathcal{C}_1$ with message length k_1, rate ρ and distance at least δ, and
2. a code $\mathcal{C}_2$ with message length k_2, rate ρ and distance at least δ,
such that for $\mathcal{T} := \mathcal{C}_1 \otimes \mathcal{C}_2$, there exists an IOR TensorReduce *from* $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ *to* $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',5)}$*, where $\varepsilon, \varepsilon'$ satisfy the conditions specified in Lemma 6.2 and $\mathcal{C}'$ is a linear-time encodable code with parameters $(\mathbb{F}, n', k', \delta_{\mathcal{C}'})$, where*

$$k' := \alpha \cdot k^{2/3} \cdot \lambda^{1/3} \ .$$

Furthermore, this IOR has the following efficiency parameters:

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$O(\log k)$	$O(\log k)$	$O(\lambda)$	$O(n)$	$O(n)$	$O(n)$	$O(\lambda + \log n)$

7.2 Selecting the codes

We will now describe how we select our codes $\mathcal{T}_1, \ldots, \mathcal{T}_{c-1}, \mathsf{RS}_c$. We think of $\mathcal{T}$ as being $\mathcal{T}_1$, with message length k. The codes $\mathcal{C}_{1,1}$ and $\mathcal{C}_{1,2}$ are chosen from the Brakedown code family to have message lengths $k_{1,1}$ and $k_{1,2}$ respectively, where $(k_{1,1}, k_{1,2}) \leftarrow \mathsf{ComputeLength}(k, \rho_0, \delta_0)$. For all $i \in [c-1]$, we define $\varepsilon_i := \min(\delta_{\mathcal{T}_i}/5, \varepsilon(\mathcal{T}_i))$ and we define $\varepsilon_c := \varepsilon(\mathsf{RS}_c)$.

The next few codes. The next few codes will also be obtained by tensoring codes from the Brakedown code family. For each $i \in [2, 3, \ldots, c]$, we define the message length

$$k_i := \alpha_i \cdot k_{i-1}^{2/3} \cdot \lambda^{1/3} \ ,$$

where α_i is some constant given by Lemma 7.3. For each $i \in [2, 3, \ldots, c-1]$, we choose $\mathcal{C}_{i,1}$ and $\mathcal{C}_{i,2}$ from the Brakedown code with message lengths $k_{i,1}$ and $k_{i,2}$ respectively, where $(k_{i,1}, k_{i,2}) \leftarrow \mathsf{ComputeLength}(k_i, \rho_0, \delta_0)$. We will now prove the following lemma.

Claim. There exists a constant c such that $k_c = O(k^{1/3})$.

Proof. Using the fact that $\lambda = O(n^{1-\gamma})$ for some $\gamma > 2/3$, the recurrence relation solves to give

$$\forall i \in [c] : k_i = O\left(k^{(1-\gamma+\gamma\cdot(2/3)^{i-1})}\right) \ .$$

Since $\gamma > 2/3$, it is easy to see that there exists a constant c such that $1 - \gamma + \gamma \cdot (2/3)^{c-1} \leq 1/3$. □

The final code. We use the fact that $k_c = O(k^{1/3})$ to set the final code RS_c to be a Reed–Solomon code with message length k_c, rate $\rho_c = 1/n^{1/3}$ and distance $\delta_c = 1 - 1/n^{1/3}$. Notice that this code has block length $n_c = O(n^{2/3})$, and can thus be encoded in $O(n_c k_c) = O(n)$ time via the naive encoding algorithm. Utilizing the fact that this code has inverse polynomial rate, we instantiate the WHIR IOPP [ACFY25] with the appropriate parameters to obtain an IOPP for $\mathsf{MEP}^{(\mathsf{RS}_c, \varepsilon_c, 1)}$ with $O(n)$ prover time and $O(\lambda)$ queries, which we call $\mathsf{FinishingIOPP}$.

We present the details in Section 7.3, but at a high level, this IOPP iteratively reduces the message length by a constant factor. Typically this requires $O(\lambda)$ queries per reduction (at least for a non-constant number of rounds), leading to $\omega(\lambda)$ queries overall. However, since our rate is $O(1/n^{1/3})$, we only require $O(\lambda/\log n)$ queries per reduction and $O(\lambda)$ queries overall.

7.3 The $\mathsf{FinishingIOPP}$

We now present the details of $\mathsf{FinishingIOPP}$ for $\mathsf{MEP}^{(\mathsf{RS}_c, \varepsilon_c, 1)}$ with $O(n)$ prover time and $O(\lambda)$ query complexity, where RS_c is a Reed–Solomon code with parameters $(\mathbb{F}, n_c = O(n^{2/3}), k_c = O(k^{1/3}), \delta_c = 1 - 1/n^{1/3})$.

The WHIR IOPP. While the WHIR IOPP is presented for constrained Reed–Solomon codes (discussed also in Section 8), it is easy to see that $\mathsf{MEP}^{(\mathsf{RS}, \epsilon, 1)}$ is a special case of constrained Reed–Solomon codes (with the weight polynomial set to express an evaluation constraint), where RS is a Reed–Solomon code.

At a high level, given a Reed–Solomon code RS with parameters $(\mathbb{F}, n, k, \delta_{\mathsf{RS}})$ and rate ρ_0, the WHIR IOPP for $\mathsf{MEP}^{(\mathsf{RS}, \epsilon, 1)}$ works as follows: for $m = \log k$ rounds, the IOPP reduces the message length by a factor of 2^v in each round, where v is a constant folding parameter. In every round $i \in [m]$, the prover sends a new oracle $[\![\boldsymbol{w}_i]\!]$ to the verifier (purported to be a Reed–Solomon codeword of rate ρ_i), and the verifier makes $t_i = O\left(\frac{\lambda}{\log(1/\rho_{i-1})}\right)$ queries to $[\![\boldsymbol{w}_{i-1}]\!]$ and sets up a sumcheck claim over $[\![\boldsymbol{w}_i]\!]$.

Since we are starting with a Reed–Solomon code RS_c with parameters $(\mathbb{F}, n_c = O(n^{2/3}), k_c = O(k^{1/3}), \delta_c = 1 - 1/n^{1/3})$, the number of rounds will be $m = \log k_c = O(\log k)$, and we set $\rho_i := 1/n^{1/3}$ for all $i \in [0, 1, \ldots, m]$. Thus, the number of queries in each round is $t_i = O(\lambda/\log(n^{1/3})) = O(\lambda/\log n)$, and the total query complexity of the IOPP is $O(m \cdot t_i) = O(\lambda + \log n)$, where the $\log n$ term stems from the fact that each of the m oracles must be queried at least once. We note that in a sense we do not utilize the full power of the WHIR IOPP, since we do not reduce the rate ρ_i in each round. This setting of parameters leads to the following lemma.

Lemma 7.4. *Let RS_c be the Reed–Solomon code with parameters $(\mathbb{F}, n_c = O(n^{2/3}), k_c = O(k^{1/3}), \delta_c = 1 - 1/n^{1/3})$. There exists an RBRTE IOPP* FinishingIOPP *for* $\mathsf{MEP}^{(\mathsf{RS}_c, \varepsilon_c, 1)}$ *with the following efficiency measures:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$O(\log n)$	$O(\lambda + \log n)$	$O(\lambda + \log n)$	$O(n)$	$\perp$	$O(n)$	$O(\lambda \log n)$

Proof. The IOPP requires no indexer, and the completeness and round-by-round knowledge soundness follow from [ACFY25, Theorem 5.2]. This further implies that this IOPP is RBRTE, since [BMMS25, Lemma 4.8] shows that round-by-round knowledge soundness implies RBRTE. The query complexity follows from the foregoing discussion, and the verifier time from [ACFY25, Theorem 5.2]. Thus all that is left is to analyze the prover time in a field-agnostic setting.

For each round $i \in [m]$, the prover prover time is dominated by encoding a Reed–Solomon codeword of rate $\rho_i = 1/n^{1/3}$ and length $k_{\mathsf{RS},i} = O(k_c/2^{vi}) = O(k^{1/3}/2^i)$. Thus the encoding time in round $i \in [m]$, using the naive encoding algorithm, is $O(k_{\mathsf{RS},i}^2/\rho_i) = O(k/4^i)$, and the total prover time is $\sum_{i=1}^{m} O(k/4^i) = O(k)$. □

7.4 The funneling lemma

The output of TensorReduce is an $\mathsf{SEP}^{(\mathcal{T}', \varepsilon', 5)}$ claim. While we can use Sumcheck to reduce this to an $\mathsf{HEP}^{(\mathcal{T}', \varepsilon', 5)}$ claim, this still cannot be used as an input to the next TensorReduce IOR. We address this type mismatch by imagining we have a 'virtual oracle' that is the random linear combination of the 5 oracles in the HEP claim. This virtual oracle is then an $\mathsf{MEP}^{(\mathcal{T}', \varepsilon', 1)}$ claim, which serves as the input to the next TensorReduce IOR. Every query to this virtual oracle can be simulated by one query to each of the 5 oracles in the HEP claim. We formalize this in the following 'funneling lemma'.

Lemma 7.5. *Given an RBRTE IOR* $\mathsf{IOR} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ *from* $\mathsf{MEP}^{(\mathcal{C}, \varepsilon, 1)}$ *to $\mathcal{R}'$ with the following efficiency measures*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
k	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$

and $\ell \in \mathbb{N}$, there exists a corresponding RBRTE IOR $\mathsf{IOR}' = (\mathbf{I}', \mathbf{P}', \mathbf{V}')$ *from* $\mathsf{HEP}^{(\mathcal{C}, \varepsilon, \ell)}$ *to $\mathcal{R}'$ with the following efficiency measures*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
k + 1	PLen	$\ell \cdot$ Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$O(\ell \cdot \mathsf{T}_{\mathcal{P}})$	$O(\ell \cdot \mathsf{T}_{\mathcal{V}})$

where $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$.

We defer the proof of this lemma to Appendix B.4. At a high level, $\mathcal{V}$ sends a random challenge $\boldsymbol{r} \in \mathbb{F}^{\log \ell}$. As opposed to proving $\hat{m}_i(\boldsymbol{z}) = y_i$ for each $i \in [\ell]$, $\mathcal{P}$ proves that $\sum_{i \in [\ell]} \mathrm{eq}(\langle i - 1 \rangle, \boldsymbol{r}) \cdot \hat{m}_i(\boldsymbol{z}) = \sum_{i \in [\ell]} \mathrm{eq}(\langle i - 1 \rangle, \boldsymbol{r}) \cdot y_i$.

7.5 Construction of TensorIOPP using TensorReduce and FinishingIOPP

Our final IOPP for $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$, TensorIOPP, is obtained by composing the following IORs:

1. The TensorReduce IOR from $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ to $\mathsf{SEP}^{(\mathcal{T}_2,\varepsilon_2,5)}$.
2. The Sumcheck IOR from $\mathsf{SEP}^{(\mathcal{T}_2,\varepsilon_2,5)}$ to $\mathsf{HEP}^{(\mathcal{T}_2,\varepsilon_2,5)}$.
3. For each $i \in \{3,\ldots,c-1\}$: the $\mathsf{TensorReduce}'_i$ IOR from $\mathsf{HEP}^{(\mathcal{T}_{i-1},\varepsilon_{i-1},5)}$ to $\mathsf{HEP}^{(\mathcal{T}_i,\varepsilon_i,5)}$, obtained by applying Lemma 7.5 to $\mathsf{TensorReduce}_i$ from $\mathsf{MEP}^{(\mathcal{T}_{i-1},\varepsilon_{i-1},1)}$ to $\mathsf{SEP}^{(\mathcal{T}_i,\varepsilon_i,5)}$ and then composing with the Sumcheck IOR from $\mathsf{SEP}^{(\mathcal{T}_i,\varepsilon_i,5)}$ to $\mathsf{HEP}^{(\mathcal{T}_i,\varepsilon_i,5)}$.
4. The $\mathsf{TensorReduce}'_c$ IOR from $\mathsf{HEP}^{(\mathcal{T}_{c-1},\varepsilon_{c-1},5)}$ to $\mathsf{HEP}^{(\mathsf{RS}_c,\varepsilon_c,5)}$, obtained by applying Lemma 7.5 to $\mathsf{TensorReduce}_c$ from $\mathsf{MEP}^{(\mathcal{T}_{c-1},\varepsilon_{c-1},1)}$ to $\mathsf{SEP}^{(\mathsf{RS}_c,\varepsilon_c,5)}$ and then composing with the Sumcheck IOR from $\mathsf{SEP}^{(\mathsf{RS}_c,\varepsilon_c,5)}$ to $\mathsf{HEP}^{(\mathsf{RS}_c,\varepsilon_c,5)}$.
5. The $\mathsf{FinishingIOPP}'$ IOPP for $\mathsf{HEP}^{(\mathsf{RS}_c,\varepsilon_c,5)}$ obtained by applying Lemma 7.5 to FinishingIOPP, the IOPP for $\mathsf{MEP}^{(\mathsf{RS}_c,\varepsilon_c,1)}$.

Theorem 7.6. *Let $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ be a linear-time encodable code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{T}})$ such that:*

- *$\mathcal{C}_2 = \mathcal{B} \otimes \mathcal{B}$ for some code $\mathcal{B}$, and*
- *$\mathcal{C}_1$ and $\mathcal{C}_2$ are both linear-time encodable codes with constant rate ρ_0 and distance $\geq \delta_0$, and have parameters $(\mathbb{F}, n_1, k_1, \delta_0)$ and $(\mathbb{F}, n_2, k_2, \delta_0)$ respectively, where $(k_1, k_2) \leftarrow \mathsf{ComputeLength}(k, \rho_0, \delta_0)$.*

For $\varepsilon = \min(\delta_{\mathcal{T}}/5, \varepsilon(\mathcal{T}))$, assuming $\lambda = O(n^{1-c})$ for some $c > 0$, there exists an RBRTE IOPP for $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ with the following efficiency parameters:

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$O(\log n)$	$O(\lambda + \log n)$	$O(\lambda + \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(\lambda \log n)$

Proof. The proof follows from Lemmas 4.8, 6.2, 7.4 and 7.5. □

7.6 IOPPs for codeswitchable codes

The following lemma shows that a prover and query optimal IOPP for any linear-time encodable code implies the existence of a prover and query optimal IOPP for all codeswitchable codes.

Theorem 7.7. *Let $\mathcal{C}$ be a codeswitchable code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. For all $\varepsilon \in (\delta_{\mathcal{C}}/8, \delta_{\mathcal{C}}/4)$, assuming $\lambda = O(n^{1-c})$ for some $c > 0$, there exists an RBRTE IOPP for $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ with the following efficiency parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$O(\log n)$	$O(\lambda + \log n)$	$O(\lambda + \log n)$	$O(n)$	$O(n)$	$O(n)$	$O(\lambda \log n)$

Proof. Such an IOPP can be obtained by composing the CodeSwitch IOR (see Definition 3.14) from $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ to $\mathsf{MEP}^{(\mathcal{T},\varepsilon',1)}$ with the TensorIOPP IOPP for $\mathsf{MEP}^{(\mathcal{T},\varepsilon',1)}$, for $\mathcal{T}$ and ε' from Theorem 7.6. The rest of the claims follow from Theorem 7.6. □

8 Query complexity lower bound for IOPPs

Prior work reduces the task of constructing IOPPs for NP-complete relations constructing IOPPs for certain indexed oracle pair relations, which we refer to as 'IOPP relations'. While there are multiple such IOPP relations that have been considered, all of them share a common structure: an underlying 'proximity claim' with respect to a linear code $\mathcal{C}$. That is, they contain $[\![\boldsymbol{w}]\!]$ as part of their instance oracle, where $\boldsymbol{w} \in \mathbb{F}^n$ is a purported codeword of $\mathcal{C}$ which has parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, and $\mathcal{P}$ proves to $\mathcal{V}$ that $\delta(\boldsymbol{w}, \mathcal{C}) \leq \varepsilon$ for some proximity parameter $\varepsilon \in (0, 1 - k/n)$.

In this section, we formalize this underlying claim as the *proximity relation*, $\mathsf{Prox}^{(\mathcal{C},\varepsilon)}$, and show that any perfectly complete IOPP for $\mathsf{Prox}^{(\mathcal{C},\varepsilon)}$ requires $\mathcal{V}$ to make at least $\Omega(\lambda/\log(n/k))$ queries to $[\![\boldsymbol{w}]\!]$, in order to obtain a soundness error of $2^{-\lambda}$. Before we present the lower bound, we define Prox and review all of the IOPP relations considered in the literature, showing that they inherit the lower bound from Prox.

Indexed oracle relation 8.1. The *proximity relation* $\mathsf{Prox} = (\mathcal{R}, \tilde{\mathcal{R}})$ is an indexed oracle pair relation with $\mathcal{R}$ and $\tilde{\mathcal{R}}$ both consisting of tuples of the following form:

$$(\mathtt{i} \coloneqq (\mathcal{C}, \varepsilon) \ , \ \mathtt{x} \coloneqq \bot \ , \ \vec{\mathtt{y}} \coloneqq \boldsymbol{w} \ , \ \mathtt{w} \coloneqq \boldsymbol{m}) \ ,$$

where $\mathcal{C}$ is a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$, $\varepsilon \in (0, 1 - k/n)$ is the proximity parameter, $\boldsymbol{w} \in \mathbb{F}^n$ is the purported codeword, and $\boldsymbol{m} \in \mathbb{F}^k$ is the underlying message. The tuples satisfy the following conditions for $\mathcal{R}$ and $\tilde{\mathcal{R}}$ respectively

$$\begin{aligned} \mathcal{R} : &\quad \boldsymbol{w} = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}), \\ \tilde{\mathcal{R}} : &\quad \boldsymbol{w} \sim_{\varepsilon} \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}) \ . \end{aligned}$$

The Prox relation was the first IOPP relation (implicitly) considered in the literature, and is exclusively used for polynomial-based codes like Reed–Solomon or Reed–Muller codes. The IOPPs from [BBHR18; BGKS20; BCIKS23; ACFY24; MZ25] can all be viewed as IOPPs for $\mathsf{Prox}^{(\mathcal{C},\varepsilon)}$.

MEP. The IOPPs of [ZCF24; BCFRRZ25; NST24; BMMS25] as well as this work can be viewed as IOPPs for $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ for some linear code $\mathcal{C}$. Any IOPP for $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)} = (\mathcal{R}, \tilde{\mathcal{R}})$ that makes q queries to $[\![\boldsymbol{w}]\!]$ implies an IOPP $\mathsf{IOPP} = (\mathcal{P}, \mathcal{V})$ for $\mathsf{Prox}^{(\mathcal{C},\varepsilon)} = (\mathcal{R}', \tilde{\mathcal{R}}')$ with the same completeness and soundness errors, that also makes q queries to $[\![\boldsymbol{w}]\!]$. This can be seen as follows: on input $(\mathtt{x} = \bot, \vec{\mathtt{y}} = \boldsymbol{w}, \mathtt{w} = \boldsymbol{m})$, $\mathcal{P}$ sends $y \coloneqq \hat{m}(0^k)$ to $\mathcal{V}$, and $\mathcal{P}$ and $\mathcal{V}$ run the IOPP for $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ with the following inputs:

$$\mathtt{i} = (\mathcal{C}, \varepsilon, 1), \quad \mathtt{x} = (0^k, y), \quad \vec{\mathtt{y}} = \boldsymbol{w}, \quad \mathtt{w} = \boldsymbol{m} \ .$$

The completeness and soundness claims follow from the fact that $((0^k, y), \boldsymbol{w}, \boldsymbol{m}) \in \mathcal{R}$ if and only if $(\bot, \boldsymbol{w}, \boldsymbol{m}) \in \mathcal{R}'$, and if $(\bot, \boldsymbol{w}) \notin \mathcal{L}(\tilde{\mathcal{R}}')$, then there is no value of y such that $((0^k, y), \boldsymbol{w}) \in \mathcal{L}(\tilde{\mathcal{R}})$.

Constrained Reed–Solomon codes. [ACFY25] construct an IOPP for constrained Reed–Solomon codes, which is essentially an IOPP for $\mathsf{SEP}^{(\mathsf{RS},\varepsilon,1)}$, where RS is a Reed–Solomon code, except that the constraint polynomial $f(\boldsymbol{X}, \boldsymbol{Z})$ is allowed to be an arbitrary multivariate polynomial. As noted by [ACFY25], by setting f to be the 0 polynomial and $\sigma = 0$, we obtain precisely an IOPP for $\mathsf{Prox}^{(\mathsf{RS},\varepsilon)}$. As such, a query lower bound for IOPPs for $\mathsf{Prox}^{(\mathcal{C},\varepsilon)}$ is inherited by IOPPs for constrained Reed–Solomon codes.

Remark 8.2 (Upper bound on ε)**.** The proximity parameter ε in Indexed oracle relation 8.1 can be as large as $1 - k/n$, which only strengthens the lower bound by implying that the lower bound holds even if list decodability and mutual correlated agreement hold up to capacity.

The lower bound. We now present our query complexity lower bound for IOPPs for $\mathsf{Prox}^{(\mathcal{C},\varepsilon)}$.

Theorem 8.3. *Let* $\mathsf{IOPP} = (\mathcal{P}, \mathcal{V})$ *be a perfectly complete IOPP for* $\mathsf{Prox}^{(\mathcal{C},\varepsilon)} = (\mathcal{R}, \tilde{\mathcal{R}})$ *with soundness error at most* $2^{-\lambda}$*, where* $\mathcal{C}$ *is a linear code with parameters* $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ *and* $\varepsilon \in (0, 1 - k/n)$*. Then,* $\mathcal{V}$ *must make at least* $q \geq \lambda / \log(n/(k-q))$ *queries to the instance oracle.*

Proof. Fix an arbitrary message $\boldsymbol{m} \in \mathbb{F}^k$. Consider the following set of instance oracles:

$$\mathcal{Y} \coloneqq \{\boldsymbol{w} \in \mathbb{F}^n : \Delta(\boldsymbol{w}, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})) = \lfloor \varepsilon \cdot n \rfloor + 1\} \ ,$$

and consider the following set of 'bad' tuples:

$$\mathcal{B} \coloneqq \{(\mathbb{x} = \bot, \vec{\mathbb{y}} = \boldsymbol{w}, \mathbb{w} = \boldsymbol{m})\}_{\boldsymbol{w} \in \mathcal{Y}} \ .$$

For any $(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \in \mathcal{B}$, we have $(\mathbb{x}, \vec{\mathbb{y}}, \mathbb{w}) \notin \tilde{\mathcal{R}}$, and thus the verifier $\mathcal{V}$ must only accept this tuple with probability $\leq 2^{-\lambda}$.

Consider an interaction where the instance oracle is some $\boldsymbol{w} \in \mathcal{Y}$. Defining $S(\boldsymbol{w}) \coloneqq \{i \in [n] : \boldsymbol{w}[i] \neq \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})[i]\}$, observe that if $\mathcal{V}$ does not query $[\![\boldsymbol{w}]\!]$ at any location $i \in S(\boldsymbol{w})$, then it must accept. This follows from the fact that IOPP is perfectly complete, and since $\mathcal{V}$ does not query $[\![\boldsymbol{w}]\!]$ at any location $i \in S(\boldsymbol{w})$, its view of the instance oracle is indistinguishable from the case when $\boldsymbol{w} = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})$.

Now consider an instance oracle chosen uniformly at random, $\boldsymbol{w} \in \mathcal{Y}$. Any deterministic verifier $\mathcal{V}$ that queries q locations in $[\![\boldsymbol{w}]\!]$ does not query a location in $S(\boldsymbol{w})$ with probability (over the choice of the input) at least

$$\left(1 - \frac{|S(\boldsymbol{w})|}{n}\right) \cdot \left(1 - \frac{|S(\boldsymbol{w})|}{n-1}\right) \cdots \left(1 - \frac{|S(\boldsymbol{w})|}{n-q+1}\right) \geq \left(1 - \frac{|S(\boldsymbol{w})|}{n-q+1}\right)^q = L \ .$$

Therefore, by Yao's minimax principle [Yao77], for any randomized $\mathcal{V}$ that makes q queries, there exists $\boldsymbol{w} \in \mathcal{Y}$ such that $\mathcal{V}$ does not query $[\![\boldsymbol{w}]\!]$ at any location in $S(\boldsymbol{w})$ with probability at least L. Since $\mathcal{V}$ would then have to accept, even though this is an invalid instance, this event must occur with probability at most $2^{-\lambda}$. Thus, we have that $2^{-\lambda} \geq L$, which we can simplify to get

$$\lambda \leq q \cdot \log\left(\frac{1}{1 - \frac{|S(\boldsymbol{w})|}{n-q+1}}\right) \leq q \cdot \log\left(\frac{n}{n - q + 1 - \lfloor \varepsilon \cdot n \rfloor - 1}\right) \leq q \cdot \log\left(\frac{n}{k-q}\right) \ .$$

□

Acknowledgements

Pratyush Mishra and Matan Shtepel are partially supported by a Sui Academic Research Award. Pratyush Mishra and Tushar Mopuri are partially supported by a Sony Research Award. Part of this work was conducted while the authors were visiting the Simons Institute for the Theory of Computing.

References

[ACFY24] G. Arnon, A. Chiesa, G. Fenzi, and E. 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] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. "WHIR: Reed–Solomon proximity testing with super-fast verification". In: *Proceedings of the 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '25. 2025, pp. 214–243.

[AER24] G. Angeris, A. Evans, and G. Roh. *A Note on Ligero and Logarithmic Randomness*. Cryptology ePrint Archive, Paper 2024/1399. 2024. URL: https://eprint.iacr.org/2024/1399.

[AHIV17] S. Ames, C. Hazay, Y. Ishai, and M. 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.

[BBHR18] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. ICALP '18. 2018, 14:1–14:17.

[BCFRRZ25] M. Brehm, B. Chen, B. Fisch, N. Resch, R. D. Rothblum, and H. Zeilberger. "Blaze: Fast SNARKs from Interleaved RAA Codes". In: *Proceedings of the 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '25. 2025, pp. 123–152.

[BCFW25] B. Bünz, A. Chiesa, G. Fenzi, and W. Wang. "Linear-Time Accumulation Schemes". In: *Proceedings of the 23rd Theory of Cryptography Conference*. TCC '25. 2025.

[BCG20] J. Bootle, A. Chiesa, and J. 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. URL: https://eprint.iacr.org/2020/1426.

[BCGGRS19] E. Ben-Sasson, A. Chiesa, L. Goldberg, T. Gur, M. Riabzev, and N. Spooner. "Linear-Size Constant-Query IOPs for Delegating Computation". In: *Proceedings of the 17th Theory of Cryptography Conference*. TCC '19. 2019.

[BCIKS23] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf. "Proximity Gaps for Reed–Solomon Codes". In: *J. ACM* 70.5 (2023), 31:1–31:57. URL: https://dblp.org/rec/journals/iacr/Ben-SassonCIKS20.html?view=bibtex.

[BCL22] J. Bootle, A. Chiesa, and S. Liu. "Zero-Knowledge IOPs with Linear-Time Prover and Polylogarithmic-Time Verifier". In: *Proceedings of the 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '22. 2022, pp. 275–304. URL: https://eprint.iacr.org/2020/1527.

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. "Interactive Oracle Proofs". In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC '16-B. 2016, pp. 31–60.

[BFKTWZ24] A. R. Block, Z. Fang, J. Katz, J. Thaler, H. Waldner, and Y. Zhang. "Field-Agnostic SNARKs from Expand-Accumulate Codes". In: *CRYPTO (10)*. Lecture Notes in Computer Science. 2024.

[BGKS20] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. 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. URL: https://eprint.iacr.org/2019/336.

[BMMS25] A. Baweja, P. Mishra, T. Mopuri, and M. Shtepel. *FICS and FACS: Fast IOPPs and Accumulation via Code-Switching*. Cryptology ePrint Archive, Paper 2025/737. 2025. URL: https://eprint.iacr.org/2025/737.

[BMNW25] B. Bünz, P. Mishra, W. Nguyen, and W. Wang. "Arc: Accumulation for Reed–Solomon Codes". In: *Proceedings of the 45th Annual International Cryptology Conference*. CRYPTO '25. 2025.

[DG25] B. E. Diamond and A. Gruen. "Proximity Gaps in Interleaved Codes". In: *IACR Communications in Cryptology* 1.4 (2025).

[DJM98] D. Divsalar, H. Jin, and R. J. McEliece. "Coding theorems for" turbo-like" codes". In: *Proceedings of the annual Allerton Conference on Communication control and Computing*. Vol. 36. 1998, pp. 201–210.

[DP24] B. E. Diamond and J. Posen. "Proximity Testing with Logarithmic Randomness". In: *IACR Communications in Cryptology* 1.1 (2024).

[GKL24] Y. Gao, H. Kan, and Y. Li. *Linear Proximity Gap for Linear Codes within the 1.5 Johnson Bound*. Cryptology ePrint Archive, Paper 2024/1810. 2024. URL: https://eprint.iacr.org/2024/1810.

[GLSTW23] A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, and R. S. Wahby. "Brakedown: Linear-Time and Field-Agnostic SNARKs for R1CS". In: *Proceedings of the 43rd Annual International Cryptology Conference*. CRYPTO '23. 2023, pp. 193–226. URL: https://eprint.iacr.org/2021/1043.

[GRS12] V. Guruswami, A. Rudra, and M. Sudan. *Essential coding theory*. Vol. 2. 2012.

[LFKN92] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. "Algebraic Methods for Interactive Proof Systems". In: *Journal of the ACM* 39.4 (1992), pp. 859–868.

[MZ25] D. Minzer and K. Z. 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.

[NST24] V. Nair, A. Sharma, and B. Thankey. *BrakingBase - a linear prover, poly-logarithmic verifier, field agnostic polynomial commitment scheme*. Cryptology ePrint Archive, Paper 2024/1825. 2024. URL: https://eprint.iacr.org/2024/1825.

[RR24] N. Ron-Zewi and R. Rothblum. "Local Proofs Approaching the Witness Length". In: *J. ACM* 71.3 (2024), p. 18. URL: https://eprint.iacr.org/2019/1062.

[RR25] N. Ron-Zewi and R. Rothblum. "Proving as fast as computing: succinct arguments with constant prover overhead". In: *J. ACM* 72.2 (2025), pp. 1–54.

[Yao77] A. C.-C. Yao. "Probabilistic computations: Toward a unified measure of complexity". In: *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '77. 1977, pp. 222–227.

[ZCF24] H. Zeilberger, B. Chen, and B. Fisch. "BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes". In: *Proceedings of the 44th Annual International Cryptology Conference*. CRYPTO '24. 2024, pp. 138–169. URL: https://eprint.iacr.org/2023/1705.

[Zei24] H. Zeilberger. *Khatam: Reducing the Communication Complexity of Code-Based SNARKs*. Cryptology ePrint Archive, Paper 2024/1843. 2024. URL: https://eprint.iacr.org/2024/1843.

A Mutual correlated agreement for multilinear proximity generators

In this section we present the deferred proofs from Section 3.2. But first we must present some prerequisites, starting with the result from [Zei24].

Theorem A.1 ([Zei24, Theorem 2]). *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. Given $\boldsymbol{w}_L, \boldsymbol{w}_R \in \mathbb{F}^n$, $\epsilon \in (0,1)$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$, define $\boldsymbol{w} \coloneqq (\boldsymbol{w}_L, \boldsymbol{w}_R)$. Define the set $A(\boldsymbol{w}, \epsilon, \beta_\epsilon)$ as*

$$\left\{ r \in \mathbb{F} : \exists S \subset [n], \boldsymbol{c} \in \mathcal{C} \quad \text{s.t.} \quad \begin{array}{l} |S| > \beta_\epsilon n \ , \\ \boldsymbol{w}_L[S] + r \cdot \boldsymbol{w}_R[S] = \boldsymbol{c}[S] \ , \\ \forall (\boldsymbol{c}_L, \boldsymbol{c}_R) \in \mathcal{C} \times \mathcal{C} : \\ |\{ i \in S : \boldsymbol{w}_L[i] = \boldsymbol{c}_L[i] \wedge \boldsymbol{w}_R[i] = \boldsymbol{c}_R[i] \}| < (\beta_\epsilon - \epsilon) n \end{array} \right\} .$$

Then we have

$$|A(\boldsymbol{w}, \epsilon, \beta_\epsilon)| \leq \tfrac{2}{\epsilon^2} \ .$$

We now prove the following corollary of Theorem A.1, which is more suited to our setting.

Corollary A.2. *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. Given $\boldsymbol{w}_L, \boldsymbol{w}_R \in \mathbb{F}^n$, $\epsilon \in (0,1)$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$, define $\boldsymbol{w} \coloneqq (\boldsymbol{w}_L, \boldsymbol{w}_R)$. For the set $B(\boldsymbol{w}, \epsilon, \beta_\epsilon)$ defined to be*

$$\left\{ r \in \mathbb{F} : \exists S \subset [n], \boldsymbol{c} \in \mathcal{C} \quad \text{s.t.} \quad \begin{array}{l} |S| > \beta_\epsilon n \ , \\ (1-r) \cdot \boldsymbol{w}_L[S] + r \cdot \boldsymbol{w}_R[S] = \boldsymbol{c}[S] \ , \\ \forall (\boldsymbol{c}_L, \boldsymbol{c}_R) \in \mathcal{C} \times \mathcal{C} : \\ |\{ i \in S : \boldsymbol{w}_L[i] = \boldsymbol{c}_L[i] \wedge \boldsymbol{w}_R[i] = \boldsymbol{c}_R[i] \}| < (\beta_\epsilon - \epsilon) n \end{array} \right\} .$$

Then we have

$$|B(\boldsymbol{w}, \epsilon, \beta_\epsilon)| \leq \tfrac{2}{\epsilon^2} + 1 \ .$$

Proof. We proceed by contradiction and assume that Corollary A.2 does not hold. Let $t(\epsilon) \coloneqq \lfloor 2/\epsilon^2 + 1 \rfloor$. There exists a tuple $(\boldsymbol{w}, \epsilon, \beta_\epsilon)$ such that $|B(\boldsymbol{w}, \epsilon, \beta_\epsilon)| > t(\epsilon)$. Consider $r_1, r_2, \ldots, r_{t(\epsilon)+1}$ in $B(\boldsymbol{w}, \epsilon, \beta_\epsilon)$. For each $i \in [t(\epsilon)+1]$, there exists a set $S_i \subset [n]$ and a codeword $\boldsymbol{c}_i \in \mathcal{C}$ such that

$$|S_i| > \beta_\epsilon n \quad \text{and} \quad \forall j \in S_i : (1 - r_i) \cdot \boldsymbol{w}_L[j] + r_i \cdot \boldsymbol{w}_R[j] = \boldsymbol{c}_i[j] \ , \tag{20}$$

and yet for all codewords $(\boldsymbol{c}_L, \boldsymbol{c}_R) \in \mathcal{C} \times \mathcal{C}$, we have $|\{ j \in S : \boldsymbol{w}_L[j] = \boldsymbol{c}_L[j] \wedge \boldsymbol{w}_R[j] = \boldsymbol{c}_R[j] \}| < (\beta_\epsilon - \epsilon) n$. If $1 \notin B(\boldsymbol{w}, \epsilon, \beta_\epsilon)$, then $(1 - r_i)$ must be invertible for each $i \in [t(\epsilon)+1]$. Thus, for all $i \in [t(\epsilon)+1]$, we can define $\boldsymbol{c}'_i \in \mathcal{C}$ as $\boldsymbol{c}'_i \coloneqq (1 - r_i)^{-1} \cdot \boldsymbol{c}_i$ and $r'_i \coloneqq (1 - r_i)^{-1} \cdot r_i$. Therefore, we can rewrite Eq. (20) as

$$|S_i| > \beta_\epsilon n \quad \text{and} \quad \forall j \in S_i : \boldsymbol{w}_L[j] + r'_i \cdot \boldsymbol{w}_R[j] = \boldsymbol{c}_i[j]. \tag{21}$$

Since all the r_i's are distinct and non-zero, so are the r'_i's. Therefore, $|A(\boldsymbol{w}, \epsilon, \beta_\epsilon)| \geq t(\epsilon) + 1$, contradicting Theorem A.1 If $1 \in B(\boldsymbol{w}, \epsilon, \beta_\epsilon)$, then the same argument holds for all but one value of r_i, and therefore we get $|A(\boldsymbol{w}, \epsilon, \beta_\epsilon)| \geq t(\epsilon)$, which is still a contradiction. □

This corollary can be further improved (by removing the 'loss' in the size of the agreement set) to obtain the following lemma.

Lemma A.3. *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$. If $\epsilon \in (0, 1/2n]$ and $\beta_\epsilon \geq (1-\delta_{\mathcal{C}}+\epsilon)^{1/3}+\epsilon$ is such that $\{\beta_\epsilon n\} > 1/2$, then for all $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{F}^n$ the following holds:*

$$\Pr_{r \xleftarrow{\$} \mathbb{F}} \left[\exists S \subset [n] \quad \text{s.t.} \quad \begin{array}{l} |S| = \lceil \beta_\epsilon n \rceil \ , \\ \exists \boldsymbol{c} \in \mathcal{C} : (1-r) \cdot \boldsymbol{w}_1[S] + r \cdot \boldsymbol{w}_2[S] = \boldsymbol{c}[S] \ , \\ \forall \boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C} : \boldsymbol{w}_1[S] \neq \boldsymbol{c}_1[S] \vee \boldsymbol{w}_2[S] \neq \boldsymbol{c}_2[S] \end{array} \right] \leq \left(\tfrac{2}{\epsilon^2} + 1\right) \cdot \frac{1}{|\mathbb{F}|} \ .$$

Proof. Follows from Corollary A.2 and the observation that since $\{\beta_\epsilon n\} > 1/2$ and $\epsilon \in (0, 1/2n]$, it must be the case that $\lfloor \beta_\epsilon n \rfloor < (\beta_\epsilon - \epsilon)n < \beta_\epsilon n < \lceil \beta_\epsilon n \rceil$. □

A.1 Extending to more than two input words

Lemma A.3 proves mutual correlated agreement for multilinear proximity generators with two input words. This can be extended to 2^m input words for any $m \geq 1$ as follows.

Theorem 3.11. *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ and $m \in \mathbb{N}$, with $M \coloneqq 2^m$. Additionally, let $\epsilon \in (0, 1/2n]$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$ be such that $\{\beta_\epsilon n\} > 1/2$.*[14] *For every $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \in \mathbb{F}^n$, the following holds:*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m} \left[\exists S \subset [n] \quad \text{s.t.} \quad \begin{array}{l} |S| = \lceil \beta_\epsilon n \rceil \ , \\ \exists \boldsymbol{c} \in \mathcal{C} : \sum_{j \in [M]} \boldsymbol{w}_j[S] \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}) = \boldsymbol{c}[S] \ , \\ \forall [\boldsymbol{c}_i]_{i \in [M]} \in \mathcal{C}^M : \exists j \in [M] \text{ s.t. } \boldsymbol{w}_j[S] \neq \boldsymbol{c}_j[S] \end{array} \right] \leq \tfrac{3^m}{\epsilon^2 |\mathbb{F}|} \ .$$

Proof. We prove the theorem via induction on m. The base case where $m = 1$ follows directly from Lemma A.3. We now show that the claim holds for $m+1$ assuming it holds for m. Let $M \coloneqq 2^m$ and $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2M} \in \mathbb{F}^n$. Define $A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M), m, \epsilon, \beta_\epsilon)$ as

$$\left\{ \boldsymbol{r} \in \mathbb{F}^m : \exists S \subset [n], \boldsymbol{c} \in \mathcal{C} \quad \text{s.t.} \quad \begin{array}{l} |S| = \lceil \beta_\epsilon n \rceil \ , \\ \sum_{j \in [M]} \boldsymbol{w}_j[S] \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}) = \boldsymbol{c}[S] \ , \\ \forall [\boldsymbol{c}_i]_{i \in [M]} \in \mathcal{C}^M : \exists j \in [M] \text{ s.t. } \boldsymbol{w}_j[S] \neq \boldsymbol{c}_j[S] \end{array} \right\} \ .$$

We can define $A((\boldsymbol{w}_{M+1}, \ldots, \boldsymbol{w}_{2M}), m, \epsilon, \beta_\epsilon)$ and $A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2M}), m+1, \epsilon, \beta_\epsilon)$ analogously. The induction hypothesis gives that $|A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M), m, \epsilon, \beta_\epsilon)| \leq 3^m |\mathbb{F}|^{m-1}/\epsilon^2$ and $|A((\boldsymbol{w}_{M+1}, \ldots, \boldsymbol{w}_{2M}), m, \epsilon, \beta_\epsilon)| \leq 3^m |\mathbb{F}|^{m-1}/\epsilon^2$. We will now proceed to show that

$$|A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2M}), m+1, \epsilon, \beta_\epsilon)| \leq 3^{m+1} |\mathbb{F}|^m / \epsilon^2 \ .$$

Let $(r_1, \ldots, r_{m+1}) \in A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2M}), m+1, \epsilon, \beta_\epsilon)$. We make the following claim.

Claim. At least one of the following conditions must hold:
- $(r_1, \ldots, r_m) \in A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M), m, \epsilon, \beta_\epsilon)$.
- $(r_1, \ldots, r_m) \in A((\boldsymbol{w}_{M+1}, \ldots, \boldsymbol{w}_{2M}), m, \epsilon, \beta_\epsilon)$.
- $r_{m+1} \in A((\boldsymbol{w}_L, \boldsymbol{w}_R), 1, \epsilon, \beta_\epsilon)$, where

$$\boldsymbol{w}_L \coloneqq \sum_{i \in [M]} \boldsymbol{w}_i \cdot \mathrm{eq}(\langle i-1 \rangle, (r_1, \ldots, r_m)) \text{ and } \boldsymbol{w}_R \coloneqq \sum_{i \in [M]} \boldsymbol{w}_{i+M} \cdot \mathrm{eq}(\langle i-1 \rangle, (r_1, \ldots, r_m)) \tag{22}$$

[14] We use $\{x\}$ denotes the fractional part of x.

Proof. Assume for the sake of contradiction that none of the conditions hold. For the $(r_1, \ldots, r_{m+1}) \in A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2M}), m+1, \epsilon, \beta_\epsilon)$ in question, consider an agreement set $S \subset [n]$ and a codeword $\boldsymbol{c} \in \mathcal{C}$ such that

$$|S| = \lceil \beta_\epsilon n \rceil \quad \text{and} \quad \sum_{j \in [2M]} \boldsymbol{w}_j[S] \cdot \mathrm{eq}(\langle j-1 \rangle, (r_1, \ldots, r_{m+1})) = \boldsymbol{c}[S] \tag{23}$$

Our goal is to show that $\exists [\boldsymbol{c}_i]_{i \in [2M]} \in \mathcal{C}^{2M}$ s.t. $\forall i \in [2M] : \boldsymbol{w}_i[S] = \boldsymbol{c}_i[S]$, which would be a contradiction. Rewriting Eq. (23), we have $(1 - r_{m+1}) \cdot \boldsymbol{w}_L[S] + r_{m+1} \cdot \boldsymbol{w}_R[S] = \boldsymbol{c}[S]$, where $\boldsymbol{w}_L$ and $\boldsymbol{w}_R$ are defined as in Eq. (22). Since $r_{m+1} \notin A((\boldsymbol{w}_L, \boldsymbol{w}_R), 1, \epsilon, \beta_\epsilon)$, there must exist codewords $(\boldsymbol{c}_L, \boldsymbol{c}_R) \in \mathcal{C}^2$ such that $\boldsymbol{w}_L[S] = \boldsymbol{c}_L[S]$ and $\boldsymbol{w}_R[S] = \boldsymbol{c}_R[S]$. Expanding the definitions of $\boldsymbol{w}_L$ and $\boldsymbol{w}_R$, we have

$$\sum_{i \in [M]} \boldsymbol{w}_i[S] \cdot \mathrm{eq}(\langle i-1 \rangle, (r_1, \ldots, r_m)) = \boldsymbol{c}_L[S] \quad \text{and} \quad \sum_{i \in [M]} \boldsymbol{w}_{i+M}[S] \cdot \mathrm{eq}(\langle i-1 \rangle, (r_1, \ldots, r_m)) = \boldsymbol{c}_R[S] \ . \tag{24}$$

Since $(r_1, \ldots, r_m) \notin A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M), m, \epsilon, \beta_\epsilon)$ and $(r_1, \ldots, r_m) \notin A((\boldsymbol{w}_{M+1}, \ldots, \boldsymbol{w}_{2M}), m, \epsilon, \beta_\epsilon)$, there must exist codewords $[\boldsymbol{c}_i]_{i \in [2M]}$ such that

$$\forall i \in [M] : \boldsymbol{w}_i[S] = \boldsymbol{c}_i[S] \quad \text{and} \quad \forall i \in [M] : \boldsymbol{w}_{i+M}[S] = \boldsymbol{c}_{i+M}[S] \ .$$

This completes the proof of the claim. □

Given that the claim holds, we know from the induction hypothesis that the number of choices for $(r_1, \ldots, r_{m+1})$ that satisfy at least one of the 3 conditions in the claim can be upper bounded by

$$\begin{aligned}
&= |\mathbb{F}| \cdot (|A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M), m, \epsilon, \beta_\epsilon)| + |A((\boldsymbol{w}_{M+1}, \ldots, \boldsymbol{w}_{2M}), m, \epsilon, \beta_\epsilon)|) + |\mathbb{F}|^m \cdot |A((\boldsymbol{w}_L, \boldsymbol{w}_R), 1, \epsilon, \beta_\epsilon)| \\
&\leq |\mathbb{F}| \cdot \left(\frac{3^m |\mathbb{F}|^{m-1}}{\epsilon^2} + \frac{3^m |\mathbb{F}|^{m-1}}{\epsilon^2} \right) + |\mathbb{F}|^m \cdot \frac{3}{\epsilon^2} \\
&\leq \frac{3^{m+1} |\mathbb{F}|^m}{\epsilon^2} \ .
\end{aligned}$$

This is precisely $|A((\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{2M}), m+1, \epsilon, \beta_\epsilon)|$, thus completing the proof. □

A.2 Mutual correlated agreement preserves list decoding

In this section we restate and prove Lemma 3.13, which shows that mutual correlated agreement for multilinear proximity generators preserves list decoding.

Lemma 3.13. *Let $\mathcal{C}$ be a linear code with parameters $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$ and $m \in \mathbb{N}$, with $M := 2^m$. Additionally, let $\epsilon \in (0, 1/2n]$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}} + \epsilon)^{1/3} + \epsilon$ be such that $\{\beta_\epsilon n\} > 1/2$. Define $\delta := 1 - \lceil \beta_\epsilon n \rceil / n$. For every $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_M \in \mathbb{F}^n$, the following holds:*

$$\Pr_{\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m} \left[\begin{array}{c} \Lambda\left(\mathcal{C}, \sum_{j \in [M]} \boldsymbol{w}_j \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}), \delta\right) \\ \neq \\ \left\{ \sum_{j \in [M]} \boldsymbol{c}_j \cdot \mathrm{eq}(\langle j-1 \rangle, \boldsymbol{r}) : [\boldsymbol{c}_j]_{j \in [M]} \in \Lambda\left(\mathcal{C}^M, [\boldsymbol{w}_j]_{j \in [M]}, \delta\right) \right\} \end{array} \right] \leq \tfrac{3^m}{\epsilon^2 |\mathbb{F}|} \ .$$

Proof. The proof follows the template of a similar proof of [ACFY25, Lemma 4.13]. For any $\boldsymbol{z} \in \mathbb{F}^M$ of the form $\boldsymbol{z} = [\mathrm{eq}(j-1, \boldsymbol{r})]_{j \in [M]}$, for some $\boldsymbol{r} \in \mathbb{F}^m$, define

$$T_{\boldsymbol{z}} := \Lambda\left(\mathcal{C}, \sum_{j \in [M]} \boldsymbol{w}_j \cdot z_j, \delta\right)$$

$$S_{\boldsymbol{z}} \coloneqq \left\{ \sum_{j\in[M]} \boldsymbol{c}_j \cdot z_j : [\boldsymbol{c}_j]_{j\in[M]} \in \Lambda\left(\mathcal{C}^M, [\boldsymbol{w}_j]_{j\in[M]}, \delta\right) \right\}$$

We prove each direction of the required inclusion separately.

1. $S_{\boldsymbol{z}} \subseteq T_{\boldsymbol{z}}$: Let $\boldsymbol{c} \in S_{\boldsymbol{z}}$, this implies that $\boldsymbol{c} = \sum_{j\in[M]} \boldsymbol{c}_j \cdot z_j$ for some $[\boldsymbol{c}_j]_{j\in[M]} \in \Lambda\left(\mathcal{C}^M, [\boldsymbol{w}_j]_{j\in[M]}, \delta\right)$. Then there exists a set $S \subseteq [n]$ with $|S| = \lceil \beta_\epsilon n \rceil$ such that for each $j \in [M]$, $\boldsymbol{w}_j[S] = \boldsymbol{c}_j[S]$. Then, for every $i \in S$ it must be that

$$\sum_{j\in[M]} \boldsymbol{w}_j[i] \cdot z_j = \sum_{j\in[M]} \boldsymbol{c}_j[i] \cdot z_j = \boldsymbol{c}[i] \ ,$$

   and thus $\delta(\boldsymbol{c}, \sum_{j\in[M]} \boldsymbol{w}_j \cdot z_j) \leq \delta$ and $\boldsymbol{c} \in T_{\boldsymbol{z}}$.

2. $T_{\boldsymbol{z}} \subseteq S_{\boldsymbol{z}}$ with high probability: Consider $\boldsymbol{z}$ of the form $[\mathrm{eq}(j-1, \boldsymbol{r})]_{j\in[M]}$ such that for every $S \subseteq [n]$ of size $\lceil \beta_\epsilon n \rceil$, one of the following holds true:
   (a) for every $\boldsymbol{c} \in \mathcal{C}$, $\boldsymbol{c}[S] \neq \sum_{j\in[M]} \boldsymbol{w}_j[S] \cdot z_j$, or
   (b) for every $j \in [M]$, $\exists \boldsymbol{c}_j \in \mathcal{C}$ such that $\boldsymbol{c}_j[S] = \boldsymbol{w}_j[S]$.

   By Theorem 3.11, at least a $(1 - \frac{3^m}{\epsilon^2|\mathbb{F}|})$ fraction of choices of $\boldsymbol{r}$ result in such a $\boldsymbol{z}$.

   Consider $\boldsymbol{c} \in T_{\boldsymbol{z}}$ for such a $\boldsymbol{z}$. By definition of $T_{\boldsymbol{z}}$, $\boldsymbol{c} \in \mathcal{C}$ and $\delta(\boldsymbol{c}, \sum_{j\in[M]} \boldsymbol{w}_j \cdot z_j) \leq \delta$. This implies that there exists a set S of size $\lceil \beta_\epsilon n \rceil$ such that $\boldsymbol{c}[S] = \sum_{j\in[M]} \boldsymbol{w}_j[S] \cdot z_k$ implying that the first condition above does not hold. Thus for every $j \in [M]$, there exists $\boldsymbol{c}_j \in \mathcal{C}$ such that $\boldsymbol{c}_j[S] = \boldsymbol{w}_j[S]$. Thus

$$\boldsymbol{c}[S] = \sum_{j\in[M]} \boldsymbol{w}_j[S] \cdot z_j = \sum_{j\in[M]} \boldsymbol{c}_j[S] \cdot z_j \ ,$$

   and so $\delta(\boldsymbol{c}, \sum_{j\in[M]} \boldsymbol{c}_j \cdot z_j) \leq \delta < \delta_{\mathcal{C}}$. This implies that $\boldsymbol{c} = \sum_{j\in[M]} \boldsymbol{c}_j \cdot z_j$, and as a result $\boldsymbol{c} \in S_{\boldsymbol{z}}$.

   The above argument simultaneously holds for all $\boldsymbol{c} \in T_{\boldsymbol{z}}$, and since at least a $(1 - \frac{3^m}{\epsilon^2|\mathbb{F}|})$ fraction of all possible $\boldsymbol{z}$'s satisfy the required properties, this completes the proof. □

B Deferred proofs

In this section we restate and prove all the results whose proofs were deferred in the main body of the paper.

B.1 Proof of Lemma 5.3

Lemma 5.3. *Let $\varepsilon \in (0, \varepsilon(\mathcal{C})]$ and $\varepsilon' \in (0, \varepsilon(\mathcal{C}')]$, where $\varepsilon(\mathcal{C})$ is the proximity threshold of $\mathcal{C}$ (see Definition 4.7).* LossyBatch *is an RBRTE IOR from* $\mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$ *to* $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',2)}$ *with the following parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
3	$O(1)$	$O(\ell/\gamma_{\mathsf{lost}})$	$O(nk\ell)$	$O(nk\ell + T_{\mathcal{C}'})$	$O(nk\ell + T_{\mathcal{C}'})$	$O(\ell/\gamma_{\mathsf{lost}} + \log n)$

Proof. Completeness and the efficiency parameters follow directly from the construction in Fig. 6. We now present the proof of RBRTE. Given $\mathbb{i} := (\mathcal{C}, \varepsilon, \ell, \gamma_{\mathsf{lost}})$, we will show that the IOR is $(\vec{a}, \varphi, \theta)$–RBRTE, for $\vec{a} := 1^3$ and φ, θ defined below. Throughout, we denote $\mathbb{x} = (\boldsymbol{z}, [y_i]_{i\in[\ell]})$, $\vec{\mathbb{y}} = ([\boldsymbol{w}_i]_{i\in[\ell]})$, and also use inst to denote $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$.

Vertical predicate. Since the vertical predicate only changes on transcripts that end with the verifier's challenge, we only consider such transcripts. We describe φ as a tuple $[\varphi_i]_{i\in[3]}$ such that for the i-th round, $\varphi(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_i, \rho_i) = 1$ if and only if $\forall j \in [i] : \varphi_j(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_j, \rho_j) = 1$.

1. φ_1: The transcript includes $\boldsymbol{w}'$ sent by $\mathcal{P}$, and the out-of-domain evaluation point $\boldsymbol{z}_{\mathsf{ood}}$ sent by $\mathcal{V}$. We define $\varphi_1(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}', \boldsymbol{z}_{\mathsf{ood}})) = 0$ if and only if there exist two messages $\boldsymbol{m}'_1, \boldsymbol{m}'_2 \in \mathbb{F}^{k'}$ such that $\boldsymbol{m}'_1 \neq \boldsymbol{m}'_2$ but $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_1), \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_2) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$ and $\hat{m}'_1(\boldsymbol{z}_{\mathsf{ood}}) = \hat{m}'_2(\boldsymbol{z}_{\mathsf{ood}})$.

   By Lemma 3.5, the probability of this event is at most $|\Lambda(\mathcal{C}', \varepsilon')|^2 \cdot \log k'/2|\mathbb{F}|$. Thus the predicate value can flip from 1 to 0 in this round with probability at most $|\Lambda(\mathcal{C}', \varepsilon')|^2 \cdot \log k'/2|\mathbb{F}|$.

2. φ_2: The transcript now includes the out-of-domain evaluation value σ_{ood} sent by $\mathcal{P}$, and the random challenge $\boldsymbol{r}_R$ sent by $\mathcal{V}$ for the repetition check. We define $\varphi_2(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}', \boldsymbol{z}_{\mathsf{ood}}, \sigma_{\mathsf{ood}}, \boldsymbol{r}_R)) = 0$ if and only if there exists some $\boldsymbol{m}' \in \mathbb{F}^n$ such that $\boldsymbol{w}' \sim_{\varepsilon'} \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}')$ and $\hat{m}'(\boldsymbol{x}, \boldsymbol{y}) \neq \hat{m}'(\boldsymbol{x}', \boldsymbol{y})$ for some $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n}$ and $\boldsymbol{y} \in \{0,1\}^{\log k + \log \ell}$, but $\hat{m}'(\boldsymbol{x}, \boldsymbol{r}_R) = \hat{m}'(\boldsymbol{x}', \boldsymbol{r}_R)$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n}$.

   Consider an arbitrary $\boldsymbol{v} \in \mathbb{F}^{k'}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{v}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$ and there exists $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n}$ and $\boldsymbol{y} \in \{0,1\}^{\log k + \log \ell}$ such that $\hat{v}(\boldsymbol{x}, \boldsymbol{y}) \neq \hat{v}(\boldsymbol{x}', \boldsymbol{y})$. By Lemma 3.2, the probability that $\hat{v}(\boldsymbol{x}, \boldsymbol{r}_R) = \hat{v}(\boldsymbol{x}', \boldsymbol{r}_R)$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n}$ is at most $(\log k + \log \ell)/|\mathbb{F}|$. Applying the union bound over all such $\boldsymbol{v}$, the probability of the predicate flipping in this round is at most $|\Lambda(\mathcal{C}', \varepsilon')| \cdot (\log k + \log \ell)/|\mathbb{F}|$.

3. φ_3: For the last round, we define

$$\varphi_3(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}', \boldsymbol{z}_{\mathsf{ood}}, \sigma_{\mathsf{ood}}, \boldsymbol{r}_R, \beta_{\mathsf{eval}}, [\xi_{i,i'}]_{i\in[\ell],i'\in[\nu]}, \beta_{\mathsf{prox}})) = 0$$

   if and only if there exists some witness $(\boldsymbol{m}'_{\mathsf{bad}}, \boldsymbol{m}_G)$ that is 'bad', i.e. $\boldsymbol{m}'_{\mathsf{bad}}$ is not a valid witness for $\mathsf{Spot}^{(\mathcal{C},\mathcal{C}',\varepsilon,\ell)}$ but the transcript still accepts. We describe the exact cases in which the witness can be bad below, but first note that our conditions do not involve $\boldsymbol{m}_G$ since it corresponds to an 'honestly generated' word, due to $\boldsymbol{w}_G$ being output by the indexer. The witness is bad if at least one of the following is true:

   (a) ρ is bad: if $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ is such that
   - $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, and

- $\sum_{\boldsymbol{x}\in\{0,1\}^{\log k'}} f(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}), \hat{m}_G(\boldsymbol{x})) = \sigma$,

yet at least one of the following is not true:

$$\sum_{\boldsymbol{x}\in\{0,1\}^{\log k'}} f_{\mathsf{ood}}(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x})) = \sigma_{\mathsf{ood}} \ , \qquad \sum_{\boldsymbol{x}\in\{0,1\}^{\log k'}} f_{\mathsf{rep}}(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x})) = \sigma_{\mathsf{rep}} \ ,$$
$$\sum_{\boldsymbol{x}\in\{0,1\}^{\log k'}} f_{\mathsf{eval}}(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x})) = \sigma_{\mathsf{eval}} \ , \qquad \sum_{\boldsymbol{x}\in\{0,1\}^{\log k'}} f_{\mathsf{prox}}(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}), \boldsymbol{m}_G(\boldsymbol{x})) = \sigma_{\mathsf{prox}} \ .$$

For a fixed $\boldsymbol{m}'_{\mathsf{bad}}$, this occurs with probability at most $3/|\mathbb{F}|$ by Lemma 3.1. Applying the union bound over all $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, the probability of this event is at most $3 \cdot |\Lambda(\mathcal{C}', \varepsilon')|/|\mathbb{F}|$.

(b) $\boldsymbol{r}_L$ is bad: We now assume that ρ is not bad. We say $\boldsymbol{r}_L$ is bad if there exists $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that

- $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, and
- $\hat{m}'_{\mathsf{bad}}(\boldsymbol{r}_L, \boldsymbol{r}_R) = \sigma_{\mathsf{rep}}$ (since we can assume that ρ was not bad), yet
- there exists $\boldsymbol{x} \in \{0,1\}^{\log n}$ such that $\hat{m}'_{\mathsf{bad}}(\boldsymbol{x}, \boldsymbol{r}_R) \neq \sigma_{\mathsf{rep}}$ (i.e. $\hat{m}'_{\mathsf{bad}}(\boldsymbol{X}, \boldsymbol{r}_R) - \sigma_{\mathsf{rep}}$ is a non-zero polynomial.)

For a fixed $\boldsymbol{m}'_{\mathsf{bad}}$, by Lemma 3.1, the probability that $\hat{m}'_{\mathsf{bad}}(\boldsymbol{r}_L, \boldsymbol{r}_R) = \sigma_{\mathsf{rep}}$ is at most $(\log n)/|\mathbb{F}|$. Applying the union bound over all $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, the probability of this event is at most $(\log n) \cdot |\Lambda(\mathcal{C}', \varepsilon')|/|\mathbb{F}|$.

(c) β_{eval} is bad: We now assume that ρ and $\boldsymbol{r}_L$ are not bad. We say β_{eval} is bad if there exists $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that

- $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, and
- $\sum_{\boldsymbol{x}\in\{0,1\}^{\log k'}} f_{\mathsf{eval}}(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x})) = \sigma_{\mathsf{eval}}$ (since we can assume that ρ was not bad), yet
- there exists $i \in [\ell]$ such that $\hat{m}'_{\mathsf{bad}}(0^{\log n}, \langle i-1 \rangle, \boldsymbol{z}) \neq y_i$.

By Lemma 3.1, the probability that this occurs for a fixed $\boldsymbol{m}'_{\mathsf{bad}}$ is at most $\ell/|\mathbb{F}|$. Applying the union bound over all $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, the probability of this event is at most $\ell \cdot |\Lambda(\mathcal{C}', \varepsilon')|/|\mathbb{F}|$.

(d) Before we discuss the remaining challenges, we note that since $\boldsymbol{r}_R$, ρ, $\boldsymbol{r}_L$, and β_{eval} are not bad, it must be the case that $\hat{m}'_{\mathsf{bad}}(\boldsymbol{x}, \boldsymbol{y}) = \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}', \boldsymbol{y})$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n}$ and $\boldsymbol{y} \in \{0,1\}^{\log k + \log \ell}$. We refer to such bad messages as having a 'repeated structure'.

(e) β_{prox} is bad: We now assume that ρ, $\boldsymbol{r}_L$, and β_{eval} are not bad. We say β_{prox} is bad if there exists $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that

- $\boldsymbol{w}' \sim_{\varepsilon'} \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}})$, and
- $\sum_{i\in[\ell], i'\in[\nu]} \beta_{\mathsf{prox}}^{i+(i'-1)\ell} \cdot v_{i,i'} = \langle \boldsymbol{m}'_{\mathsf{bad}}, \boldsymbol{m}_G, \boldsymbol{q}_{\mathsf{prox}} \rangle$ (since we can assume that ρ is not bad), and
- $\boldsymbol{m}'_{\mathsf{bad}}$ has repeated structure, i.e. there exists $\boldsymbol{m}_{\mathsf{conc}} \in \mathbb{F}^{k\ell}$ such that $\boldsymbol{m}'_{\mathsf{bad}} = \boldsymbol{m}^n_{\mathsf{conc}}$, yet
- there exists $i \in [\ell]$ and $i' \in [\nu]$ such that $v_{i,i'} \neq \langle \boldsymbol{g}_{\xi_{i,i'}}, \boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik] \rangle$, where $\boldsymbol{m}'_{\mathsf{bad}} = \boldsymbol{m}^n_{\mathsf{conc}}$.

By Lemma 3.1, the probability that this occurs for a fixed $\boldsymbol{m}'_{\mathsf{bad}}$ is at most $\ell \cdot \nu/|\mathbb{F}|$. Applying the union bound over all $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, the probability of this event is at most $\ell \cdot \nu \cdot |\Lambda(\mathcal{C}', \varepsilon')|/|\mathbb{F}|$.

(f) $[\xi_{i,i'}]_{i\in[\ell],i'\in[\nu]}$ is bad: Finally, we assume that all the previous challenges are not bad. We say $[\xi_{i,i'}]_{i\in[\ell],i'\in[\nu]}$ is bad if there exists $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that

- $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, and
- $\boldsymbol{m}'_{\mathsf{bad}}$ has repeated structure, i.e. there exists $\boldsymbol{m}_{\mathsf{conc}} \in \mathbb{F}^{k\ell}$ such that $\boldsymbol{m}'_{\mathsf{bad}} = \boldsymbol{m}^n_{\mathsf{conc}}$, and
- for all $i \in [\ell], i' \in [\nu]$, we have $v_{i,i'} = \langle \boldsymbol{g}_{\xi_{i,i'}}, \boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik]\rangle$ (since β_{prox} is not bad), yet
- there exists a set $B \subseteq [\ell]$ such that $|B| > \gamma_{\mathsf{lost}} \cdot \ell$ and $\delta(\boldsymbol{w}_i, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik])) > \varepsilon$ for all $i \in B$.

Consider a fixed $i \in B$. If $\delta(\boldsymbol{w}_i, \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik])) > \varepsilon$ the probability over the choice of $\{\xi_{i,i'}\}_{i'\in[\nu]}$ that $v_{i,i'} = \langle \boldsymbol{g}_{\xi_{i,i'}}, \boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik]\rangle = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_{\mathsf{conc}}[(i-1)k+1 : ik])[\xi_{i,i'}]$ for all $i' \in [\nu]$ is at most $(1-\varepsilon)^\nu$. This occurs independently for each $i \in B$, and therefore the probability of this event is at most $(1-\varepsilon)^{\nu\cdot|B|} \le (1-\varepsilon)^{\nu\cdot\gamma_{\mathsf{lost}}\cdot\ell} \le 2^{-\lambda}$ if $\ell \ge -\lambda/\log(1-\varepsilon)$ and $\nu \ge 1/\gamma_{\mathsf{lost}}$. Applying the union bound over all $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, the probability of this event is at most $2^{-\lambda} \cdot |\Lambda(\mathcal{C}', \varepsilon')|$.

We can apply the union bound over all of these possibilities to show that the probability of the predicate flipping from 1 to 0 in this round is negligible.

Challenge predicate. The challenge predicate always outputs 1.

Tree extractor. Notice that the arity parameters are 1^3, and thus the extractor is straight-line. Given a tree with the 'output' witness $\mathbb{w}' = (\boldsymbol{m}', \boldsymbol{m}_G)$, the extractor defines $\boldsymbol{v} = \boldsymbol{m}'[1 : k\ell]$ and outputs

$$\mathbb{w} \coloneqq [\boldsymbol{m}_i \coloneqq \boldsymbol{v}[(i-1)k+1 : ik]]_{i\in[\ell]}$$

By definition of the vertical predicate, it must be the case that $(\mathsf{inst}, \mathbb{w}) \in \mathsf{LHEP}^{(\mathcal{C},\varepsilon,\ell,\gamma_{\mathsf{lost}})}$. □

B.2 Proof of Lemma 6.2

Lemma 6.2. *Let $\mathcal{C}$ and $\mathcal{C}'$ be as defined in Fig. 7. Additionally, let $\varepsilon \in (0, \min(\delta_{\mathcal{C}}/5, \varepsilon(\mathcal{C}))]$ and $\varepsilon' \in (0, \varepsilon(\mathcal{C}')]$, where $\varepsilon(\mathcal{C})$ is the proximity threshold of $\mathcal{C}$ (see Definition 4.7).* TensorReduce *is an RBRTE IOR from* $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ *to* $\mathsf{SEP}^{(\mathcal{C}',\varepsilon',5)}$ *with the following efficiency parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$\log k_1 + 2\log k_2 + 8$	$O(\log k)$	$O(\lambda/\varepsilon)$	$O(n')$	$O(n + T_{\mathcal{C}'})$	$O(n + T_{\mathcal{C}'})$	$O(\lambda/\varepsilon + \log n)$

where $\mathcal{T} = \mathcal{C}_1 \otimes \mathcal{C}_2$ is a tensor code such that $\mathcal{C}_1$ has parameters $(\mathbb{F}, n_1, k_1, \delta_1)$ and $\mathcal{C}_2$ has parameters $(\mathbb{F}, n_2, k_2, \delta_2)$, and $k = k_1 k_2$ and $n = n_1 n_2$.

Proof. Completeness and efficiency parameters follow from Lemma 6.2 and Lemma 6.4.

We now prove that Fig. 7 is RBRTE. Given $\mathbb{i} \coloneqq (\mathcal{T}, \varepsilon, 1)$, we will show that the IOR is $(\vec{a}, \varphi, \theta)$–RBRTE, where $\vec{a} \coloneqq ([2]^{\log k_1} \| [1]^{2\log k_2 + 8})$. φ and θ are now described in detail. Throughout, we have $\mathbb{i} \coloneqq (\mathcal{T}, \varepsilon, 1)$, $\mathbb{x} = (\boldsymbol{z}, y)$, $\vec{\mathbb{y}} = (\boldsymbol{w})$, and also use inst to denote $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$.

Vertical predicate. Since the vertical predicate only changes on transcripts that end with the verifier's challenge, we only consider such transcripts. We describe φ as a tuple $[\varphi_i]_{i\in[\log k_1 + 2\log k_2 + 8]}$ such that for the i-th round, $\varphi(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_i, \rho_i) = 1$ if and only if $\forall j \in [i] : \varphi_j(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_j, \rho_j) = 1$.

1. φ_i for $i \in [\log k_1]$: These are the first $\log k_1$ rounds of $\mathsf{RandomizeEval}$. Thus, the transcript up to this point is $\mathsf{Tr} = ([v_j(X), r_j]_{j\in[i]})$, where $v_j(X)$ is the j-th univariate polynomial sent by $\mathcal{P}$ and r_j is the j-th challenge sent by $\mathcal{V}$. Recall that $\boldsymbol{r}_1$ not only needs to be 'good' as sumcheck challenges, but it also needs to be a 'good' challenge for the mutual correlated agreement. Thus, we define $\varphi_i(\mathsf{inst}, \mathsf{Tr}) = 0$ if and only if at least one of the following is true:

   (a) There exists $\boldsymbol{m} \in \mathbb{F}^k$ such that $\mathsf{Enc}_{\mathcal{T}}(\boldsymbol{m}) \in \Lambda(\mathcal{T}, \boldsymbol{w}, \varepsilon)$ and we have

   $$v_i(r_i) = \sum_{\boldsymbol{b}\in\{0,1\}^{\log k - i}} \hat{m}(r_1, \ldots, r_i, \boldsymbol{b}) \cdot \mathrm{eq}((r_1, \ldots, r_i, \boldsymbol{b}), \boldsymbol{z}) \ .$$

   Yet the round polynomial $v_i(X)$ sent by $\mathcal{P}$ is not correct, i.e.,

   $$v_i(X) \neq \sum_{\boldsymbol{b}\in\{0,1\}^{\log k - i}} \hat{m}(r_1, \ldots, r_{i-1}, X, \boldsymbol{b}) \cdot \mathrm{eq}((r_1, \ldots, r_{i-1}, X, \boldsymbol{b}), \boldsymbol{z}) \ .$$

   (b) r_i is a bad challenge with respect to Lemma A.3. where $\epsilon = 1/2n_2$ and $\beta_\epsilon \geq (1 - \delta_{\mathcal{C}_2} + \epsilon)^{1/3} + \epsilon$ is the largest possible value such that $\beta_\epsilon \leq 1 - \varepsilon'$, and $\{\beta_\epsilon n_2\} > 1/2$. Since $\varepsilon' \in \varepsilon(\mathcal{C}_2)$, we know that this β_ϵ satisfies $\lceil (1-\varepsilon')n_2 \rceil - 1 < \beta_\epsilon n_2 \leq \lceil (1-\varepsilon')n_2 \rceil$.

   By Lemma 3.1, the probability of Item 1a is at most $2/|\mathbb{F}|$ for a fixed $\boldsymbol{m}$. Applying the union bound over all $\boldsymbol{m} \in \Lambda(\mathcal{T}, \boldsymbol{w}, \varepsilon)$, the probability of this event is at most $2 \cdot |\Lambda(\mathcal{T}, \varepsilon)|/|\mathbb{F}|$. By Lemma A.3, the probability of Item 1b is at most $3/(\epsilon^2|\mathbb{F}|)$.

2. φ_i for $i \in [\log k_1 + 1, \log k]$: These are the last $\log k_2$ rounds of $\mathsf{RandomizeEval}$. These do not have additional constraints imposed by mutual correlated agreement not failing, and therefore $\varphi_i(\mathsf{inst}, \mathsf{Tr} = [v_j(X), r_j]_{j\in[i]}) = 0$ if and only if Item 1a is true.

3. $\varphi_{\log k+1}$: The transcript now includes $\boldsymbol{w}'$ sent by $\mathcal{P}$, and the out-of-domain evaluation point $\boldsymbol{z}_{\mathsf{ood}}$ sent by $\mathcal{V}$. $\varphi_{\log k+1}(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}, \boldsymbol{z}_{\mathsf{ood}})) = 0$ if and only if there exists $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^{n_2}$ such that $\boldsymbol{m}'_{\mathsf{bad}} \neq \boldsymbol{m}'$ and $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$ and $\hat{m}'_{\mathsf{bad}}(\boldsymbol{z}_{\mathsf{ood}}) = \hat{m}'(\boldsymbol{z}_{\mathsf{ood}})$. By Lemma 3.5, the probability of this event occurring is at most $|\Lambda(\mathcal{C}', \varepsilon')|^2 \cdot \log n/2|\mathbb{F}|$.

4. $\varphi_{\log k+2}$: The transcript now includes the out-of-domain evaluation value σ_{ood} sent by $\mathcal{P}$, and the random indices $[\xi_i]_{i\in[\ell]}$ sent by $\mathcal{V}$.

   Since $\varphi_{\log k+1}(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}, \boldsymbol{z}_{\mathsf{ood}})) = 1$, there exists a unique $\boldsymbol{m}' \in \mathbb{F}^{n_2}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}') \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$ and $\hat{m}'(\boldsymbol{z}_{\mathsf{ood}}) = \sigma_{\mathsf{ood}}$. Let $\boldsymbol{m}_{\mathsf{first}} \in \mathbb{F}^{k_2}$ be the first k_2 entries of $\boldsymbol{m}'$. Define $\boldsymbol{W}_{\mathsf{int}} \in \mathbb{F}^{k_1 \times n_2}$ such that for all $j \in [n_2]$ we have

   $$\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j]) \sim_{\varepsilon_{\mathsf{int}}} \boldsymbol{W}[\cdot, j] \ .$$

   Since $\varepsilon_{\mathsf{int}} < \varepsilon < \delta_{\mathcal{C}_1}/2$, there exists a unique such $\boldsymbol{W}_{\mathsf{int}}[\cdot, j]$ for each $j \in n_2$. It is important to note that not all columns of $\boldsymbol{W}$ are guaranteed to be $\varepsilon_{\mathsf{int}}$-close to a codeword of $\mathcal{C}_1$. However, for simplicity of exposition, we use $\hat{W}_{\mathsf{int}}[\cdot, j]$ to denote the multilinear extension of $\boldsymbol{W}_{\mathsf{int}}[\cdot, j]$, whether or not such a $\boldsymbol{W}_{\mathsf{int}}[\cdot, j]$ exists that satisfies the above property. If no such $\boldsymbol{W}_{\mathsf{int}}[\cdot, j]$ exists, $\boldsymbol{W}_{\mathsf{int}}[\cdot, j] = \perp^{k_1}$ and $\hat{W}_{\mathsf{int}}[\cdot, j]$ evaluates to $\perp$ at all points.

   Finally, we define the following set of 'bad' indices:

   $$B \coloneqq \{j \in [n_2] : \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\mathsf{first}})[j] \neq \hat{W}_{\mathsf{int}}[\cdot, j](\boldsymbol{r}_1)\} \ .$$

That is, B is the set of indices where the folded columns do not equal the expected folded value. Thus, we now say $\varphi_{\log k+2}(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}, \boldsymbol{z}_{\mathsf{ood}}, \sigma_{\mathsf{ood}}, [\xi_i]_{i\in[\ell]})) = 0$ if and only if

$$\left|B \cap \{\xi_i\}_{i\in[\ell]}\right| < \gamma_{\mathsf{lost}} \cdot \ell \quad \text{and yet,} \quad |B| \geq \varepsilon_{\mathsf{int}} \cdot n_2 \ .$$

That is, even though there are a lot of columns that do not fold correctly to the expected values, the number of such columns among the sampled columns is still a small fraction: small enough to evade detection by the invocation of LossyBatch.

We analyze the probability of $\varphi_{\log k+2}$ evaluating to 0 using a concentration argument. Define indicator random variables X_j for each $j \in B$ such that $X_j = 1$ if there exists $i \in [\ell]$ such that $\xi_i = j$, and 0 otherwise. The query indices are chosen without replacement, so $\Pr[X_j = 1] = \ell/n_2$ for all $j \in B$. If $X := \sum_{j\in B} X_j$ is the total number of bad indices, then $\mathbb{E}[X] = \ell|B|/n_2 \geq \varepsilon_{\mathsf{int}} \cdot \ell$. Applying the Chernoff bound (which applies to negatively correlated random variables like X_j), we get

$$\begin{aligned}
\Pr[X < \gamma_{\mathsf{lost}} \cdot \ell] &\leq \Pr[X < \gamma_{\mathsf{lost}}/\varepsilon_{\mathsf{int}} \cdot \mathbb{E}[X]] \\
&\leq \exp(-(1 - \gamma_{\mathsf{lost}}/\varepsilon_{\mathsf{int}})^2 \mathbb{E}[X]/2) \\
&\leq \exp(-(1 - \gamma_{\mathsf{lost}}/\varepsilon_{\mathsf{int}})^2 \cdot \varepsilon_{\mathsf{int}} \cdot \ell/2) \\
&= \exp(-\ell) \leq 2^{-\lambda} \ ,
\end{aligned}$$

where the equality follows from $\gamma_{\mathsf{lost}} = \varepsilon_{\mathsf{int}} - \sqrt{2\varepsilon_{\mathsf{int}}}$ and the last inequality follows from $\ell \geq \lambda \ln 2$.

5. φ_i for $i \in [\log k + 3, \log k + \log k_2 + 5]$: These rounds correspond to the invocation of SpotCheckReduce, and therefore the vertical predicate is exactly as defined in Lemma 6.4.

6. φ_i for $i \in [\log k + \log k_2 + 6, \log k + \log k_2 + 8]$: These rounds correspond to the invocation of LossyBatch, and therefore the vertical predicate is exactly as defined in Lemma 5.3. However, the final round $\log k + \log k_2 + 8$ has the additional batching random challenge ρ sent by $\mathcal{V}$. Therefore, the vertical predicate $\varphi_{\log k+\log k_2+8}$ evaluates to 0 if the vertical predicate defined by LossyBatch evaluates to 0, or if there exist $\boldsymbol{m}', \boldsymbol{m}_{\mathsf{Spot}}, \boldsymbol{m}'_{\mathsf{Spot}}, \boldsymbol{m}_{\mathsf{LHEP}}, \boldsymbol{m}'_{\mathsf{LHEP}}$ such that
   - $\boldsymbol{m}' \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon')$, $\boldsymbol{m}_{\mathsf{Spot}} \in \Lambda(\mathcal{C}', \boldsymbol{w}_{\mathsf{Spot}}, \varepsilon')$, $\boldsymbol{m}'_{\mathsf{Spot}} \in \Lambda(\mathcal{C}', \boldsymbol{w}_{\mathsf{Spot}}, \varepsilon')$, $\boldsymbol{m}_{\mathsf{LHEP}} \in \Lambda(\mathcal{C}', \boldsymbol{w}_{\mathsf{LHEP}}, \varepsilon')$, $\boldsymbol{m}'_{\mathsf{LHEP}} \in \Lambda(\mathcal{C}', \boldsymbol{w}_{\mathsf{LHEP}}, \varepsilon')$,
   - $\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} f'(\boldsymbol{x}, \hat{m}'(\boldsymbol{x}), \hat{m}_{\mathsf{Spot}}(\boldsymbol{x}), \hat{m}'_{\mathsf{Spot}}(\boldsymbol{x}), \hat{m}_{\mathsf{LHEP}}(\boldsymbol{x}), \hat{m}'_{\mathsf{LHEP}}(\boldsymbol{x})) = \sigma'$, and yet
   - at least one of the following is not true:

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \mathrm{eq}(\boldsymbol{x}, (0^{\log n - \log k} \,\|\, \boldsymbol{r}_2)) \cdot \hat{m}'(\boldsymbol{x}) = y_{\boldsymbol{r}} \ , \\
&\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \mathrm{eq}(\boldsymbol{x}, \boldsymbol{z}_{\mathsf{ood}}) \cdot \hat{m}'(\boldsymbol{x}) = \sigma_{\mathsf{ood}} \ , \\
&\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} f_{\mathsf{Spot}}(\boldsymbol{x}, \hat{m}'(\boldsymbol{x}), \hat{m}_{\mathsf{Spot}}(\boldsymbol{x}), \hat{m}'_{\mathsf{Spot}}(\boldsymbol{x})) = \sigma_{\mathsf{Spot}} \ , \\
&\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} f_{\mathsf{LHEP}}(\boldsymbol{x}, \hat{m}_{\mathsf{LHEP}}(\boldsymbol{x}), \hat{m}'_{\mathsf{LHEP}}(\boldsymbol{x})) = \sigma_{\mathsf{LHEP}} \ . \qquad (25)
\end{aligned}$$

   By Lemma 3.1, this occurs with probability at most $3/|\mathbb{F}|$ for a fixed $(\boldsymbol{m}', \boldsymbol{m}_{\mathsf{Spot}}, \boldsymbol{m}'_{\mathsf{Spot}}, \boldsymbol{m}_{\mathsf{LHEP}}, \boldsymbol{m}'_{\mathsf{LHEP}})$. Applying the union bound over all $(\boldsymbol{m}', \boldsymbol{m}_{\mathsf{Spot}}, \boldsymbol{m}'_{\mathsf{Spot}}, \boldsymbol{m}_{\mathsf{LHEP}}, \boldsymbol{m}'_{\mathsf{LHEP}})$ that satisfy the above conditions, the probability of this event is at most $3 \cdot |\Lambda(\mathcal{C}', \varepsilon')|^5/|\mathbb{F}|$.

Challenge predicate. The challenge predicate takes as input all the verifier challenges across all root-to-leaf paths in T. Consider all the possible values of $\boldsymbol{r}_1$ in the k_1 rewindings, and let these be $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k_1}$. The challenge predicate θ outputs 0 if and only if $\mathbf{eq}(\boldsymbol{v}_1), \ldots, \mathbf{eq}(\boldsymbol{v}_{k_1})$ are linearly dependent. We compute this probability as follows:

1. There exist $i, i' \in [k_1]$ such that $\boldsymbol{v}_i = \boldsymbol{v}_{i'}$. That means that the last random challenge in $\boldsymbol{v}_i = \boldsymbol{v}_{i'}$ is equal, which occurs with probability $1/|\mathbb{F}|$. Applying the union bound over all $\binom{k_1}{2}$ pairs of indices, we get that the probability of this event is at most $k_1^2/2|\mathbb{F}|$.

2. Now, assuming all $\boldsymbol{v}_i$ are distinct, $\mathbf{eq}(\boldsymbol{v}_1), \ldots, \mathbf{eq}(\boldsymbol{v}_{k_1})$ are also distinct. Now, consider the matrix $\boldsymbol{V}$ such that $\boldsymbol{V}[i, j] = \mathbf{eq}(\boldsymbol{v}_i)[j]$ for all $i, j \in [k_1]$. $\mathbf{eq}(\boldsymbol{v}_1), \ldots, \mathbf{eq}(\boldsymbol{v}_{k_1})$ are linearly dependent if and only if $\det(\boldsymbol{V}) = 0$. Now, $\det(\boldsymbol{V})$ is a multivariate polynomial in $k_1 \log k_1$ variables, so we can write this as a polynomial $\hat{D}(X_1, \ldots, X_{k_1 \log k_1})$. The degree of $\hat{D}$ is at most $k_1 \log k_1$ since each entry of $\boldsymbol{V}$ has degree at most $\log k_1$, and the determinant is a sum of products of k_1 entries. Furthermore, clearly $\hat{D}$ is not the zero polynomial since if $\boldsymbol{v}_i = \langle i \rangle$, then $\boldsymbol{V}$ is a permutation matrix and therefore $\det(\boldsymbol{V}) = 1$. Thus, by Lemma 3.1, we have that $\Pr[\det(\boldsymbol{V}) = 0] \leq \deg(\hat{D})/|\mathbb{F}| = k_1 \log k_1/|\mathbb{F}|$.

Thus, by applying the union bound, the probability that the challenge predicate outputs 0 is at most $(k_1^2/2 + k_1 \log k_1)/|\mathbb{F}|$.

Tree extractor. $\mathcal{E}^{\mathrm{Tree}}$ takes as input a constrained transcript tree T such that the vertical predicate is satisfied for every path from the root to a leaf, and the challenge predicate is satisfied. $\mathcal{E}^{\mathrm{Tree}}$ then behaves as follows.

1. For each root-to-leaf path of T, $\mathcal{E}^{\mathrm{Tree}}$ parses the output witness $\mathbb{w}' = (\boldsymbol{m}', \boldsymbol{m}_{\mathsf{Spot}}, \boldsymbol{m}'_{\mathsf{Spot}}, \boldsymbol{m}_{\mathsf{LHEP}}, \boldsymbol{m}'_{\mathsf{LHEP}})$. Then, $\mathcal{E}^{\mathrm{Tree}}$ computes $\boldsymbol{m}_{\boldsymbol{r}_1} \in \mathbb{F}^{k_2}$ as $\boldsymbol{m}'[1 : k_2]$ and also computes $\boldsymbol{c}_{\boldsymbol{r}_1} \coloneqq \mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})$.

2. Let $\boldsymbol{m}_{\boldsymbol{r}_1,1}, \ldots, \boldsymbol{m}_{\boldsymbol{r}_1,k_1} \in \mathbb{F}^{k_2}$ be the output witnesses obtained across all k_1 rewindings, and define $\boldsymbol{M}' \in \mathbb{F}^{k_1 \times k_2}$ such that $\boldsymbol{M}'[i, \cdot] = \boldsymbol{m}_{\boldsymbol{r}_1,i}$ for all $i \in [k_1]$. Also define $\boldsymbol{C} \in \mathbb{F}^{k_1 \times k_1}$ such that $\boldsymbol{C}[i, \cdot] = \mathbf{eq}(\boldsymbol{v}_i)$ for all $i \in [k_1]$, where $\boldsymbol{v}_i$ is the value of $\boldsymbol{r}_1$ in the i-th rewinding.

3. $\mathcal{E}^{\mathrm{Tree}}$ outputs $\boldsymbol{M} \coloneqq \boldsymbol{C}^{-1} \cdot \boldsymbol{M}'$. Since the challenge predicate is satisfied, $\boldsymbol{C}$ is invertible, and thus $\boldsymbol{M}$ is computable in polynomial time.

We now show that $\mathcal{E}^{\mathrm{Tree}}$'s output is a valid witness for the $\mathsf{MEP}^{(\mathcal{T},\varepsilon,1)}$ claim.

1. Since $\varphi_{\log k + \log k_2 + 8}$ evaluates to 1, it must be that Eq. (25) are all true.

2. Since $\varphi_{\log k + \log k_2 + 6}$ to $\varphi_{\log k + \log k_2 + 8}$ all evaluate to 1, it must be that the guarantee of the $\mathsf{LHEP}^{(\mathcal{C}',\varepsilon_{\mathsf{int}},\gamma_{\mathsf{lost}},\ell)}$ claim is satisfied. That is, there exists $I \subseteq [\ell]$ such that $\hat{W}_{\mathsf{int}}[\cdot, \xi_j](\boldsymbol{r}_1) = y_j$ and $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, \xi_j]) \sim_{\varepsilon_{\mathsf{int}}} \boldsymbol{W}[\cdot, \xi_j]$ for all $j \in I$, where $|I| \geq (1 - \gamma_{\mathsf{lost}}) \cdot \ell$.

3. Since $\varphi_{\log k + 3}$ to $\varphi_{\log k + \log k_2 + 5}$ all evaluate to 1, it must be that the guarantee of the $\mathsf{Spot}^{(\mathcal{C}_2,\mathcal{C}',\varepsilon',\ell)}$ claim is satisfied. Thus, there exists $\boldsymbol{m}_{\boldsymbol{r}_1} \in \mathbb{F}^{k_2}$ such that $\boldsymbol{m}' = \boldsymbol{m}_{\boldsymbol{r}_1}^{n_2/k_2}$, and $\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})[\xi_j] = y_j$ for all $j \in [\ell]$. Since $\boldsymbol{m}_{\mathsf{first}}$ was simply defined to be the first k_2 entries of $\boldsymbol{m}'$, we get that $\boldsymbol{m}_{\boldsymbol{r}_1} = \boldsymbol{m}_{\mathsf{first}}$.

4. Combining the above two points, we get that there exists $I \subseteq [\ell]$ such that $\mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, \xi_j]) \sim_{\varepsilon_{\mathsf{int}}} \boldsymbol{W}[\cdot, \xi_j]$, and $\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})[\xi_j] = \hat{W}_{\mathsf{int}}[\cdot, \xi_j](\boldsymbol{r}_1)$ for all $j \in I$, where $|I| \geq (1 - \gamma_{\mathsf{lost}})\ell$. Thus, if B is the set of 'bad' indices as defined in Item 4, then $|B \cap \{\xi_j\}_{j \in [\ell]}| < \gamma_{\mathsf{lost}} \cdot \ell$.

5. Since $\varphi_{\log k+2}$ evaluates to 1, and $|B \cap \{\xi_j\}_{j\in[\ell]}| < \gamma_{\mathsf{lost}} \cdot \ell$, we have that $|B| < \varepsilon_{\mathsf{int}} n_2$. In other words, there exists an agreement set $A \subseteq [n_2]$ such that $|A| \geq (1-\varepsilon_{\mathsf{int}})n_2$ and for all $j \in A$, $\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})[j] = \hat{W}_{\mathsf{int}}[\cdot, j](\boldsymbol{r}_1)$. Furthermore, since the first equation in Eq. (25) is true, we have that $\hat{m}'(0^{\log n - \log k} \,\|\, \boldsymbol{r}_2) = y_{\boldsymbol{r}}$, which gives us that $\hat{m}_{\boldsymbol{r}_1}(\boldsymbol{r}_2) = y_{\boldsymbol{r}}$.

6. Since $\varphi_{\log k+1}$ evaluates to 1, there is a unique $\boldsymbol{m}' \in \mathbb{F}^{n_2}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}') \sim_{\varepsilon'} \boldsymbol{w}'$ and the second claim of Eq. (25) is satisfied. Thus, $\boldsymbol{m}_{\boldsymbol{r}_1}$ is also the unique message such that for all $j \in A$, $\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{\boldsymbol{r}_1})[j] = \hat{W}_{\mathsf{int}}[\cdot, j](\boldsymbol{r}_1)$, and $\hat{m}_{\boldsymbol{r}_1}(\boldsymbol{r}_2) = y_{\boldsymbol{r}}$.

7. Since φ_1 to $\varphi_{\log k}$ all evaluate to 1, we have the following two guarantees:
   - By Item 1b, $\boldsymbol{r}_1$ is good with respect to mutual correlated agreement (Lemma 3.13), and thus there exist messages $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_{k_1} \in \mathbb{F}^{k_2}$ such that for all $i \in [k_1]$ and $j \in A$, $\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_i)[j] = \boldsymbol{W}_{\mathsf{int}}[i, j]$, and $\sum_{i\in[k_1]} \boldsymbol{m}_i \cdot \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}_1) = \boldsymbol{m}_{\boldsymbol{r}_1}$. If $\boldsymbol{m} \coloneqq (\boldsymbol{m}_1 \,\|\, \cdots \,\|\, \boldsymbol{m}_{k_1}) \in \mathbb{F}^k$, then clearly $\hat{m}_{\boldsymbol{r}_1}(\boldsymbol{r}_2) = y_{\boldsymbol{r}}$ implies that $\hat{m}(\boldsymbol{r}) = y_{\boldsymbol{r}}$. Furthermore, for each $j \in A$ there are at least $(1-\varepsilon_{\mathsf{int}})n_1$ values of $i \in [n_1]$ such that $\boldsymbol{W}[i, j] = \mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j])[i]$. Thus, for each such pair (i, j), we have

$$\boldsymbol{W}[i, j] = \mathsf{Enc}_{\mathcal{C}_1}(\boldsymbol{W}_{\mathsf{int}}[\cdot, j])[i] = \mathsf{Enc}_{\mathcal{C}_1}([\mathsf{Enc}_{\mathcal{C}_2}(\boldsymbol{m}_{i'})[j]]_{i'\in[k_1]})[i] = \mathsf{Enc}_{\mathcal{T}}(\boldsymbol{M})[i, j] \ ,$$

     where $\boldsymbol{M} \in \mathbb{F}^{k_1 \times k_2}$ is the matrix form of $\boldsymbol{m}$. The number of such pairs (i, j) is at least $(1-\varepsilon_{\mathsf{int}})n_1 \cdot |A| \geq (1-\varepsilon_{\mathsf{int}})^2 n_1 n_2 = (1-\varepsilon)n_1 n_2$ as desired.
   - Finally, by Item 1a, $\boldsymbol{r}$ is a good sumcheck challenge, and therefore $\hat{m}(\boldsymbol{r}) = y_{\boldsymbol{r}}$ implies that $\hat{m}(\boldsymbol{z}) = y$ (the details of this argument are nearly identical to the tree extractor correctness proof in Appendix B.3, so we refer the reader to that proof for details).

□

B.3 Proof of Lemma 6.4

Lemma 6.4. *Let $\varepsilon \in [0, \varepsilon(\mathcal{C}'))$, where $\varepsilon(\mathcal{C})$ is the proximity threshold of $\mathcal{C}$ (see Definition 4.7). Fig. 8 is an IOR from $\mathsf{Spot}^{(\mathcal{C},\mathcal{C}',\varepsilon,\ell)}$ to $\mathsf{SEP}^{(\mathcal{C}',\varepsilon,3)}$ with the following efficiency parameters:*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$\log k + 3$	$O(\log k)$	0	$O(n')$	$O(T_{\mathcal{C}'})$	$O(T_{\mathcal{C}'})$	$O(\log n)$

where $\mathcal{C} = \mathcal{B} \otimes \mathcal{B}$ is a tensor code such that $\mathcal{B}$ has parameters $(\mathbb{F}, \sqrt{n}, \sqrt{k}, \delta_{\mathcal{B}})$ and $\mathcal{C}'$ is a linear code with parameters $(\mathbb{F}, n', n, \delta_{\mathcal{C}'})$.

Proof. Completeness and all efficiency parameters other than prover time immediately follow from Fig. 8. We briefly argue why Step 10 and Step 13 only require $O(n')$ time for the prover, since this is not straightforward.

- Step 10: $\mathcal{P}$ precomputes $\hat{G}_{\mathcal{B}}(\boldsymbol{x}_R, \boldsymbol{t}_R)$ for all $\boldsymbol{x}_R \in \{0,1\}^{\log k/2}$, which takes $O(k)$ time. Then for each round $i \in [\log k/2]$: $\mathcal{P}$ computes and stores $\hat{m}(\boldsymbol{r}_i, X, \boldsymbol{x}_L, \boldsymbol{x}_R)$ and $\hat{G}_{\mathcal{B}}(\boldsymbol{r}_i, X, \boldsymbol{x}_L, \boldsymbol{t}_L)$ for all $X \in \{0,1\}$, $\boldsymbol{x}_L \in \{0,1\}^{\log k/2-i}$ and $\boldsymbol{x}_R \in \{0,1\}^{\log k/2}$. Since $\mathcal{P}$ explicitly stores these values for the previous round, in the current round these values can be computed by 'folding' 2 previously computed values. This way, the number of field operations needed in round i is $O(k/2^i)$, and the total number of operations sums to $O(k)$.
- Step 13: $\mathcal{P}$ precomputes $\hat{G}_{\mathcal{B}}(\boldsymbol{r}_L, \boldsymbol{t}_L)$ and uses the folded polynomials for $\hat{G}_{\mathcal{B}}$ and $\hat{m}$ from previous rounds to compute the new folded polynomials and $h_i(X)$. Similarly, all $\log k/2$ rounds for this step also only require $O(k)$ field operations in total.

We now prove that Fig. 8 is RBRTE. Given $\mathbb{i} := (\mathcal{C}, \mathcal{C}', \varepsilon, \ell)$, we will show that the IOR is $(\vec{a}, \varphi, \theta)$–RBRTE, for $\vec{a} := 1^{\log k+3}$ and φ, θ defined below. Throughout, we denote $\mathbb{x} = ([\xi_i]_{i\in[\ell]}, [y_i]_{i\in[\ell]})$, $\vec{\mathbb{y}} = (\boldsymbol{w}')$, and also use inst to denote $(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}})$.

Vertical predicate. Since the vertical predicate only changes on transcripts that end with the verifier's challenge, we only consider such transcripts. We describe φ as a tuple $[\varphi_i]_{i\in[\log k+3]}$ such that for the i-th round, $\varphi(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_i, \rho_i) = 1$ if and only if $\forall j \in [i] : \varphi_j(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_j, \rho_j) = 1$.

1. φ_1: The transcript includes $\boldsymbol{w}_c$ sent by $\mathcal{P}$, and $\boldsymbol{t}$ sent by $\mathcal{V}$. We define $\varphi_1(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}_c, \boldsymbol{t})) = 0$ if and only if there exist two messages $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{F}^n$ such that $\boldsymbol{c}_1 \neq \boldsymbol{c}_2$ but $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{c}_1), \mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{c}_2) \in \Lambda(\mathcal{C}', \boldsymbol{w}_c, \varepsilon)$ and $\boldsymbol{c}_1(\boldsymbol{z}_{\mathsf{ood}}) = \boldsymbol{c}_2(\boldsymbol{z}_{\mathsf{ood}})$.

   By Lemma 3.5, the probability of this event is at most $\frac{|\Lambda(\mathcal{C}',\varepsilon)|^2}{2} \cdot \frac{\log n}{|\mathbb{F}|}$. Thus the predicate value can flip from 1 to 0 in this round with probability at most $\frac{|\Lambda(\mathcal{C}',\varepsilon)|^2}{2} \cdot \frac{\log n}{|\mathbb{F}|}$.

2. φ_2: The transcript now includes τ sent by $\mathcal{P}$, and the random challenge $\boldsymbol{u}_R$ sent by $\mathcal{V}$ for the repetition check. $\varphi_2(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}_c, \boldsymbol{t}, \tau, \boldsymbol{u}_R)) = 0$ if and only if there exists some $\boldsymbol{m}' \in \mathbb{F}^n$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}') \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon)$ and $\hat{m}'(\boldsymbol{x}, \boldsymbol{y}) \neq \hat{m}'(\boldsymbol{x}', \boldsymbol{y})$ for some $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n - \log k}$ and $\boldsymbol{y} \in \{0,1\}^{\log k}$, but $\hat{m}'(\boldsymbol{x}, \boldsymbol{u}_R) = \hat{m}'(\boldsymbol{x}', \boldsymbol{u}_R)$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n - \log k}$.

   Consider an arbitrary $\boldsymbol{v} \in \mathbb{F}^{\log n}$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{v}) \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon)$ and there exists $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n - \log k}$ and $\boldsymbol{y} \in \{0,1\}^{\log k}$ such that $\hat{v}(\boldsymbol{x}, \boldsymbol{y}) \neq \hat{v}(\boldsymbol{x}', \boldsymbol{y})$. By Lemma 3.2, the probability that $\hat{v}(\boldsymbol{x}, \boldsymbol{u}_R) = \hat{v}(\boldsymbol{x}', \boldsymbol{u}_R)$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \{0,1\}^{\log n - \log k}$ is at most $(\log k)/|\mathbb{F}|$. Applying the union bound over all such $\boldsymbol{v}$, the probability of the predicate flipping from 1 to 0 is at most $|\Lambda(\mathcal{C}', \varepsilon)| \cdot (\log k)/|\mathbb{F}|$.

3. φ_{i+2} for $i \in [\log k]$: Firstly, given a possible witness $\boldsymbol{m}' \in \mathbb{F}^n$, consider its first k entries $\boldsymbol{m}_{\mathsf{first}} \in \mathbb{F}^k$ and define $g_{\boldsymbol{m}'} : \mathbb{F}^{\log k} \to \mathbb{F}$ as the polynomial on the left-hand side of Eq. (19). That is,

$$g_{\boldsymbol{m}'}(\boldsymbol{y}) := \hat{G}_{\mathcal{B}}(\boldsymbol{t}_L, \boldsymbol{y}_L) \cdot \hat{G}_{\mathcal{B}}(\boldsymbol{t}_R, \boldsymbol{y}_R) \cdot \hat{m}_{\mathsf{first}}(\boldsymbol{y}) \ .$$

   We define $\varphi_{i+2}(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}_c, \boldsymbol{t}, \tau, \boldsymbol{u}_R, \sigma_{\mathsf{rep}}, [h_{i'}(X), r_{i'}]_{i'\in[i]})) = 0$ if and only if there exists some $\boldsymbol{m}' \in \mathbb{F}^n$ such that $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}') \in \Lambda(\mathcal{C}', \boldsymbol{w}', \varepsilon)$, and for some $i' \in [i]$ we have

$$h_{i'}(r_{i'}) = \sum_{\boldsymbol{y}\in\{0,1\}^{\log k - i'}} g_{\boldsymbol{m}'}(r_1, \ldots, r_{i'}, \boldsymbol{y}) \ ,$$

   but $h_{i'}$ is not the correct round polynomial, i.e. $h_{i'}(X) \neq \sum_{\boldsymbol{y}\in\{0,1\}^{\log k - i'}} g_{\boldsymbol{m}'}(r_1, \ldots, r_{i'-1}, X, \boldsymbol{y})$.

   For a fixed $\boldsymbol{m}'$, the probability of this happening can be upper bounded by upper bounding the probability that $h_i(X) \neq \sum_{\boldsymbol{y}\in\{0,1\}^{\log k - i}} g(r_1, \ldots, r_{i-1}, X, \boldsymbol{y})$, despite $h_i(r_i) = \sum_{\boldsymbol{y}\in\{0,1\}^{\log k - i}} g(r_1, \ldots, r_{i-1}, r_i, \boldsymbol{y})$. By Lemma 3.1, this can happen with probability at most $3/|\mathbb{F}|$. We can apply the union bound over all such $\boldsymbol{m}'$, to upper bound the probability of the predicate flipping from 1 to 0 in this round by $3 \cdot |\Lambda(\mathcal{C}', \varepsilon)|/|\mathbb{F}|$.

4. $\varphi_{\log k+3}$: For the last round, we define

$$\varphi_{\log k+3}(\mathsf{inst}, \mathsf{Tr} = (\boldsymbol{w}_c, \boldsymbol{t}, \tau, \boldsymbol{u}_R, \sigma_{\mathsf{rep}}, [h_{i'}(X), r_{i'}]_{i'\in[i]}, \boldsymbol{u}_L, \beta, \rho)) = 0$$

   if and only if there exists some witness $(\boldsymbol{m}'_{\mathsf{bad}}, \boldsymbol{c}_{\mathsf{bad}}, \boldsymbol{m}_{\mathcal{B}})$ that is 'bad', i.e. $\boldsymbol{m}'_{\mathsf{bad}}$ is not a valid witness for $\mathsf{Spot}^{(\mathcal{C},\mathcal{C}',\varepsilon,\ell)}$ but $\mathcal{V}$ does not reject. We describe the exact cases in which the witness can be bad below, but first note that our conditions do not involve $\boldsymbol{m}_{\mathcal{B}}$ since it corresponds to an 'honestly generated' word, due to $\boldsymbol{w}_{\mathcal{B}}$ being output by the indexer. We define the witness to be bad if:

(a) ρ is bad, that is:

- $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda\left(\mathcal{C}', \boldsymbol{w}', \varepsilon\right)$, and $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{c}_{\mathsf{bad}}) \in \Lambda\left(\mathcal{C}', \boldsymbol{w}_c, \varepsilon\right)$,
- $\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} f(\boldsymbol{x}, \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}), \hat{c}_{\mathsf{bad}}(\boldsymbol{x}), \hat{m}_{\mathcal{B}}(\boldsymbol{x})) = \sigma$,

yet at least one of the following 6 equalities is not true:

$$\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \mathrm{eq}(\boldsymbol{x}, (\boldsymbol{u}_L \parallel \boldsymbol{u}_R)) \cdot \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}) = \sigma_{\mathsf{rep}} \ , \quad \sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{s}(\boldsymbol{x}) \cdot \hat{c}_{\mathsf{bad}}(\boldsymbol{x}) = \left(\sum_{i=1}^{\ell} \beta^{i-1} \cdot y_i\right) \ ,$$
$$\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \mathrm{eq}(\boldsymbol{x}, \boldsymbol{t}) \cdot \hat{c}_{\mathsf{bad}}(\boldsymbol{x}) = \tau \ , \quad \sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{q}_4(\boldsymbol{x}) \cdot \hat{m}_{\mathcal{B}}(\boldsymbol{x}) = \tau_L \ ,$$
$$\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{q}_5(\boldsymbol{x}) \cdot \hat{m}_{\mathcal{B}}(\boldsymbol{x}) = \tau_R \ , \quad \sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{q}_6(\boldsymbol{x}) \cdot \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}) = \tau_m \ .$$

By Lemma 3.1, the probability of this event is at most $6/|\mathbb{F}|$ for a fixed $\boldsymbol{m}'_{\mathsf{bad}}, \boldsymbol{c}_{\mathsf{bad}}$. Applying the union bound over all such $\boldsymbol{m}'_{\mathsf{bad}}, \boldsymbol{c}_{\mathsf{bad}}$, the probability of this event is at most $6 \cdot |\Lambda\left(\mathcal{C}', \varepsilon\right)|^2/|\mathbb{F}|$.

(b) β is bad: We now assume that ρ is not bad. We discuss the probability that there exists $\boldsymbol{c}_{\mathsf{bad}} \in \mathbb{F}^{k'}$ such that

- $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{c}_{\mathsf{bad}}) \in \Lambda\left(\mathcal{C}', \boldsymbol{w}_c, \varepsilon\right)$,
- $\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{s}(\boldsymbol{x}) \cdot \hat{c}_{\mathsf{bad}}(\boldsymbol{x}) = \sum_{i=1}^{\ell} \beta^{i-1} \cdot y_i$ (which we can assume since ρ is not bad), and
- there exists $i \in [\ell]$ such that $\boldsymbol{c}_{\mathsf{bad}}[\xi_i] \neq y_i$.

Note that by definition of $\hat{s}$ (see Eq. (13)), the second item above implies that $\sum_{i=1}^{\ell} \beta^{i-1} \cdot \boldsymbol{c}_{\mathsf{bad}}[\xi_i] = \sum_{i=1}^{\ell} \beta^{i-1} \cdot y_i$. By Lemma 3.1, the probability of this event is at most $\ell/|\mathbb{F}|$ for a fixed $\boldsymbol{c}_{\mathsf{bad}}$. Applying the union bound over all such $\boldsymbol{c}_{\mathsf{bad}}$, the probability of this event is at most $\ell \cdot |\Lambda\left(\mathcal{C}', \varepsilon\right)|/|\mathbb{F}|$.

(c) $\boldsymbol{u}_L$ is bad: We now assume that ρ, β are not bad. We discuss the probability that there exists $\boldsymbol{m}'_{\mathsf{bad}} \in \mathbb{F}^n$ such that

- $\mathsf{Enc}_{\mathcal{C}'}(\boldsymbol{m}'_{\mathsf{bad}}) \in \Lambda\left(\mathcal{C}', \boldsymbol{w}', \varepsilon\right)$,
- $\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \mathrm{eq}(\boldsymbol{x}, (\boldsymbol{u}_L \parallel \boldsymbol{u}_R)) \cdot \hat{m}'_{\mathsf{bad}}(\boldsymbol{x}) = \sigma_{\mathsf{rep}}$ (which we can assume since ρ is not bad), and
- there exists $\boldsymbol{x} \in \{0,1\}^{\log n - \log k}$ such that $\hat{m}'_{\mathsf{bad}}(\boldsymbol{x}, \boldsymbol{u}_R) \neq \sigma_{\mathsf{rep}}$ (i.e. $\hat{m}'_{\mathsf{bad}}(\boldsymbol{X}, \boldsymbol{u}_R) - \sigma_{\mathsf{rep}}$ is a non-zero polynomial).

By definition of eq, the second item above implies that $\hat{m}'_{\mathsf{bad}}(\boldsymbol{u}_L, \boldsymbol{u}_R) - \sigma_{\mathsf{rep}} = 0$. If $\hat{m}'_{\mathsf{bad}}(\boldsymbol{X}, \boldsymbol{u}_R) - \sigma_{\mathsf{rep}}$ is a non-zero polynomial, then by Lemma 3.1, the probability of this event is at most $(\log n - \log k)/|\mathbb{F}|$ for a fixed $\boldsymbol{m}'_{\mathsf{bad}}$. Applying the union bound over all such $\boldsymbol{m}'_{\mathsf{bad}}$, the probability of this event is at most $|\Lambda\left(\mathcal{C}', \varepsilon\right)| \cdot (\log n - \log k)/|\mathbb{F}|$.

We can apply the union bound over all of these possibilities to show that the probability of the predicate flipping from 1 to 0 in this round is negligible.

Challenge predicate. The challenge predicate always outputs 1.

Tree extractor. Notice that the arity parameters are $1^{\log k+3}$, and thus the extractor is straight-line. Given a tree with the 'output' witness $\mathbb{w}' = (\boldsymbol{m}', \boldsymbol{c}, \boldsymbol{m}_{\mathcal{B}})$, the extractor just outputs $\boldsymbol{m}'$.

We now show that if the vertical predicate is satisfied, $\boldsymbol{m}'$ is a valid witness for inst. Notice that if Eq. (17), Eq. (18), and Eq. (19) were all true with respect to $\mathbb{w}'$, then we have that:

1. Given that β is not bad, Eq. (17) implies that $\boldsymbol{c}[\xi_i] = y_i$ for all $i \in [\ell]$.

2. Given that $\boldsymbol{t}$ is not bad, Eq. (18), and Eq. (19) together imply that $\boldsymbol{c} = \mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m})$.

3. Given that ρ is not bad, we have $\hat{m}'(\boldsymbol{u}) = \sigma_{\mathsf{rep}}$. Given that $\boldsymbol{u}_L$ is not bad and $\boldsymbol{u}_R$ is not bad, we have $\boldsymbol{m}' = \boldsymbol{m}^{n/k}$.

Thus if Eq. (17), Eq. (18), and Eq. (19) all held, then the extractor succeeds in outputting a valid witness. The fact that Eq. (17) and Eq. (18) hold follows immediately from the fact that ρ is not bad. Thus, all that remains is to prove that Eq. (19) holds. Since ρ is not bad, we have

$$\sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{q}_4(\boldsymbol{x}) \cdot \hat{m}_{\mathcal{B}}(\boldsymbol{x}) = \tau_L \ , \quad \sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{q}_5(\boldsymbol{x}) \cdot \hat{m}_{\mathcal{B}}(\boldsymbol{x}) = \tau_R \ , \quad \sum_{\boldsymbol{x}\in\{0,1\}^{\log n}} \hat{q}_6(\boldsymbol{x}) \cdot \hat{m}'(\boldsymbol{x}) = \tau_m \ .$$

Thus, by definition of $\hat{q}_4, \hat{q}_5, \hat{q}_6$, we have $\hat{G}_{\mathcal{B}}(\boldsymbol{r}_L, \boldsymbol{t}_L) = \tau_L$, $\hat{G}_{\mathcal{B}}(\boldsymbol{r}_R, \boldsymbol{t}_R) = \tau_R$, and $\hat{m}_{\mathsf{first}}(\boldsymbol{r}) = \tau_m$, where $\hat{m}_{\mathsf{first}}$ is the multilinear extension of $\boldsymbol{m}_{\mathsf{first}} \in \mathbb{F}^k$, the first k entries of $\boldsymbol{m}'$. Since $\boldsymbol{m}' = \boldsymbol{m}^{n/k}$, we have $\boldsymbol{m}_{\mathsf{first}} = \boldsymbol{m}$, and thus $\hat{m}(\boldsymbol{r}) = \tau_m$.

We can now prove Eq. (19) via the standard soundness proof of sumcheck. Firstly, we know that $h_{\log k}(r_{\log k}) = \tau_L \cdot \tau_R \cdot \tau_m = g(r_1, \ldots, r_{\log k})$ (otherwise $\mathcal{V}$ would have rejected in Item 20). Since all the 'sumcheck predicates' are satisfied, this implies that $h_{\log k}(X) = g(r_1, \ldots, r_{\log k - 1}, X)$. Thus, $h_{\log k}(0) + h_{\log k}(1) = \sum_{b\in\{0,1\}} g(r_1, \ldots, r_{\log k - 1}, b)$. Since $\mathcal{V}$ did not reject in Item 16, we have $h_{\log k - 1}(r_{\log k - 1}) = h_{\log k}(0) + h_{\log k}(1)$. Combining these two equalities, we have $h_{\log k - 1}(r_{\log k - 1}) = \sum_{b\in\{0,1\}} g(r_1, \ldots, r_{\log k - 1}, b)$. By repeating this argument inductively, we have for all $i \in [\log k - 1]$ that

$$h_i(X) = \sum_{\boldsymbol{y}\in\{0,1\}^{\log k - i}} g(r_1, \ldots, r_{i-1}, X, \boldsymbol{y}) \ .$$

In particular, for $i = 1$, we have $h_1(X) = \sum_{\boldsymbol{y}\in\{0,1\}^{\log k - 1}} g(X, \boldsymbol{y})$, which implies that $h_1(0) + h_1(1) = \sum_{\boldsymbol{y}\in\{0,1\}^{\log k}} g(\boldsymbol{y})$. Finally, since $\mathcal{V}$ did not reject in Item 16, we have $h_1(0) + h_1(1) = \tau$, which implies that $\sum_{\boldsymbol{y}\in\{0,1\}^{\log k}} g(\boldsymbol{y}) = \tau$ as desired. □

B.4 Proof of Lemma 7.5

Lemma 7.5. *Given an RBRTE IOR* $\mathsf{IOR} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ *from* $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ *to* $\mathcal{R}'$ *with the following efficiency measures*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
k	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$

and $\ell \in \mathbb{N}$*, there exists a corresponding RBRTE IOR* $\mathsf{IOR}' = (\mathbf{I}', \mathbf{P}', \mathbf{V}')$ *from* $\mathsf{HEP}^{(\mathcal{C},\varepsilon,\ell)}$ *to* $\mathcal{R}'$ *with the following efficiency measures*

Rounds	PLen	Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$\mathsf{T}_{\mathcal{P}}$	$\mathsf{T}_{\mathcal{V}}$
$\mathsf{k}+1$	PLen	$\ell \cdot$ Queries	OLen	$\mathsf{T}_{\mathcal{I}}$	$O(\ell \cdot \mathsf{T}_{\mathcal{P}})$	$O(\ell \cdot \mathsf{T}_{\mathcal{V}})$

where $\mathcal{C}$ *is a linear code with parameters* $(\mathbb{F}, n, k, \delta_{\mathcal{C}})$.

Proof. Given $\mathsf{IOR} = (\mathcal{P}, \mathcal{V})$, we construct $\mathsf{IOR}' = (\mathcal{P}', \mathcal{V}')$ as follows. The indexer of IOR' is identical to that of IOR. Let the input to IOR' be

$$\mathbb{x}' = (\boldsymbol{z}, [y_i]_{i\in[\ell]}), \quad \vec{\mathbb{y}}' = [\boldsymbol{w}_i]_{i\in[\ell]}, \quad \mathbb{w}' = [\boldsymbol{m}_i]_{i\in[\ell]} \ .$$

We assume ℓ is a power of 2, because if it is not, we can pad with 'dummy oracles' which are of the form $\boldsymbol{w}_i = 0^n$, $\boldsymbol{m}_i = 0^k$ and $y_i = 0$.

In the first round of IOR', $\mathcal{V}'$ samples $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{\log \ell}$ and sends it to $\mathcal{P}'$. Define $y' \coloneqq \sum_{i\in[\ell]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}) \cdot y_i$, $\boldsymbol{w}' \coloneqq \sum_{i\in[\ell]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}) \cdot \boldsymbol{w}_i$ and $\boldsymbol{m}' \coloneqq \sum_{i\in[\ell]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}) \cdot \boldsymbol{m}_i$. For the next k rounds, $\mathcal{P}'$ and $\mathcal{V}'$ simulate $\mathcal{P}$ and $\mathcal{V}$ respectively on the following inputs:

$$\mathbb{x} = (\boldsymbol{z}, y'), \quad \vec{\mathbb{y}} = \boldsymbol{w}', \quad \mathbb{w} = \boldsymbol{m}' \ ,$$

where every time $\mathcal{V}$ wants to query $\boldsymbol{w}'$ at index j, $\mathcal{V}'$ queries $\boldsymbol{w}_i$ at index j to obtain σ_i, for all $i \in [\ell]$, and gives $\mathcal{V}$ the value $\sum_{i\in[\ell]} \mathrm{eq}(\langle i-1\rangle, \boldsymbol{r}) \cdot \sigma_i$.

Soundness analysis. Completeness and the efficiency claims follow by construction of IOR'. We now prove that the IOR is RBRTE. Given an index $\mathbb{i}$, let IOR be $(\vec{a}, \varphi, \theta)$-RBRTE with respect to $\mathbb{i}$. We will now define $\vec{a}', \varphi', \theta'$ such that IOR' is $(\vec{a}', \varphi', \theta')$-RBRTE with respect to $\mathbb{i}$. The arity parameters $\vec{a}' \coloneqq (1, \vec{a})$.

Vertical predicate. Since the vertical predicate only changes on transcripts that end with the verifier's challenge, we only consider such transcripts. We describe φ as a tuple $[\varphi_i]_{i\in[\mathsf{k}+1]}$ such that for the i-th round, $\varphi(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_i, \rho_i) = 1$ if and only if $\forall j \in [i] : \varphi_j(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \alpha_1, \rho_1, \ldots, \alpha_j, \rho_j) = 1$.

Let $\epsilon = 1/2n$ and $\beta_\epsilon \geq (1-\delta_{\mathcal{C}}+\epsilon)^{1/3}+\epsilon$ be the largest value such that $\beta_\epsilon \leq 1-\varepsilon$, and $\{\beta_\epsilon n\} > 1/2$. Since $\varepsilon \in \varepsilon(\mathcal{C})$, we know that this β_ϵ satisfies $\lceil (1-\varepsilon)n \rceil - 1 < \beta_\epsilon n \leq \lceil (1-\varepsilon)n \rceil$. We define $\varphi'_1(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \boldsymbol{r}) = 0$ if and only if at least one of the following conditions holds:

- $\boldsymbol{r}$ is a bad challenge for $[\boldsymbol{w}_i]_{i\in[\ell]}$ with respect to Lemma 3.13 using the agreement parameter β_ϵ.
- There exist messages $\boldsymbol{m}_{\mathsf{bad},1}, \ldots, \boldsymbol{m}_{\mathsf{bad},\ell} \in \mathbb{F}^k$ such that $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_{\mathsf{bad},j}) \sim_\varepsilon \boldsymbol{w}_j$ for all $j \in [\ell]$ and $\sum_{j\in[\ell]} \mathrm{eq}(\langle j-1\rangle, \boldsymbol{r}) \cdot \hat{m}_{\mathsf{bad},j}(\boldsymbol{z}) = y'$, and yet $\sum_{j\in[\ell]} \mathrm{eq}(\langle j-1\rangle, \boldsymbol{X}) \cdot (\hat{m}_{\mathsf{bad},j}(\boldsymbol{z}) - y_i)$ is not the zero polynomial.

The first condition above occurs with probability at most $3^\ell/\epsilon^2|\mathbb{F}|$ by Lemma 3.13. The second condition above occurs with probability at most $\log \ell/|\mathbb{F}|$ by Lemma 3.1 for a fixed set of messages $\boldsymbol{m}_{\mathsf{bad},1}, \ldots, \boldsymbol{m}_{\mathsf{bad},\ell}$. Applying the union bound over all possible sets of messages $\boldsymbol{m}_{\mathsf{bad},1}, \ldots, \boldsymbol{m}_{\mathsf{bad},\ell}$, we have that the second condition occurs with probability at most $|\Lambda(\mathcal{C}, \varepsilon)|^\ell \cdot \log \ell/|\mathbb{F}|$. Applying the union bound over both conditions, we have that $\varphi'_1(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \boldsymbol{r}) = 0$ with negligible probability.

On input the full transcript $\mathsf{Tr}' = (\boldsymbol{r}, \mathsf{Tr})$, $\varphi'(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathsf{Tr}') = 1$ if and only if $\varphi'_1(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \boldsymbol{r}) = 1$ and $\varphi(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \mathsf{Tr}) = 1$. Since φ'_1 and φ are valid vertical predicates, φ' is also a valid vertical predicate.

Challenge predicate. The challenge predicate outputs 1 iff θ, the challenge predicate of IOR, outputs 1 on the corresponding sub-tree.

Tree extractor. The tree extractor $\mathcal{E}'$ for IOR' on input a tree $\mathsf{T}' = (\boldsymbol{r}, \mathsf{T})$ behaves as follows:

1. Use the tree extractor of IOR, $\mathcal{E}^{\mathrm{Tree}}$, on T to extract a valid witness $\boldsymbol{m}'$ for the $\mathsf{MEP}^{(\mathcal{C},\varepsilon,1)}$ claim, such that $\hat{m}'(\boldsymbol{z}) = y'$ and $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}') \sim_\varepsilon \boldsymbol{w}'$. Thus, there exists a set S such that $|S| = \lceil \beta_\epsilon n \rceil = (1-\varepsilon)n$ and $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}')[S] = \boldsymbol{w}'[S]$.
2. Compute S using $\boldsymbol{w}'$ and $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}')$.
3. Use erasure correction (see Lemma 3.9) to obtain $[\boldsymbol{c}_i]_{i\in[\ell]}$ from $[\boldsymbol{w}_i]_{i\in[\ell]}$ and S, where each $\boldsymbol{c}_i$ is the unique codeword that agrees with $\boldsymbol{w}_i$ on S (this codeword is unique because $|S| > (1-\delta_{\mathcal{C}})n$). Furthermore, these codewords exist since $\boldsymbol{r}$ is a good challenge with respect to Lemma 3.13 as $\varphi'_1(\mathbb{i}, \mathbb{x}, \vec{\mathbb{y}}, \boldsymbol{r}) = 1$.

4. Output the underlying messages $[\boldsymbol{m}_i]_{i\in[\ell]}$ such that for all $i \in [\ell]$, $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) = \boldsymbol{c}_i$.
Clearly, by construction of $\mathcal{E}'$, we have $\mathsf{Enc}_{\mathcal{C}}(\boldsymbol{m}_i) \sim_{\varepsilon} \boldsymbol{w}_i$ for all $i \in [\ell]$.

Thus, all that is left to show is that the output messages $[\boldsymbol{m}_i]_{i\in[\ell]}$ satisfy the evaluation claims. Since $\hat{m}' = \sum_{j\in[\ell]} \mathrm{eq}(\langle j-1\rangle, \boldsymbol{r}) \cdot \hat{m}_i$, we have $\sum_{j\in[\ell]} \mathrm{eq}(\langle j-1\rangle, \boldsymbol{r}) \cdot \hat{m}_i(\boldsymbol{z}) = y'$. Combining this with the fact that φ_1' outputs 1, we have that $\sum_{j\in[\ell]} \mathrm{eq}(\langle j-1\rangle, X) \cdot (\hat{m}_j(\boldsymbol{z}) - y_i) = 0$, which means that $\hat{m}_j(\boldsymbol{z}) = y_j$ for all $j \in [\ell]$. □