Polylogarithmic Proofs for Multilinears over Binary Towers

Benjamin E. Diamond
Irreducible
bdiamond@irreducible.com

Jim Posen
Irreducible
jposen@irreducible.com

Abstract

We present a succinct polynomial commitment scheme for multilinears over tiny binary fields, including that with just 2 elements. To achieve this, we develop two main ideas. Our first adapts Zeilberger, Chen and Fisch's *BaseFold* ('23) PCS to the binary setting; it uses FRI (ICALP '18)'s lesser-known binary variant, and reveals a new connection between that work and Lin, Chung and Han (FOCS '14)'s *additive NTT*. We moreover present a novel large-field-to-small-field compiler for polynomial commitment schemes. Using a technique we call "ring-switching", our compiler generically bootstraps any multilinear PCS over a large, power-of-two degree extension field into a further PCS over that field's ground field. The resulting small-field PCS lacks "embedding overhead", in that its commitment cost is identical to that of the large-field scheme on each input size (measured in *bits*). We attain concretely small proofs for enormous binary multilinears, shrinking the proofs of Diamond and Posen ('23) by an order of magnitude.

1 Introduction

In recent work, Diamond and Posen [DP23] introduce a sublinear argument designed to capture certain efficiencies available in towers of binary fields. Using a "block-level encoding" technique, that work evades, at least during its commitment phase, the *embedding overhead* prone to arise whenever tiny fields are used, especially in those protocols that critically use Reed–Solomon codes. That work's key polynomial commitment scheme features opening proofs whose size and verifier complexity both grow on the order of the square root of the size (i.e., measured in total data bits) of the committed polynomial.

In this work, we present a multilinear polynomial commitment scheme—again designed for small binary fields—whose proof size and verifier complexity grow only polylogarithmically in the size of the committed polynomial. Our starting point is Zeilberger, Chen and Fisch's *BaseFold PCS* [ZCF23, § 5], a multilinear PCS for large prime fields. BaseFold's polynomial commitment scheme [ZCF23, § 5] identifies a new connection between Ben-Sasson, Bentov, Horesh and Riabzev's [BBHR18] celebrated *FRI* IOP of proximity and *multilinear* polynomials. Specifically, that work observes that, in the setting of prime-field FRI, when the FRI folding arity is fixed at 2, the constant *value* of the prover's final FRI oracle relates to the *univariate* coefficients of its FRI message just as a multilinear's *evaluation* relates to its *multilinear* coefficients in the monomial basis. This fact underlies BaseFold's use of FRI within its multilinear polynomial commitment scheme. On the other hand, since the scheme's evaluation point is typically known in advance to the prover, whereas FRI's folding challenges of course must not be, BaseFold moreover interleaves into its FRI folding phase an execution of the *sumcheck* protocol, thereby reducing the evaluation of the multilinear at the *known* query point to its evaluation at the *random* point sampled during FRI. We describe BaseFold further in Subsection 1.2 below.

We note that FRI has figured in both univariate and multilinear commitment schemes previously. All prior such uses of FRI, however—with the exception of [ZCF23, § 5]—invoke "quotienting", and so suffer from embedding overhead, a phenomenon described at length in [DP23]. We refer to Haböck [Hab22] for a description of FRI's use as a *univariate* commitment scheme. In the multilinear setting, we note briefly an approach proposed by Chen, Bünz, Boneh and Zhang [CBBZ23, § B], which itself makes blackbox use of a univariate commitment scheme such as FRI. Interestingly, that scheme—assuming the FRI-based univariate scheme—resembles [ZCF23, § 5], at least during its commitment phase. Its evaluation phase, however, generically invokes the underlying univariate scheme's evaluation protocol logarithmically many times.

1.1 Our Contributions

To make something like BaseFold PCS [ZCF23, § 5] work in our setting, we must overcome two main obstacles. Firstly, while BaseFold accepts any large field and any "foldable" code, that work emphasizes the odd-characteristic case, and especially that of Reed–Solomon codes over large prime fields. It does not examine the case of binary fields. Though FRI itself works over binary fields—and was in fact originally presented in that setting [BBHR18, § 3]—the even–odd coefficient-folding behavior essential to [ZCF23, § 5] fails to persist in the binary arena, at least for FRI-folding maps chosen generically. In Section 3 below, we discuss how to choose those maps in such a way as to cause the desired folding behavior to reemerge. Our treatment makes necessary a careful analysis of Lin, Chung and Han's *additive NTT* [LCH14], and demonstrates a hitherto-unremarked connection between those two works.

Our second main obstacle pertains to the fact that BaseFold is designed for *large-field* multilinears. Indeed, as the work [DP23] explains in detail, it is difficult—given only a large-field scheme—to define a small-field scheme that inherits the "right" security and efficiency guarantees. Specifically, the small-field scheme should commit polynomials as efficiently as the large-field scheme commits *equally-sized* polynomials (i.e., in bits). (In particular, embedding the small-field polynomial coefficient-wise into the large field wouldn't suffice.) In Section 4 below, we describe a generic and extremely efficient reduction from large-field schemes to small-field schemes. Given an extension $L \,/\, K$ of degree 2^κ, our technique reduces the problem of committing an ℓ-variate multilinear over K to that of committing an $\ell := \ell - \kappa$-variate multilinear over L (which contains the "same amount of data"). Our scheme's commitment phase leverages a *packing* technique present already in [DP23]. Its evaluation phase introduces a new idea that we call "ring-switching", loosely inspired by Ron-Zewi and Rothblum [RR24, Fig. 2]'s "code-switching" technique. We roughly describe our approach as follows, deferring our full treatment to Section 4. Informally, we observe that the evaluation of the given K-polynomial at a desired point can be reduced to that of its packed L-polynomial at a random point, up to the execution of an ℓ-round sumcheck in the *tensor algebra* $L \otimes_K L$, a ring which contains L in "two different ways". (That object can be viewed as a generalization of the *tower algebra* data structure of [DP23, § 3.4].)

In Section 5 below, we combine the ideas of Sections 3 and 4 into a single, streamlined construction. That construction yields an extremely efficient small-field PCS for binary multilinears. In Subsection 5.2 below, we benchmark our combined scheme in detail, showing that its proofs are smaller than those of [DP23, Cons. 3.11] by between fourfold and a hundredfold.

1.2 Technical Overview

We sketch in advance various aspects of our technical approach.

Remarks on FRI-folding. Each honest FRI prover begins with the evaluation of some polynomial $P(X) \coloneqq \sum_{j=0}^{2^\ell - 1} a_j \cdot X^j$ over the initial domain $S^{(0)}$. Under certain mild conditions—specifically, if the folding factor η divides ℓ, and the recursion is carried out to its end—the prover's final oracle will be *identically constant* over its domain (and in fact, the prover will rather send the verifier this latter constant in the clear). What will the *value* of this constant be, as a function of $P(X)$ and of the verifier's folding challenges?

In the setting of *prime field* multiplicative FRI, the folding maps $q^{(i)}$ all take the especially simple form $X \mapsto X^{2^\eta}$. BaseFold [ZCF23, § 5] makes the interesting observation whereby—again, in the prime field setting, for η now moreover set to 1, and for $q^{(0)}, \ldots, q^{(\ell-1)}$ defined in just this way—the prover's final FRI response will be nothing other than $a_0 + a_1 \cdot r_0 + a_2 \cdot r_1 + \cdots + a_{2^\ell - 1} \cdot r_0 \cdot \cdots r_{\ell-1}$, where $r_0, \ldots, r_{\ell-1}$ are the verifier's FRI folding challenges. That is, it will be exactly the evaluation of the multilinear polynomial $a_0 + a_1 \cdot X_0 + a_2 \cdot X_1 + \cdots + a_{2^\ell - 1} \cdot X_0 \cdot \cdots X_{\ell-1}$ at the point $(r_0, \ldots, r_{\ell-1})$.

What about in the binary field setting? In this setting, the simple folding maps $X \mapsto X^{2^\eta}$ no longer suffice, as [BBHR18, § 2.1] already remarks; rather, we must choose for the maps $q^{(i)}$ a certain sequence of *linear subspace polynomials* of degree 2^η. FRI does *not* suggest precise values for these polynomials, beyond merely demanding that they feature the right linear-algebraic syntax. Roughly, each $q^{(i)}$'s kernel must reside entirely inside the domain $S^{(i)}$; we discuss this requirement further in Subsection 2.4 below. Given syntactically valid subspace polynomials $q^{(i)}$ chosen otherwise arbitrarily—and, we emphasize, FRI does *not* suggest a choice—the constant value of the prover's final oracle will relate in a complicated way to the coefficient vector $(a_0, \ldots, a_{2^\ell - 1})$ and to the verifier's folding challenges r_i.

The additive NTT and FRI. We recall briefly the "additive NTT" of Lin, Chung, and Han [LCH14] (we refer to Subsection 2.3 below for a thorough description). We fix a binary field L of degree more than ℓ. The work [LCH14] defines a "novel polynomial basis" $(X_j(X))_{j=0}^{2^\ell-1}$ of the L-vector space consisting of polynomials over L of degree less than 2^ℓ (which, of course, is *not* the standard monomial L-basis $\left(X^j\right)_{j=0}^{2^\ell-1}$ of that space). The work [LCH14] then gives an algorithm which, on input a polynomial $P(X) \coloneqq \sum_{j=0}^{2^\ell-1} a_j \cdot X_j(X)$ expressed *with respect to the novel basis*, computes $P(X)$'s "additive NTT". That is, the algorithm computes from $P(X)$'s coefficient vector $(a_0, \ldots, a_{2^\ell-1})$ the set of $P(X)$'s evaluations over some appropriately chosen affine $\mathbb{F}_2$-vector subspace $S \subset L$, and in quasilinear time in the size of S, no less.

We recover as follows the "classical" FRI folding pattern identified above in the binary-field setting. For expository purposes, we fix $\eta = 1$ (though cf. Subsection 3.2 below). We stipulate first of all that the prover use the coefficients $(a_0, \ldots, a_{2^\ell-1})$ of its input multilinear as the coefficients in Lin, Chung and Han [LCH14]'s *novel polynomial basis* of its initial univariate FRI polynomial $P(X) \coloneqq \sum_{j=0}^{2^\ell-1} a_j \cdot X_j(X)$. (This choice of basis has the crucial additional effect of making the prover's evaluation of $P(X)$ over $S^{(0)}$ computable in quasilinear time.) Essentially, our insight is that, if we choose the FRI subspace maps $q^{(0)}, \ldots, q^{(\ell-1)}$ appropriately, then the prover's final FRI oracle becomes meaningfully related to $P(X)$'s initial coefficient vector $(a_0, \ldots, a_{2^\ell-1})$; that is, it becomes once again $a_0 + a_1 \cdot r_0 + a_2 \cdot r_1 + \cdots + a_{2^\ell-1} \cdot r_0 \cdots r_{\ell-1}$. Specifically, our construction—which we explain in detail in Subsection 3.1 below—opts to define the maps $q^{(0)}, \ldots, q^{(\ell-1)}$ precisely so that they *factor* Lin, Chung and Han [LCH14, § II. C.]'s "normalized subspace vanishing polynomials" $\left(\widehat{W}_i(X)\right)_{i=0}^{\ell}$, in the sense that $\widehat{W}_i(X) = q^{(i-1)} \circ \cdots \circ q^{(0)}$ holds for each $i \in \{0, \ldots, \ell\}$ (see Corollary 3.4).

Upon defining the maps $q^{(0)}, \ldots, q^{(\ell-1)}$ in this way, we recover a familiar, Fourier-theoretic characterization of the novel basis polynomials $(X_j(X))_{j=0}^{2^\ell-1}$, as well as a reinterpretation of Lin, Chung and Han's algorithm [LCH14, § III.] along more classical lines (see Remark 3.15). We reproduce here this Fourier-theoretic identity, which appears as (2) below:

$$P^{(i)}(X) = P_0^{(i+1)}(q^{(i)}(X)) + X \cdot P_1^{(i+1)}(q^{(i)}(X)).$$

Here, the index $i \in \{0, \ldots, \ell-1\}$ is arbitrary. We write $P^{(i)}(X) = \sum_{j=0}^{2^{\ell-i}-1} a_j \cdot X_j^{(i)}(X)$ for an arbitrary polynomial expressed with respect to the basis $\left(X_j^{(i)}(X)\right)_{j=0}^{2^{\ell-i}-1}$, a so-called i^{th}*-order* analogue of the novel polynomial basis (we refer to Theorem 3.12 below for a thorough definition of our "higher-order" polynomial bases, as well as to Remark 3.14 for further discussion). The polynomials $P_0^{(i+1)}(X)$ and $P_1^{(i+1)}(X)$ are $P^{(i)}(X)$'s even and odd refinements (these are expressed with respect to the $i+1^{\text{st}}$-order basis $\left(X_j^{(i+1)}(X)\right)_{i=0}^{2^{\ell-i-1}-1}$).

Our particular choice of the maps $q^{(0)}, \ldots, q^{(\ell-1)}$ serves to recover the coefficient-folding behavior of prime-field FRI (i.e., which was exploited by [ZCF23, § 5]). Indeed, by using the polynomial identity above, and by expressing each polynomial at hand with respect to the appropriate higher-order novel polynomial basis, we are able to establish the required FRI folding pattern (see in particular Lemma 3.13).

A new FRI folding mechanism. As it happens, we opt moreover to modify FRI itself, so as to induce a *Lagrange-style*, as opposed to a monomial-style, folding pattern in the coefficient domain. In our FRI variant, the value of the prover's final oracle becomes $a_0 \cdot (1 - r_0) \cdot \cdots \cdot (1 - r_{\ell-1}) + \cdots + a_{2^\ell-1} \cdot r_0 \cdots r_{\ell-1}$, the evaluation at $(r_0, \ldots, r_{\ell-1})$ of the polynomial whose coefficients in the *multilinear Lagrange basis* are $(a_0, \ldots, a_{2^\ell-1})$. We moreover introduce a *multilinear* style of many-to-one FRI folding, which contrasts with FRI's univariate approach [BBHR18, § 3.2]. We describe our FRI folding variant in Subsection 3.2 below (see in particular Definitions 3.6 and 3.8, as well as Remark 3.7). Interestingly, our FRI-folding variant necessitates a sort of proximity gap different from that invoked by the original FRI protocol. Indeed, while the soundness proof [Ben+23, § 8.2] of FRI uses the proximity gap result [Ben+23, Thm. 1.5] for *low-degree parameterized curves*, our security treatment below instead uses a *tensor-folding* proximity gap of the sort recently established by Diamond and Posen [DP24, Thm. 2] (see also Theorem 2.4 below).

Miscellanea. Throughout Subsection 3.1, we examine in detail various further aspects of binary-field FRI. For example—even in the abstract IOP model—we must necessarily fix $\mathbb{F}_2$-bases of the respective Reed–Solomon domains $S^{(i)}$, in order to interpret committed functions $f^{(i)} : S^{(i)} \to L$ as L-valued *strings* (that is, must implicitly lexicographically "flatten" each domain $S^{(i)}$ using some ordered $\mathbb{F}_2$-basis of it, known to both the prover and the verifier). The choice of these bases matters. Indeed, for $\mathbb{F}_2$-bases of $S^{(i)}$ and $S^{(i+1)}$ chosen arbitrarily, the fundamental operation which associates to each $y \in S^{(i+1)}$ its fiber ${q^{(i)}}^{-1}(\{y\}) \subset S^{(i)}$—which both the prover and the verifier must perform repeatedly—could come to assume complexity on the order of $\dim\big(S^{(i)}\big)^2$ bit-operations, even after a linear-algebraic preprocessing phase.

We moreover suggest a family of bases for the respective domains $S^{(i)}$ with respect to which the maps $q^{(i)}$ come to act simply by *projecting* away their first η coordinates. In particular, the application of each map $q^{(i)}$—in coordinates—becomes free; the preimage operation ${q^{(i)}}^{-1}(\{y\})$ comes to amount simply to that of prepending η arbitrary boolean coordinates to y's coordinate representation. While bases with these properties can of course be constructed in FRI even for maps $q^{(i)}$ chosen arbitrarily, our procedure yields a basis of the initial domain $S^{(0)}$ which coincides with that expected by the additive NTT of [LCH14]. In particular, our prover may use *as is* the output of the additive NTT as its 0^{th} FRI oracle, without first subjecting that output to the permutation induced by an appropriate change-of-basis transformation on $S^{(0)}$. We believe that these observations stand to aid all implementers of binary-field FRI.

Ring-switching. We sketch in slightly more detail our ring-switching technique (we again refer also to Section 4 below). We fix an input multilinear $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$ over K and a desired evaluation point $(r_0, \ldots, r_{\ell-1}) \in L^\ell$; we moreover set $\ell' := \ell - \kappa$, and write $t'(X_0, \ldots, X_{\ell'-1}) \in L[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$ for $t(X_0, \ldots, X_{\ell-1})$'s *packed polynomial* in the sense of Definition 2.1. (We recall from [DP23, § 3] that $t'(X_0, \ldots, X_{\ell'-1})$'s vector of Lagrange coefficients arises from $t(X_0, \ldots, X_{\ell-1})$'s by a process which interprets each successive chunk consisting of 2^κ of that latter vector's components as a single L-element.) To learn the desired datum $t(r_0, \ldots, r_{\ell-1})$, it certainly doesn't suffice *merely* to learn the single evaluation $t'(r'_0, \ldots, r'_{\ell'-1})$, either for $(r'_0, \ldots, r'_{\ell'-1}) := (r_\kappa, \ldots, r_{\ell-1})$ or else for $(r'_0, \ldots, r'_{\ell'-1})$ otherwise chosen. Indeed, to learn that latter quantity alone would entail a sort of information loss, as is noted in [DP23, § 3].

To mitigate this information loss, we work in a ring which *contains* L as a subring. Specifically, we work in the ring $A := L \otimes_K L$, the tensor product of L with itself over K (we view here L as a K-algebra). Concretely, this ring represents a sort of two-dimensional data structure, which contains two isomorphic copies of L (in its leftmost column and its topmost row, respectively); we refer also to Figure 1 below. We first examine in detail the structure of $A := L \otimes_K L$ (see also Subsection 2.5). Indeed, we note the "vertical" and "horizontal" embeddings $\varphi_0 : L \to A$ and $\varphi_1 : L \to A$ of L into A; we also argue that the images of these subrings act on the algebra respectively "column-wise" and "row-wise".

Using these observations, we argue further that the *desired* evaluation $t(r_0, \ldots, r_{\ell-1})$ can be obtained—up to a $\Theta(2^\kappa)$-time postprocessing step on the part of the verifier—*instead* from the A-element $\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$. Indeed, we explain this complicated expression as follows. The point is that we embed $t'(X_0, \ldots, X_{\ell'-1})$'s coefficients horizontally into the algebra, and embed the components $(r_\kappa, \ldots, r_{\ell-1})$ vertically. Exploiting the ring structure of A, we conclude that this latter sort of embedding captures exactly the partial evaluation of our original multilinear $t(X_0, \ldots, X_{\ell-1})$ at the trailing suffix $(r_\kappa, \ldots, r_{\ell-1}) \in L^{\ell'}$. (Our detailed correctness proof appears as Theorem 4.2 below.)

Though this maneuver doesn't alone solve our problem, it makes applicable the sumcheck protocol over the ring A. In the second step of our reduction, we take exactly this step, drawing our sumcheck challenges from the *horizontal* copy of L in A. In this way, we reduce the desired evaluation $\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ to the further evaluation $\varphi_1(t')\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big) = \varphi_1\big(t'(r'_0, \ldots, r'_{\ell'-1})\big)$, for random coordinates $(r'_0, \ldots, r'_{\ell'-1}) \in L^{\ell'}$. The key point is that this latter expression depends solely on the horizontal subring $\varphi_1(L) \subset A$, and so has nothing to do with A proper. To learn the desired quantity $t'(r'_0, \ldots, r'_{\ell'-1})$, we may thus make black-box use of the underlying large-field PCS over L.

We note a final benefit of our technique. Our ring-switching compiler reduces the problem of K-evaluation to that of L-evaluation. Applied to simultaneously to many instances—which themselves pertain to *different* subfields K of L, say—our compiler thus reduces its initial batch of claims to a set of further claims over the *single* field L. These latter claims can then be batch-evaluated. Informally, our ring-switching technique makes possible batching "across different fields". We discuss batching further in Subsection 5.3 below.

Performance. In Subsection 5.2 below, we thoroughly benchmark our scheme, measuring both its proof size and its prover and verifier complexities. Though our scheme's commitment and proving times still lag behind those of the state-of-the-art `Plonky3` scheme on polynomials with 32-bit coefficients, our scheme outperforms that one by a factor of roughly ten on 1-bit polynomials. Our scheme's proof sizes are comparable to those of Plonky3, and beat those of [DP23] by an order of magnitude.

1.3 Prior Work

The works most relevant to this one are Zeilberger, Chen and Fisch's *BaseFold* [ZCF23] and Diamond and Posen [DP23]. BaseFold's PCS [ZCF23, § 5] introduces the connection between FRI folding and multilinear evaluation upon which our large-field PCS rests. That work uses only prime-field FRI, and does not attempt to support small fields (with or without embedding overhead).

The work [DP23] initiates the use of towers of binary fields in SNARKs, and moreover develops several ideas which presage ours, including its tower algebra [DP23, Def. 3.8]. That work moreover isolates the notion of *small-field polynomial commitment schemes*, and supplies a key instantiation [DP23, Cons. 3.11]. We note that [DP23] presents not just a multilinear polynomial commitment scheme, but moreover an entire toolbox of "virtual polynomial protocols" [DP23, § 4] and a high-level SNARK [DP23, § 5]. This work presents *only* an improved polynomial commitment scheme. The higher-level content of [DP23] remains perfectly applicable in our setting; indeed, our scheme serves as a drop-in replacement for that of [DP23, § 3], and serves the purposes of [DP23, §§ 4–5] exactly as [DP23, § 3] does.

During our large-field scheme's security proof (see Theorems 3.16), we draw variously on the works [BBHR18] and [Ben+23]. Neither of those works contain results which serve *as stated* to achieve our purposes; rather, we must instead selectively extract and adapt their ideas. Indeed, our soundness proof must concern itself not merely with the prover's oracles' proximity, but moreover with their *consistency*. In any case, the essential ideas of our Lemmas 3.24 and 3.25 below are implicit in [BBHR18, § 4.2.2]; moreover, our Proposition 3.20 below can be viewed as an adaptation to our setting of a technique of [Ben+23, § 8.2].

Our ring-switching reduction (see Construction 4.1 below) is loosely inspired by Ron-Zewi and Rothblum [RR24, Fig. 2]'s "code-switching" technique, as we now explain. Indeed, our respective techniques share a few structural similarities. Both begin with an encoding procedure on the part of the prover. That work, which aims throughout to bring the total length of its oracles close to that of its witness, uses a high-rate tensor code; we simply use a standard, low-rate Reed–Solomon code (albeit in conjunction with our "packing" technique). During its inner IOPP, [RR24, Fig. 2] uses a special "sumcheck"-like protocol for high-rate tensor codes [RR24, Fig. 4]. Finally, both our protocol and [RR24, Fig. 2] decide the appropriate reduction target using a suitable "internal" protocol. Informally, both our protocol and Ron-Zewi and Rothblum [RR24, Fig. 2]'s uses a sumcheck to translate an "unfavorable" environment into a "favorable" one. We believe that that the analogy between that work's sumcheck and ours is loose; our sumcheck much-more-closely resembles the standard sumcheck for multilinear polynomials (though it operates over an unusual ring).

We do not yet, in this work, use the list-decoding proximity gap [Ben+23, Thm. 5.1]. We leave as a future research direction the adaptation of this work to the list-decoding regime.

Acknowledgements. We would like to acknowledge our colleagues at Irreducible for their insights and contributions to the *Binius* implementation of these techniques. We would like to gratefully thank Guillermo Angeris, Alex Evans, Angus Gruen, Ulrich Haböck, Gyumin Roh, Justin Thaler and Benjamin Wilson, whose collective comments and suggestions contributed significantly to this work. We thank Ron Rothblum for patiently explaining code-switching to us.

2 Background and Notation

We write $\mathbb{N}$ for the nonnegative integers. All fields in this work are finite. We fix a binary field L. For each $\ell \in \mathbb{N}$, we write $\mathcal{B}_\ell$ for the ℓ-dimensional *boolean hypercube* $\{0,1\}^\ell \subset L^\ell$. We occasionally identify $\mathcal{B}_\ell$ with the integer range $\{0, \dots 2^\ell - 1\}$ by mapping $v \mapsto \{v\} \coloneqq \sum_{i=0}^{\ell-1} 2^i \cdot v_i$. The *rings* we treat are nonzero and commutative with unit. For our purposes, an *algebra* A over a field L is a commutative ring A together with an embedding of rings $L \hookrightarrow A$. For L a field and $R \subset L^\vartheta$ a subset, we write $\mu(R) \coloneqq \frac{|R|}{|L|^\vartheta}$.

2.1 Lagrange and Monomial Forms

We review various normal forms for multilinear polynomials, following [DP23, § 2.1]. An ℓ-variate polynomial in $K[X_0, \ldots, X_{\ell-1}]$ is *multilinear* if each of its indeterminates appears with individual degree at most 1; we write $K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$ for the set of multilinear polynomials over K in ℓ indeterminates. Clearly, the set of monomials $(1, X_0, X_1, X_0 \cdot X_1, \ldots, X_0 \cdot \cdots \cdot X_{\ell-1})$ yields a K-basis for $K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$; we call this basis the *multilinear monomial basis* in ℓ variables.

We introduce the $2 \cdot \ell$-variate polynomial

$$\widetilde{\mathtt{eq}}(X_0, \ldots, X_{\ell-1}, Y_0, \ldots, Y_{\ell-1}) \coloneqq \prod_{i=0}^{\ell-1} (1 - X_i) \cdot (1 - Y_i) + X_i \cdot Y_i.$$

It is essentially the content of Thaler [Tha22, Fact. 3.5]) that the set $\left(\widetilde{\mathtt{eq}}(X_0, \ldots, X_{\ell-1}, v_0, \ldots, v_{\ell-1})\right)_{v \in \mathcal{B}_\ell}$ yields a K-basis of the space $K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$.

For each fixed $(r_0, \ldots, r_{\ell-1}) \in L^\ell$, the vector $\left(\widetilde{\mathtt{eq}}(r_0, \ldots, r_{\ell-1}, v_0, \ldots, v_{\ell-1})\right)_{v \in \mathcal{B}_\ell}$ takes the form

$$\left(\prod_{i=0}^{\ell-1} r_i \cdot v_i + (1 - r_i) \cdot (1 - v_i)\right)_{v \in \mathcal{B}_\ell} = ((1 - r_0) \cdot \cdots \cdot (1 - r_{\ell-1}), \ldots, r_0 \cdot \cdots \cdot r_{\ell-1}).$$

We call this vector the *tensor product expansion* of the point $(r_0, \ldots, r_{\ell-1}) \in L^\ell$, and denote it by $\bigotimes_{i=0}^{\ell-1}(1 - r_i, r_i)$. We note that this latter vector can be computed in $\Theta(2^\ell)$ time (see e.g. [Tha22, Lem. 3.8]).

As a notational device, we introduce the further $2 \cdot \ell$-variate polynomial:

$$\widetilde{\mathtt{mon}}(X_0, \ldots, X_{\ell-1}, Y_0, \ldots, Y_{\ell-1}) \coloneqq \prod_{i=0}^{\ell-1} 1 + (X_i - 1) \cdot Y_i;$$

we note that $\left(\widetilde{\mathtt{mon}}(X_0, \ldots, X_{\ell-1}, v_0, \ldots, v_{\ell-1})\right)_{v \in \mathcal{B}_\ell}$ yields the multilinear monomial basis in ℓ indeterminates.

2.2 Error-Correcting Codes

We recall details on codes, referring throughout to Guruswami [Gur06]. A *code* of block length n over the alphabet Σ is a subset of Σ^n. In Σ^n, we write d for the Hamming distance between two vectors (i.e., the number of components at which they differ). We fix a field L. A linear $[n, k, d]$-*code* over L is a k-dimensional linear subspace $C \subset L^n$ for which $d(v_0, v_1) \geq d$ holds for each unequal pair of elements v_0 and v_1 of C. The *unique decoding radius* of the $[n, k, d]$-code $C \subset L^n$ is $\left\lfloor \frac{d-1}{2} \right\rfloor$; indeed, we note that, for each word $u \in L^n$, at most one codeword $v \in C$ satisfies $d(u, v) < \frac{d}{2}$ (this fact is a direct consequence of the triangle inequality). For $u \in L^n$ arbitrary, we write $d(u, C) \coloneqq \min_{v \in C} d(u, v)$ for the *distance* between u and the code C.

Given a linear code $C \subset L^n$ and an integer $m \geq 1$, we have C's *m-fold interleaved code*, defined as the subset $C^m \subset (L^n)^m \cong (L^m)^n$. We understand this latter set as a length-n block code over the alphabet L^m. In particular, its elements are essentially matrices in $L^{m \times n}$ each of whose rows is a C-element. We write matrices $(u_i)_{i=0}^{m-1} \in L^{m \times n}$ row-wise. By definition of C^m, two matrices in $L^{m \times n}$ *differ* at a column if they differ at *any* of that column's components. That a matrix $(u_i)_{i=0}^{m-1} \in L^{m \times n}$ is within distance e to the code C^m—in which event we write $d^m\left((u_i)_{i=0}^{m-1}, C^m\right) \leq e$—thus entails precisely that there exists a subset $D \coloneqq \Delta^m\left((u_i)_{i=0}^{m-1}, C^m\right)$, say, of $\{0, \ldots, n-1\}$, of size at most e, for which, for each $i \in \{0, \ldots, m-1\}$, the row u_i admits a codeword $v_i \in C$ for which $u_i|_{\{0,\ldots,n-1\} \setminus D} = v_i|_{\{0,\ldots,n-1\} \setminus D}$.

We recall Reed–Solomon codes (see [Gur06, Def. 2.3]). For notational convenience, we consider only Reed–Solomon codes whose message and block lengths are powers of two. We fix nonnegative *message length* and *rate* parameters ℓ and $\mathcal{R}$, as well as a subset $S \subset L$ of size $2^{\ell+\mathcal{R}}$. We write $C \subset L^{2^{\ell+\mathcal{R}}}$ for the *Reed–Solomon* code $\mathsf{RS}_{L,S}[2^{\ell+\mathcal{R}}, 2^\ell]$ is defined to be the set $\left\{(P(x))_{x \in S} \;\middle|\; P(X) \in L[X]^{\prec 2^\ell}\right\}$; that is, $\mathsf{RS}_{L,S}[2^{\ell+\mathcal{R}}, 2^\ell]$ is the set of those $2^{\ell+\mathcal{R}}$-tuples which arise as the evaluations of some polynomial of degree less than 2^ℓ over S. The distance of $\mathsf{RS}_{L,S}[n, k]$ is $d = 2^{\ell+\mathcal{R}} - 2^\ell + 1$. We write $\mathsf{Enc} : L[X]^{\prec 2^\ell} \to L^S$ for the *encoding function* which maps a polynomial $P(X)$ of degree less than 2^ℓ to its tuple of evaluations over S.

We recall the *Berlekamp–Welch* algorithm for Reed–Solomon decoding within the unique decoding radius (see [Gur06, Rem. 4]).

Algorithm 1 (Berlekamp–Welch [Gur06, Rem. 4].)

procedure DecodeReedSolomon$\big((f(x))_{x\in S}\big)$
  allocate $A(X)$ and $B(X)$ of degrees $\lfloor\frac{d-1}{2}\rfloor$ and $2^{\ell+\mathcal{R}}-\lfloor\frac{d-1}{2}\rfloor-1$; write $Q(X,Y)\coloneqq A(X)\cdot Y+B(X)$.
  interpret the equalities $Q(x,f(x))=0$, for $x\in S$, as a system of $2^{\ell+\mathcal{R}}$ equations in $2^{\ell+\mathcal{R}}+1$ unknowns.
  by finding a nonzero solution of this linear system, obtain values for the polynomials $A(X)$ and $B(X)$.
  if $A(X)\nmid B(X)$ **then return** $\perp$.
  write $P(X)\coloneqq -B(X)/A(X)$.
  return $P(X)$.

We note that the unknown polynomial $Q(X,Y)$ above indeed has $\lfloor\frac{d-1}{2}\rfloor+1+2^{\ell+\mathcal{R}}-\lfloor\frac{d-1}{2}\rfloor=2^{\ell+\mathcal{R}}+1$ coefficients, as required.

Upon being given an input word $f:S\to L$ for which $d(f,C)<\frac{d}{2}$, Algorithm 1 necessarily returns the unique polynomial $P(X)$ of degree less than 2^ℓ for which $d(f,\mathsf{Enc}(P(X)))<\frac{d}{2}$ holds. Indeed, this is simply the correctness of Berlekamp–Welch algorithm on input assumed to reside within the unique decoding radius; we refer to [Gur06, Rem. 4] for a thorough treatment. We discuss this algorithm further in Remark **??** below.

2.3 The Novel Polynomial Basis

We recall in detail the *novel polynomial basis* of Lin, Chung and Han [LCH14, § II.]. We fix again a binary field L, of degree r, say, over $\mathbb{F}_2$. For our purposes, a *subspace polynomial* over L is a polynomial $W(X)\in L[X]$ which splits completely over L, and whose roots, each of multiplicity 1, form an $\mathbb{F}_2$-linear subspace of L. For a detailed treatment of subspace polynomials, we refer to Berlekamp [Ber15, § 11]. We recall that, for each subspace polynomial $W(X)\in L[X]$, the evaluation map $W:L\to L$ is $\mathbb{F}_2$-*linear*.

For each fixed $\ell\in\{0,\dots,r-1\}$, the set $L[X]^{\prec 2^\ell}$ of polynomials of degree less than 2^ℓ is a 2^ℓ-dimensional vector space over L. Of course, the set $(1,X,X^2,\dots,X^{2^\ell-1})$ yields a natural L-basis of $L[X]^{\prec 2^\ell}$. Lin, Chung and Han define a further L-basis of $L[X]^{\prec 2^\ell}$—called the *novel polynomial basis*—in the following way. We fix once and for all an $\mathbb{F}_2$-basis $(\beta_0,\dots,\beta_{r-1})$ of L (which we view as an r-dimensional vector space over its subfield $\mathbb{F}_2$). For each $i\in\{0,\dots,\ell-1\}$, we write $U_i\coloneqq\langle\beta_0,\dots,\beta_{i-1}\rangle$ for the $\mathbb{F}_2$-linear span of the prefix $(\beta_0,\dots,\beta_{i-1})$, and define the *subspace vanishing polynomial* $W_i(X)\coloneqq\prod_{u\in U_i}X-u$, as well as its *normalized* variant $\widehat{W}_i(X)\coloneqq\frac{W_i(X)}{W_i(\beta_i)}$ (we note that $\beta_i\notin U_i$, so that $W_i(\beta_i)\neq 0$). In words, for each $i\in\{0,\dots,\ell-1\}$, $W_i(X)$ vanishes precisely on $U_i\subset L$; $\widehat{W}_i(X)$ moreover satisfies $\widehat{W}_i(X)(\beta_i)=1$. Finally, for each $j\in\{0,\dots,2^\ell-1\}$, we write $(j_0,\dots,j_{\ell-1})$ for the bits of j—so that $j=\sum_{k=0}^{\ell-1}2^k\cdot j_k$ holds—and set $X_j(X)\coloneqq\prod_{i=0}^{\ell-1}\widehat{W}_i(X)^{j_i}$. We note that, for each $j\in\{0,\dots,2^\ell-1\}$, $X_j(X)$ is of degree j. We conclude that the change-of-basis matrix from $(1,X,\dots,X^{2^\ell-1})$ to $(X_0(X),X_1(X),\dots,X_{2^\ell-1}(X))$ is triangular (with an everywhere-nonzero diagonal), so that this latter list indeed yields a L-basis of $L[X]^{\prec 2^\ell}$.

As in Subsection 2.4 above, we now fix moreover a rate parameter $\mathcal{R}\in\{1,\dots,r-\ell\}$ and an $\mathbb{F}_2$-subspace $S\subset L$ of dimension $\ell+\mathcal{R}$; now, we require moreover that S *contain* the $\mathbb{F}_2$-subspace $U_\ell\coloneqq\langle\beta_0,\dots,\beta_{\ell-1}\rangle$. For each subspace $S\subset L$ of this form, Lin, Chung and Han [LCH14, § III.]'s $\Theta(\ell\cdot 2^{\ell+\mathcal{R}})$-time algorithm serves to compute, on input the polynomial $P(X)\coloneqq\sum_{j=0}^{2^\ell-1}a_j\cdot X_j(X)$ (expressed in coordinates with respect to the novel polynomial basis), its encoding $(P(x))_{x\in S}$.

In Remark 3.15 below, we suggest a new interpretation of Lin, Chung and Han's algorithm [LCH14, § III.], based on the techniques of this paper. For now, for self-containedness, we record here the key algorithm in full, in our notation. We note that Algorithm 2's equivalence with [LCH14, § III.] is not obvious; we explain the correctness of our description in Remark 3.15 below. In what follows, we fix as above the degree and rate parameters ℓ and $\mathcal{R}$. We finally fix a polynomial $P(X)=\sum_{j=0}^{2^\ell-1}a_j\cdot X_j(X)$; we write $b:\mathcal{B}_{\ell+\mathcal{R}}\to L$ for $(a_j)_{j=0}^{2^\ell-1}$'s $2^{\mathcal{R}}$-fold tiling; in other words, for each $v\in\mathcal{B}_{\ell+\mathcal{R}}$, we set $b(v_0,\dots,v_{\ell+\mathcal{R}-1})\coloneqq a_{\{(v_0,\dots,v_{\ell-1})\}}$.

Algorithm 2 (Lin–Chung–Han [LCH14, § III.].)

procedure AdditiveNTT$\Big((b(v))_{v\in\mathcal{B}_{\ell+\mathcal{R}}}\Big)$
  for $i \in \{\ell-1,\dots,0\}$ (i.e., in downward order) **do**
    for $(u,v) \in \mathcal{B}_{\ell+\mathcal{R}-i-1} \times \mathcal{B}_i$ **do**
      define the twiddle factor $t \coloneqq \sum_{k=0}^{\ell+\mathcal{R}-i-2} u_k \cdot \widehat{W}_i(\beta_{i+1+k})$.
      overwrite first $b(u \parallel 0 \parallel v) \mathrel{+}= t \cdot b(u \parallel 1 \parallel v)$ and then $b(u \parallel 1 \parallel v) \mathrel{+}= b(u \parallel 0 \parallel v)$.
  return $(b(v))_{v\in\mathcal{B}_{\ell+\mathcal{R}}}$.

We note that the twiddle factor t above depends only on u, and not on v, and can be reused accordingly. Finally, in the final return statement above, we implicitly identify $\mathcal{B}_{\ell+\mathcal{R}} \cong S$ using the standard basis $\beta_0,\dots,\beta_{\ell+\mathcal{R}-1}$ of the latter space (see also Subsection 3.1 below).

2.4 FRI

We recall Ben-Sasson, Bentov, Horesh and Riabzev's [BBHR18] *Fast Reed–Solomon Interactive Oracle Proof of Proximity* (FRI). For L a binary field, and size and rate parameters ℓ and $\mathcal{R}$ fixed, FRI yields an *IOP of proximity* for the Reed–Solomon code $\mathsf{RS}_{L,S}[2^{\ell+\mathcal{R}}, 2^\ell]$; here, we require that $S \subset L$ be an $\mathbb{F}_2$-*linear* subspace (of dimension $\ell+\mathcal{R}$, of course). That is, FRI yields an IOP for the claim whereby some oracle $[f]$—i.e., representing a function $f : S \to L$—is close to a codeword $(P(x))_{x\in S}$ (here, $P(X) \in L[X]^{\prec 2^\ell}$ represents a polynomial of degree less than 2^ℓ). FRI's verifier complexity is polylogarithmic in 2^ℓ. We abbreviate $\rho \coloneqq 2^{-\mathcal{R}}$, so that $\mathsf{RS}_{L,S}[2^{\ell+\mathcal{R}}, 2^\ell]$ is of rate ρ.

Internally, FRI makes use of a folding constant η—which we fix to be 1—as well as a fixed, global *sequence* of subspaces and maps of the form:

$$S = S^{(0)} \xrightarrow{q^{(0)}} S^{(1)} \xrightarrow{q^{(1)}} S^{(2)} \xrightarrow{q^{(2)}} \cdots \xrightarrow{q^{(\ell-1)}} S^{(\ell)}. \tag{1}$$

Here, for each $i \in \{0,\dots,\ell-1\}$, $q^{(i)}$ is a subspace polynomial of degree $2^\eta = 2$, whose kernel, which is 1-dimensional, is moreover contained in $S^{(i)}$. By linear-algebraic considerations, we conclude that $S^{(i+1)}$'s $\mathbb{F}_2$-dimension is 1 less than $S^{(i)}$'s is; inductively, we conclude that each $S^{(i)}$ is of dimension $\ell+\mathcal{R}-i$.

2.5 Tensor Products of Fields

We record algebraic preliminaries, referring throughout to Lang [Lan02, Ch. XVI]. We fix a field extension $L \,/\, K$. We define the *tensor product* $A \coloneqq L \otimes_K L$ of L with itself over K as in [Lan02, Ch. XVI § 6]. Here, we view L as a K-algebra; the resulting object $A \coloneqq L \otimes_K L$ is likewise a K-algebra. We would like to sincerely thank Benjamin Wilson for first suggesting to us this tensor-theoretic perspective on the tower algebra [DP23, § 3.4].

We recall from [Lan02, Ch. XVI, § 1] the natural K-bilinear mapping $\varphi : L \times L \to L \otimes_K L$ which sends $\varphi : (\alpha_0, \alpha_1) \mapsto \alpha_0 \otimes \alpha_1$. We write φ_0 and φ_1 for φ's restrictions to the subsets $L \times \{1\}$ and $\{1\} \times L$ of $L \times L$, and moreover identify these latter subsets with L. That is, we write $\varphi_0 : \alpha \mapsto \alpha \otimes 1$ and $\varphi_1 : \alpha \mapsto 1 \otimes \alpha$, both understood as maps $L \to A$. We claim that these maps are injective (i.e., that they're not identically zero). We follow Lang [Lan02, Ch. XVI, § 2, Prop. 2.3]. The mapping $f : L \times L \to L$ sending $f : (\alpha_0, \alpha_1) \mapsto \alpha_0 \cdot \alpha_1$ is K-bilinear; by the universal property of the tensor product, f induces a K-linear map $h : L \otimes_K L \to L$, for which, for each $\alpha \in L$, $h(\alpha \otimes 1) = f(\alpha, 1) = \alpha \cdot 1 = \alpha$ holds; we see that $\alpha \otimes 1 = 0$ if and only if $\alpha = 0$.

We assume once and for all that $\deg(L \,/\, K)$ is a power of 2, say 2^κ. We fix a K-basis $(\beta_v)_{v\in\mathcal{B}_\kappa}$ of L. We moreover impose the simplifying assumption whereby $\beta_{(0,\dots,0)} = 1$. By [Lan02, Ch. XVI, § 2, Cor. 2.4], the set $(\beta_u \otimes \beta_v)_{(u,v)\in\mathcal{B}_\kappa\times\mathcal{B}_\kappa}$ yields a K-basis of A. We thus see that each A-element is, concretely, a $2^\kappa \times 2^\kappa$ array of K-elements. For each $a \in A$ given, there is a unique 2^κ-tuple of L-elements $(a_v)_{v\in\mathcal{B}_\kappa}$ for which $a = \sum_{v\in\mathcal{B}_\kappa} a_v \otimes \beta_v$ holds. (Indeed, this is just [Lan02, Ch. XVI, § 2, Prop. 2.3].) Similarly, there is a unique 2^κ-tuple of L-elements $(a_u)_{u\in\mathcal{B}_\kappa}$ for which $a = \sum_{u\in\mathcal{B}_\kappa} \beta_u \otimes a_u$ holds. We call the tuples $(a_v)_{v\in\mathcal{B}_\kappa}$ and $(a_u)_{u\in\mathcal{B}_\kappa}$ a's *column* and *row* representations, respectively.

We depict the tensor algebra in Figure 1 below.

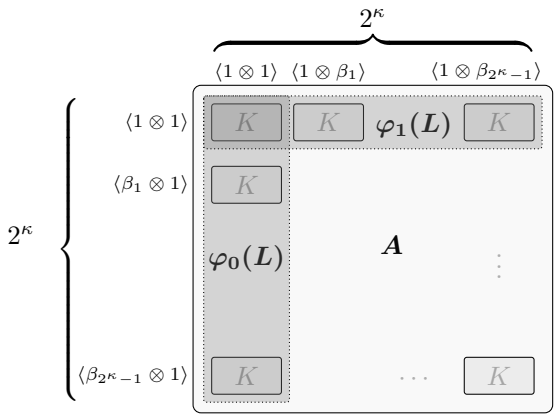

Figure 1: A depiction of our "tensor algebra" data structure.

The maps φ_0 and φ_1 respectively embed L into A's left-hand *column* and top *row*. That is, the image of $\varphi_0 : L \hookrightarrow A$ is the set of K-arrays which are 0 except in their respective left-most columns; the image of $\varphi_1 : L \hookrightarrow A$ is the set of K-arrays which are 0 outside of their top rows. We finally characterize concretely the products $\varphi_0(\alpha) \cdot a$ and $\varphi_1(\alpha) \cdot a$, for elements $\alpha \in L$ and $a \in A$ arbitrary. It is a straightforward to show that $\varphi_0(\alpha) \cdot a = \sum_{v \in \mathcal{B}_\kappa} (\alpha \cdot a_v) \otimes \beta_v$ and $\varphi_1(\alpha) \cdot a = \sum_{u \in \mathcal{B}_\kappa} \beta_u \otimes (\alpha \cdot a_u)$ both hold; here, we again write $(a_v)_{v \in \mathcal{B}_\kappa}$ and $(a_u)_{u \in \mathcal{B}_\kappa}$ for a's column and row representations. That is, $\varphi_0(\alpha) \cdot a$ differs from a by column-wise multiplication by α; $\varphi_1(\alpha) \cdot a$ differs from a by row-wise multiplication by α. In short, φ_0 operates on columns; φ_1 operates on rows.

Below, the tensor algebra $A := L \otimes_K L$ plays a critical role in our "ring-switching" technique (see Section 4). For now, we record a simple polynomial-packing operation, which is implicit in [DP23, § 3.4]. We obtain a natural K-isomorphism $K^{2^\kappa} \to L$ via the basis combination procedure $(\alpha_v)_{v \in \mathcal{B}_\kappa} \to \sum_{v \in \mathcal{B}_\kappa} \alpha_v \cdot \beta_v$. By applying this map in "chunks", we can replace any ℓ-variate K-polynomial by an $\ell - \kappa$-variate L-polynomial.

Definition 2.1. For each extension L/K, with K-basis $(\beta_v)_{v \in \mathcal{B}_\kappa}$ say, and each multilinear $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, we write $\ell' := \ell - \kappa$, and define the *packed polynomial* $t'(X_0, \ldots, X_{\ell'-1}) \in L[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$ by declaring, for each $v \in \mathcal{B}_{\ell'}$, that $t' : v \mapsto \sum_{u \in \mathcal{B}_\kappa} t(u_0, \ldots, u_{\kappa-1}, v_0, \ldots, v_{\ell'-1}) \cdot \beta_u$.

Definition 2.1 replaces each little-endian chunk—consisting of 2^κ adjacent K-elements—of $t(X_0, \ldots, X_{\ell-1})$'s Lagrange coefficient vector with a *single* L-element, by basis-combining that chunk.

We emphasize that Definition 2.1's packing procedure is *reversible* (see also [DP23, Thm. 3.9]); that is, $t'(X_0, \ldots, X_{\ell'-1})$ can be "unpacked". We note that Definition 2.1 is essentially the same as [DP23, § 4.3].

We finally write $\varphi_1(t')(X_0, \ldots, X_{\ell'-1}) \in A[X_0, \ldots, X_{\ell'-1}]$ for the result of embedding $t'(X_0, \ldots, X_{\ell'-1})$, componentwise, along the inclusion $\varphi_1 : L \hookrightarrow A$.

2.6 Binary Towers

We recall towers of binary fields, referring throughout to [DP23, § 2.3]. For simplicity, we present only Wiedemann's tower [Wie88]; on the other hand, our results go through without change on other binary towers (cf. e.g. the *Cantor tower* given in Li et al. [Li+18, § 2.1]). That is, we set $\mathcal{T}_0 := \mathbb{F}_2$ and $\mathcal{T}_1 := \mathbb{F}_2[X_0]/(X_0^2 + X_0 + 1)$, as well as, for each $\iota > 1$, $\mathcal{T}_\iota := \mathcal{T}_{\iota-1}/(X_{\iota-1}^2 + X_{\iota-2} \cdot X_{\iota-1} + 1)$. Fan and Paar [FP97] observe that the multiplication and inversion operations in Wiedemann's tower admit $O\big(2^{\log(3) \cdot \iota}\big)$-time algorithms.

The *monomial* $\mathbb{F}_2$*-basis* of the binary tower $\mathcal{T}_\tau$ is $(\beta_v)_{v \in \mathcal{B}_\tau} := (\widetilde{\texttt{mon}}(X_0, \ldots, X_{\tau-1}, v_0, \ldots, v_{\tau-1}))_{v \in \mathcal{B}_\tau}$. More generally, for each pair of integers $\iota \geq 0$ and $\tau \geq \iota$, the set $(\widetilde{\texttt{mon}}(X_\iota, \ldots, X_{\tau-1}, v_0, \ldots, v_{\tau-\iota-1})_{v \in \mathcal{B}_{\tau-\iota}}$ likewise yields a $\mathcal{T}_\iota$-basis of $\mathcal{T}_\tau$; we again write $(\beta_v)_{v \in \mathcal{B}_{\tau-\iota}}$ for this basis.

The *tower algebra* data structure of Diamond and Posen [DP23, § 3.4] is essentially nothing other than $\mathcal{T}_\tau \otimes_{\mathcal{T}_\iota} \mathcal{T}_{\iota+\kappa}$. We use tensor-notation in this work; we thus avoid referring to that algebra directly. In this work, we moreover only consider "square" tensors (i.e., of the same field with itself). That work's "constant" and "synthetic" embeddings correspond to our embeddings φ_0 and φ_1, respectively.

2.7 Proximity Gaps

We turn to proximity gaps, following Ben-Sasson, et al. and [Ben+23] and Diamond and Posen [DP24]. As above, we fix a Reed–Solomon code $C := \mathsf{RS}_{L,S}[2^{\ell+\mathcal{R}}, 2^{\ell}]$; we moreover write $d := 2^{\ell+\mathcal{R}} - 2^{\ell} + 1$ for C's distance. In the following results, for notational convenience, we abbreviate $n := 2^{\ell+\mathcal{R}}$ for the Reed–Solomon code C's block length.

Theorem 2.2. *Fix a proximity parameter $e \in \left\{0, \ldots, \left\lfloor \frac{d-1}{2} \right\rfloor\right\}$. If the words u_0 and u_1 in $L^{2^{\ell+\mathcal{R}}}$ satisfy*

$$\Pr_{r \in L}[d((1-r) \cdot u_0 + r \cdot u_1, C) \leq e] > \frac{n}{|L|},$$

then $d^2\Big((u_i)_{i=0}^{1}, C^2\Big) \leq e$.

Proof. This result is exactly [Ben+23, Thm. 4.1], though we use a slightly different parameterization; that is, our line is of the form $(1-r_0) \cdot u_0 + r_0 \cdot u_1$, while that result considers lines of the form $u_0' + r_0 \cdot u_1'$. The difference between these conventions is immaterial, up to the reparameterization which sets $u_0' := u_0$ and $u_1' := u_1 - u_0$ (this reparameterization moreover doesn't affect the conclusion). □

In the following result, we fix an integer $\vartheta > 1$. The following analogue of Theorem 2.2 takes place in the *interleaved* code $C^{2^{\vartheta-1}}$.

Theorem 2.3. *Fix a proximity parameter $e \in \left\{0, \ldots, \left\lfloor \frac{d-1}{2} \right\rfloor\right\}$. If the words U_0 and U_1 in $L^{2^{\vartheta-1} \times 2^{\ell+\mathcal{R}}}$ satisfy*

$$\Pr_{r \in L}\left[d^{2^{\vartheta-1}}\left((1-r) \cdot U_0 + r \cdot U_1, C^{2^{\vartheta-1}}\right) \leq e\right] > \frac{n}{|L|},$$

then $d^2\left((U_i)_{i=0}^{1}, \left(C^{2^{\vartheta-1}}\right)^2\right) \leq e$.

Proof. The theorem's hypothesis implies that at least two parameters $r \in L$—say, r_0^* and r_1^*—satisfy $d^{2^{\vartheta-1}}\left((1-r) \cdot U_0 + r \cdot U_1, C^{2^{\vartheta-1}}\right) \leq e$. We may therefore assume, up to replacing $U_0 := (1-r_0^*) \cdot U_0 + r_0^* \cdot U_1$ and $U_1 := (1-r_1^*) \cdot U_0 + r_1^* \cdot U_1$, that $d^{2^{\vartheta-1}}\left(U_0, C^{2^{\vartheta-1}}\right) \leq e$ and $d^{2^{\vartheta-1}}\left(U_1, C^{2^{\vartheta-1}}\right) \leq e$ both hold. We fix interleaved *codewords* ${V_0}^*$ and ${V_1}^*$ in $C^{2^{\vartheta-1}}$ for which $d^{2^{\vartheta-1}}(U_0, {V_0}^*) \leq e$ and $d^{2^{\vartheta-1}}(U_1, {V_1}^*) \leq e$ hold; we moreover write $E_0^* := \Delta^{2^{\vartheta-1}}(U_0, {V_0}^*)$ and $E_1^* := \Delta^{2^{\vartheta-1}}(U_1, {V_1}^*)$.

We write $U_0 = (u_{0,i})_{i=0}^{2^{\vartheta-1}-1}$ and $U_1 = (u_{1,i})_{i=0}^{2^{\vartheta-1}-1}$ for the rows of the matrices of the theorem's hypothesis. For each *individual* row $i \in \{0, \ldots, 2^{\vartheta-1}-1\}$, the theorem's hypothesis implies *a fortiori* that the hypothesis of Theorem 2.2 is fulfilled with respect to the affine line $(1-r) \cdot u_{0,i} + r \cdot u_{1,i}$. That theorem yields codewords $v_{0,i}$ and $v_{1,i}$ in C and a subset $E_i := \Delta^2\Big((u_{k,i})_{k=0}^{1}, C^2\Big)$ of $\{0, \ldots, 2^{\ell+\mathcal{R}} - 1\}$, of cardinality at most e, for which, for each $j \notin E_i$ $(u_{0,i})_j = (v_{0,i})_j$ and $(u_{1,i})_j = (v_{1,i})_j$ both hold. We write $V_0 := (v_{0,i})_{i=0}^{2^{\vartheta-1}-1}$ and $V_1 = (v_{1,i})_{i=0}^{2^{\vartheta-1}-1}$. Certainly, $\Delta^2\Big((U_i)_{i=0}^{1}, (V_i)_{i=0}^{1}\Big) \subset \bigcup_{i=0}^{2^{\vartheta-1}-1} E_i$ holds; indeed, for each $j \notin \bigcup_{i=0}^{2^{\vartheta-1}-1} E_i$, by definition of the sets E_i, we have $(u_{b,i})_j = (v_{b,i})_j$ for each $b \in \{0,1\}$ and each $i \in \{0, \ldots, 2^{\vartheta-1}-1\}$. On the other hand, we claim that $\bigcup_{i=0}^{2^{\vartheta-1}-1} E_i \subset E_0^* \cap E_1^*$. This claim suffices to prove the theorem, since $|E_0^* \cap E_1^*| \leq e$.

To prove the claim, we first argue that both $V_0 = {V_0}^*$ and $V_1 = {V_1}^*$ hold. To this end, we fix indices $b \in \{0,1\}$ and $i \in \{0, \ldots, 2^{\vartheta-1}-1\}$ arbitrarily. The triangle inequality implies that $d(v_{b,i}, {v_{b,i}}^*) \leq d(v_{b,i}, u_{b,i}) + d(u_{b,i}, {v_{b,i}}^*)$; the former distance is at most $|E_i| \leq e$, while the latter distance is at most $|E_b^*| \leq e$. We conclude that $d(v_{b,i}, {v_{b,i}}^*) \leq 2 \cdot e < d$, so that $v_{b,i} = {v_{b,i}}^*$, as required.

We now let $b \in \{0,1\}$ be arbitrary; we moreover fix an index $j \notin E_b^*$. By definition of E_b^*, j satisfies $(u_{b,i})_j = ({v_{b,i}}^*)_j$ for each $i \in \{0, \ldots, 2^{\vartheta-1}-1\}$; exploiting the equality $V_b = {V_b}^*$ just proven, we conclude in turn that $(u_{b,i})_j = (v_{b,i})_j$ holds for each $i \in \{0, \ldots, 2^{\vartheta-1}-1\}$. By definition of the sets E_i, we conclude finally in turn that $j \notin \bigcup_{i=0}^{2^{\vartheta-1}-1} E_i$ holds; this argument proves that $\bigcup_{i=0}^{2^{\vartheta-1}-1} E_i \subset E_b^*$, as required. □

We note that the doubly interleaved code $\left(C^{2^{\vartheta-1}}\right)^2$ of Theorem 2.3's conclusion is the same as C^{2^ϑ}.

Theorem 2.4. *Fix a proximity parameter $e \in \left\{0, \ldots, \left\lfloor \frac{d-1}{2} \right\rfloor\right\}$. If the words $u_0, \ldots, u_{2^\vartheta-1}$ in $L^{2^{\ell+\mathcal{R}}}$ satisfy*

$$\Pr_{(r_0,\ldots,r_{\vartheta-1})\in L^\vartheta}\left[d\left(\begin{bmatrix} \bigotimes_{i=0}^{\vartheta-1}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \vdots & \\ — & u_{2^\vartheta-1} & — \end{bmatrix}, C\right) \leq e\right] > \vartheta \cdot \frac{n}{|L|},$$

then $d^m\left((u_i)_{i=0}^{2^\vartheta-1}, C^{2^\vartheta}\right) \leq e$.

Proof. We prove the result by induction on ϑ. In the base case $\vartheta = 1$, the theorem's statement is exactly Theorem 2.2. We turn to the case $\vartheta > 1$. That the case $\vartheta = 1$ of the theorem implies the case $\vartheta > 1$ is essentially the content of [DP24, Thm. 2]. We present a variant of that result which eliminates a certain factor of 2 present in that result's soundness bound (see also [DP24, Rem. 3]). We would like to sincerely thank Guillermo Angeris, Alex Evans and Gyumin Roh for suggesting to us this argument.

We write U_0 and U_1 for $(u_i)_{i=0}^{2^\vartheta-1}$'s lower and upper halves. We first note a variant of the recursive substructure given in [DP24, Thm. 2]:

$$\begin{bmatrix} \bigotimes_{i=0}^{\vartheta-1}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \vdots & \\ — & u_{2^\vartheta-1} & — \end{bmatrix} = \begin{bmatrix} \bigotimes_{i=0}^{\vartheta-2}(1-r_i, r_i) \end{bmatrix} \cdot \left(\begin{bmatrix} (1-r_{\vartheta-1}) \cdot U_0 \end{bmatrix} + \begin{bmatrix} r_{\vartheta-1} \cdot U_1 \end{bmatrix}\right).$$

For each $r_{\vartheta-1} \in L$, we moreover abbreviate:

$$p(r_{\vartheta-1}) \coloneqq \Pr_{(r_0,\ldots,r_{\vartheta-2})\in L^{\vartheta-1}}\left[d\left(\begin{bmatrix} \bigotimes_{i=0}^{\vartheta-2}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} (1-r_{\vartheta-1}) \cdot U_0 + r_{\vartheta-1} \cdot U_1 \end{bmatrix}, C\right) \leq e\right].$$

we finally write $R^* \coloneqq \left\{r_{\vartheta-1} \in L \;\middle|\; p(r_{\vartheta-1}) > (\vartheta-1) \cdot \frac{n}{|L|}\right\}$. We first claim that $|R^*| > n$ holds. Indeed, under the hypothesis of the theorem, we have the following probability decomposition, which evokes [DP24, Lem. 2] (though it proceeds in the "opposite direction"):

$$\begin{aligned}
\vartheta \cdot \frac{n}{|L|} &< \Pr_{(r_0,\ldots,r_{\vartheta-1})\in L^\vartheta}\left[d\left(\begin{bmatrix} \bigotimes_{i=0}^{\vartheta-1}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \vdots & \\ — & u_{2^\vartheta-1} & — \end{bmatrix}, C\right) \leq e\right] \\
&= \Pr_{(r_0,\ldots,r_{\vartheta-1})\in L^\vartheta}\left[d\left(\begin{bmatrix} \bigotimes_{i=0}^{\vartheta-2}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} (1-r_{\vartheta-1}) \cdot U_0 + r_{\vartheta-1} \cdot U_1 \end{bmatrix}, C\right) \leq e\right] \\
&\leq (\vartheta-1) \cdot \frac{n}{|L|} + \Pr_{r_{\vartheta-1}\in L}[r_{\vartheta-1} \in R^*].
\end{aligned}$$

By subtraction, this calculation clearly implies that $\mu(R^*) > \frac{n}{|L|}$, as required. To achieve the final step, we slice the space L^ϑ along its last coordinate $r_{\vartheta-1}$. For *each* slice $r_{\vartheta-1} \in L$, we upper-bound the proportion of elements $(r_0, \ldots, r_{\vartheta-2}) \in L^{\vartheta-1}$ for which $d\left(\bigotimes_{i=0}^{\vartheta-2}(1-r_i, r_i) \cdot ((1-r_{\vartheta-1}) \cdot U_0 + r_{\vartheta-1} \cdot U_1), C\right) \leq e$ holds, either trivially by 1 (if $r_{\vartheta-1} \in R^*$) or else by $(\vartheta-1) \cdot \frac{n}{|L|}$ (if $r_{\vartheta-1} \notin R^*$).

We let $r^*_{\vartheta-1} \in R^*$ be arbitrary. By definition of R^*, we note that the hypothesis of this theorem is fulfilled with respect to the parameter $\vartheta - 1$ and the half-length matrix $(1-r^*_{\vartheta-1}) \cdot U_0 + r^*_{\vartheta-1} \cdot U_1$. Applying the theorem inductively to that matrix, we conclude that $d^{2^{\vartheta-1}}\left((1-r^*_{\vartheta-1}) \cdot U_0 + r^*_{\vartheta-1} \cdot U_1, C^{2^{\vartheta-1}}\right) \leq e$ holds for each $r^*_{\vartheta-1} \in R^*$. In view of our above guarantee whereby $|R^*| > n$, this fact in turn implies that the hypothesis of Theorem 2.3 is fulfilled with respect to the matrices U_0 and U_1. Applying that theorem, we conclude finally that the conclusion of this theorem holds with respect to $(u_i)_{i=0}^{2^\vartheta-1}$, as required. □

2.8 Security Definitions

We record security definitions. We begin by defining various abstract oracle machines, following [DP23].

FUNCTIONALITY 2.5 ($\mathcal{F}_{\mathsf{Vec}}^{L}$—vector oracle)**.**
An arbitrary alphabet L is given.

- Upon receiving $(\texttt{submit}, m, f)$ from $\mathcal{P}$, where $m \in \mathbb{N}$ and $f \in L^{\mathcal{B}_m}$, output $(\texttt{receipt}, L, [f])$ to all parties, where $[f]$ is some unique handle onto the vector f.
- Upon receiving $(\texttt{query}, [f], v)$ from $\mathcal{V}$, where $v \in \mathcal{B}_m$, send $\mathcal{V}$ $(\texttt{result}, f(v))$.

FUNCTIONALITY 2.6 ($\mathcal{F}_{\mathsf{Poly}}^{\lambda,\ell}$—polynomial oracle)**.**
A security parameter $\lambda \in \mathbb{N}$ and a number-of-variables parameter $\ell \in \mathbb{N}$ are given. The functionality constructs and fixes a field L (allowed to depend on λ and ℓ).

- Upon receiving $(\texttt{submit}, t)$ from $\mathcal{P}$, where $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, output $(\texttt{receipt}, [t])$ to all parties, where $[t]$ is some unique handle onto the polynomial t.
- On input $(\texttt{query}, [t], r)$ from $\mathcal{V}$, where $r \in L^{\ell}$, send $\mathcal{V}$ $(\texttt{result}, t(r_0, \ldots, r_{\ell-1}))$.

FUNCTIONALITY 2.7 ($\mathcal{F}_{\mathsf{SFPoly}}^{\lambda,K,\ell}$—small-field polynomial oracle)**.**
A security parameter $\lambda \in \mathbb{N}$, a number-of-variables parameter $\ell \in \mathbb{N}$, and a ground field K are given. The functionality constructs and fixes a field extension $L \,/\, K$ (allowed to depend on λ, ℓ and K).

- Upon receiving $(\texttt{submit}, t)$ from $\mathcal{P}$, where $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, output $(\texttt{receipt}, [t])$ to all parties, where $[t]$ is some unique handle onto the polynomial t.
- On input $(\texttt{query}, [t], r)$ from $\mathcal{V}$, where $r \in L^{\ell}$, send $\mathcal{V}$ $(\texttt{result}, t(r_0, \ldots, r_{\ell-1}))$.

An *IOP*, by definition, is a protocol in which $\mathcal{P}$ and $\mathcal{V}$ may make free use of the abstract Functionality 2.5; in a *PIOP*, the parties may instead use Functionality 2.6. *Interactive oracle polynomial commitment schemes* (IOPCSs) serve to bridge these two models. They're IOPs; that is, they operate within the abstract computational model in which Functionality 2.5 is assumed to exist. On the other hand, they "emulate" the more-powerful Functionality 2.6, in the sense that each given PIOP—by inlining in place of each of its calls to Functionality 2.6 an execution of the IOPCS—stands to yield an equivalently secure IOP.

Our approach contrasts with that taken by various previous works (we note e.g. Diamond and Posen [DP23] and Setty [Set20]). Those works opt to define polynomial commitment schemes in the *plain* (random oracle) model, noting that a plain PCS, upon being inlined into a secure PIOP, yields a sound argument. This latter approach absorbs the Merklization process both into the PCS and into the composition theorem. Our approach bypasses this technicality, and separates the relevant concerns. Indeed, given a PIOP, we may first inline our IOPCS into it; on the resulting *IOP*, we may finally invoke generically the compiler of Ben-Sasson, Chiesa and Spooner [BCS16]. This "two-step" compilation process serves to transform any secure PIOP into a secure argument in the random oracle model.

We also define the security of IOPCS differently than do [Set20, Def. 2.11] and [DP23, § 3.5]; our definition is closer to Marlin's [Chi+20, Def. 6.2]. Our definition below requires that $\mathcal{E}$ extract t *immediately* after seeing $\mathcal{A}$'s commitment (that is, before seeing r, or observing any evaluation proofs on the part of $\mathcal{A}$). This work's IOPCS constructions indeed meet this stricter requirement, owing essentially to their use of Reed–Solomon codes, which are efficiently *decodable*. (In the setting of *general*—that is, not-necessarily-decodable—codes, extraction becomes much more complicated, and requires rewinding.) On the other hand, our strict rendition of the *IOPCS* notion makes its key composability property—that is, the fact whereby a secure IOPCS, upon being inlined into a secure PIOP, yields a secure IOP—*easier* to prove. (We believe that this composability property should, on the other hand, nonetheless hold even under various weakenings of Definition 2.9.)

Definition 2.8. An *interactive oracle polynomial commitment scheme* (IOPCS) is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ with the following syntax:

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, \ell)$. On input the security parameter $\lambda \in \mathbb{N}$ and a number-of-variables parameter $\ell \in \mathbb{N}$, outputs params, which includes, among other things, a field L.
- $[f] \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input params and a multilinear polynomial $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, outputs a handle $[f]$ to a vector.
- $b \leftarrow \langle \mathcal{P}([f], s, r; t), \mathcal{V}([f], s, r) \rangle$ is an IOP, in which the parties may jointly leverage the machine $\mathcal{F}^L_{\mathsf{Vec}}$. The parties have as common input a vector handle $[f]$, an evaluation point $(r_0, \ldots, r_{\ell-1}) \in L^\ell$, and a claimed evaluation $s \in L$. $\mathcal{P}$ has as further input a multilinear polynomial $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$. $\mathcal{V}$ outputs a success bit $b \in \{0, 1\}$.

The IOPCS Π is *complete* if the obvious correctness property holds. That is, for each multilinear polynomial $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$ and each honestly generated commitment $[f] \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$, it should hold that, for each $r \in L^\ell$, setting $s := t(r_0, \ldots, r_{\ell-1})$, the honest prover algorithm induces the verifier to accept with probability 1, so that $\langle \mathcal{P}([f], s, r; t), \mathcal{V}([f], s, r) \rangle = 1$.

We now define the security of IOPCSs.

Definition 2.9. For each interactive oracle polynomial commitment scheme Π, security parameter $\lambda \in \mathbb{N}$, and number-of-variables parameter $\ell \in \mathbb{N}$, PPT query sampler $\mathcal{Q}$, PPT adversary $\mathcal{A}$, and PPT emulator $\mathcal{E}$, we define the following experiment:

- The experimenter samples $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, \ell)$, and gives params, including L, to $\mathcal{A}$ and $\mathcal{E}$.
- The adversary, after interacting arbitrarily with the vector oracle, outputs a handle $[f] \leftarrow \mathcal{A}(\mathsf{params})$.
- On input $\mathcal{A}$'s record of interactions with the oracle, $\mathcal{E}$ outputs $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$.
- The query sampler outputs $(r_0, \ldots, r_{\ell-1}) \leftarrow \mathcal{Q}(\mathsf{params})$; $\mathcal{A}$ responds with an evaluation claim $s \leftarrow \mathcal{A}(r)$.
- The experimenter defines the following two random bits:
  - By running the evaluation IOP with $\mathcal{A}$ as $\mathcal{V}$, obtain the bit $b \leftarrow \langle \mathcal{A}(s, r), \mathcal{V}([f], s, r) \rangle$.
  - Obtain the further bit $b' := t(r_0, \ldots, r_{\ell-1}) \stackrel{?}{=} s$.

The IOPCS Π is *secure* if, for each PPT adversary $\mathcal{A}$, there is a PPT emulator $\mathcal{E}$ and a negligible function negl such that, for each $\lambda \in \mathbb{N}$, each $\ell \in \mathbb{N}$, and each PPT query sampler $\mathcal{Q}$, $\Pr[b = 1 \wedge b' = 0] \leq \mathsf{negl}(\lambda)$.

We finally record a variant of Definition 2.8 in which the parties may fix a small coefficient field K.

Definition 2.10. A *small-field interactive oracle polynomial commitment scheme* (small-field IOPCS) is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ with the following syntax:

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, \ell, K)$. On input the security parameter $\lambda \in \mathbb{N}$, a number-of-variables parameter $\ell \in \mathbb{N}$ and a field K, outputs params, which includes, among other things, a field extension $L \,/\, K$.
- $[f] \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input params and a multilinear polynomial $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, outputs a handle $[f]$ to a vector.
- $b \leftarrow \langle \mathcal{P}([f], s, r; t), \mathcal{V}([f], s, r) \rangle$ is an IOP, in which the parties may jointly leverage the machine $\mathcal{F}^L_{\mathsf{Vec}}$. The parties have as common input a vector handle $[f]$, an evaluation point $(r_0, \ldots, r_{\ell-1}) \in L^\ell$, and a claimed evaluation $s \in L$. $\mathcal{P}$ has as further input a multilinear polynomial $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$. $\mathcal{V}$ outputs a success bit $b \in \{0, 1\}$.

We define the *security* of small-field IOPCSs Π exactly as in Definition 2.9, except that we require that $\mathcal{E}$ output a polynomial $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$.

3 Simple Binary-Field IOPCS

In this section, we develop a basic form of our main IOPCS (see Definition 2.8). This section's IOPCS does *not* operate over small fields; its goal is essentially to make BaseFold's IOPCS [ZCF23, § 5] work in the binary setting. To achieve this, we must develop a degree of machinery. Indeed, we must first use FRI's *binary-field* variant [BBHR18]. In the binary setting, in order to recover the even–odd FRI folding behavior essential to BaseFold, we must further specialize binary-field FRI. That is, we must carefully choose that protocol's codeword domains $S^{(i)} \subset L$ (for $i \in \{0, \ldots, l\}$) and its two-to-one collapsing maps $q^{(i)} : S^{(i)} \to S^{(i+1)}$ (for $i \in \{0, \ldots, l-1\}$). Our choice serves to make FRI compatible with Lin, Chung and Han's [LCH14] *additive NTT*.

3.1 Using FRI in Novel Polynomial Basis

We begin by proposing a *specific* construction of those subspace polynomials $q^{(0)}, \ldots, q^{(\ell-1)}$ invoked internally by FRI.

Throughout this section, we fix a binary field L, with $\mathbb{F}_2$-basis $(\beta_0, \ldots, \beta_{r-1})$, say, as well as a size parameter $\ell \in \{0, \ldots, r-1\}$ and a rate parameter $\mathcal{R} \in \{1, \ldots, r-\ell\}$. We finally recall the (non-normalized) subspace vanishing polynomials $W_i(X) \in L[X]$, for $i \in \{0, \ldots, \ell-1\}$, which we now view as $\mathbb{F}_2$-*linear maps* $W_i : L \to L$ (see Subsection 2.3).

Definition 3.1. We initialize $S^{(0)} \coloneqq \langle \beta_0, \ldots, \beta_{\ell+\mathcal{R}-1} \rangle$. Moreover, for each $i \in \{0, \ldots, \ell-1\}$, we set

$$q^{(i)} \coloneqq \frac{W_i(\beta_i)^2}{W_{i+1}(\beta_{i+1})} \cdot X \cdot (X+1),$$

as well as, inductively, $S^{(i+1)} \coloneqq \mathrm{im}\big(q^{(i)}\big|_{S^{(i)}}\big)$.

The following lemma demonstrates that this construction fulfills the template demanded by (1).

Lemma 3.2. *For each $i \in \{0, \ldots, \ell-1\}$, $\ker\big(q^{(i)}\big) \subset S^{(i)}$ holds.*

Proof. We note that, trivially, $\ker\big(q^{(i)}\big) = \{0, 1\}$ for each $i \in \{0, \ldots, \ell-1\}$. For each $i \in \{0, \ldots, \ell-1\}$, we claim in fact that the inductive invariant $S^{(i)} = \mathrm{im}\Big(\widehat{W}_i\Big|_{S^{(0)}}\Big)$ holds. Assuming this invariant, the conclusion of the lemma certainly follows; indeed, we see immediately that $1 = \widehat{W}_i(\beta_i)$, while of course $\beta_i \in S^{(0)}$.

It thus suffices to argue that the inductive invariant holds throughout. In the base case $i = 0$, the claim is a triviality, since $\widehat{W}_0(X) = X$ is the identity. We thus fix an index $i \in \{0, \ldots, \ell-1\}$, and show that the assignment $S^{(i+1)} \coloneqq \mathrm{im}\big(q^{(i)}\big|_{S^{(i)}}\big)$ preserves the inductive invariant; in other words, we must show that $S^{(i+1)} \coloneqq \mathrm{im}\big(q^{(i)}\big|_{S^{(i)}}\big) \stackrel{?}{=} \mathrm{im}\Big(\widehat{W}_{i+1}\Big|_{S^{(0)}}\Big)$. Unrolling the assumed inductive invariant on this equality's left-hand side, we reduce it in turn to the equality $\mathrm{im}\Big(q^{(i)} \circ \widehat{W}_i\Big|_{S^{(0)}}\Big) \stackrel{?}{=} \mathrm{im}\Big(\widehat{W}_{i+1}\Big|_{S^{(0)}}\Big)$. This latter equality itself follows from the following direct calculation:

$$\begin{aligned}
\Big(q^{(i)} \circ \widehat{W}_i\Big)(X) &= \frac{W_i(\beta_i)^2}{W_{i+1}(\beta_i)} \cdot \widehat{W}_i(X) \cdot \Big(\widehat{W}_i(X) + 1\Big) && \text{(by definition of } q^{(i)}.) \\
&= \frac{W_i(\beta_i)^2}{W_{i+1}(\beta_{i+1})} \cdot \frac{W_i(X)}{W_i(\beta_i)} \cdot \frac{W_i(X) + W_i(\beta_i)}{W_i(\beta_i)} && \text{(by definition of } \widehat{W}_i.) \\
&= \frac{W_i(X) \cdot (W_i(X) + W_i(\beta_i))}{W_{i+1}(\beta_{i+1})} && \text{(cancellation of } W_i(\beta_i)^2.) \\
&= \frac{W_{i+1}(X)}{W_{i+1}(\beta_{i+1})} && \text{(recursive characterization of } W_{i+1}(X).) \\
&= \widehat{W}_{i+1}(X) && \text{(by definition of } \widehat{W}_{i+1}(X).)
\end{aligned}$$

in the second-to-last step, we exploit the recursive identity $W_{i+1}(X) = W_i(X) \cdot (W_i(X) + W_i(\beta_i))$, itself a basic consequence of the definitions of W_{i+1} and W_i and of the linearity of W_i. □

Lemma 3.2 shows that the maps $q^{(0)}, \ldots, q^{(\ell-1)}$ and the spaces $S^{(0)}, \ldots, S^{(\ell)}$ yield a *valid* global parameterization, suitable for use in FRI.

We extract and state separately a few corollaries of the proof of Lemma 3.2.

Corollary 3.3. *For each $i \in \{0, \ldots, \ell\}$, $S^{(i)} = \mathrm{im}\Big(\widehat{W}_i\Big|_{S^{(0)}}\Big)$.*

Proof. This fact is shown explicitly in the course of Lemma 3.2. □

As a further side effect, Lemma 3.2 shows that the polynomials $q^{(0)}, \ldots, q^{(\ell-1)}$ collectively "factor" the normalized subspace polynomials $\widehat{W}_0, \ldots, \widehat{W}_{\ell-1}$, in the following sense:

Corollary 3.4. *For each $i \in \{0, \ldots, \ell\}$, $\widehat{W}_i = q^{(i-1)} \circ \cdots \circ q^{(0)}$.*

Proof. This fact admits a simple inductive proof. In the base case $i = 0$, there's nothing to prove (the empty composition is the identity). Letting $i \in \{0, \ldots, \ell - 1\}$ be arbitrary, the proof of Lemma 3.2 shows that $\widehat{W}_{i+1} = q^{(i)} \circ \widehat{W}_i$. Applying induction, we conclude that this latter map in turn equals $q^{(i)} \circ \cdots \circ q^{(0)}$. □

We note finally the following result.

Corollary 3.5. *For each $i \in \{0, \ldots, \ell\}$, the set $\Big(\widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{\ell+\mathcal{R}-1})\Big)$ is an $\mathbb{F}_2$-basis of the space $S^{(i)}$.*

Proof. Indeed, the subspace $V_i \coloneqq \langle \beta_i, \ldots, \beta_{\ell+\mathcal{R}-1} \rangle$ clearly satisfies $V_i \subset S^{(0)}$, so that $\widehat{W}_i(V_i) \subset \widehat{W}_i(S^{(0)})$, which itself equals $S^{(i)}$ (by Corollary 3.3). On the other hand, the restriction of $\widehat{W}_i$ to V_i is necessarily injective, since $\widehat{W}_i$'s kernel $\langle \beta_0, \ldots, \beta_{i-1} \rangle$ intersects V_i trivially. Since $S^{(i)}$ is $\ell + \mathcal{R} - i$-dimensional, we conclude by a dimension count that $\Big(\widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{\ell+\mathcal{R}-1})\Big)$ spans $S^{(i)}$. □

The bases $\Big\langle \widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{\ell+\mathcal{R}-1}) \Big\rangle = S^{(i)}$, for $i \in \{0, \ldots, \ell\}$, allow us to simplify various aspects of our protocol's implementation. For example, expressed in coordinates with respect to these bases, each map $q^{(i)} : S^{(i)} \to S^{(i+1)}$ acts simply by projecting away its 0^{th}-indexed component (indeed, for each $i \in \{0, \ldots, \ell - 1\}$, $q^{(i)}$ maps $(\widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{\ell+\mathcal{R}-1}))$ to $(0, \widehat{W}_{i+1}(\beta_{i+1}), \ldots, \widehat{W}_{i+1}(\beta_{\ell+\mathcal{R}-1}))$). Similarly, for each $i \in \{0, \ldots, \ell - 1\}$ and each $y \in S^{(i+1)}$, the two L-elements $x \in S^{(i)}$ for which $q^{(i)}(x) = y$ differ precisely at their 0^{th} components, and elsewhere agree with y's coordinate representation. Below, we often identify $S^{(i)} \cong \mathcal{B}_{\ell+\mathcal{R}-i}$ as sets, using these bases; moreover, where possible, we eliminate altogether the maps $q^{(0)}, \ldots, q^{(\ell-1)}$ from our descriptions. These measures make our protocol's description and implementation more transparent.

3.2 FRI Folding, Revisited

We now introduce a new FRI-like folding mechanism. We recall that FRI [BBHR18, § 3.2] makes use of a folding arity constant η. FRI stipulates that, to fold a given oracle, the prover interpolate a *univariate* polynomial of degree less than 2^η on each coset of the given oracle, and finally evaluate the resulting polynomials collectively at the verifier's challenge point. We introduce a new, *multilinear* folding mechanism as follows. Informally, we stipulate that the verifier send a fixed and positive—and yet arbitrary—number ϑ of folding challenges, and that the prover fold its oracle, again coset-wise, using a length-2^ϑ *tensor combination* (in the sense of Subsection 2.1) of the verifier's challenges over each coset. Below, we again write L for a binary field.

Definition 3.6. We fix an index $i \in \{0, \ldots, \ell - 1\}$ and a map $f^{(i)} : S^{(i)} \to L$. For each $r \in L$, we define the map $\mathsf{fold}\big(f^{(i)}, r\big) : S^{(i+1)} \to L$ by setting, for each $y \in S^{(i+1)}$:

$$\mathsf{fold}\Big(f^{(i)}, r\Big) : y \mapsto \begin{bmatrix} 1 - r & r \end{bmatrix} \cdot \begin{bmatrix} x_1 & -x_0 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} f^{(i)}(x_0) \\ f^{(i)}(x_1) \end{bmatrix},$$

where we write $(x_0, x_1) \coloneqq {q^{(i)}}^{-1}(\{y\})$ for the fiber of $q^{(i)}$ over $y \in S^{(i+1)}$.

Remark 3.7. Definition 3.6's quantity $\mathsf{fold}\big(f^{(i)}, r\big)(y)$ is closely related—and yet *not* equivalent—to FRI's expression $\mathsf{interpolant}\Big(f^{(i)}\big|_{q^{(i)^{-1}}(\{y\})}\Big)(r)$. (FRI's variant, however, admits a similar matrix expression.) The essential point is that FRI's variant induces a *monomial* fold, as opposed to a Lagrange fold; that is, if we were to use FRI's variant instead of our own, then our Lemma 3.13 below would remain true, albeit with the alternate conclusion $P^{(i+1)}(X) = \sum_{j=0}^{2^{\ell-i-1}-1}(a_{2j} + r'_i \cdot a_{2j+1}) \cdot X_j^{(i+1)}(X)$. Our entire theory admits a parallel variant in this latter setting, though that variant introduces further complications.

We finally record the following iterated extension of Definition 3.6.

Definition 3.8. We fix a positive folding factor ϑ, an index $i \in \{0, \ldots, \ell - \vartheta\}$, and a map $f^{(i)} : S^{(i)} \to L$. For each tuple $(r_0, \ldots, r_{\vartheta-1}) \in L^\vartheta$, we abbreviate $\mathsf{fold}\big(f^{(i)}, r_0, \ldots, r_{\vartheta-1}\big) \coloneqq \mathsf{fold}\big(\cdots \mathsf{fold}\big(f^{(i)}, r_0\big), \cdots, r_{\vartheta-1}\big)$.

We have the following mathematical characterization of this iterated folding operation:

Lemma 3.9. *For each positive folding factor ϑ, each index $i \in \{0, \ldots, \ell - \vartheta\}$, and each $y \in S^{(i+\vartheta)}$, there is a $2^\vartheta \times 2^\vartheta$ invertible matrix M_y,which depends only on $y \in S^{(i+\vartheta)}$, such that, for each function $f^{(i)} : S^{(i)} \to L$ and each tuple $(r_0 \ldots, r_{\vartheta-1}) \in L^\vartheta$ of folding challenges, we have the matrix identity:*

$$\mathsf{fold}\Big(f^{(i)}, r_0, \ldots, r_{\vartheta-1}\Big)(y) = \begin{bmatrix} \quad \bigotimes_{j=0}^{\vartheta-1}(1 - r_j, r_j) \quad \end{bmatrix} \cdot \begin{bmatrix} \qquad M_y \qquad \end{bmatrix} \cdot \begin{bmatrix} f^{(i)}(x_0) \\ \vdots \\ f^{(i)}(x_{2^\vartheta-1}) \end{bmatrix},$$

where the right-hand vector's values $(x_0, \ldots, x_{2^\vartheta-1})$ represent the fiber $\big(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{y\}) \subset S^{(i)}$.

Proof. We prove the result by induction on ϑ. In the base case $\vartheta = 1$, the claim is a tautology, in view of Definition 3.6. We note that that definition's matrix $\begin{bmatrix} x_1 & -x_0 \\ -1 & 1 \end{bmatrix}$ is invertible, since its determinant $x_1 - x_0$ is nonzero (and in fact equals 1, a fact we shall use below).

We thus fix a folding factor $\vartheta > 1$, and suppose that the claim holds for $\vartheta - 1$. We write $(z_0, z_1) \coloneqq q^{(i+\vartheta-1)^{-1}}(\{y\})$, as well as $(x_0, \ldots, x_{2^\vartheta-1}) \coloneqq \big(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{y\})$. Unwinding Definition 3.8, we recursively express the relevant quantity $\mathsf{fold}\big(f^{(i)}, r_0, \ldots, r_{\vartheta-1}\big)(y)$—which, for typographical reasons, we call $\mathfrak{f}$—in the following way:

$$\begin{aligned}
\mathfrak{f} &= \begin{bmatrix} 1 - r_{\vartheta-1} & r_{\vartheta-1} \end{bmatrix} \cdot \begin{bmatrix} z_1 & -z_0 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathsf{fold}\big(f^{(i)}, r_0, \ldots, r_{\vartheta-2}\big)(z_0) \\ \mathsf{fold}\big(f^{(i)}, r_0, \ldots, r_{\vartheta-2}\big)(z_1) \end{bmatrix} \\
&= \begin{bmatrix} 1 - r_{\vartheta-1} & r_{\vartheta-1} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} z_1 & -z_0 \\ -1 & 1 \end{bmatrix} \cdot \left[\begin{array}{c|c} \bigotimes_{j=0}^{\vartheta-2}(1 - r_j, r_j) & \\ \hline & \bigotimes_{j=0}^{\vartheta-2}(1 - r_j, r_j) \end{array}\right]}_{\text{these matrices may be interchanged.}} \cdot \left[\begin{array}{c|c} M_{z_0} & \\ \hline & M_{z_1} \end{array}\right] \cdot \begin{bmatrix} f^{(i)}(x_0) \\ \vdots \\ f^{(i)}(x_{2^\vartheta-1}) \end{bmatrix}.
\end{aligned}$$

In the second step above, we apply the inductive hypothesis on both z_0 and z_1. That hypothesis furnishes the nonsingular, $2^{\vartheta-1} \times 2^{\vartheta-1}$ matrices M_{z_0} and M_{z_1}; we note moreover that the union of the fibers $\big(q^{(i+\vartheta-2)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{z_0\})$ and $\big(q^{(i+\vartheta-2)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{z_1\})$ is precisely $\big(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{y\})$. Interchanging the two matrices bracketed above, we further reexpress this quantity as:

$$= \begin{bmatrix} 1 - r_{\vartheta-1} & r_{\vartheta-1} \end{bmatrix} \cdot \left[\begin{array}{c|c} \bigotimes_{j=0}^{\vartheta-2}(1 - r_j, r_j) & \\ \hline & \bigotimes_{j=0}^{\vartheta-2}(1 - r_j, r_j) \end{array}\right] \cdot \left[\begin{array}{c|c} \mathsf{diag}(z_1) & \mathsf{diag}(-z_0) \\ \hline \mathsf{diag}(-1) & \mathsf{diag}(1) \end{array}\right] \cdot \left[\begin{array}{c|c} M_{z_0} & \\ \hline & M_{z_1} \end{array}\right] \cdot \begin{bmatrix} f^{(i)}(x_0) \\ \vdots \\ f^{(i)}(x_{2^\vartheta-1}) \end{bmatrix}.$$

By the standard recursive substructure of the tensor product, the product of the left-hand two matrices equals exactly $\bigotimes_{j=0}^{\vartheta-1}(1 - r_j, r_j)$. On the other hand, the product of the two $2^\vartheta \times 2^\vartheta$ nonsingular matrices above is itself nonsingular, and supplies the required $2^\vartheta \times 2^\vartheta$ matrix M_y. □

We emphasize that, in Lemma 3.9, the matrix M_y depends *only* on $y \in S^{(i+\vartheta)}$—and of course on ϑ and $i \in \{0, \ldots, \ell - \vartheta\}$—but *not* on the map $f^{(i)}$ or the folding challenges $(r_0, \ldots, r_{\vartheta-1}) \in L^\vartheta$.

Remark 3.10. Interestingly, the matrix M_y of Lemma 3.9 is nothing other than that of the *inverse* additive NTT [LCH14, § III. C.] on the coset $(x_0, \ldots, x_{2^\vartheta-1})$; i.e., it's the matrix which, on input the evaluations of some polynomial of degree less than 2^ϑ on the set of elements $(x_0, \ldots, x_{2^\vartheta-1})$, returns the coefficients—with respect to the i^{th}-order novel basis (see Remark 3.14 below)—of that polynomial.

3.3 Simple IOPCS

We now present our "simple" IOPCS construction, an instantiation of Definition 2.8 based on BaseFold's IOPCS [ZCF23, § 5] and on the material of our Subsections 3.1 and 3.2 above. In order to present a notationally simpler variant of our protocol, we assume below that $\vartheta \mid \ell$; this requirement is not necessary.

CONSTRUCTION 3.11 (Simple Large-Field IOPCS)**.**
We define $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ as follows.

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, \ell)$. On input 1^λ and ℓ, choose a constant, positive rate parameter $\mathcal{R} \in \mathbb{N}$ and a binary field $L/\mathbb{F}_2$ whose degree r (say) over $\mathbb{F}_2$ satisfies $r = \omega(\log \lambda)$ and $r \geq \ell + \mathcal{R}$. Initialize the vector oracle $\mathcal{F}^L_{\mathsf{Vec}}$. Fix a folding factor $\vartheta \mid \ell$ and a repetition parameter $\gamma = \omega(\log \lambda)$. Fix an arbitrary $\mathbb{F}_2$-basis $(\beta_0, \ldots, \beta_{r-1})$ of L. Write $(X_0(X), \ldots, X_{2^\ell-1}(X))$ for the resulting novel L-basis of $L[X]^{\prec 2^\ell}$, and fix the domains $S^{(0)}, \ldots, S^{(\ell)}$ and the polynomials $q^{(0)}, \ldots, q^{(\ell-1)}$ as in Subsection 3.1. Write $C^{(0)} \subset L^{2^{\ell+\mathcal{R}}}$ for the Reed–Solomon code $\mathsf{RS}_{L,S^{(0)}}[2^{\ell+\mathcal{R}}, 2^l]$.

- $[f] \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, use t's Lagrange coefficients $(t(v))_{v \in \mathcal{B}_\ell}$ as the coefficients, in the novel polynomial basis, of a univariate polynomial $P(X) \coloneqq \sum_{v \in \mathcal{B}_\ell} t(v) \cdot X_{\{v\}}(X)$, say. Using Algorithm 2, compute the Reed–Solomon codeword $f : S^{(0)} \to L$ defined by $f : x \mapsto P(x)$. Submit $(\texttt{submit}, \ell + \mathcal{R}, f)$ to the vector oracle $\mathcal{F}^L_{\mathsf{Vec}}$. Upon receiving $(\texttt{receipt}, \ell + \mathcal{R}, [f])$ from $\mathcal{F}^L_{\mathsf{Vec}}$, output the handle $[f]$.

We define $(\mathcal{P}, \mathcal{V})$ as the following IOP, in which both parties have the common input $[f]$, $s \in L$, and $(r_0, \ldots, r_{\ell-1}) \in L^\ell$, and $\mathcal{P}$ has the further input $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$.

- $\mathcal{P}$ writes $h(X_0, \ldots, X_{\ell-1}) \coloneqq t(X_0, \ldots, X_{\ell-1}) \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, X_0, \ldots, X_{\ell-1})$.

- $\mathcal{P}$ and $\mathcal{V}$ both abbreviate $f^{(0)} \coloneqq f$ and $s_0 \coloneqq s$, and execute the following loop:
  1: **for** $i \in \{0, \ldots, \ell - 1\}$ **do**
  2: $\quad\mathcal{P}$ sends $\mathcal{V}$ the univariate polynomial $h_i(X) \coloneqq \sum_{v \in \mathcal{B}_{\ell-i-1}} h(r'_0, \ldots, r'_{i-1}, X, v_0, \ldots, v_{\ell-i-2})$.
  3: $\quad\mathcal{V}$ requires $s_i \overset{?}{=} h_i(0) + h_i(1)$. $\mathcal{V}$ samples $r'_i \leftarrow L$, sets $s_{i+1} \coloneqq h_i(r'_i)$, and sends $\mathcal{P}$ r'_i.
  4: $\quad\mathcal{P}$ defines $f^{(i+1)} : S^{(i+1)} \to L$ as the function $\mathsf{fold}\big(f^{(i)}, r'_i\big)$ of Definition 3.6.
  5: $\quad$**if** $i + 1 = \ell$ **then** $\mathcal{P}$ sends $c \coloneqq f^{(\ell)}(0, \ldots, 0)$ to $\mathcal{V}$.
  6: $\quad$**else if** $\vartheta \mid i + 1$ **then** $\mathcal{P}$ submits $(\texttt{submit}, \ell + \mathcal{R} - i - 1, f^{(i+1)})$ to the oracle $\mathcal{F}^L_{\mathsf{Vec}}$.

- $\mathcal{V}$ requires $s_\ell \overset{?}{=} c \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, r'_0, \ldots, r'_{\ell-1})$.

- $\mathcal{V}$ executes the following querying procedure:
  1: **for** γ repetitions **do**
  2: $\quad\mathcal{V}$ samples $v \leftarrow \mathcal{B}_{\ell+\mathcal{R}}$ randomly.
  3: $\quad$**for** $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$ (i.e., taking ϑ-sized steps) **do**
  4: $\quad\quad$for each $u \in \mathcal{B}_\vartheta$, $\mathcal{V}$ sends $\big(\texttt{query}, [f^{(i)}], (u_0, \ldots, u_{\vartheta-1}, v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})\big)$ to the oracle.
  5: $\quad\quad$**if** $i > 0$ **then** $\mathcal{V}$ requires $c_i \overset{?}{=} f^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1})$.
  6: $\quad\quad\mathcal{V}$ defines $c_{i+\vartheta} \coloneqq \mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$.
  7: $\quad\mathcal{V}$ requires $c_\ell \overset{?}{=} c$.

In our commitment procedure above, we give meaning to the commitment of f by implicitly identifying $S^{(0)} \cong \mathcal{B}_{\ell+\mathcal{R}}$ as sets (as discussed above); similarly, in the prover's line 6 above, we identify $\mathcal{B}_{\ell+\mathcal{R}-i-1} \cong S^{(i+1)}$. Conversely, in its lines 4 and 6 above, the verifier must implicitly identify the $\mathcal{B}_{\ell+\mathcal{R}-i}$-elements $(u_0, \ldots, u_{\vartheta-1}, v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})_{u \in \mathcal{B}_\vartheta}$ with $S^{(i)}$-elements—and the $\mathcal{B}_{\ell+\mathcal{R}-i-\vartheta}$-element $(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$ with an $S^{(i+\vartheta)}$-element—in order to appropriately apply Definition 3.8. We note that, in line 6, $\mathcal{V}$ has precisely the information it needs to compute $\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$ (namely, the values of $f^{(i)}$ on the fiber $(u_0, \ldots, u_{\vartheta-1}, v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})_{u \in \mathcal{B}_\vartheta} \cong \big(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})\})$.

The completness of Construction 3.11's evaluation IOP is not straightforward. For instance, it is simply not obvious what the folding operation of line 4 does to the *coefficients* of the low-degree polynomial $P^{(i)}(X)$ underlying $f^{(i)}$. (Though our folding operation departs slightly from FRI's—we refer to Remark 3.7 for a discussion of this fact—the conceptual obstacle is essentially the same.) Indeed, the completeness proof of generic FRI [BBHR18, § 4.1.1] tells us that the folded function $f^{(i+1)}$ represents the evaluations of *some* polynomial $P^{(i+1)}(X)$ of appropriate degree on the domain $S^{(i+1)}$. But which one? The proof of [BBHR18, § 4.1.1] fails to *constructively* answer this question, in that it invokes the generic characteristics of the multivariate reduction—called $Q^{(i)}(X, Y)$—of $P^{(i)}(X)$ by $Y - q^{(i)}(X)$. (We refer to e.g. von zur Gathen and Gerhard [GG13, Alg. 21.11] for a thorough treatment of multivariate division.) It seems simply infeasible to analyze by hand the execution of the multivariate division algorithm with sufficient fidelity as to determine with any precision the result $P^{(i+1)}(Y) = Q^{(i)}(r'_i, Y)$ (though we don't rule out that a proof could in principle be achieved by this means).

Instead, we introduce certain, carefully-selected L-bases of the spaces $L[X]^{\prec 2^{\ell-i}}$, for $i \in \{0, \ldots, \ell\}$ (so-called "higher-order" novel polynomial bases). As it turns out, the respective coefficients of $P^{(i)}(X)$ and $P^{(i+1)}(X)$ *with respect to these bases* are tractably related; their relationship amounts to an even–odd tensor-fold by the FRI challenge r'_i. Proceeding by induction, we obtain the desired characterization of c.

Theorem 3.12. *The IOPCS $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ of Construction 3.11 is complete.*

Proof. Provided that $\mathcal{P}$ is honest, $s = t(r_0, \ldots, r_{\ell-1})$ will hold. Since $t(r_0 \ldots, r_{\ell-1}) = \sum_{v \in \mathcal{B}_\ell} h(v)$, this guarantee in turn implies that $s = s_0 = \sum_{v \in \mathcal{B}_\ell} h(v)$ will hold, so that, by the completeness of the sumcheck, $\mathcal{V}$'s checks $s_i \stackrel{?}{=} h_i(0) + h_i(1)$ will pass. Finally, $s_\ell = h(r'_0, \ldots, r'_{\ell-1}) = t(r'_0, \ldots, r'_{\ell-1}) \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, r'_0, \ldots, r'_{l-1})$ too will hold. To argue the completeness of $\mathcal{V}$'s check $s_\ell \stackrel{?}{=} c \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, r'_0, \ldots, r'_{\ell-1})$ above, it thus suffices to argue that, for $\mathcal{P}$ honest, $c = t(r'_0, \ldots, r'_{\ell-1})$ will hold.

We introduce a family of further polynomial bases. For each $i \in \{0, \ldots, \ell-1\}$, we define the *$i^{th}$-order subspace vanishing polynomials* $\widehat{W}_0^{(i)}, \ldots, \widehat{W}_{\ell-i-1}^{(i)}$ as the polynomials $X, q^{(i)}, q^{(i+1)} \circ q^{(i)}, \ldots, q^{(\ell-2)} \circ \cdots \circ q^{(i)}$, respectively (that is, $\widehat{W}_k^{(i)} \coloneqq q^{(i+k-1)} \circ \cdots \circ q^{(i)}$, for each $k \in \{0, \ldots, \ell-i-1\}$). Finally, we define the *i^{th}-order novel polynomial basis* by setting $X_j^{(i)} \coloneqq \prod_{k=0}^{\ell-i-1} \widehat{W}_k^{(i)^{j_k}}$, for each $j \in \{0, \ldots, 2^{\ell-i}-1\}$ (here, again, we write $(j_0, \ldots, j_{\ell-i-1})$ for the bits of j). We adopt the notational convention whereby the ℓ^{th} basis consists simply of the constant polynomial $X_0^{(\ell)}(X) = 1$. Our lemma below relies on the following inductive relationship between the bases $\left(X_j^{(i)}(X)\right)_{j=0}^{2^{\ell-i}-1}$ and $\left(X_j^{(i+1)}(X)\right)_{j=0}^{2^{\ell-i-1}-1}$. Indeed, for each $j \in \{0, \ldots, 2^{\ell-i-1}-1\}$, the polynomials $X_{2j}^{(i)}(X)$ and $X_{2j+1}^{(i)}(X)$ respectively equal $X_j^{(i+1)}\big(q^{(i)}(X)\big)$ and $X \cdot X_j^{(i+1)}\big(q^{(i)}(X)\big)$.

Lemma 3.13. *Fix an index $i \in \{0, \ldots, \ell-1\}$. If $f^{(i)} : S^{(i)} \to L$ is exactly the evaluation over $S^{(i)}$ of the polynomial $P^{(i)}(X) = \sum_{j=0}^{2^{\ell-i}-1} a_j \cdot X_j^{(i)}(X)$, then, under honest prover behavior, $f^{(i+1)} : S^{(i+1)} \to L$ is exactly the evaluation over $S^{(i+1)}$ of the polynomial $P^{(i+1)}(X) = \sum_{j=0}^{2^{\ell-i-1}-1} ((1-r'_i) \cdot a_{2j} + r'_i \cdot a_{2j+1}) \cdot X_j^{(i+1)}(X)$.*

Proof. Given $P^{(i)}(X)$ as in the hypothesis of the lemma, we introduce the *even and odd refinements* $P_0^{(i+1)}(X) \coloneqq \sum_{j=0}^{2^{\ell-i-1}-1} a_{2j} \cdot X_j^{(i+1)}(X)$ and $P_1^{(i+1)}(X) \coloneqq \sum_{j=0}^{2^{\ell-i-1}-1} a_{2j+1} \cdot X_j^{(i+1)}(X)$ of $P^{(i)}(X)$. We note the following key polynomial identity:

$$P^{(i)}(X) = P_0^{(i+1)}(q^{(i)}(X)) + X \cdot P_1^{(i+1)}(q^{(i)}(X)); \tag{2}$$

This identity is a direct consequence of the definitions of the higher-order novel polynomial bases.

We turn to the proof of the lemma. We claim that $f^{(i+1)}(y) = P^{(i+1)}(y)$ holds for each $y \in S^{(i+1)}$, where $P^{(i+1)}(X)$ is as in the lemma's hypothesis. To this end, we let $y \in S^{(i+1)}$ be arbitrary; we moreover write $(x_0, x_1) \coloneqq {q^{(i)}}^{-1}(\{y\})$ for the fiber of $q^{(i)}$ over y. We begin by examining the values $P^{(i)}(x_0)$ and $P^{(i)}(x_1)$. For each $b \in \{0,1\}$ we have:

$$\begin{aligned} P^{(i)}(x_b) &= P_0^{(i+1)}\Big(q^{(i)}(x_b)\Big) + x_b \cdot P_1^{(i+1)}\Big(q^{(i)}(x_b)\Big) && \text{(by the identity (2).)} \\ &= P_0^{(i+1)}(y) + x_b \cdot P_1^{(i+1)}(y). && \text{(using } q^{(i)}(x_b) = y.\text{)} \end{aligned}$$

Using now our assumption whereby $f^{(i)}(x_b) = P^{(i)}(x_b)$ for each $b \in \{0,1\}$, and unwinding the prescription of Definition 3.6, we obtain:

$$\begin{aligned} f^{(i+1)}(y) &= \begin{bmatrix} 1 - r_i' & r_i' \end{bmatrix} \cdot \begin{bmatrix} x_1 & -x_0 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} P^{(i)}(x_0) \\ P^{(i)}(x_1) \end{bmatrix} && \text{(by our hypothesis on } f^{(i)}\text{, and by Definition 3.6.)} \\ &= \begin{bmatrix} 1 - r_i' & r_i' \end{bmatrix} \cdot \begin{bmatrix} x_1 & -x_0 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} \cdot \begin{bmatrix} P_0^{(i+1)}(y) \\ P_1^{(i+1)}(y) \end{bmatrix} && \text{(by the calculation just performed above.)} \\ &= \begin{bmatrix} 1 - r_i' & r_i' \end{bmatrix} \cdot \begin{bmatrix} P_0^{(i+1)}(y) \\ P_1^{(i+1)}(y) \end{bmatrix} && \text{(cancellation of inverse matrices.)} \\ &= P^{(i+1)}(y). && \text{(by the definitions of } P_0^{(i+1)}(X)\text{, } P_1^{(i+1)}(X)\text{, and } P^{(i+1)}(X)\text{.)} \end{aligned}$$

To achieve the third equality above, we note that the matrices $\begin{bmatrix} x_1 & -x_0 \\ -1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix}$ are inverses; there, we use the guarantee $x_1 - x_0 = 1$, a basic consequence of Definition 3.1 (or rather of $\ker\big(q^{(i)}\big) = \{0,1\}$). □

Applying Corollary 3.4, we note finally that $\left(\widehat{W}_k^{(0)}\right)_{k=0}^{\ell-1}$ and $\left(X_j^{(0)}\right)_{j=0}^{2^\ell-1}$ themselves yield precisely the *standard* subspace vanishing and novel basis polynomials, respectively. It follows that in the base case $i = 0$ of Lemma 3.13—and again assuming honest behavior by the prover—we have that $f^{(0)}$ will equal the evaluation over $S^{(0)}$ of $P^{(0)}(X) \coloneqq P(X) = \sum_{v \in \mathcal{B}_\ell} t(v) \cdot X_{\{v\}}^{(0)}(X)$. Applying Lemma 3.13 repeatedly, we conclude by induction that $f^{(\ell)}$ will equal the evaluation over $S^{(\ell)}$ of the constant polynomial $\sum_{v \in \mathcal{B}_\ell} t(v) \cdot \widetilde{\mathsf{eq}}(r_0', \ldots, r_{\ell-1}', v_0, \ldots, v_{\ell-1}) = t(r_0', \ldots, r_{\ell-1}')$, so that $c = t(r_0', \ldots, r_{\ell-1}')$ will hold, as desired.

The completeness of the verifier's query phase is self-evident (and is just as in [BBHR18, § 4.1.1]); we note that $\mathcal{V}$ applies to each oracle $f^{(i)}$ the same folding procedure that $\mathcal{P}$ does. This completes the proof of completeness. □

Remark 3.14. Though it seems inessential to the proof of Theorem 3.12, it is interesting to note that, for each $i \in \{0, \ldots, \ell - 1\}$, the i^{th}-order basis $\left(X_j^{(i)}\right)_{i=0}^{2^{\ell-i}-1}$ is *itself* a novel polynomial basis in its own right, namely that attached to the set of vectors $\Big(\widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{\ell-1})\Big)$. Equivalently, the i^{th}-order subspace vanishing polynomials $\left(\widehat{W}_k^{(i)}\right)_{k=0}^{\ell-i-1}$ are simply the subspace vanishing polynomials attached to this latter set of vectors. Indeed, for each $k \in \{0, \ldots, \ell - i - 1\}$, $\Big\langle \widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{i+k-1}) \Big\rangle \subset \ker\Big(\widehat{W}_k^{(i)}\Big)$ certainly holds, since $\widehat{W}_k^{(i)} \circ \widehat{W}_i = q^{(i+k-1)} \circ \cdots \circ q^{(i)} \circ \widehat{W}_i = \widehat{W}_{i+k}$, which annihilates $\langle \beta_0, \ldots, \beta_{i+k-1} \rangle$ (here, we use the definition of $\widehat{W}_k^{(i)}$ and Corollary 3.4). On the other hand, $\widehat{W}_k^{(i)} = q^{(i+k-1)} \circ \cdots \circ q^{(i)}$'s kernel can be of dimension at most k (say by degree considerations), while the vectors $\widehat{W}_i(\beta_i), \ldots, \widehat{W}_i(\beta_{i+k-1})$ are linearly independent (a consequence of Corollary 3.5). We conclude that the above containment is an equality. Finally, the subspace polynomials $\left(\widehat{W}_k^{(i)}\right)_{k=0}^{\ell-i-1}$ are normalized. Indeed, using Corollary 3.4 again, we see that, for each $k \in \{0, \ldots, \ell - i - 1\}$, $\widehat{W}_k^{(i)}\Big(\widehat{W}_i(\beta_{i+k})\Big) = \Big(q^{(i+k-1)} \circ \cdots \circ q^{(i)} \circ \widehat{W}_i\Big)(\beta_{i+k}) = \widehat{W}_{i+k}(\beta_{i+k}) = 1$.

Remark 3.15. Using the techniques of Subsection 3.1 and of Theorem 3.12 above, we are able to suggest a new explanation of the additive NTT algorithm of Lin, Chung and Han [LCH14, § III.], and of its correctness; we note also our Algorithm 2 above. (We refer finally to Li, et al. [Li+18, Alg. 2] for a further perspective.) We fix an index $i \in \{0, \ldots, \ell-1\}$ and a polynomial $P^{(i)}(X) \coloneqq \sum_{j=0}^{2^{\ell-i}-1} a_j \cdot X_j^{(i)}(X)$, expressed with respect to the *i^{th}-order* novel basis. The key idea is that the values of $P^{(i)}(X)$ on the domain $S^{(i)}$ can be derived—using only $\Theta\big(2^{\ell+\mathcal{R}-i}\big)$ K-operations—given the values of $P^{(i)}(X)$'s even and odd refinements $P_0^{(i+1)}(X)$ and $P_1^{(i+1)}(X)$ (as in the proof of Lemma 3.13) over the domain $S^{(i+1)}$. This is a direct consequence of the identity (2) above. Indeed, applying that identity, we see that, for $y \in S^{(i+1)}$ arbitrary, with fiber $(x_0, x_1) \coloneqq {q^{(i)}}^{-1}(\{y\})$, say, we have the equalities $P^{(i)}(x_0) \coloneqq P_0^{(i+1)}(y) + x_0 \cdot P_1^{(i+1)}(y)$ and $P^{(i)}(x_1) \coloneqq P_0^{(i+1)}(y) + x_1 \cdot P_1^{(i+1)}(y)$. Since x_0 and x_1 in fact differ by exactly 1, we see that $P^{(i)}(x_1)$ can be computed from $P^{(i)}(x_0)$ using a single further K-addition. We recover the key butterfly diagram of [LCH14, Fig. 1. (a)] (see also Algorithm 2 above) upon carrying out this procedure recursively, with the convention whereby we flatten (using the space's canonical basis) and *interleave* the two copies of $S^{(i+1)}$ at each instance. The base case of the recursion consists of the 2^ℓ-fold interleaving of the domain $S^{(\ell)}$, into which $P^{(0)}$'s coefficients are tiled $2^{\mathcal{R}}$ times. The final stage of the butterfly diagram yields the desired evaluation of $P^{(0)}(X)$ on $S^{(0)}$. Algorithm 2's twiddle factors in its i^{th} stage, then, are nothing other than the respective first lifts x_0 of y, as the image $y = q^{(i)}(x_0)$ *varies* throughout $S^{(i+1)}$. These latter elements x_0, in turn, take precisely the form $\sum_{k=0}^{\ell+\mathcal{R}-i-2} u_k \cdot \widehat{W}_i(\beta_{i+1+k})$, for $u \in \mathcal{B}_{\ell+\mathcal{R}-i-1} \cong S^{(i+1)}$ arbitrary; indeed, we suppress throughout the 0^{th} canonical basis element $\widehat{W}_i(\beta_i) = 1$ of $S^{(i)}$, since that element is subsumed into each butterfly. We find it interesting that the *same* polynomial identity underlies both the correctness of [LCH14, § III.] and our above analysis of FRI's folding.

We now prove the security of Construction 3.11. Our key technical results below (see Propositions 3.20 and 3.23), essentially, jointly constitute a variant of FRI's soundness statement [BBHR18, § 4.2.2]. Our proofs of these results incorporate—in an attenuated form—various ideas present in [BBHR18, § 4.2.2] and [Ben+23, § 8.2]. We also introduce a number of new ideas, which, by and large, pertain to our new folding technique (see Subsection 3.2).

Theorem 3.16. *The IOPCS $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ of Construction 3.11 is secure (see Definition 2.9).*

Proof. We define a straight-line emulator $\mathcal{E}$ as follows.

1. By inspecting $\mathcal{A}$'s messages to the vector oracle, $\mathcal{E}$ immediately recovers the function $f : S^{(0)} \to L$ underlying the handle $[f]$ output by $\mathcal{A}$.
2. $\mathcal{E}$ runs the Berlekamp–Welch decoder (i.e., Algorithm 1) on the word $f : S^{(0)} \to L$. If that algorithm outputs $P(X) = \perp$ or if $\deg(P(X)) \geq 2^\ell$, then $\mathcal{E}$ sets $t(X_0, \ldots, X_{\ell-1}) \coloneqq 0$. Otherwise, $\mathcal{E}$ expresses $P(X) = \sum_{v \in \mathcal{B}_\ell} t(v) \cdot X_{\{v\}}(X)$ in coordinates with respect to the novel polynomial basis. $\mathcal{E}$ writes $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$ for the multilinear whose Lagrange coordinates are $\big(t(v)\big)_{v \in \mathcal{B}_\ell}$.
3. $\mathcal{E}$ outputs $t(X_0, \ldots, X_{\ell-1})$ and terminates.

We now argue that $\mathcal{E}$ fulfills the requirements of Definition 2.9 with respect to the protocol Π. We write $(r_0, \ldots, r_{\ell-1}) \in L^\ell$ for the evaluation point output by $\mathcal{Q}$ and $s \in L$ for $\mathcal{A}$'s response. We must show that the probability with which $s \neq t(r_0, \ldots, r_{\ell-1})$ *and* $\mathcal{V}$ accepts is negligible. It suffices to assume that $s \neq t(r_0, \ldots, r_{\ell-1})$, and to argue conditionally that $\mathcal{V}$ accepts with negligible probability. We thus assume that $s \neq t(r_0, \ldots, r_{\ell-1})$.

As in Construction 3.11, we write $h(X_0, \ldots, X_{\ell-1}) \coloneqq t(X_0, \ldots, X_{\ell-1}) \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, X_0, \ldots, X_{\ell-1})$. Since, as before, $t(r_0, \ldots, r_{\ell-1}) = \sum_{v \in \mathcal{B}_\ell} h(v)$, our assumption $s \neq t(r_0, \ldots, r_{\ell-1})$ amounts to the condition $s \neq \sum_{v \in \mathcal{B}_\ell} h(v)$. The soundness analysis of the sumcheck (we refer to Thaler [Tha22, § 4.1]) states that, under this very assumption, the probability that the verifier accepts its checks $s_i \stackrel{?}{=} h_i(0) + h_i(1)$ *and* $s_\ell = h(r'_0, \ldots, r'_{\ell-1})$ holds is at most $\frac{2 \cdot \ell}{|L|}$ over $\mathcal{V}$'s choice of its folding challenges $(r'_0, \ldots, r'_{\ell-1})$. We thus assume that $s_\ell \neq h(r'_0, \ldots, r'_{\ell-\kappa-1}) = t(r'_0, \ldots, r'_{\ell-1}) \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, r'_0, \ldots, r'_{\ell-1})$.

This assumption implies that $\mathcal{V}$ will *reject* its check $s_\ell \stackrel{?}{=} c \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\ell-1}, r'_0, \ldots, r'_{\ell-1})$ so long as $c = t(r'_0, \ldots, r'_{\ell-1})$ holds. We argue that, conditioned on $s \neq t(r_0, \ldots, r_{\ell-1})$ as above *and* on $c = t(r'_0, \ldots, r'_{\ell-1})$, $\mathcal{V}$ will accept its FRI-querying phase with negligible probability. To this end, we turn to FRI.

We begin by defining various notions, adapting [BBHR18, § 4.2.1]. For each $i \in \{0, \vartheta, \ldots, \ell\}$ (i.e., ascending in ϑ-sized steps), we write $C^{(i)} \subset L^{2^{\ell+\mathcal{R}-i}}$ for the Reed–Solomon code $\mathsf{RS}_{L,S^{(i)}}[2^{\ell+\mathcal{R}-i}, 2^{\ell-i}]$. We recall that $C^{(i)}$ is of distance $d_i \coloneqq 2^{\ell+\mathcal{R}-i} - 2^{\ell-i} + 1$. We write $f^{(0)}, f^{(\vartheta)}, \ldots, f^{(\ell-\vartheta)}$ for the oracles committed by $\mathcal{A}$; we moreover write $f^{(\ell)} : S^{(\ell)} \to L$ for the identically-c function (here, $c \in L$ is $\mathcal{A}$'s final FRI message). For each $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$, we write $\Delta\big(f^{(i+\vartheta)}, g^{(i+\vartheta)}\big) \subset S^{(i+\vartheta)}$ for the *disagreement* set between the elements $f^{(i+\vartheta)}$ and $g^{(i+\vartheta)}$ of $L^{2^{\ell+\mathcal{R}-i-\vartheta}}$; that is, $\Delta\big(f^{(i+\vartheta)}, g^{(i+\vartheta)}\big)$ is the set of elements $y \in S^{(i+\vartheta)}$ for which $f^{(i+\vartheta)}(y) \neq g^{(i+\vartheta)}(y)$. We moreover write $\Delta^{(i)}\big(f^{(i)}, g^{(i)}\big) \subset S^{(i+\vartheta)}$ for the *fiber-wise disagreement set* of the elements $f^{(i)}$ and $g^{(i)}$ of $L^{2^{\ell+\mathcal{R}-i}}$. That is, $\Delta^{(i)}\big(f^{(i)}, g^{(i)}\big) \subset S^{(i+\vartheta)}$ denotes the set of elements $y \in S^{(i+\vartheta)}$ for which the respective restrictions of $f^{(i)}$ and $g^{(i)}$ to the fiber $\big(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{y\}) \subset S^{(i)}$ are not identically equal. We define $d^{(i)}\big(f^{(i)}, C^{(i)}\big) \coloneqq \min_{g^{(i)} \in C^{(i)}} \big|\Delta^{(i)}\big(f^{(i)}, g^{(i)}\big)\big|$. We note that, if $d^{(i)}\big(f^{(i)}, C^{(i)}\big) < \frac{d_{i+\vartheta}}{2}$, then $d\big(f^{(i)}, C^{(i)}\big) < \frac{d_i}{2}$ *a fortiori* holds. (Each offending fiber contributes at most 2^ϑ errors; on the other hand, $2^\vartheta \cdot \left\lfloor \frac{d_{i+\vartheta}-1}{2} \right\rfloor \leq \left\lfloor \frac{d_i - 1}{2} \right\rfloor$.) In any case, in case the oracle $f^{(i)} : S^{(i)} \to L$ is such that $d\big(f^{(i)}, L^{(i)}\big) < \frac{d_i}{2}$ happens to hold, we write $\overline{f}^{(i)} \in L^{(i)}$ for the unique codeword for which $d\big(f^{(i)}, \overline{f}^{(i)}\big) < \frac{d_i}{2}$.

We record the following key compliance condition:

Definition 3.17. For each index $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$, we say that $\mathcal{A}$'s i^{th} oracle $f^{(i)}$ is *compliant* if the conditions $d^{(i)}\big(f^{(i)}, C^{(i)}\big) < \frac{d_i}{2}$, $d\big(f^{(i+\vartheta)}, C^{(i+\vartheta)}\big) < \frac{d_{i+\vartheta}}{2}$, and $\overline{f}^{(i+\vartheta)} = \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)$ all hold.

We first argue that if $\mathcal{A}$ submits oracles which *are* all compliant—and assuming, as usual, that $s \neq t(r_0, \ldots, r_{\ell-1})$—then $\mathcal{V}$ will reject. Indeed, under this assumption, we note first that $d\big(f^{(0)}, C^{(0)}\big) < \frac{d_0}{2}$ will hold. We see that Algorithm 1 will terminate successfully in step 2 above; we write $t(X_0, \ldots, X_{\ell-1}) \in L[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$ for the polynomial output by $\mathcal{E}$ in that step. We now apply Definition 3.17 repeatedly. In the base case $i = 0$, we note that $\overline{f}^{(0)}$ will be the encoding of $P^{(0)}(X) = \sum_{v \in \mathcal{B}_\ell} t(v) \cdot X^{(0)}_{\{v\}}(X)$, precisely by $\mathcal{E}$'s construction of $(t(v))_{v \in \mathcal{B}_\ell}$. On the other hand, for each $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$, writing $P^{(i)}(X) \in L[X]^{\prec 2^{\ell-i}}$ for the polynomial for which $\mathsf{Enc}(P^{(i)}) = \overline{f}^{(i)}$ holds, our assumption $\overline{f}^{(i+\vartheta)} = \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)$ implies that $\overline{f}^{(i+\vartheta)}$ will be exactly the encoding of that polynomial $P^{(i+\vartheta)}(X) \in L[X]^{\prec 2^{\ell-i-\vartheta}}$ which results from repeatedly applying to $P^{(i)}(X)$ the conclusion of Lemma 3.13 (with the folding challenges $r'_i, \ldots, r'_{i+\vartheta-1}$). Carrying out the induction, we see that $\overline{f}^{(\ell)}$ will itself be identically equal to $\sum_{v \in \mathcal{B}_\ell} t(v) \cdot \widetilde{\mathsf{eq}}\big(r'_0, \ldots, r'_{\ell-1}, v_0, \ldots, v_{\ell-1}\big) = t(r'_0, \ldots, r'_{\ell-1})$, so that $c = t(r'_0, \ldots, r'_{\ell-1})$ will hold. We have just seen above that, in precisely this setting, $\mathcal{V}$ will reject.

It thus suffices to argue that if any among $\mathcal{A}$'s oracles $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$ is *not* compliant, then $\mathcal{V}$ will again reject (except perhaps with negligible probability). This is exactly Proposition 3.23 below. In order to prepare for that proposition, we record a sequence of lemmas. We begin with the following elementary fact.

Lemma 3.18. *For each $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$, if $d\big(f^{(i)}, C^{(i)}\big) < \frac{d_i}{2}$, then, for each tuple of folding challenges $(r'_i, \ldots, r'_{i+\vartheta-1}) \in L^\vartheta$, we have that $\Delta\big(\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big), \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)\big) \subset \Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)$.*

Proof. We proceed by contraposition; we fix an element $y \notin \Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)$. By definition of that latter set, we conclude immediately that the restrictions $f^{(i)}\big|_{\left(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\right)^{-1}(\{y\})} = \overline{f}^{(i)}\big|_{\left(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\right)^{-1}(\{y\})}$ are identically equal. Applying Definition 3.8, we see under this guarantee that, regardless of the challenges $(r'_i, \ldots, r'_{i+\vartheta-1})$, $\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)(y) = \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)(y)$ necessarily also holds. □

We now define a sequence of bad folding events. Our definition of E_i is case-based, and depends on the status of $f^{(i)}$. If $f^{(i)}$ is within the (fiber-wise) unique decoding radius, then E_i captures the event whereby the generic inclusion of Lemma 3.18 becomes *strict*. Otherwise, E_i captures the "bad batching" event whereby $\mathsf{fold}(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1})$ becomes close to $C^{(i+\vartheta)}$.

Definition 3.19. For each $i \in \{0, \vartheta, \ldots, \ell - \vartheta\}$, we define the *bad subset* $E_i \subset L^\vartheta$ as the set of tuples $(r'_i, \ldots, r'_{i+\vartheta-1}) \in L^\vartheta$ for which, as the case may be:

in case $\boldsymbol{d^{(i)}\big(f^{(i)}, C^{(i)}\big) < \frac{d_{i+\vartheta}}{2}}$: $\Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big) \not\subset \Delta\big(\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big), \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)\big)$.

in case $\boldsymbol{d^{(i)}\big(f^{(i)}, C^{(i)}\big) \geq \frac{d_{i+\vartheta}}{2}}$: $d\big(\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big), C^{(i+\vartheta)}\big) < \frac{d_{i+\vartheta}}{2}$.

We now bound the bad subsets E_i of Definition 3.19. We recall that $\mu(E_i) \coloneqq \frac{|E_i|}{|L|^\vartheta}$ denotes the probability mass of the set $E_i \subset L^\vartheta$.

Proposition 3.20. *For each $i \in \{0, \vartheta, \dots, \ell - \vartheta\}$, $\mu(E_i) \leq \vartheta \cdot \frac{\left|S^{(i+\vartheta)}\right|}{|L|}$ holds.*

Proof. We treat separately the two cases of Definition 3.19.

We begin with the first case. We fix an element $y \in \Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)$, we moreover write $E_i^y \subset L^\vartheta$ for the set of tuples $(r'_i, \dots, r'_{i+\vartheta-1}) \in L^\vartheta$ for which $y \notin \Delta\big(\mathsf{fold}\big(f^{(i)}, r'_i, \dots, r'_{i+\vartheta-1}\big), \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \dots, r'_{i+\vartheta-1}\big)\big)$. We argue that $\mu(E_i^y) \leq \frac{\vartheta}{|L|}$. This latter claim suffices to complete the proof of the first case; indeed, since $E_i = \bigcup_{y \in \Delta^{(i)}\left(f^{(i)}, \overline{f}^{(i)}\right)} E_i^y$, assuming the claim, we conclude that $\mu(E_i) \leq \left|\Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)\right| \cdot \frac{\vartheta}{|L|} \leq |S^{(i+\vartheta)}| \cdot \frac{\vartheta}{|L|}$.

For $y \in \Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)$ chosen as above, we apply Lemma 3.9 to the words $f^{(i)}$ and $\overline{f}^{(i)}$. Applying that lemma, we see that $(r'_i, \dots, r'_{i+\vartheta-1}) \in E_i^y$ holds if and only if we have the following matrix identity:

$$0 = \begin{bmatrix} \quad \bigotimes_{j=0}^{\vartheta-1}(1 - r'_{i+j}, r'_{i+j}) \quad \end{bmatrix} \cdot \begin{bmatrix} \\ \quad M_y \quad \\ \\ \end{bmatrix} \cdot \begin{bmatrix} f^{(i)}(x_0) - \overline{f}^{(i)}(x_0) \\ \vdots \\ f^{(i)}(x_{2^\vartheta-1}) - \overline{f}^{(i)}(x_{2^\vartheta-1}) \end{bmatrix}, \tag{3}$$

where we again write $(x_0, \dots, x_{2^\vartheta-1}) \coloneqq \big(q^{(i+\vartheta-1)} \circ \cdots \circ q^{(i)}\big)^{-1}(\{y\})$. Our hypothesis $y \in \Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)$ entails precisely that the right-hand vector of (3) is not identically zero. By Lemma 3.9, M_y is non-singular; we conclude that the image of the right-hand vector of (3) under M_y is likewise not identically zero. Writing $(a_0, \dots, a_{2^\vartheta-1})$ for this latter vector—which, we repeat, is not zero—we conclude that $E_i^y \subset L^\vartheta$ is precisely the vanishing locus in L^ϑ of the ϑ-variate polynomial $s(X_0, \dots, X_{\vartheta-1}) \coloneqq \sum_{v \in \mathcal{B}_\vartheta} a_{\{v\}} \cdot \widetilde{\mathsf{eq}}(X_0, \dots, X_{\vartheta-1}, v_0, \dots, v_{\vartheta-1})$ over L. Since $s(X_0, \dots, X_{\vartheta-1})$'s values on the cube $\{0,1\}^\vartheta \subset L^\vartheta$ are exactly $(a_0, \dots, a_{2^\vartheta-1})$, $s(X_0, \dots, X_{\vartheta-1})$ is certainly not zero. Applying the Schwartz–Zippel lemma to $s(X_0, \dots, X_{\vartheta-1})$, we conclude that the relevant locus $E_i^y \subset L^\vartheta$ is of mass at most $\mu(E_i^y) \leq \frac{\vartheta}{|L|}$, as required.

We turn to the second case of Definition 3.19; in particular, we assume that $d^{(i)}\big(f^{(i)}, C^{(i)}\big) \geq \frac{d_{i+\vartheta}}{2}$. We define an interleaved word $\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$—i.e., a $2^\vartheta \times 2^{\ell+\mathcal{R}-i-\vartheta}$ matrix, with entries in L—in the following way. For each $y \in S^{(i+\vartheta)}$, writing M_y for the matrix guaranteed to exist by Lemma 3.9, we define the column:

$$\begin{bmatrix} f_0^{(i+\vartheta)}(y) \\ \vdots \\ f_{2^\vartheta-1}^{(i+\vartheta)}(y) \end{bmatrix} \coloneqq \begin{bmatrix} \\ \quad M_y \quad \\ \\ \end{bmatrix} \cdot \begin{bmatrix} f^{(i)}(x_0) \\ \vdots \\ f^{(i)}(x_{2^\vartheta-1}) \end{bmatrix}. \tag{4}$$

We note that the resulting $2^\vartheta \times 2^{\ell+\mathcal{R}-i-\vartheta}$ matrix $\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$—i.e., that whose columns are given by the respective left-hand sides of (4), for $y \in S^{(i+\vartheta)}$—satisfies, for each $(r'_i, \dots, r'_{i+\vartheta-1}) \in L^\vartheta$,

$$\mathsf{fold}\Big(f^{(i)}, r'_i, \dots, r'_{i+\vartheta-1}\Big) = \begin{bmatrix} \quad \bigotimes_{j=i}^{i+\vartheta-1}(1 - r'_j, r'_j) \quad \end{bmatrix} \cdot \begin{bmatrix} — & f_0^{(i+\vartheta)} & — \\ & \vdots & \\ — & f_{2^\vartheta-1}^{(i+\vartheta)} & — \end{bmatrix}. \tag{5}$$

Indeed, this is essentially the content of Lemma 3.9, which we apply here jointly to *all* elements $y \in S^{(i+\vartheta)}$.

We claim that the interleaved word $\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$ constructed in this way is far from the interleaved code $C^{(i+\vartheta)^{2^\vartheta}}$.

Lemma 3.21. *Under our hypothesis $d^{(i)}\big(f^{(i)}, C^{(i)}\big) \geq \frac{d_{i+\vartheta}}{2}$, we have $d^{2^\vartheta}\left(\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}, C^{(i+\vartheta)^{2^\vartheta}}\right) \geq \frac{d_{i+\vartheta}}{2}$.*

Proof. We fix an arbitrary interleaved *codeword* $\left(g_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1} \in C^{(i+\vartheta)^{2^\vartheta}}$. We define a "lift" $g^{(i)} \in C^{(i)}$ of $\left(g_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$ in the following way. Writing, for each $j \in \{0,\dots,2^\vartheta-1\}$, $P_j^{(i+\vartheta)}(X) \coloneqq \sum_{k=0}^{2^{\ell-i-\vartheta}-1} a_{j,k} \cdot X_k^{(i+\vartheta)}(X)$ for the polynomial—expressed in coordinates with respect to the $i+\vartheta^{\text{th}}$-order novel polynomial basis—for which $g_j^{(i+\vartheta)} = \mathsf{Enc}(P_j^{(i+\vartheta)})$ holds, we define

$$P^{(i)}(X) \coloneqq \sum_{j=0}^{2^\vartheta-1} \sum_{k=0}^{2^{\ell-i-\vartheta}-1} a_{j,k} \cdot X_{k\cdot 2^\vartheta+j}^{(i)};$$

that is, $P^{(i)}$'s list of i^{th}*-order* coefficients is precisely the 2^ϑ-fold interleaving of the polynomials $P_0^{(i+\vartheta)}(X),\dots,P_{2^\vartheta-1}^{(i+\vartheta)}(X)$'s respective lists of $i+\vartheta^{th}$*-order* coefficients. Finally, we define $g^{(i)} \coloneqq \mathsf{Enc}(P^{(i)})$.

We argue that the codeword $g^{(i)} \in C^{(i)}$ constructed in this way stands in relation to $\left(g_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$ just as $f^{(i)}$ does to $\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$ (i.e., it also satisfies a matrix identity analogous to (4) for each $y \in S^{(i+\vartheta)}$). To prove this, we fix an arbitrary element $y \in S^{(i+\vartheta)}$; we moreover fix a row-index $j \in \{0,\dots,2^\vartheta-1\}$. We write $(j_0,\dots,j_{\vartheta-1})$ for the bits of j (i.e., so that $j = \sum_{k=0}^{\vartheta-1} 2^k \cdot j_k$ holds). We first note that the functions $g_j^{(i+\vartheta)}$ and $\mathsf{fold}\big(g^{(i)}, j_0,\dots,j_{\vartheta-1}\big)$ agree identically over the domain $S^{(i+\vartheta)}$. Indeed, this is a direct consequence of Lemma 3.13 and of the construction of $g^{(i)}$ ($g_j^{(i+\vartheta)}(y)$'s underlying polynomial's coefficients are the j^{th} refinement of $g^{(i)}$'s underlying polynomial's). On the other hand, applying Lemma 3.9 to $y \in S^{(i+\vartheta)}$ and $g^{(i)}$, with the folding tuple $(j_0,\dots,j_{\vartheta-1})$, we see that the dot product between M_y's j^{th} row and $\big(g^{(i)}(x_0),\dots,g^{(i)}(x_{2^\vartheta-1})\big)$ is exactly $\mathsf{fold}\big(g^{(i)}, j_0,\dots,j_{\vartheta-1}\big)(y) = g_j^{(i+\vartheta)}(y)$, where the latter equality was just argued.

Since $g^{(i)} \in C^{(i)}$ is a codeword, our hypothesis $d^{(i)}\big(f^{(i)}, C^{(i)}\big) \geq \frac{d_{i+\vartheta}}{2}$ applies to it. That hypothesis entails precisely that, for *at least* $\frac{d_{i+\vartheta}}{2}$ elements $y \in S^{(i+\vartheta)}$, the restrictions $f^{(i)}\big|_{\left(q^{(i+\vartheta-1)}\circ\cdots\circ q^{(i)}\right)^{-1}(\{y\})}$ and $g^{(i)}\big|_{\left(q^{(i+\vartheta-1)}\circ\cdots\circ q^{(i)}\right)^{-1}(\{y\})}$ are *not* identically equal. For each such $y \in S^{(i+\vartheta)}$, since M_y is nonsingular (and since both $f^{(i)}$ and $g^{(i)}$ satisfy (4)), we conclude that the columns $\left(f_j^{(i+\vartheta)}(y)\right)_{j=0}^{2^\vartheta-1}$ and $\left(g_j^{(i+\vartheta)}(y)\right)_{j=0}^{2^\vartheta-1}$ are in turn unequal. Since $\left(g_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$ was arbitrary, we conclude that $d^{2^\vartheta}\left(\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}, C^{(i+\vartheta)^{2^\vartheta}}\right) \geq \frac{d_{i+\vartheta}}{2}$. □

Applying Lemma 3.21, we conclude directly that the contraposition of Theorem 2.4 is fulfilled with respect to the code $C^{(i+\vartheta)} \subset L^{2^{\ell+\mathcal{R}-i-\vartheta}}$, the proximity parameter $e \coloneqq \left\lfloor \frac{d_{i+\vartheta}-1}{2} \right\rfloor$, and the interleaved word $\left(f_j^{(i+\vartheta)}\right)_{j=0}^{2^\vartheta-1}$. That theorem's contraposition immediately implies that the set $E_i \subset L^\vartheta$ consisting of those tuples $(r'_i,\dots,r'_{i+\vartheta-1}) \in L^\vartheta$ for which $d\big(\mathsf{fold}\big(f^{(i)}, r'_i,\dots,r'_{i+\vartheta-1}\big), C^{(i+\vartheta)}\big) < \frac{d_{i+\vartheta}}{2}$ holds—and here, we use (5)—is of mass at most $\mu(E_i) \leq \vartheta \cdot \frac{2^{\ell+\mathcal{R}-i-\vartheta}}{|L|} = \vartheta \cdot \frac{|S^{(i+\vartheta)}|}{|L|}$, as required. □

Proposition 3.22. *The probability that any among the bad events $E_0, E_\vartheta, \dots, E_{\ell-\vartheta}$ occurs is at most $\frac{2^{\ell+\mathcal{R}}}{|L|}$.*

Proof. Applying Proposition 3.20, we upper-bound the quantity of interest as:

$$\frac{\vartheta}{|L|} \cdot (|S_\vartheta| + \cdots + |S_\ell|) = \frac{\vartheta}{|L|} \cdot \left(2^{\ell+\mathcal{R}-\vartheta} + \cdots + 2^{\mathcal{R}}\right) \leq \frac{\vartheta}{|L|} \cdot \frac{2^\vartheta}{2^\vartheta-1} \cdot 2^{\ell+\mathcal{R}-\vartheta} \leq \frac{2^{\ell+\mathcal{R}}}{|L|},$$

which completes the proof. In the last two steps, we use the geometric series formula and the inequality $\frac{\vartheta}{2^\vartheta-1} \leq 1$ (which holds for each $\vartheta \geq 1$), respectively. □

In light of Proposition 3.22, we freely assume that none of the events $E_0, E_\vartheta, \dots, E_{\ell-\vartheta}$ occurs. Under this assumption, we finally turn to the following key proposition.

Proposition 3.23. *If any of $\mathcal{A}$'s oracles is not compliant, then $\mathcal{V}$ accepts with at most negligible probability.*

Proof. We suppose that at least one of $\mathcal{A}$'s oracles is not compliant; we write $i^* \in \{0, \vartheta, \ldots, \ell - \vartheta\}$ for $\mathcal{A}$'s *highest-indexed* noncompliant oracle.

Lemma 3.24. *For $i^* \in \{0, \vartheta, \ldots, \ell - \vartheta\}$ as above, we have $d\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big), \overline{f}^{(i^*+\vartheta)}\big) \geq \frac{d_{i^*+\vartheta}}{2}$.*

Proof. Assuming first that $d^{(i^*)}\big(f^{(i^*)}, C^{(i^*)}\big) < \frac{d_{i^*+\vartheta}}{2}$, we write $\overline{f}^{(i^*)} \in C^{(i^*)}$ for the codeword for which $\big|\Delta^{(i^*)}\big(f^{(i^*)}, \overline{f}^{(i^*)}\big)\big| < \frac{d_{i^*+\vartheta}}{2}$. We note that $d\big(f^{(i^*)}, \overline{f}^{(i^*)}\big) < \frac{d_{i^*}}{2}$ *a fortiori* holds; by Definition 3.17 and our choice of i^*, we thus must have in turn $\overline{f}^{(i^*+\vartheta)} \neq \mathsf{fold}\big(\overline{f}^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big)$. On the other hand, by Lemma 3.18, $\big|\Delta^{(i^*)}\big(f^{(i^*)}, \overline{f}^{(i^*)}\big)\big| < \frac{d_{i^*+\vartheta}}{2}$ implies that $d\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big), \mathsf{fold}\big(\overline{f}^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big)\big) < \frac{d_{i^*+\vartheta}}{2}$. Finally, by the reverse triangle inequality, $d\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big), \overline{f}^{(i^*+\vartheta)}\big)$ is at least:

$$d\Big(\overline{f}^{(i^*+\vartheta)}, \mathsf{fold}\Big(\overline{f}^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\Big)\Big) - d\Big(\mathsf{fold}\Big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\Big), \mathsf{fold}\Big(\overline{f}^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\Big)\Big).$$

Since $\overline{f}^{(i^*+\vartheta)}$ and $\mathsf{fold}\big(\overline{f}^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big)$ are unequal codewords in $C^{(i^*+\vartheta)}$, this quantity in turn is greater than $d_{i^*+\vartheta} - \frac{d_{i^*+\vartheta}}{2} \geq \frac{d_{i^*+\vartheta}}{2}$, and the proof of the first case is complete.

In the case $d^{(i^*)}\big(f^{(i^*)}, C^{(i^*)}\big) \geq \frac{d_{i^*+\vartheta}}{2}$, our assumption whereby E_{i^*} didn't occur implies, by definition, that $d\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big), C^{(i^*+\vartheta)}\big) \geq \frac{d_{i^*+\vartheta}}{2}$. Since $\overline{f}^{(i^*+\vartheta)} \in C^{(i^*+\vartheta)}$ is a codeword, $d\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big), \overline{f}^{(i^*+\vartheta)}\big) \geq \frac{d_{i^*+\vartheta}}{2}$ in particular holds, and the proof is again complete. □

Lemma 3.25. *Whenever its suffix $(v_{i^*+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1}) \in \Delta\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i+\vartheta-1}\big), \overline{f}^{(i^*+\vartheta)}\big)$, $\mathcal{V}$ rejects.*

Proof. We fix an iteration of the query phase's outer loop for which the lemma's hypothesis holds. We fix an arbitrary index $i \in \{i^*, i^* + \vartheta, \ldots, \ell - \vartheta\}$. If $\mathcal{V}$ rejects before finishing the inner loop 3's i^{th} iteration, then there's nothing to prove. We argue that, conditioned on $\mathcal{V}$ reaching the end of its i^{th} iteration, we have the inductive conclusion $c_{i+\vartheta} \neq \overline{f}^{(i+\vartheta)}(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$ as of the end of that iteration.

In the base case $i = i^*$, $\mathcal{V}$ assigns $c_{i^*+\vartheta} := \mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big)(v_{i^*+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$ inline on line 6. On the other hand, the hypothesis of the lemma is precisely $\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i+\vartheta-1}\big)(v_{i^*+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1}) \neq \overline{f}^{(i^*+\vartheta)}(v_{i^*+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$; we conclude immediately that $c_{i^*+\vartheta} \neq \overline{f}^{(i^*+\vartheta)}(v_{i^*+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$ will hold as of the end of the $i^{*\text{th}}$ iteration, as desired.

We fix an index $i \in \{i^*+\vartheta, \ldots, \ell-\vartheta\}$. As of the beginning of the i^{th} iteration, by induction, we have the hypothesis $c_i \neq \overline{f}^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1})$. If $\overline{f}^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1}) = f^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1})$ moreover holds, then we see immediately that $\mathcal{V}$ will reject on line 5; indeed, in this case $c_i \neq \overline{f}^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1}) = f^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1})$ will hold. We conclude that, conditioned on $\mathcal{V}$ reaching the end of its i^{th} iteration, we necessarily have $\overline{f}^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1}) \neq f^{(i)}(v_i, \ldots, v_{\ell+\mathcal{R}-1})$, or in other words $(v_i, \ldots, v_{\ell+\mathcal{R}-1}) \in \Delta\big(f^{(i)}, \overline{f}^{(i)}\big)$. This guarantee implies *a fortiori* that $(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1}) \in \Delta^{(i)}\big(f^{(i)}, \overline{f}^{(i)}\big)$, by definition of this latter set. Using our assumption whereby the event E_i *didn't* occur, we conclude in turn that $(v_{i+\vartheta}, \ldots, v_{\ell-1}) \in \Delta\big(\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big), \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)\big)$. Since $\overline{f}^{(i+\vartheta)} = \mathsf{fold}\big(\overline{f}^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)$ (a consequence of the maximality of i^*), this latter set itself equals $\Delta\big(\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big), \overline{f}^{(i+\vartheta)}\big)$. We conclude that $\mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1}) \neq \overline{f}^{(i+\vartheta)}(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$, so that, after its assignment on line 6, $\mathcal{V}$ will obtain $c_{i+\vartheta} \neq \overline{f}^{(i+\vartheta)}(v_{i+\vartheta}, \ldots, v_{\ell+\mathcal{R}-1})$, thereby preserving the inductive hypothesis.

Carrying through the induction, we see finally that either $\mathcal{V}$ will abort before finishing its inner loop 3 or else it will have $c_\ell \neq \overline{f}^{(\ell)}(v_\ell, \ldots, v_{\ell+\mathcal{R}-1})$ as of its final check 7. Since $c = \overline{f}^{(\ell)}(v_\ell, \ldots, v_{\ell+\mathcal{R}-1})$ holds identically for each $v \in \mathcal{B}_\mathcal{R}$ (by definition of this latter oracle), we see that $\mathcal{V}$ will reject its check $c_\ell \stackrel{?}{=} c$. □

We return to the proposition. Lemma 3.24 guarantees (i.e., assuming E_{i^*} doesn't occur) that $c_{i^*+\vartheta} \in \Delta\big(\mathsf{fold}\big(f^{(i^*)}, r'_{i^*}, \ldots, r'_{i^*+\vartheta-1}\big), \overline{f}^{(i^*+\vartheta)}\big)$ will hold with probability at least $\frac{1}{|S^{(i^*+\vartheta)}|} \cdot \frac{d_{i^*+\vartheta}}{2} \geq \frac{1}{2} - \frac{1}{2 \cdot 2^\mathcal{R}}$ in each of the verifier's query iterations. By Lemma 3.25, the verifier will reject in each such iteration (i.e., assuming none of the events $E_{i^*+\vartheta}, \ldots, E_{\ell-\vartheta}$ occurs). We see that $\mathcal{V}$ will accept with probability at most $\big(\frac{1}{2} + \frac{1}{2 \cdot 2^\mathcal{R}}\big)^\gamma$, which is negligible (we recall that $\mathcal{R}$ is constant). This completes the proof of the proposition. □

Our proof of Proposition 3.23 completes the proof of the theorem. □

We postpone our analysis of Construction 3.11's efficiency to Section 5, in which we present our unified small-field scheme. For now, we remark that Construction 3.11's commitment phase entails a Reed–Solomon encoding on the part of the prover; using the additive NTT (see Algorithm 2 above), the prover can compute this encoding in $\Theta(\ell \cdot 2^\ell)$ time. Construction 3.11's evaluation phase entails, for both parties, an execution of ℓ-variate FRI *and* of an ℓ-variate sumcheck, both over L. In view of standard algorithms (we refer to [BBHR18, § 4.4] Thaler [Tha22, Lem. 4.5]), these tasks amount collectively to $\Theta(2^\ell)$ L-operations for the prover and $\Theta(\ell)$ L-operations for the verifier. The BCS-compiled [BCS16] variant of the protocol imposes rather $\Theta(\lambda \cdot \log^2(\ell))$ work on the verifier.

4 Ring-Switching

In this section, we introduce a generic small-field-to-large-field compiler, suitable for any finite field K and any extension $L \,/\, K$ of power-of-two degree (not necessarily of characteristic 2). Our reduction technique serves to "bootstrap" any *standard* interactive oracle polynomial commitment scheme over L (in the sense of Definition 2.8) into a *small-field* scheme over K (in the sense of Definition 2.10).

It would certainly be possible, on input a small-field polynomial $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, simply to embed $t(X_0, \ldots, X_{\ell-1})$ along $K \subset L$ and to use a large-field IOPCS generically on the result. This strategy, however, would suffer from at least two defects. (We refer to [DP23] for a thorough discussion of this matter, and merely summarize it here.) For one, it would be inefficient; it would treat each K-element as an L-element, thereby blowing up by a factor of $\deg(L \,/\, K)$ the sizes of the witness data and of the various other internal values at play. Further, it would yield an inadequate security guarantee. Indeed, it would allow the emulator to extract an L-polynomial, as opposed to a K-polynomial; that is, it would fail to guarantee to the verifier that the prover actually committed a polynomial defined over K.

Our approach, in contrast, reduces the problem of committing a K-polynomial to that of committing an L-polynomial of *identical size in bits* (i.e., on fewer variables). Moreover, our K-polynomial evaluation protocol adds to that of the underlying L-polynomial only a small, logarithmic overhead. We informally summarize our approach. We write $\kappa \geq 0$ for the integer for which $2^\kappa = \deg(L \,/\, K)$ holds, and write $\ell' := \ell - \kappa$. We take for granted the contents of Subsection 2.5 above. In particular, we recall the tensor algebra $A := L \otimes_K L$, as well as the two embeddings $\varphi_0 : L \hookrightarrow A$ and $\varphi_1 : L \hookrightarrow A$. We finally fix a polynomial $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, and write $t'(X_0, \ldots, X_{\ell'-1}) \in L[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$ for its *packed polynomial* in the sense of Definition 2.1. During our commitment phase, we stipulate simply that the prover commit to the packed polynomial $t'(X_0, \ldots, X_{\ell'-1})$ over L.

Our evaluation phase is based on two reductions. The first is that, in order to learn $t(r_0, \ldots, r_{\ell-1})$, for some evaluation point $(r_0, \ldots, r_{\ell-1}) \in L^\ell$ given, it suffices—up to a small, $\Theta(2^\kappa)$-time postprocessing step on the part of the verifier—instead to learn $\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$. Here, we again write $\varphi_1(t')(X_0, \ldots, X_{\ell'-1})$ for the coefficientwise *horizontal* embedding of $t'(X_0, \ldots, X_{\ell'-1})$ along $\varphi_1 : L \hookrightarrow A$. That is, we may chunk and *horizontally* embed $t(X_0, \ldots, X_{\ell-1})$'s coordinates, and moreover *vertically* embed all but κ of $(r_0, \ldots, r_{\ell-1})$'s components, and evaluate the resulting thing over A. By dotting the columns of the resulting A-element $\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ with $\bigotimes_{i=0}^{\kappa-1}(1-r_i, r_i)$, $\mathcal{V}$ may learn its desired evaluation $t(r_0, \ldots, r_{\ell-1})$. (We rigorously demonstrate the correctness of this reduction in Theorem 4.2 below.)

Our second reduction begins with the standard multilinear expansion

$$\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1})) = \sum_{v \in \mathcal{B}_{\ell'}} \varphi_1(t')(v_0, \ldots, v_{\ell'-1}) \cdot \widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), v_0, \ldots, v_{\ell'-1}),$$

and observes that—up to an execution of *a special sort of sumcheck*, which we explain presently—the desired evaluation of $\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ may in turn be reduced to that of $\varphi_1(t')(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1}))$, for random elements $(r'_0, \ldots, r'_{\ell'-1}) \in L^{\ell'}$ sampled during the sumcheck. Indeed, our sumcheck is of the polynomial $h(X_0, \ldots, X_{\ell'-1}) := \varphi_1(t')(X_0, \ldots, X_{\ell'-1}) \cdot \widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), X_0, \ldots, X_{\ell'-1}))$ over A. On the other hand, we draw our sumcheck challenges from the *subring* $\varphi_1(L) \subset A$. The essential point is that the *final* reduction target $\varphi_1(t')(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})) = \varphi_1\big(t'(r'_0, \ldots, r'_{\ell'-1})\big)$ of this latter sumcheck "lives" entirely in the subring $\varphi_1(L) \subset A$, which is itself isomorphic to L. To learn $\varphi_1(t')(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1}))$, the parties may thus finally make blackbox use of the large-field scheme over L. (The verifier can compute $\widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1}))$ itself, by performing just $\Theta(\ell')$ local work.)

4.1 Ring-Switching Protocol

We now record our ring-switching reduction.

CONSTRUCTION 4.1 (Ring-Switching Compiler)**.**
A large-field scheme $\Pi' = \big(\mathsf{Setup}', \mathsf{Commit}', \mathcal{P}', \mathcal{V}'\big)$ is given. We define the small-field scheme $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ as follows.

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, \ell, K)$. On input 1^λ, ℓ, and K, run and output $\Pi'.\mathsf{Setup}'(1^\lambda, \ell')$, where ℓ' is such that the field $L \,/\, K$ returned by that routine, of degree 2^κ over K say, satisfies $\ell' = \ell - \kappa$.
- $[f] \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, fix the packed polynomial $t'(X_0, \ldots, X_{\ell'-1}) \in L[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$ as in Definition 2.1; output $\Pi'.\mathsf{Commit}'(\mathsf{params}, t')$.

We define $(\mathcal{P}, \mathcal{V})$ as the following IOP, in which both parties have the common input $[f]$, $s \in L$, and $(r_0, \ldots, r_{\ell-1}) \in L^\ell$, and $\mathcal{P}$ has the further input $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$.

- $\mathcal{P}$ again writes $t'(X_0, \ldots, X_{\ell'-1}) \in L[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$ for $t(X_0, \ldots, X_{\ell-1})$'s packed polynomial; $\mathcal{P}$ moreover sets $h(X_0, \ldots, X_{\ell'-1}) := \varphi_1(t')(X_0, \ldots, X_{\ell'-1}) \cdot \widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), X_0, \ldots, X_{\ell'-1})$.
- $\mathcal{P}$ computes $s_0 := \varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ and sends $\mathcal{V}$ s_0.
- $\mathcal{V}$ writes $(s_{0,v})_{v \in \mathcal{B}_\kappa}$ for s_0's column representation (see Subsection 2.5). $\mathcal{V}$ requires $s \stackrel{?}{=} \sum_{v \in \mathcal{B}_\kappa} s_{0,v} \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\kappa-1}, v_0, \ldots, v_{\kappa-1})$.
- $\mathcal{P}$ and $\mathcal{V}$ execute the following loop:
  1: **for** $i \in \{0, \ldots, \ell' - 1\}$ **do**
  2: $\quad\mathcal{P}$ sends $\mathcal{V}$ the polynomial $h_i(X) := \sum_{w \in \mathcal{B}_{\ell'-i-1}} h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{i-1}), X, w_0, \ldots, w_{\ell'-i-2}\big)$.
  3: $\quad\mathcal{V}$ requires $s_i \stackrel{?}{=} h_i(0) + h_i(1)$. $\mathcal{V}$ samples $r'_i \leftarrow L$, sets $s_{i+1} := h_i(\varphi_1(r'_i))$, and sends $\mathcal{P}$ r'_i.
- $\mathcal{P}$ computes $s' := t'(r'_0, \ldots, r'_{\ell'-1})$ and sends $\mathcal{V}$ s'.
- $\mathcal{V}$ requires $s_{\ell'} \stackrel{?}{=} \varphi_1(s') \cdot \widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$.
- $\mathcal{P}$ and $\mathcal{V}$ engage in the evaluation protocol $b \leftarrow \langle \mathcal{P}'([f], s', r'; t'), \mathcal{V}'([f], s', r') \rangle$; $\mathcal{V}$ outputs b.

In our completeness proof below, we rely on the content of Subsection 2.5. In particular, we use the fact whereby, for $a \in A$ arbitrary, with column representation $(a_v)_{v \in \mathcal{B}_\kappa}$ say, and for $\alpha \in L$ also arbitrary, the column representation of $\varphi_0(\alpha) \cdot a$ is $(\alpha \cdot a_v)_{v \in \mathcal{B}_\kappa}$.

Theorem 4.2. *If* $\Pi' = \big(\mathsf{Setup}', \mathsf{Commit}', \mathcal{P}', \mathcal{V}'\big)$ *is complete, then* $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ *also is.*

Proof. If $\mathcal{P}$ operates as prescribed, then its initial message $s_0 \in A$ will satisfy:

$$s_0 := \varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1})) = \sum_{w \in \mathcal{B}_{\ell'}} \varphi_1(t')(w) \cdot \widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), w_0, \ldots, w_{\ell'-1}). \tag{6}$$

By the definitions of $t'(X_0, \ldots, X_{\ell'-1})$ and of φ_1, for each $w \in \mathcal{B}_{\ell'}$, the A-element $\varphi_1(t')(w)$ has the column representation $\big(t(v_0, \ldots, v_{\kappa-1}, w_0, \ldots, w_{\ell'-1})\big)_{v \in \mathcal{B}_\kappa}$ (these coordinates are *a posteriori* K-elements, not L-elements). On the other hand, for each $w \in \mathcal{B}_{\ell'}$, we have $\widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), w_0, \ldots, w_{\ell'-1}) = \varphi_0(\widetilde{\mathsf{eq}}(r_\kappa, \ldots, r_{\ell-1}, w_0, \ldots, w_{\ell'-1}))$. We thus see that for each $w \in \mathcal{B}_{\ell'}$, the column representation of the w^{th} term $\varphi_0(\widetilde{\mathsf{eq}}(r_\kappa, \ldots, r_{\ell-1}, w_0, \ldots, w_{\ell'-1})) \cdot \varphi_1(t')(w)$ of the sum (6) above is exactly

$$\big(\widetilde{\mathsf{eq}}(r_\kappa, \ldots, r_{\ell-1}, w_0, \ldots, w_{\ell'-1}) \cdot t(v_0, \ldots, v_{\kappa-1}, w_0, \ldots, w_{\ell'-1})\big)_{v \in \mathcal{B}_\kappa}. \tag{7}$$

For each arbitrary $v \in \mathcal{B}_\kappa$, upon adding the respective v^{th} components of (7) across all summands $w \in \mathcal{B}_{\ell'}$ of (6), we obtain:

$$s_{0,v} = \sum_{w \in \mathcal{B}_{\ell'}} \widetilde{\mathsf{eq}}(r_\kappa, \ldots, r_{\ell-1}, w_0, \ldots, w_{\ell'-1}) \cdot t(v_0, \ldots, v_{\kappa-1}, w_0, \ldots, w_{\ell'-1}) = t(v_0, \ldots, v_{\kappa-1}, r_\kappa, \ldots, r_{\ell-1}).$$

We conclude finally that $\mathcal{V}$ will itself compute

$$\sum_{v\in\mathcal{B}_\kappa} s_{0,v}\cdot\widetilde{\mathsf{eq}}(r_0,\dots,r_{\kappa-1},v_0,\dots,v_{\kappa-1}) = \sum_{v\in\mathcal{B}_\kappa} t(v_0,\dots,v_{\kappa-1},r_\kappa,\dots,r_{\ell-1})\cdot\widetilde{\mathsf{eq}}(r_0,\dots,r_{\kappa-1},v_0,\dots,v_{\kappa-1})$$
$$= t(r_0,\dots,r_{\ell-1}),$$

which itself equals s, provided that $\mathcal{P}$'s initial claim $s \stackrel{?}{=} t(r_0,\dots,r_{\ell-1})$ is true. We thus see that $\mathcal{V}$ will accept its check $s \stackrel{?}{=} \sum_{v\in\mathcal{B}_\kappa} s_{0,v}\cdot\widetilde{\mathsf{eq}}(r_0,\dots,r_{\kappa-1},v_0,\dots,v_{\kappa-1})$.

We note moreover that, by construction of $h(X_0,\dots,X_{\ell'-1})$,

$$\sum_{w\in\mathcal{B}_{\ell'}} h(w) = \sum_{w\in\mathcal{B}_{\ell'}} \varphi_1(t')(w)\cdot\widetilde{\mathsf{eq}}(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}),w_0,\dots,w_{\ell'-1}) = \varphi_1(t')(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}))$$

holds. Assuming again that $\mathcal{P}$ computes $s_0 := \varphi_1(t')(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}))$ honestly, we see, by the completeness of the sumcheck, that $\mathcal{V}$ will accept its checks $s_i \stackrel{?}{=} h_i(0)+h_i(1)$. Finally, $\mathcal{V}$ will set $s_{\ell'} := h_{\ell'-1}(r'_{\ell'-1}) = h\big(\varphi_1(r'_0),\dots,\varphi_1(r'_{\ell'-1})\big)$. By definition of this latter function, we see in turn that:

$$s_{\ell'} = \varphi_1(t')(\varphi_1(r'_0),\dots,\varphi_1(r'_{\ell'-1}))\cdot\widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}),\varphi_1(r'_0),\dots,\varphi_1(r'_{\ell'-1})\big)$$

will hold. Since $\varphi_1(s') = \varphi_1(t'(r'_0,\dots,r'_{\ell'-1})) = \varphi_1(t')(\varphi_1(r'_0),\dots,\varphi_1(r'_{\ell'-1}))$ will further hold for $\mathcal{P}$ honest, we conclude finally that

$$s_{\ell'} = \varphi_1(s')\cdot\widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}),\varphi_1(r'_0),\dots,\varphi_1(r'_{\ell'-1})\big)$$

will hold, and that $\mathcal{V}$ will accept its final check. Finally, assuming again that $s' = t'(r'_0,\dots,r'_{\ell'-1})$, by the completeness of Π', $\mathcal{V}'$—and hence $\mathcal{V}$—will necessarily accept its sub-evaluation protocol. □

Remark 4.3. We explain in slightly more rigorous terms the "information loss" which would result if the parties *merely* evaluated $t'(r_\kappa,\dots,r_{\ell-1})$, as opposed to using the tensor algebra. It is shown during the proof of Theorem 4.2 above that the prover's quantity $s_0 := \varphi_1(t')(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}))$ is such that, for each $v\in\mathcal{B}_\kappa$, $s_{0,v} = t(v_0,\dots,v_{\kappa-1},r_\kappa,\dots,r_{\ell-1})$ holds. On the other hand,

$$\begin{aligned}
t'(r_\kappa,\dots,r_{\ell-1}) &= \sum_{w\in\mathcal{B}_{\ell'}} t'(w)\cdot\widetilde{\mathsf{eq}}(r_\kappa,\dots,r_{\ell-1},w_0,\dots,w_{\ell'-1})\\
&= \sum_{w\in\mathcal{B}_{\ell'}}\left(\sum_{v\in\mathcal{B}_\kappa} t(v_0,\dots,v_{\kappa-1},w_0,\dots,w_{\ell'-1})\cdot\beta_v\right)\cdot\widetilde{\mathsf{eq}}(r_\kappa,\dots,r_{\ell-1},w_0,\dots,w_{\ell'-1})\\
&= \sum_{v\in\mathcal{B}_\kappa}\left(\sum_{w\in\mathcal{B}_{\ell'}} t(v_0,\dots,v_{\kappa-1},w_0,\dots,w_{\ell'-1})\cdot\widetilde{\mathsf{eq}}(r_\kappa,\dots,r_{\ell-1},w_0,\dots,w_{\ell'-1})\right)\cdot\beta_v\\
&= \sum_{v\in\mathcal{B}_\kappa} t(v_0,\dots,v_{\kappa-1},r_\kappa,\dots,r_{\ell-1})\cdot\beta_v\\
&= \sum_{v\in\mathcal{B}_\kappa} s_{0,v}\cdot\beta_v.
\end{aligned}$$

We see that while the information contained in $s_0 = (s_{0,v})_{v\in\mathcal{B}_\kappa}$ suffices to derive $t(r_0,\dots,r_{\ell-1})$—as the proof of Theorem 4.2 above demonstrates—the datum $t(r_\kappa,\dots,r_{\ell-1})$ would yield, rather, the *basis-combination* $\sum_{v\in\mathcal{B}_\kappa} s_{0,v}\cdot\beta_v$ of s_0's columns. Since the basis $(\beta_v)_{v\in\mathcal{B}_\kappa}$ is certainly not linearly independent over L, this latter combination reflects s_0 only "lossfully". We note that, interestingly, $\sum_{v\in\mathcal{B}_\kappa} s_{0,v}\cdot\beta_v = h(s_0)$ holds; here, $h: L\otimes_K L\to L$ is the canonical K-linear map defined on simple tensors by multiplication (we recall Subsection 2.5 above). That is, $t(r_\kappa,\dots,r_{\ell-1})$ relates to $\varphi_1(t')(\varphi_0(r_\kappa),\dots,\varphi_0(r_{\ell-1}))$ exactly by the map h, which is of course not injective. We would like to thank Raju Krishnamoorthy for explaining this fact.

We now prove the security of ring-switching.

Theorem 4.4. *If* $\Pi' = \big(\mathsf{Setup}', \mathsf{Commit}', \mathcal{P}', \mathcal{V}'\big)$ *is secure, then* $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ *also is.*

Proof. We write $\mathcal{E}'$ for the emulator for Π' (guaranteed to exist by assumption). We define an emulator $\mathcal{E}$ for Π as follows.

1. On input $\mathcal{A}$'s record of interactions with the vector oracle, $\mathcal{E}$ internally runs $t'(X_0, \ldots, X_{\ell'-1}) \leftarrow \mathcal{E}'$.
2. By reversing Definition 2.1, $\mathcal{E}$ obtains $t(X_0, \ldots, X_{\ell-1}) \in K[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, which it outputs.

We argue that the emulator $\mathcal{E}$ defined in this way is secure. In particular, the probability with which $s \neq t(r_0, \ldots, r_{\ell-1})$ holds and $\mathcal{V}$ accepts is negligible. We suppose that $s \neq t(r_0, \ldots, r_{\ell-1})$, and argue, conditioned on this assumption, that $\mathcal{V}$ will accept with at most negligible probability.

We first note that we may assume that $\mathcal{P}$'s message $s_0 \neq \varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$. Indeed, it is shown directly in the course of our proof of Theorem 4.2 above that, *if* $s_0 = \varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ holds, then $\sum_{v \in \mathcal{B}_\kappa} s_{0,v} \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\kappa-1}, v_0, \ldots, v_{\kappa-1}) = t(r_0, \ldots, r_{\ell-1})$ too will hold; under our above assumption whereby $s \neq t(r_0, \ldots, r_{\ell-1})$, we see that $s \neq t(r_0, \ldots, r_{\ell-1}) = \sum_{v \in \mathcal{B}_\kappa} s_{0,v} \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\kappa-1}, v_0, \ldots, v_{\kappa-1})$ in turn will hold, so that $\mathcal{V}$ will reject and we're done. We thus assume that $s_0 \neq \varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$.

We write $h(X_0, \ldots, X_{\ell'-1}) := \varphi_1(t')(X_0, \ldots, X_{\ell'-1}) \cdot \widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), X_0, \ldots, X_{\ell'-1})$ as above; since $\varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1})) = \sum_{w \in \mathcal{B}_{\ell'}} h(w)$, our assumption $s_0 \neq \varphi_1(t')(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ entails exactly that $s_0 \neq \sum_{w \in \mathcal{B}_{\ell'}} h(w)$. We argue that, by the soundness of the sumcheck, $s_{\ell'} \neq h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$ in turn will hold, except with low probability. Since our sumcheck is slightly nonstandard, we prove this result explicitly (though our proof is the same as the usual one, up to its use of Lemma 4.5). We go through the details now. Indeed, we fix an index $i \in \{0, \ldots, \ell' - 1\}$, and assume by induction that $s_i \neq \sum_{w \in \mathcal{B}_{\ell'-i}} h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{i-1}), w_0, \ldots, w_{\ell'-i-1}\big)$ holds. Under this assumption, we may ignore the case in which the prover constructs its i^{th} univariate polynomial $h_i(X) = \sum_{w \in \mathcal{B}_{\ell'-i-1}} h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{i-1}), X, w_0, \ldots, w_{\ell'-i-2}\big)$ as prescribed. Indeed, if it *does*, then we immediately obtain, under our inductive assumption, that

$$s_i \neq \sum_{w \in \mathcal{B}_{\ell'-i}} h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{i-1}), w_0, \ldots, w_{\ell'-i-1}\big) = h_i(0) + h_i(1)$$

will hold, so that $\mathcal{V}$ will reject outright and we're done. We thus assume the polynomial inequality $h_i(X) \neq \sum_{w \in \mathcal{B}_{\ell'-i-1}} h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{i-1}), X, w_0, \ldots, w_{\ell'-i-2}\big)$; here, both sides of this inequality are degree-two, univariate polynomials with coefficients in A. We argue that, under this assumption, except with probability at most $\frac{2}{|L|}$ over $\mathcal{V}$'s choice of $r'_i \leftarrow L$, we will in turn have $h_i(\varphi_1(r'_i)) \neq \sum_{w \in \mathcal{B}_{\ell'-i-1}} h(\varphi_1(r'_0), \ldots, \varphi_1(r'_i), w_0, \ldots, w_{\ell'-i-2})$. Since $\mathcal{V}$ sets $s_{i+1} := h_i(\varphi_1(r'_i))$ by definition, this latter inequality implies that $s_{i+1} \neq \sum_{w \in \mathcal{B}_{\ell'-i-1}} h(\varphi_1(r'_0), \ldots, \varphi_1(r'_i), w_0, \ldots, w_{\ell'-i-2})$ will in turn hold; this is exactly the claim we need in order to preserve the inductive hypothesis.

It thus suffices to argue that each nonzero polynomial of degree at most 2 in $A[X]$ has at most 2 roots in the subring $\varphi_1(L) \subset A$. While the standard univariate factorization result [Lan02, Ch. IV, § 1, Thm. 1.4] can fail spectacularly, in general, for commutative rings which aren't integral domains, we below salvage that result in our more-general setting, granting that we consider only roots in a subfield of the algebra.

Lemma 4.5. *Each nonzero* $p(X) \in A[X]$ *of degree at most* d *has at most* d *zeros in the subring* $\varphi_1(L) \subset A$.

Proof. We adapt [Lan02, Ch. IV, § 1, Thm. 1.4]. For contradiction, we fix distinct elements $a_0, \ldots, a_d$ of $\varphi_1(L) \subset A$, each satisfying $p(a_i) = 0$. Applying [Lan02, Ch. IV, § 1, Thm. 1.1], we see that there exist polynomials $q(X)$ and $r(X)$ in $A[X]$, where $r(X)$ is of degree less than 1, for which $p(X) = (X - a_0) \cdot q(X) + r(X)$ holds. Since $0 = p(a_0) = r(a_0)$, we see that $r(X)$ is in fact identically zero, so that $(X - a_0) \cdot q(X) = p(X)$. On the other hand, for each $i \in \{1, \ldots, d\}$, we moreover have $0 = p(a_i) = (a_i - a_0) \cdot q(a_i)$. Since $a_i - a_0 \in \varphi_1(L)$ is a unit in A, we deduce further that $q(a_i) = 0$. We conclude that $q(X)$ vanishes identically on each of the points $a_1, \ldots, a_d$. This fact justifies our use of induction on $q(X)$. Proceeding in exactly that way, we establish the divisibility $(X - a_1) \cdot \cdots \cdot (X - a_d) \mid q(X)$, which implies $(X - a_0) \cdot \cdots \cdot (X - a_d) \mid p(X)$. By degree considerations, we finally conclude that $p(X)$ is identically zero, contradicting our hypothesis. $\square$

This lemma proves as desired that, except with probability at most $\frac{2}{|L|}$ over $\mathcal{V}$'s choice of $r'_i \leftarrow L$, we have $s_{i+1} \neq \sum_{w \in \mathcal{B}_{\ell'-i-1}} h(\varphi_1(r'_0), \ldots, \varphi_1(r'_i), w_0, \ldots, w_{\ell'-i-2})$. Carrying through the induction, we conclude that the probability with which $\mathcal{V}$ accepts *and* $s_{\ell'} = h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$ holds is at most $\ell' \cdot \frac{2}{|L|}$. We thus ignore those executions, and assume instead that $s_{\ell'} \neq h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$, or in other words that:

$$s_{\ell'} \neq \varphi_1(t')(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})) \cdot \widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big).$$

Under this latter assumption, we argue further that if $\mathcal{P}$ correctly sends $s' = t'(r'_0, \ldots, r'_{\ell'-1})$, then $\mathcal{V}$ will again reject. Indeed, if it does, then $\varphi_1(s') = \varphi_1(t'(r'_0, \ldots, r'_{\ell'-1})) = \varphi_1(t')(\varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1}))$ will hold; using our assumption above, we see in turn that

$$s_{\ell'} \neq \varphi_1(s') \cdot \widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$$

will hold, so that $\mathcal{V}$ will reject. We thus assume that $s' \neq t'(r'_0, \ldots, r'_{\ell'-1})$. Under exactly this assumption, the probability with which $\mathcal{V}'$ accepts is negligible, under our hypothesis whereby Π' is secure. □

We defer our analysis of the efficiency of Construction 4.1 to Section 5. We note informally that it has no commitment overhead whatsoever; its evaluation overhead consists of an ℓ'-variate sumcheck, of degree 2, over the ring A. As we explain in Subsection 5.2 below, the ideas of Gruen [Gru24, § 3] serve to further reduce the effective degree of that protocol's sumcheck from 2 to 1.

5 Small-Field IOPCS

In this section, we describe a "combined" small-field protocol, which combines the simple scheme of Section 3 with the ring-switching reduction of Section 4. We moreover streamline and optimize the *resulting* combination, by unifying Construction 3.11's sumcheck with that required within Construction 4.1. That is, we move the algebra-element sumcheck of Construction 4.1 *inline* into Construction 3.11. We also concretely benchmark this combined scheme.

5.1 Combined Small-Field Protocol

We present our full combined protocol below. Our protocol directly instantiates the generic small-field template of Definition 2.10; we slightly specialize that template by requiring that the ground field $K = \mathcal{T}_\iota$ be a binary tower field.

We again use the tensor algebra $A := \mathcal{T}_\tau \otimes_{\mathcal{T}_\iota} \mathcal{T}_\tau$, as well as the two ring embeddings $\varphi_0 : \mathcal{T}_\tau \hookrightarrow A$ and $\varphi_1 : \mathcal{T}_\tau \hookrightarrow A$.

CONSTRUCTION 5.1 (Combined Small-Field IOPCS)**.**
We define $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ as follows.

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, \ell, \iota)$. On input 1^λ, ℓ, and ι, choose a constant, positive rate parameter $\mathcal{R} \in \mathbb{N}$ and a tower height $\tau \geq \log(\omega(\log \lambda))$ for which $\tau \geq \iota$ and $2^\tau \geq \ell - \tau + \iota + \mathcal{R}$. Write $\kappa := \tau - \iota$ and $\ell' := \ell - \kappa$. Initialize the vector oracle $\mathcal{F}_{\mathsf{Vec}}^{\mathcal{T}_\tau}$. Fix a folding factor $\vartheta \mid \ell'$ and a repetition parameter $\gamma = \omega(\log(\lambda))$. Write $(X_0(X), \ldots, X_{2^{\ell'}-1}(X))$ for the novel $\mathcal{T}_\tau$-basis of $\mathcal{T}_\tau[X]^{\prec 2^{\ell'}}$, and fix the domains $S^{(0)}, \ldots, S^{(\ell')}$ and the polynomials $q^{(0)}, \ldots, q^{(\ell'-1)}$ as in Subsection 3.1. Write $C^{(0)} \subset {\mathcal{T}_\tau}^{2^{\ell'+\mathcal{R}}}$ for the Reed–Solomon code $\mathsf{RS}_{\mathcal{T}_\tau, S^{(0)}}[2^{\ell'+\mathcal{R}}, 2^{\ell'}]$.
- $[f] \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input $t(X_0, \ldots, X_{\ell-1}) \in \mathcal{T}_\iota[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$, construct using Definition 2.1 the packed polynomial $t'(X_0, \ldots, X_{\ell'-1}) \in \mathcal{T}_\tau[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$. Write $P(X) := \sum_{v \in \mathcal{B}_{\ell'}} t'(v) \cdot X_{\{v\}}(X)$ for its univariate flattening. Using Algorithm 2, compute the Reed–Solomon codeword $f : S^{(0)} \to \mathcal{T}_\tau$ defined by $f : x \mapsto P(x)$. Submit $(\texttt{submit}, \ell' + \mathcal{R}, f)$ to the vector oracle $\mathcal{F}_{\mathsf{Vec}}^{\mathcal{T}_\tau}$. Upon receiving $(\texttt{receipt}, \ell' + \mathcal{R}, [f])$ from the oracle, output the commitment $[f]$.

We define $(\mathcal{P}, \mathcal{V})$ as the following IOP, in which both parties have the common input $[f]$, $s \in \mathcal{T}_\tau$, and

$(r_0, \ldots, r_{\ell-1}) \in \mathcal{T}_\tau^\ell$, and $\mathcal{P}$ has the further input $t(X_0, \ldots, X_{\ell-1}) \in \mathcal{T}_\iota[X_0, \ldots, X_{\ell-1}]^{\preceq 1}$.

- $\mathcal{P}$ again writes $t'(X_0, \ldots, X_{\ell'-1}) \in \mathcal{T}_\tau[X_0, \ldots, X_{\ell'-1}]^{\preceq 1}$ for $t(X_0, \ldots, X_{\ell-1})$'s packed polynomial; $\mathcal{P}$ moreover writes $h(X_0, \ldots, X_{\ell'-1}) \coloneqq t'(X_0, \ldots, X_{\ell'-1}) \cdot \widetilde{\mathsf{eq}}(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), X_0, \ldots, X_{\ell'-1})$.
- $\mathcal{P}$ writes $s_0 \coloneqq t'(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}))$ and sends $\mathcal{V}$ s_0.
- $\mathcal{V}$ destructures $(s_{0,v})_{v \in \mathcal{B}_\kappa} \coloneqq s_0$, and checks $s \overset{?}{=} \sum_{v \in \mathcal{B}_\kappa} s_{0,v} \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\kappa-1}, v_0, \ldots, v_{\kappa-1})$.
- $\mathcal{P}$ and $\mathcal{V}$ both abbreviate $f^{(0)} \coloneqq f$, and execute the following loop:
  1: **for** $i \in \{0, \ldots, \ell' - 1\}$ **do**
  2: $\quad$ $\mathcal{P}$ sends $\mathcal{V}$ the polynomial $h_i(X) \coloneqq \sum_{v \in \mathcal{B}_{\ell'-i-1}} h\big(\varphi_1(r'_0), \ldots, \varphi_1(r'_{i-1}), X, v_0, \ldots, v_{\ell'-i-2}\big)$.
  3: $\quad$ $\mathcal{V}$ requires $s_i \overset{?}{=} h_i(0) + h_i(1)$. $\mathcal{V}$ samples $r'_i \leftarrow \mathcal{T}_\tau$, sets $s_{i+1} \coloneqq h_i(\varphi_1(r'_i))$, and sends $\mathcal{P}$ r'_i.
  4: $\quad$ $\mathcal{P}$ defines $f^{(i+1)} : S^{(i+1)} \to \mathcal{T}_\tau$ as the function $\mathsf{fold}\big(f^{(i)}, r'_i\big)$ of Definition 3.6.
  5: $\quad$ **if** $i + 1 = \ell'$ **then** $\mathcal{P}$ sends $c \coloneqq f^{(\ell')}(0, \ldots, 0)$ to $\mathcal{V}$.
  6: $\quad$ **else if** $\vartheta \mid i + 1$ **then** $\mathcal{P}$ submits $(\texttt{submit}, \ell' + \mathcal{R} - i - 1, f^{(i+1)})$ to the oracle.
- $\mathcal{V}$ requires $s_{\ell'} \overset{?}{=} \varphi_1(c) \cdot \widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$.
- $\mathcal{V}$ executes the following querying procedure:
  1: **for** γ repetitions **do**
  2: $\quad$ $\mathcal{V}$ samples $v \leftarrow \mathcal{B}_{\ell'+\mathcal{R}}$ randomly.
  3: $\quad$ **for** $i \in \{0, \vartheta, \ldots, \ell' - \vartheta\}$ (i.e., taking ϑ-sized steps) **do**
  4: $\quad\quad$ for each $u \in \mathcal{B}_\vartheta$, $\mathcal{V}$ sends $\big(\texttt{query}, [f^{(i)}], (u_0, \ldots, u_{\vartheta-1}, v_{i+\vartheta}, \ldots, v_{\ell'+\mathcal{R}-1})\big)$ to the oracle.
  5: $\quad\quad$ **if** $i > 0$ **then** $\mathcal{V}$ requires $c_i \overset{?}{=} f^{(i)}(v_i, \ldots, v_{\ell'+\mathcal{R}-1})$.
  6: $\quad\quad$ $\mathcal{V}$ defines $c_{i+\vartheta} \coloneqq \mathsf{fold}\big(f^{(i)}, r'_i, \ldots, r'_{i+\vartheta-1}\big)(v_{i+\vartheta}, \ldots, v_{\ell'+\mathcal{R}-1})$.
  7: $\quad$ $\mathcal{V}$ requires $c_{\ell'} \overset{?}{=} c$.

We refrain from proving explicitly the completeness and security of Construction 5.1. Indeed, those results follow essentially directly from the ideas already developed in Sections 3 and 4 above.

5.2 Efficiency

We examine the efficiency of Construction 5.1, both asymptotic and concrete. Throughout our below analysis, we view the coefficient size parameter ι and the Reed–Solomon rate parameter $\mathcal{R}$ as constants, though we note in passing our protocol's various dependencies on these values.

We note that it's possible to achieve, for both parties, a merely-*polylogarithmic* dependence on the security parameter λ—while retaining asymptotic security—by instantiating with appropriate care the extension degree 2^τ, the random oracle digest width, and the repetition parameter γ. (Specifically, it's enough to demand that these quantities grow strictly polylogarithmically—i.e., with exponent greater than 1—in λ.) Since this fact is of essentially theoretical interest, we refrain from developing it (though we refer to [DP23, Thm. 3.14] for a related treatment). We finally assume throughout that ϑ is bounded from above by a constant (increasing ϑ is universally an option, and not a requirement).

For Construction 5.1 to be well-defined; it's necessary that $2^\tau \geq \ell - \tau + \iota + \mathcal{R}$ hold. For the sake of security, we moreover set $2^\tau \geq \Theta(\lambda)$. In sum, it suffices that $2^\tau \geq \Theta(\lambda + \ell)$ hold; we assume as much throughout. We moreover set $\gamma \coloneqq \Theta(\lambda)$, and assume that the random oracle outputs digests of size $\Theta(\lambda)$. We see that each $\mathcal{T}_\tau$-element takes $\Omega(\lambda, \ell)$ bits to represent and each $\mathcal{T}_\tau$-operation takes $\mathsf{poly}(\lambda, \ell)$ work (in fact, $O((\lambda + \ell)^{\log(3)})$ is enough). Similarly, each A-element occupies $2^\iota \cdot (2^{\tau-\iota})^2 = 2^{2\cdot\tau-\iota} = O((\lambda + \ell)^2)$ bits, and each A-operation again takes $\mathsf{poly}(\lambda, \ell)$ work.

The commitment phase of Construction 5.1 amounts to a Reed–Solomon encoding operation in the code $C^{(0)} = \mathsf{RS}_{\mathcal{T}_\tau, S^{(0)}}[2^{\ell'+\mathcal{R}}, 2^{\ell'}]$. By Lin, Chung and Han [LCH14, § III. D.] (see also Algorithm 2), this operation can be carried out in $\Theta\Big(\ell' \cdot 2^{\ell'+\mathcal{R}}\Big) = \Theta\Big(\ell' \cdot 2^{\ell'}\Big)$ $\mathcal{T}_\tau$-operations (specifically, using $\ell' \cdot 2^{\ell'+\mathcal{R}}$ and $\ell' \cdot 2^{\ell'+\mathcal{R}-1}$ $\mathcal{T}_\tau$-additions and $\mathcal{T}_\tau$-multiplications, respectively). The prover's opening protocol entails a sumcheck on the

A-polynomial $h(X_0, \ldots, X_{\ell'-1})$—whose individual degree in each variable is at most 2—and an execution of our 2^ϑ-ary multilinear FRI variant (see Subsection 3.2) on the ℓ'-variate committed word f over $\mathcal{T}_\tau$. By the sumcheck prover analysis of Thaler [Tha22, Lem. 4.5], the first task takes $\Theta(2^{\ell'})$ A-operations, which represents $\Theta(2^{\ell'}) \cdot \mathsf{poly}(\lambda, \ell)$ total work. It follows essentially by inspection that our prover's FRI-incumbent work amounts to $\Theta(2^{\ell'+\mathcal{R}}) = \Theta(2^{\ell'})$ $\mathcal{T}_\tau$-operations, and thus again represents $\Theta(2^{\ell'}) \cdot \mathsf{poly}(\lambda, \ell)$ total work. We conclude that our prover is quasilinear in the packed length $2^{\ell'}$ of its witness.

Construction 5.1's verifier complexity is essentially that of the sumcheck verifier plus that of the FRI verifier. These latter tasks entail $\Theta(\ell')$ A-operations and $\Theta(\gamma \cdot 2^\vartheta \cdot \frac{\ell'}{\vartheta}) = O(\ell') \cdot \mathsf{poly}(\lambda)$ $\mathcal{T}_\tau$-operations, respectively. These tasks thus represent total work on the order of $O(\ell') \cdot \mathsf{poly}(\lambda, \ell) = \mathsf{poly}(\lambda, \ell)$ bit-operations for the verifier. Finally, the verifier's computation of $\big(\widetilde{\mathsf{eq}}(r_0, \ldots, r_{\kappa-1}, v_0, \ldots, v_{\kappa-1})\big)_{v \in \mathcal{B}_\kappa}$ takes $\Theta(2^\kappa) = \Theta(2^\tau) = \Theta(\lambda, \ell)$ $\mathcal{T}_\tau$-operations, and so represents $\mathsf{poly}(\lambda, \ell)$ total work (we recall from Subsection 2.1 the standard algorithm for this task). The verifier's computation of $\widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big)$ takes just $\Theta(\ell')$ A-operations, and so again represents just $O(\ell') \cdot \mathsf{poly}(\lambda, \ell) = \mathsf{poly}(\lambda, \ell)$ total work.

The non-oracle communication cost of Construction 5.1 (i.e., directly between the prover and the verifier) amounts to three A-elements per round of the sumcheck, together with the final $\mathcal{T}_\tau$-element c.

The BCS transform. In the variant of Construction 5.1 in which, by means of the BCS transform [BCS16], the use of the vector oracle is eliminated, the prover must moreover Merkle-hash $f^{(0)}$ during its commitment phase, as well as the oracles $f^{(\vartheta)}, \ldots, f^{(\ell'-\vartheta)}$ during its opening proof; these commitments represent total work on the order of $\Theta\Big(2^{\ell'+\mathcal{R}}\Big) = \Theta\Big(2^{\ell'}\Big)$ hash evaluations. We note that for each query repetition $i \in \{\ell' - \vartheta, \ldots, \vartheta, 0\}$ and each $v \in \mathcal{B}_{\ell'+\mathcal{R}-i-\vartheta}$, the required leaves $\big(f^{(i)}(u \,\|\, v)\big)_{u \in \mathcal{B}_\vartheta}$ are naturally adjacent in the prover's i^{th} Merkle tree. We thus opt to send only a single shortened Merkle path, of height only $\ell' + \mathcal{R} - i - \vartheta$, as well as the 2^ϑ relevant field elements, at each such query step. The total prover work during the query phase is thus $O\Big(\gamma \cdot \Big(\lambda \cdot (\ell' + \mathcal{R})^2 + \frac{\ell'}{\vartheta} \cdot 2^\vartheta \cdot \Theta(\lambda + \ell)\Big)\Big) = O\big(\gamma \cdot (\lambda \cdot \ell^2)\big)$. Using our further assumption whereby $\gamma = \Theta(\lambda)$, we upper-bound the prover's work during the query phase as $O\big(\lambda^2 \cdot \ell^2\big)$.

In this non-oracle variant of the protocol—in which the verifier must check Merkle paths—the verifier's FRI cost becomes $O\Big(\gamma \cdot \Big(\lambda \cdot (\ell' + \mathcal{R})^2 + \frac{\ell'}{\vartheta} \cdot 2^\vartheta \cdot \mathsf{poly}(\lambda, \ell)\Big)\Big)$, which is again $O(\ell^2) \cdot \mathsf{poly}(\lambda, \ell) = \mathsf{poly}(\lambda, \ell)$.

During the protocol's query phase—and assuming again the BCS-transformed version—we encounter further a proof size cost on the order of $O\Big(\gamma \cdot \Big(\lambda \cdot (\ell' + \mathcal{R})^2 + \frac{\ell'}{\vartheta} \cdot 2^\vartheta \cdot \Theta(\lambda + \ell)\Big)\Big) = O\big(\lambda^2 \cdot \ell^2\big)$ bits.

Concrete soundness. We record proof sizes for both this work and [DP23, Cons. 3.11]. In our concrete proof size analyses below, we incorporate various further optimizations. For example, we opt to send the *entire j^{th} layer* of the Merkle tree—as opposed to only its root—in each round $i \in \{0, \vartheta, \ldots, \ell' - \vartheta\}$, where j is an appropriately chosen constant. (For those i for which $j > \ell' + \mathcal{R} - i - \vartheta$, we simply send the prover's entire oracle in the clear.) Increasing j exponentially increases the fixed cost of each Merkle tree, but also causes each among the γ subsequently sent paths to become shorter. The optimal truncation height turns out to be $j \coloneqq \lceil \log_2(\gamma) \rceil$. Finally, we incorporate a standard FRI early-termination optimization; we stipulate that our prover send its message directly to the verifier in the clear as soon as it becomes sufficiently small.

We further incorporate the various optimizations described in Gruen [Gru24, § 3]. Those optimizations serve to reduce the communication cost of each polynomial $h_i(X)$ above from three algebra-elements per round to just one. They also decrease our protocol's sumcheck-specific soundness error from $\frac{2 \cdot \ell'}{|\mathcal{T}_\tau|}$ to $\frac{\ell'}{|\mathcal{T}_\tau|}$.

In order to appropriately select the query repetition parameter γ, we must examine the concrete security of our protocol (we refer to [DP23, § 3.5] for an analogous analysis). It follows essentially from the proof of Theorem 3.16 that Construction 5.1's concrete soundness error is bounded from above by

$$\frac{\ell'}{|\mathcal{T}_\tau|} + \frac{2^{\ell'+\mathcal{R}}}{|\mathcal{T}_\tau|} + \left(\frac{1}{2} + \frac{1}{2 \cdot 2^{\mathcal{R}}}\right)^\gamma; \tag{8}$$

above, the first summand is sumcheck-specific, whereas the latter two reflect Propositions 3.22 and 3.23, respectively. For each desired *concrete security* level Ξ, we thus set γ minimally so that (8) becomes bounded from above by Ξ. (Clearly, this is possible only when $\Xi > \frac{\ell'}{|\mathcal{T}_\tau|} + \frac{2^{\ell'+\mathcal{R}}}{|\mathcal{T}_\tau|}$ holds.) We say in this case that Construction 5.1 attains $-\log_2(\Xi)$ *bits of security*.

Proof sizes. In our proof size measurements below, we use a 128-bit field, and attain 96 bits of *provable* security. In Construction 5.1, we use between $\gamma = 142$ and $\gamma = 144$ queries, as the case may be. We use the Merkle tree truncation height $j := 8$. We fix the folding factor $\vartheta := 4$, which happens to yield the smallest proofs throughout. We terminate FRI early as soon as the message size inequality $\ell' - i \leq 11$ holds. The previous work [DP23, Cons. 3.11] requires more queries—rather between $\gamma = 231$ and $\gamma = 232$, for the sizes we benchmark below—as [DP23, Rem. 3.18] explains. Our proof sizes appear in Table 1 below.

Total Data Size	Num. Variables ℓ	Coefficient Size ι	[DP23, Cons. 3.11]	Construction 5.1
32 MiB (2^{28} bits)	22	6	0.753 MiB	0.227 MiB
	25	3	1.003 MiB	0.231 MiB
	28	0	2.849 MiB	0.267 MiB
512 MiB (2^{32} bits)	26	6	4.532 MiB	0.335 MiB
	29	3	5.682 MiB	0.340 MiB
	32	0	11.300 MiB	0.383 MiB
8 GiB (2^{36} bits)	30	6	11.329 MiB	0.465 MiB
	33	3	22.572 MiB	0.471 MiB
	36	0	61.064 MiB	0.521 MiB

Table 1: Proof sizes, including oracle-skipping, Merkle caps, and early FRI termination.

We see that our Construction 5.1 beats [DP23, Cons. 3.11] by as much as a hundredfold.

Concrete performance. We concretely benchmark this work's Construction 5.1 above, as well as [DP23, Cons. 3.11] and the univariate-FRI-based scheme *Plonky3*. Our benchmarks of the first two schemes use *Binius*, an open-source implementation of both [DP23] and this work.

In our benchmarks below, we again use a 128-bit field and attain 96 bits of provable security. We work exclusively in the unique-decoding regime. We note that both [DP23, Cons. 3.11] and this work operate solely in that regime (as of yet). As for Plonky3, we note that it's impossible to obtain 96 bits of provable security in the *list-decoding* regime over a field of merely 128 bits. Indeed, the best-available proximity gap in that regime—namely, [Ben+23, Thm. 5.1]—has a false witness probability [Ben+23, (5.3)] which grows *quadratically* in its problem size. We see that each reasonably-large instance stands to overwhelm that result's 128-bit denominator (yielding a vacuous bound). Our benchmarks below thus reflect the *best-possible* proof size attainable in Plonky3, conditioned on the 96-bit security level and the use of a 128-bit field.

In [DP23] and this work, we work over the 128-bit tower field $\mathcal{T}_7$. In *Plonky3*, we use the quartic extension $\mathbb{F}_p[X]/\big(X^4 - 11\big)$ of the *Baby Bear* prime field $\mathbb{F}_p$, where $p := 2^{31} - 2^{27} + 1$. Throughout, we use the code rate $\rho = \frac{1}{4}$. We benchmark [DP23] and Construction 5.1 on ℓ-variate multilinear polynomials, where $\ell \in \{20, 24, 28\}$. In each case, we consider polynomials over $\mathcal{T}_\iota$, for $\iota \in \{0, 3, 5\}$ (i.e., with coefficients of 1 bit, 8 bits and 32 bits). As far as Plonky3, we benchmark univariate polynomials of degree $2^\ell \in \{2^{20}, 2^{24}\}$ (the degree-2^{28} size is inaccessible in Plonky3, as we explain below). In that setting, we benchmark *only* polynomials over the 31-bit Baby Bear field $\mathbb{F}_p$; indeed, that scheme would not perform any better upon being given as input a polynomial whose coefficients were "smaller" (albeit still $\mathbb{F}_p$-elements).

In our concrete benchmarks both of this work and of Plonky3 below, we omit throughout the Merkle-caps, oracle-skipping, *and* early-termination optimizations. (That is, in this work, we set $\vartheta := 1$ and $j := 0$ and use the termination threshold of 0, and moreover proceed analogously in Plonky3.) These omissions make our proofs become significantly larger (and our prover and verifier slower to boot); we refer to Table 1 above for our protocol's "true" proof sizes. On the other hand, they make our comparison to Plonky3 below more direct, since that work *also* neglects to include these optimizations, as currently written.

Finally, we have chosen to benchmark Plonky3 on *batches* of polynomials. The most natural benchmark would have compared our scheme's performance on ℓ-variate multilinear polynomials to Plonky3's on single, degree-2^ℓ univariate polynomials. We note, however, that Plonky3's FRI-PCS implementation is heavily optimized for the case of batched polynomial commitments. In order to present a fairer juxtaposition, we instead run Plonky3 on batches of 2^4 univariate polynomials, each of degree $2^{\ell-4}$, for each problem size ℓ.

In our CPU benchmarks below, we use throughout a Google Cloud machine of type `c3-standard-22` with an Intel *Xeon Scalable* ("Sapphire Rapids") processor and 22 virtual cores. Both the Binius and Plonky3 implementations leverage AVX-512 accelerated instructions; Binius moreover uses the Intel GFNI instruction set extension. We benchmark Plonky3 using both the Poseidon2 and Keccak-256 hashes (the former hash is "recursion-friendly" in that work's prime-field setting). We present our singlethreaded results in Table 2.

Commit. Scheme	Prob. Sz. ℓ	Coef. Sz. (bits)	Pf. Sz. (MiB)	Commit (s)	Prove (s)	Verify (s)
Plonky3,	20	31	0.741	0.527	0.416	0.0254
Baby Bear,	24	31	1.097	8.72	6.81	0.0367
Poseidon2	28	31	1.533	146	112	0.0499
Plonky3,	20	31	0.741	0.265	0.297	0.0125
Baby Bear,	24	31	1.097	4.67	4.88	0.0180
Keccak-256	28	31	1.533	80.5	81.0	0.0228
[DP23, Cons. 3.11]	20	1	0.183	0.00409	0.00294	0.00537
		8	0.205	0.0326	0.00406	0.00351
		32	0.281	0.128	0.0173	0.00478
	24	1	0.725	0.0457	0.0662	0.0146
		8	0.746	0.405	0.0959	0.00917
		32	1.010	1.58	0.465	0.0109
	28	1	2.849	0.740	1.56	0.0501
		8	3.870	6.27	2.99	0.0291
		32	3.884	29.3	17.0	0.0716
Construction 5.1	20	1	0.533	0.0252	0.0335	0.00870
		8	0.732	0.205	0.268	0.0123
		32	0.899	0.822	0.961	0.0176
	24	1	0.844	0.401	0.513	0.0141
		8	1.089	3.33	4.27	0.0253
		32	1.289	13.4	15.5	0.0682
	28	1	1.225	6.73	8.39	0.0357
		8	1.515	54.9	68.6	0.224
		32	1.749	222	248	0.889

Table 2: Singlethreaded benchmarks.

In Table 3 below, we present multithreaded benchmarks.

Commit. Scheme	Prob. Sz. ℓ	Coef. Sz. (bits)	Pf. Sz. (MiB)	Commit (s)	Prove (s)	Verify (s)
Plonky3,	20	31	0.741	0.0612	0.0744	0.0258
Baby Bear,	24	31	1.097	0.931	1.11	0.0372
Poseidon2	28	31	1.533	15.8	19.5	0.0506
Plonky3,	20	31	0.741	0.0366	0.0592	0.0129
Baby Bear,	24	31	1.097	0.603	0.984	0.0180
Keccak-256	28	31	1.533	11.2	17.6	0.0242
[DP23, Cons. 3.11]	20	1	0.183	0.00257	0.000976	0.00525
		8	0.205	0.00556	0.00125	0.00393
		32	0.281	0.00146	0.00248	0.00410
	24	1	0.725	0.00677	0.00688	0.0108
		8	0.746	0.0738	0.0114	0.00729
		32	1.010	0.288	0.0421	0.00760
	28	1	2.849	0.146	0.141	0.0302
		8	3.870	1.35	0.308	0.0197
		32	3.884	5.24	1.42	0.0472
Construction 5.1	20	1	0.533	0.00368	0.0145	0.0585
		8	0.732	0.0229	0.0405	0.0698
		32	0.899	0.0840	0.109	0.0825
	24	1	0.844	0.0334	0.0576	0.0754
		8	1.089	0.350	0.435	0.0966
		32	1.289	1.39	1.58	0.134
	28	1	1.225	0.684	0.851	0.116
		8	1.515	5.65	6.91	0.336
		32	1.749	22.7	25.0	1.09

Table 3: Multithreaded benchmarks.

5.3 Batching

In this subsection, we sketch our approach to the batch-evaluation of many polynomials. We note that a technique of Ron-Zewi and Rothblum [RR24, Fig. 3] serves to reduce the evaluation of some fixed polynomial at many points to that of the same polynomial at just one point. It is not difficult to extend the reduction [RR24, Fig. 3] to the case of *possibly different* polynomials at different points (we refer to Chen, Bünz, Boneh and Zhang [CBBZ23, § 3.7] for a similar extension). In this subsection, we discuss the *target* of this reduction; i.e., we discuss how to commit to a batch of polynomials, and then later to evaluate all of those polynomials at a single given point.

We fix tower heights ι and τ as in Construction 5.1, a batching factor $m \geq 0$, and a batch of ℓ-variate input multilinears $(t_u(X_0, \ldots, X_{\ell-1}))_{u \in \mathcal{B}_m}$ over $\mathcal{T}_\iota$. For simplicity, we assume in this subsection that the relevant input multilinears are defined over the *same* subfield $\mathcal{T}_\iota \subset \mathcal{T}_\tau$ (an extension of our theory in which this restriction is relaxed exists). The essential point is that our *batching* and *packing* processes interact in a fairly subtle way. We describe our approach informally as follows. To commit to the batch $(t_u(X_0, \ldots, X_{\ell-1}))_{u \in \mathcal{B}_m}$, $\mathcal{P}$ applies Definition 2.1 independently to each of its polynomials, obtaining in this way the batch $(t'_u(X_0, \ldots, X_{\ell'-1}))_{u \in \mathcal{B}_m}$ of packed polynomials. $\mathcal{P}$ defines the $\ell' + m$-variate polynomial $T(X_0, \ldots, X_{\ell'-1}, Y_0, \ldots, Y_{m-1})$ over $\mathcal{T}_\tau$ in such a way that, for each $u \in \mathcal{B}_m$ and each $v \in \mathcal{B}_{\ell'}$, $T(v_0, \ldots, v_{\ell'-1}, u_0, \ldots, u_{m-1}) = t'_u(v)$ holds. Finally, $\mathcal{P}$ flattens and RS-encodes $T(X_0, \ldots, X_{\ell'-1}, Y_0, \ldots, Y_{m-1})$, exactly as in the commitment phase of Construction 5.1; $\mathcal{P}$ writes $[f]$, say, for the resulting $\ell' + m + \mathcal{R}$-dimensional oracle over $\mathcal{T}_\tau$.

To evaluate the batch $(t_u(X_0, \ldots, X_{\ell-1}))_{u \in \mathcal{B}_m}$ collectively at the point $(r_0, \ldots, r_{\ell-1}) \in \mathcal{T}_\tau^\ell$, $\mathcal{P}$ and $\mathcal{V}$ proceed as follows, given the batch of evaluation claims $(s_u)_{u \in \mathcal{B}_m}$. First, $\mathcal{V}$ samples random scalars $(r''_0, \ldots, r''_{m-1}) \leftarrow \mathcal{T}_\tau^m$. It is a standard consequence of Schwartz–Zippel that, except for with probability at most $\frac{m}{|\mathcal{T}_\tau|}$ over $\mathcal{V}$'s choice of $(r''_0, \ldots, r''_{m-1})$, it suffices for the parties to decide rather the tensor-combined equality:

$$\sum_{u \in \mathcal{B}_m} t_u(r_0, \ldots, r_{\ell-1}) \cdot \widetilde{\mathsf{eq}}(r''_0, \ldots, r''_{m-1}, u_0, \ldots, u_{m-1}) \stackrel{?}{=} \sum_{u \in \mathcal{B}_m} s_u \cdot \widetilde{\mathsf{eq}}(r''_0, \ldots, r''_{m-1}, u_0, \ldots, u_{m-1}). \tag{9}$$

Upon receiving $(r''_0, \ldots, r''_{m-1})$ from $\mathcal{V}$, $\mathcal{P}$ defines the ℓ'-variate polynomial:

$$h(X_0, \ldots, X_{\ell'-1}) := \left(\sum_{u \in \mathcal{B}_m} \varphi_1(t'_u)(X_0, \ldots, X_{\ell'-1}) \cdot \varphi_0(\widetilde{\mathsf{eq}}(r'', u)) \right) \cdot \varphi_0(\widetilde{\mathsf{eq}}(r_\kappa, \ldots, r_{\ell-1}, X_0, \ldots, X_{\ell'-1})).$$

$\mathcal{P}$ sends $s_0 := \sum_{w \in \mathcal{B}_{\ell'}} h(w)$ to $\mathcal{V}$, who destructures $(s_{0,v})_{v \in \mathcal{B}_\kappa} := s_0$ and checks:

$$\sum_{u \in \mathcal{B}_m} s_u \cdot \widetilde{\mathsf{eq}}(r''_0, \ldots, r''_{m-1}, u_0, \ldots, u_{m-1}) \stackrel{?}{=} \sum_{v \in \mathcal{B}_\kappa} s_{0,v} \cdot \widetilde{\mathsf{eq}}(r_0, \ldots, r_{\kappa-1}, v_0, \ldots, v_{\kappa-1}). \tag{10}$$

The parties then proceed exactly in the main evaluation phase of Construction 5.1. That is, for ℓ' rounds, they sumcheck $h(X_0, \ldots, X_{\ell'-1})$ and FRI-fold $\mathcal{P}$'s initial oracle $[f]$.

After proceeding in this way for ℓ' rounds, $\mathcal{V}$ obtains the reduced sumcheck claim $s_{\ell'}$. $\mathcal{P}$ moreover sends to $\mathcal{V}$ in the clear the *entire* message underlying its FRI oracle $f^{(\ell')}$. This message is a univariate polynomial over $\mathcal{T}_\tau$ of degree less than 2^m; we write $(c_u)_{u \in \mathcal{B}_m}$ for its coordinates (with respect to the ℓ'^{th}-order novel polynomial basis). $\mathcal{V}$ finally checks:

$$s_{\ell'} \stackrel{?}{=} \left(\sum_{u \in \mathcal{B}_m} \varphi_1(c_u) \cdot \varphi_0(\widetilde{\mathsf{eq}}(r'', u)) \right) \cdot \widetilde{\mathsf{eq}}\big(\varphi_0(r_\kappa), \ldots, \varphi_0(r_{\ell-1}), \varphi_1(r'_0), \ldots, \varphi_1(r'_{\ell'-1})\big). \tag{11}$$

In its final phase, $\mathcal{V}$ reconstructs the codeword $f^{(\ell')} : S^{\ell'} \to \mathcal{T}_\tau$, by re-encoding $(c_u)_{u \in \mathcal{B}_m}$ with respect to the ℓ'^{th}-order basis. Finally, $\mathcal{V}$ launches its usual FRI querying procedure (we note, however, that $\mathcal{P}$'s initial oracle hasn't been "folded all the way").

We explain our approach in the following way. If the tensor-combined variant (9) of $\mathcal{P}$'s initial claim is false, then $\mathcal{P}$ can pass $\mathcal{V}$'s check (10) only by kicking off their sumcheck with the false initial claim $s_0 \neq \sum_{w \in \mathcal{B}_{\ell'}} h(w)$. By the standard argument, during the course of the sumcheck, this false claim will, except with low probability, induce the further false claim $s_{\ell'} \neq h(r'_0, \ldots, r'_{\ell'-1})$. In order to argue that $\mathcal{V}$ will reject (11), then, it suffices to argue that $\mathcal{P}$'s final FRI message $(c_u)_{u \in \mathcal{B}_m}$ will *equal* exactly the set of evaluations $\big(t_u(r'_0, \ldots, r'_{\ell'-1})\big)_{u \in \mathcal{B}_m}$. To argue this, we simply need a variant of Theorems 3.12 and 3.16 in which we only *partially* fold $\mathcal{P}$'s initial FRI message $(T(v \parallel u))_{(v,u) \in \mathcal{B}_{\ell'} \times \mathcal{B}_m}$.

References

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *International Colloquium on Automata, Languages, and Programming*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. Leibniz International Proceedings in Informatics. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 14:1–14:17.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *International Conference on Theory of Cryptography*. Vol. 9986. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 31–60. ISBN: 978-3-662-53644-5. DOI: 10.1007/978-3-662-53644-5_2.

[Ben+23] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. "Proximity Gaps for Reed–Solomon Codes". In: *Journal of the ACM* 70.5 (Oct. 2023). DOI: 10.1145/3614423.

[Ber15] Elwyn Berlekamp. *Algebraic Coding Theory*. Revised Edition. World Scientific Publishing, 2015.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. "HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates". In: *Advances in Cryptology – EUROCRYPT 2023*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023.

[Chi+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Lecture Notes in Computer Science. Full version. Cham: Springer International Publishing, 2020, pp. 738–768. ISBN: 978-3-030-45721-1. DOI: 10.1007/978-3-030-45721-1_26.

[DP23] Benjamin E. Diamond and Jim Posen. *Succinct Arguments over Towers of Binary Fields*. Cryptology ePrint Archive, Paper 2023/1784. 2023. URL: https://eprint.iacr.org/2023/1784.

[DP24] Benjamin E. Diamond and Jim Posen. "Proximity Testing with Logarithmic Randomness". In: *IACR Communications in Cryptology* 1.1 (Apr. 9, 2024). ISSN: 3006-5496. DOI: 10.62056/aksdkp10.

[FP97] John L. Fan and Christof Paar. "On efficient inversion in tower fields of characteristic two". In: *Proceedings of IEEE International Symposium on Information Theory*. 1997.

[GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd Edition. Cambridge University Press, 2013.

[Gru24] Angus Gruen. *Some Improvements for the PIOP for ZeroCheck*. Cryptology ePrint Archive, Paper 2024/108. https://eprint.iacr.org/2024/108. 2024. URL: https://eprint.iacr.org/2024/108.

[Gur06] Venkatesan Guruswami. *Algorithmic Results in List Decoding*. Vol. 2. Foundations and Trends in Theoretical Computer Science 2. now publishers, 2006.

[Hab22] Ulrich Haböck. *A summary on the FRI low degree test*. Cryptology ePrint Archive, Paper 2022/1216. 2022. URL: https://eprint.iacr.org/2022/1216.

[Lan02] Serge Lang. *Algebra*. Revised Third Edition. Vol. 211. Graduate Texts in Mathematics. Springer, 2002.

[LCH14] Sian-Jheng Lin, Wei-Ho Chung, and Yunghsiang S. Han. "Novel Polynomial Basis and Its Application to Reed–Solomon Erasure Codes". In: *IEEE 55th Annual Symposium on Foundations of Computer Science*. 2014, pp. 316–325. DOI: 10.1109/FOCS.2014.41.

[Li+18] Wen-Ding Li, Ming-Shing Chen, Po-Chun Kuo, Chen-Mou Cheng, and Bo-Yin Yang. "Frobenius Additive Fast Fourier Transform". In: *ACM International Symposium on Symbolic and Algebraic Computation*. 2018. ISBN: 9781450355506. DOI: 10.1145/3208976.3208998.

[RR24] Noga Ron-Zewi and Ron Rothblum. "Local Proofs Approaching the Witness Length". In: *Journal of the ACM* 71.3 (June 2024). DOI: 10.1145/3661483. URL: https://doi.org/10.1145/3661483.

[Set20] Srinath Setty. "Spartan: Efficient and General-Purpose zkSNARKs Without Trusted Setup". In: *Advances in Cryptology – CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Cham: Springer International Publishing, 2020, pp. 704–737. ISBN: 978-3-030-56877-1. DOI: 10.1007/978-3-030-56877-1_25.

[Tha22] Justin Thaler. *Proofs, Arguments and Zero-Knowledge*. Vol. 4. Foundations and Trends in Privacy and Security 2–4. now publishers, 2022.

[Wie88] Doug Wiedemann. "An Iterated Quadratic Extension of $GF(2)$". In: *The Fibonacci Quarterly* 26.4 (1988), pp. 290–295.

[ZCF23] Hadas Zeilberger, Binyi Chen, and Ben Fisch. *BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes*. Cryptology ePrint Archive, Paper 2023/1705. 2023. URL: https://eprint.iacr.org/2023/1705.