BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes

Hadas Zeilberger*[1], Binyi Chen†[2], and Ben Fisch‡[1]

[1]Yale University
[2]Espresso Systems

February 22, 2024

Abstract

This works introduces `BaseFold`, a new *field-agnostic* Polynomial Commitment Scheme (PCS) for multilinear polynomials that has $O(\log^2(n))$ verifier costs and $O(n \log n)$ prover time. An important application of a multilinear PCS is constructing Succinct Non-interactive Arguments (SNARKs) from multilinear polynomial interactive oracle proofs (PIOPs). Furthermore, field-agnosticism is a major boon to SNARK efficiency in applications that require (or benefit from) a certain field choice.

Our inspiration for `BaseFold` is the Fast Reed-Solomon Interactive-Oracle Proof of Proximity (`FRI` IOPP), which leverages two properties of Reed-Solomon (RS) codes defined over "FFT-friendly" fields: $O(n \log n)$ encoding time, and a second property that we call *foldability*. We first introduce a generalization of the `FRI` IOPP that works over any foldable linear code in linear time. Second, we construct a new family of linear codes which we call *random foldable codes*, that are a special type of punctured Reed-Muller codes, and prove tight bounds on their minimum distance. Unlike RS codes, our new codes are foldable and have $O(n \log n)$ encoding time over *any* sufficiently large field. Finally, we construct a new multilinear PCS by carefully interleaving our IOPP with the classical sumcheck protocol, which also gives a new multilinear PCS from `FRI`.

`BaseFold` is 2-3 times faster than prior multilinear PCS constructions from `FRI` when defined over the same finite field. More significantly, using Hyperplonk (Eurocrypt, 2022) as a multilinear PIOP backend for apples-to-apples comparison, we show that `BaseFold` results in a SNARK that has better concrete efficiency across a range of field choices than with any prior multilinear PCS in the literature. Hyperplonk with `BaseFold` has a proof size that is more than 10 times smaller than Hyperplonk with

*hadas.zeilberger@yale.edu
†binyi@cs.stanford.edu
‡ben.fisch@yale.edu

Brakedown and its verifier is over 30 times faster for circuits with more than 2^{20} gates. Compared to `FRI`, Hyperplonk with `BaseFold` retains efficiency over *any* sufficiently large field. For illustration, with `BaseFold` we can prove ECDSA signature verification over the secp256k1 curve more than 20 times faster than Hyperplonk with `FRI` and the verifier is also twice as fast. Proofs of signature verification have many useful applications, including offloading blockchain transactions and enabling anonymous credentials over the web.

1 Introduction

A *polynomial commitment scheme* (PCS) [50, 58] is a powerful cryptographic primitive that allows a prover to commit to a polynomial $f \in \mathbb{F}[x]$ of degree d using a short commitment and later, given $\alpha, \beta \in \mathbb{F}$, create a proof that it knows the committed polynomial f of degree at most d satisfying $\beta = f(\alpha)$. The concept was first introduced in the seminal paper by Kate, Zaverucha, and Goldberg (KZG) [50] for univariate polynomials and was later extended to multivariate polynomials [58]. A PCS is an integral building block for many important cryptographic applications including verifiable secret sharing [37], succinct non-interactive arguments of knowledge (SNARKs) [35, 30, 33, 60, 42, 11, 41], proofs of retrievability [49], data availability sampling [48][1], and other authenticated data structures.

SNARKs enable a prover to convince a resource-constrained verifier (with input x) that it knows a witness $w \in \{0,1\}^*$ such that a circuit C is satisfiable with respect to (x, w) (i.e. $C(x, w) = 0$). Plonk [41] is a widely-used SNARK that combines a univariate polynomial commitment scheme with an information-theoretic protocol called *polynomial interactive oracle proofs* (PIOPs) [30, 35]. In a Plonk circuit, each gate is represented as an algebraic formula of the input/output wires and some preprocessed constant values. The *degree* of a gate is its total degree as a multivariate polynomial (e.g., a multiplication gate $a \cdot b = c$ with input wires a, b and output wire c has degree 2). High-degree gates can be used to express the same computation in significantly fewer gates than with only multiplication and addition gates [1, 66]. (For example, Xiong et. al. [66] achieved a $50\times$ reduction in the number of gates required for a specific application by incorporating a customized gate with a maximum degree of 6.) However, for a Plonk circuit with n degree-d gates, the corresponding PIOP prover would need to evaluate the gate formula at nd points and run a few Fast Fourier Transforms (FFTs) of degree nd, which have time complexity $O(nd(d + \log nd))$. When n and d are large (e.g., $n = 2^{20}$), this becomes a dominating cost in prover complexity. STARKs [11] face similar efficiency challenges. In contrast, Hyperplonk [32] and SuperSpartan [61] use multivariate PIOPs that avoid these expensive high-degree FFTs and bring down the cost to $O(nd \log^2 d)$. These protocols require a PCS for *multilinear polynomials* in order to compile the multivariate PIOP into a SNARK. Moreover, the efficiency of this multilinear PCS is critical to the prover efficiency

[1] A planned improvement to Ethereum uses the KZG PCS for "data availability sampling", which is related to proofs of retrievability, https://www.eip4844.com/.

of the resulting SNARK.

Field-Agnosticism

A field-agnostic PCS is a PCS that works over any sufficiently large finite field. To see why field-agnosticism is important, suppose an application is confined to the finite field $\mathbb{F}_p$ but the SNARK is defined over a finite field $\mathbb{F}_q$. In this case, the "modulo p" operation is explicitly encoded into the circuit and needs to be invoked for each multiplication gate. For ECDSA verification circuits, for instance, this is the difference between a circuit having 2^{14} constraints versus a circuit having 2^{20} constraints (see Section 6 for more details). A *field-agnostic* multilinear PCS is therefore ideal as it enables the SNARK prover to avoid this large constant overhead.

1.1 Polynomial Commitment Schemes from Error Correcting Codes

Interactive Oracle Proofs of Proximity (IOPP) were introduced independently by both [15] and [59]. An IOPP is a special proof system for proving that a committed vector over a field $\mathbb{F}$ is close to a codeword in some linear error-correcting code $C \subseteq \mathbb{F}$. Polynomial commitment schemes with efficient provers can be constructed from IOPPs [52, 43, 27, 33, 53].

There are two known families of IOPP-based polynomial commitment schemes that offer fast provers and tolerable verifier and communication costs; an IOPP for Reed-Solomon codes called Fast Reed Solomon Interactive Proof of Proximity (FRI [10]) and an IOPP from tensor codes [26]. This second family of constructions is based on ideas first presented in Ligero [3], and was most recently improved upon in Brakedown [43]. While both families of IOPP-based PCS are practical, they present a rather limiting set of tradeoffs. Without relying on proof recursion (which adds significant efficiency overhead), Brakedown achieves field-agnosticism and has linear-time provers, but it has relatively slow verifier and large proof (i.e., $O(\sqrt{n})$). On the other hand, FRI achieves polylogarithmic proof size and verifier times but with a slower prover (i.e., $O(n \log n)$) and lacks field-agnosticism.

Recently, in ECFFT2 [12], the authors present a method for using `FRI` over any finite field, however to use it within a *multilinear* PCS, the runtime of their prover is asymptotically $O(n \log^2 n)$. Bordage, Lhotel, Nardi, and Randriam [28] also generalize `FRI` to a wide class of polynomial codes, i.e., codes that consist of evaluations of multivariate polynomials. They also present an IOPP for punctured Reed-Muller codes, which they call "Short Reed-Muller Codes". However they use a different "puncturing" (i.e., a different evaluation domain) and their verifier is linear in the message size. The IOPP constructed in this work has a verifier that runs in logarithmic time in the message size.

1.2 Our Contributions

We have three main contributions, each of independent interest.

First, we formalize the notion of a *foldable code* and introduce a generalization of the `FRI` IOPP (which we refer to as the `BaseFold` IOPP) that can be used with any foldable code.

Our second contribution is a new linear code family, which we call *Random Foldable Code* (RFCs). This is a special case of punctured Reed-Muller codes (see Appendix D for more details) over which messages can be encoded with approximately $n \log n$ field additions and $0.5n \log n$ field multiplications. We prove tight bounds on the minimum distance of RFCs. For instance, an RFC with message length 2^{25} and rate $\frac{1}{8}$ over a 256-bit finite field has a relative minimum distance of 0.728 with overwhelming probability (See Table 1 and Appendix C for more details on concrete bounds). It is well-known that the highest possible minimum relative distance for a code with rate $\frac{1}{8}$ is approximately 0.875 (see Definition 3). Beyond its practicality, this result also implies that with overwhelming probability, a uniformly sampled *foldable code* (which enables efficient PCS provers) also has good relative minimum distance (which enables efficient PCS verifiers). An open question is to find an explicit construction of field-agnostic foldable linear codes with a high minimum relative distance.

Our final contribution is a construction of a multilinear polynomial commitment scheme based on interleaving the `BaseFold` IOPP (using any foldable linear code) with the classical sumcheck protocol [56]. There are two key insights behind the construction. First, sumcheck can be used to transform an *arbitrary* multilinear evaluation claim into a random evaluation claim. Secondly, the BaseFold IOPP (where the verifier makes $O_\lambda(\log n)$ random queries) can be understood as an interactive protocol for proving the multilinear evaluation of a polynomial at a *random* point. Thus we can use this random evaluation to answer the last query of the verifier in the sumcheck protocol to enable proving *arbitrary* multilinear evaluation statement, so long as we interleave the IOPP and the sum-check together with shared verifier randomness. The prover time of this PCS is $O(n)$ with a small leading constant.[2] The prover of the `BaseFold` PCS is $2-3$ times faster than the existing multilinear `FRI` construction, derived from [53], which we benchmark in Section 6.

1.2.1 Improvements in Concrete Performance

In addition to the above, we also report the following concrete performance improvements of the `BaseFold` PCS. As mentioned earlier, `BaseFold` can be defined over any (sufficiently large) finite field, maintains polylogarithmic verifier costs and has a prover that runs in time $O(n \log n)$. In Section 6, we show that Hyperplonk[Basefold] can prove an ECDSA signature verification 23 times faster than HyperPlonk[ZeromorphFri] [53]. Compared to Hyperplonk[Brakedown] (which has comparable proving time to Hyperplonk[Basefold]), the proof size is 5.15 times smaller and the verifier is 33 times faster. We also compare the `BaseFold` code to the ECFFT2 code. ECFFT2 has encoding time $O(n \log^2(n))$ when

[2] The committing complexity is $O(n \log n)$ as the prover needs to compute the encoding of the message. After that, the prover can generate PCS evaluation proofs in linear time.

used within a multilinear PCS. For $n = 2^{20}$, it is about 16 times slower than `BaseFold`'s encoding algorithm.

Finally, we find that when *instantiated with the same 256-bit finite field*, the prover time of the `BaseFold` PCS is 2–3 times faster compared to ZeromorphFri [53], while the proof size is 2–3 times larger. We provide more detailed performance comparisons of `BaseFold`, Brakedown, ZeromorphFri, and ECFFT2 in Section 6.

1.3 Overview

Foldable Linear Codes and `BaseFold` IOPP. Let $d \in \mathbb{Z}$ and let C_d be a linear code with rate $\frac{1}{c}$ that encodes messages of length $k_d = 2^d$ into codewords of length $n_d = ck_d$. We consider C_d to be *foldable* if its generator matrix, G_d is a $k_d \times n_d$ matrix that is equal up to row permutation to

$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix} \tag{1}$$

where G_{d-1} is a foldable linear code that encodes messages of length $\frac{k_d}{2}$ into codewords of length $\frac{n_d}{2}$ and T_d, T'_d are both $\frac{n_d}{2} \times \frac{n_d}{2}$ diagonal matrices. For instance, Reed-Solomon codes are foldable when defined over a cyclic group H such that $|H|$ is a power of 2 (we discuss this further in Appendix E). In Section 4 we will present the `BaseFold` IOP of Proximity (IOPP), which can be used with any foldable linear code with adequate minimum distance, (and is in fact equivalent to `FRI` when instantiated with Reed-Solomon codes).

Random Foldable Linear Codes. Armed with the general IOPP for *foldable linear codes*, our goal is to design a linear code that is foldable and efficiently encodable regardless of its finite field. We accomplish this by setting T_i to a diagonal matrix whose entries are each a uniform random sample from $\mathbb{F}^\times$ and setting $T'_i = -T_i$. We show that with overwhelming probability over choice of $(T_1, .., T_d)$, this code has relative minimum distance equal to

$$1 - \left(\frac{\epsilon_{\mathbb{F}}^d}{c} + \frac{\epsilon_{\mathbb{F}}}{\log|\mathbb{F}|} \sum_{i=1}^{d} (\epsilon_{\mathbb{F}})^{d-i} \left(0.6 + \frac{2\log(n_i/2) + \lambda}{n_i} \right) \right)$$

where c is the inverse of the rate of the code, $\epsilon_{\mathbb{F}} = \frac{\log|\mathbb{F}|}{\log|\mathbb{F}| - 1.001}$, n_i is the block-length of the code, and d is the logarithm of the message length. For instance, if we set $|\mathbb{F}|$ to be equal to 2^{61}, $c = 16$, and $d = 15$, then the relative mininum distance is 0.572. If we set $|\mathbb{F}|$ to be 2^{256}, $c = 8$, and $d = 15$, then the minimum distance is .76.

The Basefold Multilinear Polynomial Commitment Scheme. It is known that by using the classic sum-check protocol with a multilinear extension, we can transform the evaluation check of a multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ at a point $\mathbf{z} \in \mathbb{F}^d$ into an evaluation check of f at a random point $\mathbf{r} \in \mathbb{F}^d$. Interestingly, the BaseFold (and

FRI-based) IOPP has a similar structure. Given an input oracle that is an encoding of a polynomial f, the last oracle sent by the honest prover in the IOPP protocol is exactly an encoding of a random evaluation $f(\mathbf{r})$. Thus a natural way to build a PCS is to set the commitment as the Merkle commitment of the encoding of f. During the evaluation phase, the prover and verifier run an IOPP protocol and a sumcheck protocol in parallel using the same set of round challanges $\mathbf{r} = (r_1, \ldots, r_d) \in \mathbb{F}^d$. Finally the verifier checks that the claimed evaluation $y \in \mathbb{F}$ in the last round of the sumcheck protocol is consistent with the last prover message of the IOPP protocol.

Proving that this construction satisfies evaluation binding and knowledge soundness, however, is non-trivial. Recall that the IOPP soundness only states that to pass the verification with high probability, the committed oracle should be close to a codeword. But it does not rule out the possibility that at some round, the prover message may shift from the folding of the previous message to the encoding of a different message. It is therefore not straightforward to argue *evaluation binding*, i.e. that the last prover message in the IOPP protocol is an encoding of the evaluation of the committed polynomial f. Moreover, even if the protocol is evaluation binding, it is still non-trivial to recover the polynomial f and obtain knowledge soundness if the linear code we use is not efficiently decodable. We solve these issues by constructing an extractor that obtains a sufficient number of *correct* evaluations of f from the PCS prover. Via a careful analysis, we argue that with high probability, the extractor can recover the polynomial f from the set of evaluations obtained. We provide more technical details in Sect. 5.1.

1.4 Other Related Work

1.4.1 Error Correcting Codes

Reed-Muller codes with *uniformly* random puncturings were proven to have good list decoding properties in [45]. However, a uniformly sampled puncturing is unlikely to have the recursive structure that enables efficient encoding. An efficiently *list-decodable* puncturing of Reed-Muller codes was discovered in [44]. However, this code is not efficiently encodable. An interrsting direction is to investigate whether techniques from [44] can be used to *explicitly* construct a field-agnostic foldable punctured Reed-Muller code.

1.4.2 Polynomial Commitment Schemes

Orion [65] presents a SNARK that has a linear-time prover and polylogarithmic proof size, but it is not field-agnostic, and the verifier runs in linear time in the non-interactive setting [39].

The KZG [50] PCS has very short evaluation proofs but can only be used to commit to polynomials (of degree at most d) over a (pairing-friendly) prime field $\mathbb{F}_p$ and requires $\Theta(d)$ scalar multiplication operations over an elliptic curve group $\mathbb{G}$. Moreover, p needs to be exponentially large in some security parameter λ so that the t-Diffie-Hellman inversion (t-DHI) and related assumptions holds in $\mathbb{G}$.

See Appendix A for a summary of other prior work on Interactive Oracle Proofs and Polynomial Commitment Schemes.

1.5 Roadmap

The remainder of this paper proceeds as follows. In Section 2, we present definitions and statements that will be useful for the remainder of the paper. In Section 3, we present our new error-correcting code and state and prove its minimum distance. In Section 4, we describe the `BaseFold` IOPP, and prove its correctness and soundness. In Section 5, we compile the `BaseFold` IOPP into a multilinear polynomial commitment scheme using the sumcheck protocol and prove its knowledge soundness. In Section 6, we analyze the performance of `BaseFold` and compare it with other multilinear polynomial commitment schemes.

2 Preliminaries

Notation. A diagonal matrix, T is a square matrix that only has non-zero entries along the diagonal. For a diagonal matrix, T, denote the vector of diagonal entries as $\mathsf{diag}(T)$ and denote the matrix with diagonal entries $-1 \cdot \mathsf{diag}(T)$ as $-T$. For a matrix $\mathbf{G}$, $\mathbf{G}^\top$ is the transpose of $\mathbf{G}$. For a vector $\mathbf{v} \in \mathbb{F}^{2n}$, we write $\mathbf{v}_l$ as the first n components of $\mathbf{v}$ and $\mathbf{v}_r$ as the last n components of $\mathbf{v}$. We will sometimes write $\mathbf{v}$ as $(\mathbf{v}_l, \mathbf{v}_r)$ or $(\mathbf{v}_l || \mathbf{v}_r)$. $\circ$ denotes the Hadamard product of two vectors. For a finite field $\mathbb{F}$, we use $\mathbb{F}^\times$ to denote $\mathbb{F} \setminus \{0\}$.

Definition 1 (Linear Code)**.** *A linear error-correcting code with message length k and codeword length n is an injective mapping from $\mathbb{F}^k$ to a linear subspace $C \subseteq \mathbb{F}^n$. C is associated with a generator matrix, $G \in \mathbb{F}^{k \times n}$ such that the rows of G are a basis of C and the encoding of a vector $\mathbf{v} \in \mathbb{F}^k$ is $\mathbf{v} \cdot G$. The minimum Hamming distance of a code is the minimum on the number of different entries between any two different codewords $c_1, c_2 \in C$. If C has a minimum distance $d \in [0, n]$, we say that C is an $[n, k, d]$ code and use Δ_C to denote d/n—the relative minimum distance.*

Next, as in [9], we define a slightly altered version of relative minimum distance, called *coset relative minimum distance*, denoted as Δ^*. We will use this definition in our proofs of IOPP and PCS soundness.

Definition 2 (Coset Relative Minimum Distance)**.** *Let n be an even integer and let C be a $[n, k, d]$ error-correcting code. Let $\mathbf{v} \in \mathbb{F}^n$ be a vector and let $c \in C$ be a codeword. The coset relative distance $\Delta^*(\mathbf{v}, c)$ between $\mathbf{v}$ and c is*

$$\Delta^*(\mathbf{v}, c) = \frac{2|\{j \in [1, n/2] : \mathbf{v}[j] \neq c[j] \vee \mathbf{v}[j + n/2] \neq c[j + n/2]\}|}{n}.$$

The relative minimum distance of $\mathbf{v} \in \mathbb{F}^n$ *to the code,* C*, which we denote as* $\Delta^*(\mathbf{v}, C)$ *is defined as follows:*

$$\Delta^*(\mathbf{v}, C) = \min_{c \in C} \Delta^*(\mathbf{v}, c)\,.$$

Definition 3 (Maximum Distance Seperable Code)**.** *Let* C *be an* $[n, k, d]$ *code. Then* C *is Maximum Distance Seperable (MDS) if* $d = n - k + 1$.

A list-decodable code is one where for a certain radius around a codeword, there is an upper bound on the number of other neighboring codewords that can be contained inside that radius. The Johnson bound gives a radius around every code in terms of its minimum distance, within which it is list-decodable. This will be useful in the soundness analysis of the `BaseFold` IOPP.

Definition 4 (Johnson Bound)**.** *For every* $\gamma \in (0, 1]$*, define* $J_\gamma : [0, 1] \to [0, 1]$ *as the function* $J_\gamma(\lambda) := 1 - \sqrt{1 - \lambda(1 - \gamma)}$.

Foldable codes. We define *foldable linear codes* that generalize Reed-Solomon codes used in FRI [9].

Definition 5 ((c, k_0, d)-foldable linear codes)**.** *Let* $c, k_0, d \in \mathbb{N}$ *and let* $\mathbb{F}$ *be a finite field. A linear code* $C_d : \mathbb{F}^{k_0 \cdot 2^d} \to \mathbb{F}^{ck_0 \cdot 2^d}$ *with generator matrix* $\mathbf{G}_d$ *is called **foldable** if there exists a list of generator matrices* $(\mathbf{G}_0, \ldots, \mathbf{G}_{d-1})$ *and diagonal matrices* $(T_0, \ldots, T_{d-1})$ *and* $(T'_0, \ldots, T'_{d-1})$*, such that for every* $i \in [1, d]$*, (i) the diagonal matrices* $T_{i-1}, T'_{i-1} \in \mathbb{F}^{ck_0 \cdot 2^{i-1} \times ck_0 \cdot 2^{i-1}}$ *satisfies that* $\mathsf{diag}(T_{i-1})[j] \neq \mathsf{diag}(T'_{i-1})[j]$ *for every* $j \in [ck_0 \cdot 2^{i-1}]$*; and (ii) the matrix* $\mathbf{G}_i \in \mathbb{F}^{k_0 \cdot 2^i \times ck_0 \cdot 2^i}$ *equals (up to row permutation)*

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{G}_{i-1} & \mathbf{G}_{i-1} \\ \mathbf{G}_{i-1} \cdot T_{i-1} & \mathbf{G}_{i-1} \cdot T'_{i-1} \end{bmatrix}.$$

2.1 Interactive Oracle Proofs and Polynomial Commitments

Interactive oracle proofs (IOPs). We briefly recall the definition of interactive oracle proofs (IOPs) [20, 59, 4]. A k-round public coin IOP, $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$, for a relation $\mathcal{R}$ runs as follows: Initially, $\mathcal{P}$ sends an oracle string π_0. In each round $i \in [1, k]$, the verifier samples and sends a random challenge α_i, and the prover replies with an oracle string π_i. After k rounds of communications, the verifier $\mathcal{V}$ queries some entries of the oracle strings $\pi_0, \pi_1, \ldots, \pi_k$ and outputs a bit b.

Definition 6 (IOPs)**.** *Let* $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$ *be a* k*-round public coin IOP protocol for a relation* $\mathcal{R}$*. We say that* IOP *is **complete** if for every* $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$*,*

$$\Pr_{\alpha_1, \ldots, \alpha_k}\left[\mathcal{V}^{\pi_0, \pi_1, \ldots, \pi_k}(\mathsf{x}, \alpha_1, \ldots, \alpha_k) = 1 \;\middle|\; \begin{array}{c} \pi_0 \leftarrow \mathcal{P}(\mathsf{x}, \mathsf{w}) \\ \pi_1 \leftarrow \mathcal{P}(\mathsf{x}, \mathsf{w}, \alpha_1) \\ \cdots \\ \pi_k \leftarrow \mathcal{P}(\mathsf{x}, \mathsf{w}, \alpha_1, \ldots, \alpha_k) \end{array} \right] = 1\,.$$

We say that IOP *has* ***soundness error*** ν *if for any* $\mathsf{x} \notin L(\mathcal{R})$ *and any unbounded adversary* $\mathcal{A}$*,*

$$\Pr_{\alpha_1,\ldots,\alpha_k}\left[\mathcal{V}^{\pi_0,\pi_1,\ldots,\pi_k}(\mathsf{x},\alpha_1,\ldots,\alpha_k)=1 \;\middle|\; \begin{array}{c}\pi_0 \leftarrow \mathcal{A}(\mathsf{x})\\ \pi_1 \leftarrow \mathcal{A}(\mathsf{x},\mathsf{w},\alpha_1)\\ \ldots\\ \pi_k \leftarrow \mathcal{A}(\mathsf{x},\alpha_1,\ldots,\alpha_k)\end{array}\right] \le \nu\,.$$

IOPs of proximity. IOP of proximity (IOPP) is similar to IOP with the specialty that the witness w is also sent as an oracle string. The verifier can query w as an oracle but will only query $q_{\mathsf{w}} \ll |\mathsf{w}|$ entries of w. The soundness states that if w is far from any valid witness, then the verifier rejects with high probability.

Definition 7 (IOPPs)**.** *A public coin IOP* $\mathsf{IOP} = (\mathcal{P},\mathcal{V})$ *for relation* $\mathcal{R}$ *is an IOP of proximity if it satisfies IOP completeness, and the* $\nu(\cdot)$*-**IOPP soundness** holds for some function* $\nu(\cdot)$*: for every* (x,w) *where* w *is* δ*-far (in relative Hamming distance) from any* w' *such that* $(\mathsf{x},\mathsf{w}') \in \mathcal{R}$*, it holds that for any unbounded adversary* $\mathcal{A}$*,*

$$\Pr_{\alpha_1,\ldots,\alpha_k}\left[\mathcal{V}^{\mathsf{w},\pi_0,\ldots,\pi_k}(\mathsf{x},\alpha_1,\ldots,\alpha_k)=1 \;\middle|\; \begin{array}{c}\pi_0 \leftarrow \mathcal{A}(\mathsf{x},\mathsf{w})\\ \pi_1 \leftarrow \mathcal{A}(\mathsf{x},\mathsf{w},\alpha_1)\\ \ldots\\ \pi_k \leftarrow \mathcal{A}(\mathsf{x},\mathsf{w},\alpha_1,\ldots,\alpha_k)\end{array}\right] \le \nu(\delta)\,.$$

Let C be any linear code. In this paper, we consider the relation $\mathcal{R}_C$ where (x,w) is in the relation $\mathcal{R}_C$ if and only if x is the code parameters and $\mathsf{w} \in C$ is a valid codeword.

Polynomial commitment schemes. We recall the definition from [43, 30].

Definition 8 (Polynomial commitment scheme)**.** *A multilinear polynomial commitment scheme* PC *over a field* $\mathbb{F}$ *consists of a tuple of algorithms* $(\mathsf{Setup},\mathsf{Commit},\mathsf{Open},\mathsf{Eval})$*:*

- $\mathsf{Setup}(1^\lambda, d) \to \mathsf{pp}$ *takes security parameter* λ *and* $d \in \mathbb{N}$ *(i.e. the number of variables in a polynomial), outputs public parameter* pp*.*
- $\mathsf{Commit}(\mathsf{pp}, f) \to C$ *takes a multilinear polynomial* $f \in \mathbb{F}[X_1,\ldots,X_d]$ *and outputs a commitment* C*.*
- $\mathsf{Open}(\mathsf{pp}, C, f) \to b$ *takes a commitment* C *and a multilinear polynomial* $f \in \mathbb{F}[X_1,\ldots,X_d]$*, outputs a bit* b*.*
- $\mathsf{Eval}(\mathsf{pp}, C, \mathbf{z}, y; f)$ *is a protocol between the prover* $\mathcal{P}$ *and the verifier* $\mathcal{V}$ *with public input a commitment* C*, an evaluation point* $\mathbf{z} \in \mathbb{F}^d$ *and a value* $y \in \mathbb{F}$*.* $\mathcal{P}$ *additionally knows a multilinear polynomial* $f \in \mathbb{F}[X_1,\ldots,X_d]$ *and* $\mathcal{P}$ *wants to convince* $\mathcal{V}$ *that* f *is an opening to* C *and* $f(\mathbf{z}) = y$*. The verifier outputs a bit* b *at the end of the protocol.*

The scheme PC *satisfies* ***completeness*** *if for any bound* $d \in \mathbb{N}$*, for any multilinear polynomial* $f \in \mathbb{F}[X_1, \ldots, X_d]$ *and any point* $\mathbf{z} \in \mathbb{F}^d$

$$\Pr\left[\ \mathsf{Eval}(\mathsf{pp}, C, \mathbf{z}, f(z); f) = 1 \ \middle|\ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ C \leftarrow \mathsf{Commit}(\mathsf{pp}, f) \end{array}\right] = 1\,.$$

PC *is* ***binding*** *if for any* $d \in \mathbb{N}$ *and PPT adversray* $\mathcal{A}$*,*

$$\Pr\left[\ b_0 = b_1 = 1 \wedge f_0 \neq f_1 \ \middle|\ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (C, f_0, f_1) \leftarrow \mathcal{A}(\mathsf{pp}) \\ b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, C, f_0) \\ b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, C, f_1) \end{array}\right] \leq \mathsf{negl}(\lambda)\,.$$

PC *satisfies* ***knowledge soundness*** *if* Eval *is an argument of knowledge for the relation*

$$\mathcal{R}_{\mathsf{Eval},\mathsf{pp}} = \{[(C, \mathbf{z}, y); f] : f(\mathbf{z}) = y \wedge \mathsf{Open}(\mathsf{pp}, C, f) = 1\}$$

where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d)$*. That is, for any* $d \in \mathbb{N}$ *and any PPT algorithms* $\mathcal{A}$ *and* P^**, there is an expected polynomial-time extractor* Ext *such that for any randomness* r*,*

$$\Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (C, \mathbf{z}, y) \leftarrow \mathcal{A}(\mathsf{pp}, r) \\ \mathsf{Eval}(\mathsf{pp}, C, \mathbf{z}, y)\langle \mathsf{P}^*(r), \mathcal{V}\rangle = 1 \end{array}\right] \approx \Pr\left[\begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d) \\ (C, \mathbf{z}, y) \leftarrow \mathcal{A}(\mathsf{pp}, r) \\ f \leftarrow \mathsf{Ext}(\mathsf{pp}, r) \\ ((C, \mathbf{z}, y); f) \in \mathcal{R}_{\mathsf{Eval},\mathsf{pp}} \end{array}\right].$$

3 Fast Linear Code from Foldable Distributions

In this section, we present a novel random linear code, which is foldable, and efficiently encodable over any sufficiently large finite field. We present its encoding algorithm and analyze its relative minimum distance.

We first define a family of *random foldable distributions.*

Definition 9 ((c, k_0)-foldable distributions)**.** *Fix finite field* $\mathbb{F}$ *and* $c, k_0 \in \mathbb{N}$*. Let* $G_0 \in \mathbb{F}^{k_0 \times ck_0}$ *be the generator matrix of a* $[ck_0, k_0]$*-linear code that is maximum distance separable (MDS)* [3]*, and let* D_0 *be the distribution that outputs* G_0 *with probability* 1*. For every* $i > 0$*, we define inductively the distribution* D_i *that samples generator matrices* $(\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_i)$ *where* $\mathbf{G}_i \in \mathbb{F}^{k_i \times n_i}$ *and* $k_i := k_0 \cdot 2^i$*,* $n_i := ck_i$*:*

1. *Sample* $(\mathbf{G}_0, \ldots, \mathbf{G}_{i-1}) \leftarrow D_{i-1}$*.*
2. *Sample* $\mathsf{diag}(T_{i-1}) \leftarrow\$ (\mathbb{F}^\times)^{n_{i-1}}$ *and define* $\mathbf{G}_i$ *as*

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{G}_{i-1} & \mathbf{G}_{i-1} \\ \mathbf{G}_{i-1} \cdot T_{i-1} & \mathbf{G}_{i-1} \cdot -T_{i-1} \end{bmatrix}.$$

[3] We require this code to be MDS in order to make the distance analysis more straightforward but it is not strictly necessary. The analysis works for any linear codes.

Encoding algorithms for linear foldable codes. Let $\{D_i\}_{i\in\mathbb{N}}$ be a family of (c, k_0)-foldable distributions. For a $d \in \mathbb{N}$, let $(\mathbf{G}_0, .., \mathbf{G}_d) \leftarrow D_d$ be the sampled generator matrices with associated diagonal matrices $(T_0, .., T_{d-1})$. Denote as C_d the (c, k_0, d)-foldable linear code[4] with generator matrix $\mathbf{G}_d \in \mathbb{F}^{k_d \times n_d}$ where $k_d = k_0 2^d$ and $n_d = ck_d$. Next, we describe an encoding algorithm Enc_d for C_d which takes $\frac{dn_d}{2}$ field multiplications and dn_d field additions.

Protocol 1 Enc_d : `BaseFold` Encoding Algorithm

Input: $\mathbf{m} \in \mathbb{F}^{k_d}$
Output: $\mathbf{w} \in \mathbb{F}^{n_d}$ such that $\mathbf{w} = \mathbf{m} \cdot \mathbf{G}_d$
Parameters: $\mathbf{G}_0$ and diagonal matrices $(T_0, T_1, \ldots, T_{d-1})$

1. If $d = 0$ (i.e. $\mathbf{m} \in \mathbb{F}^{k_0}$):
   (a) return $\mathsf{Enc}_0(\mathbf{m})$
2. else
   (a) parse $\mathbf{m} := (\mathbf{m}_l, \mathbf{m}_r)$
   (b) set $\mathbf{l} := \mathsf{Enc}_{d-1}(\mathbf{m}_l)$, $\mathbf{r} := \mathsf{Enc}_{d-1}(\mathbf{m}_r)$ and $\mathbf{t} = \mathsf{diag}(T_{d-1})$
   (c) return $(\mathbf{l} + \mathbf{t} \circ \mathbf{r}, \mathbf{l} - \mathbf{t} \circ \mathbf{r})$

Figure 1: The encoding algorithm for `BaseFold`.

Lemma 1 (Correctness)**.** *Let $c, k_0, d \in \mathbb{N}$ and let $(\mathbf{G}_0, .., \mathbf{G}_d) \leftarrow D_d$ be the sampled generator matrices with associated diagonal matrices $(T_0, .., T_{d-1})$. Then for all $\mathbf{m} \in \mathbb{F}^{k_0 \cdot 2^d}$, we have $\mathsf{Enc}_d(\mathbf{m}) = \mathbf{m} \cdot \mathbf{G}_d$.*

Proof. We proceed by induction on $i \le d$. Fix $c, k_0 \in \mathbb{N}$. Let $i = 0$. Then Protocol 1 returns $\mathsf{Enc}_0(\mathbf{m})$, which by definition equals to $\mathbf{m} \cdot \mathbf{G}_0$. Now suppose, by inductive hypothesis, that $\mathsf{Enc}_i(\mathbf{m}) = \mathbf{m} \cdot \mathbf{G}_i$ for all $\mathbf{m} \in \mathbb{F}^{k_0 \cdot 2^i}$. We now consider the execution of Enc_{i+1}: on Line 2.b of the $(i+1)$-th recursion of Protocol 1, we can replace calls to $\mathsf{Enc}_i(\mathbf{m}_l)$ and $\mathsf{Enc}_i(\mathbf{m}_r)$ with $\mathbf{m}_l \cdot \mathbf{G}_i$ and $\mathbf{m}_r \cdot \mathbf{G}_i$ respectively. Therefore,

$$\begin{aligned}\mathbf{l} + \mathbf{t} \circ \mathbf{r} &= \mathbf{m}_l \cdot \mathbf{G}_i + \mathsf{diag}(T_i) \circ (\mathbf{m}_r \cdot \mathbf{G}_i) = \mathbf{m}_l \cdot \mathbf{G}_i + \mathbf{m}_r \cdot \mathbf{G}_i \cdot T_i \\ \mathbf{l} - \mathbf{t} \circ \mathbf{r} &= \mathbf{m}_l \cdot \mathbf{G}_i - \mathsf{diag}(T_i) \circ (\mathbf{m}_r \cdot \mathbf{G}_i) = \mathbf{m}_l \cdot \mathbf{G}_i - \mathbf{m}_r \cdot \mathbf{G}_i \cdot T_i\,,\end{aligned}$$

hence for all $\mathbf{m} := (\mathbf{m}_l, \mathbf{m}_r) \in \mathbb{F}^{k_0 \cdot 2^{i+1}}$,

$$\mathsf{Enc}_{i+1}(\mathbf{m}) = (\mathbf{l} + \mathbf{t} \circ \mathbf{r}, \mathbf{l} - \mathbf{t} \circ \mathbf{r}) = (\mathbf{m}_l, \mathbf{m}_r) \cdot \begin{bmatrix} \mathbf{G}_i & \mathbf{G}_i \\ \mathbf{G}_i \cdot T_i & \mathbf{G}_i \cdot (-T_i) \end{bmatrix} = \mathbf{m} \cdot \mathbf{G}_{i+1}$$

[4]Note that C_d is foldable as we set $T_i' := -T_i$ for every $i \in [0, d-1]$.

where the last equality holds by definition of $\mathbf{G}_{i+1}$. $\square$

3.1 Proof of Relative Minimum Distance

In this section, we analyze the relative minimum distance of the `BaseFold` code C_d. We start with an overview of the proof techniques and then formally state and prove our main theorem. Finally, we demonstrate concrete bounds for typical instantiations of the code.

Before proving the result, we first recall the famous Rank-Nullity Theorem.

Theorem 1 (Rank-Nullity Theorem)**.** *For any matrix M with m columns over a field $\mathbb{F}$, the rank and the nullity (i.e., the dimension of the kernel) of M sums to m, that is,*

$$\mathsf{rank}(M) + \mathsf{nullity}(M) = m\,.$$

Next, we overview the techniques for analyzing the relative minimum distance of the random foldable code.

Technical overview. It is well known that the minimum distance of a linear code is identical to the minimum Hamming weight of non-zero codewords. Thus it is sufficient to prove that for any nonzero message $\mathbf{m}$, the encoding $\mathsf{Enc}_d(\mathbf{m})$ does not have many zeros.

We start with a strawman idea using induction. Suppose by induction hypothesis that with overwhelming probability (over diagonal matrices $T_0, \ldots, T_{i-1}$), for all $\mathbf{m} \in \mathbb{F}^{k_i} \setminus \{0^{k_i}\}$, the encoding $\mathsf{Enc}_i(\mathbf{m})$ has at most t_i zeros (for some $t_i \in \mathbb{N}$ to be clear later). Now fix any non-zero message $\mathbf{m} = (\mathbf{m}_l, \mathbf{m}_r) \in \mathbb{F}^{2k_i}$, it holds that $\mathsf{Enc}_{i+1}(\mathbf{m}) = (\mathbf{M}_l || \mathbf{M}_r)$ where

$$\mathbf{M}_l = \mathsf{Enc}_i(\mathbf{m}_l) + \mathsf{Enc}_i(\mathbf{m}_r) \circ \mathsf{diag}(T_i)\,, \qquad \mathbf{M}_r = \mathsf{Enc}_i(\mathbf{m}_l) - \mathsf{Enc}_i(\mathbf{m}_r) \circ \mathsf{diag}(T_i)\,.$$

By induction hypothesis, there are at most t_i indices $j \in [n_i]$ where $\mathsf{Enc}_i(\mathbf{m}_l)[j] = \mathsf{Enc}_i(\mathbf{m}_r)[j] = 0$. For each index j' in the rest of $n_i - t_i$ entries, i.e. in entries such that $\mathsf{Enc}_i(\mathbf{m}_l[j'] \neq 0$ or $\mathsf{Enc}_i(\mathbf{m}_r)[j'] \neq 0$, the event that one[5] of $\mathbf{M}_l[j]$, $\mathbf{M}_r[j]$ equals zero is an independent Bernoulli trial with success probability $O(1/|\mathbb{F}|)$ (where the randomness is over $\mathsf{diag}(T_i)[j]$). Thus the probability that $\mathsf{Enc}_{i+1}(\mathbf{m})$ has at least $2t_i + \ell_i$ zeros is approximately $O(n_i 2^{n_i}/|\mathbb{F}|^{\ell_i})$. Unfortunately, to argue that (with high probability over $\mathsf{diag}(T_i)$) the encoding $\mathsf{Enc}_{i+1}(\mathbf{m})$ has no more than $2t_i + \ell_i$ zeros *for any nonzero* $\mathbf{m} \in \mathbb{F}^{2k_i}$, we need to take a union bound over all non-zero messages in $\mathbb{F}^{2k_i}$, which is meaningful only when $\ell_i \gg 2k_i$. This leads to a really weak bound.

Looking more deeply, the bound is loose because we treat every nonzero message $\mathbf{m} = (\mathbf{m}_l, \mathbf{m}_r) \in \mathbb{F}^{2k_i}$ equally and always assume the worst case where for *exactly* t_i indices $j \in [n_i]$, it holds that $\mathsf{Enc}_i(\mathbf{m}_l)[j] = \mathsf{Enc}_i(\mathbf{m}_r)[j] = 0$. However, for many messages $\mathbf{m} \in \mathbb{F}^{2k_i}$, $\mathsf{Enc}_i(\mathbf{m}_l)$ and $\mathsf{Enc}_i(\mathbf{m}_r)$ actually have significantly fewer zeros and we can obtain a much better bound on the number of zeros in $\mathsf{Enc}_{i+1}(\mathbf{m})$.

[5] Note that it's impossible for both of $\mathbf{M}_l[j']$ and $\mathbf{M}_r[j']$ to be zeros as that implies $\mathsf{Enc}_i(\mathbf{m}_l)[j'] = \mathsf{Enc}_i(\mathbf{m}_r)[j'] = 0$, which we ruled out, or $\mathsf{diag}(T_i)[j] = 0$, contradicts the definition of T_i.

We have the following key observation for obtaining a tighter bound: suppose $\mathsf{Enc}_i(\mathbf{m})$ has less than t_i zeros for all non-zero $\mathbf{m} \in \mathbb{F}^{k_i}$, then for any subset $S \subseteq [n_i]$ where $|S| \leq t_i$, the kernel of $\mathsf{Enc}_i[S]$ has size at most $\mathbb{F}^{t_i - |S|}$. Here the kernel of $\mathsf{Enc}_i[S]$ denotes the set of messages in $\mathbb{F}^{k_i}$ whose encoding equals zeros on set S. Intuitively, for a larger set S, there will be fewer messages whose encodings equal zeros on S. To prove it, observe that if T is a subset $[n_i]$ that is *larger than* t_i, the kernel of $\mathsf{Enc}_i[T]$ must be 0^{k_i}, as by assumption, 0^{k_i} is the only message whose encoding has at least t_i zeroes. Note that the partial encoding $\mathsf{Enc}_i[T]$ is essentially a linear map represented by a submatrix, G', of the generator matrix and $\mathsf{Enc}_i[S]$ is a linear map represented by a submatrix of G'. Thus by the rank-nullity theorem, the kernel of $\mathsf{Enc}_i[S]$ has size at most $|\mathbb{F}|^{t_i - |S|}$. We refer to Lemma 2 for more proof details.

Given the above, for any nonzero message $\mathbf{m} = (\mathbf{m}_l, \mathbf{m}_r)$ in $\mathbb{F}^{2k_i}$, let $S \subseteq [n_i]$ be the *maximal set* where both $\mathbf{m}_l$ and $\mathbf{m}_r$ are in the kernel of $\mathsf{Enc}_i[S]$. Using the same argument as in the strawman idea, the probability that $\mathsf{Enc}_{i+1}(\mathbf{m})$ has $2t_i + \ell_i$ zeros is approximately $O(n_i 2^{n_i} / |\mathbb{F}|^{2t_i + \ell_i - 2|S|})$. Note that there are only 2^{n_i} choices of set S and for each S there are at most $|\mathbb{F}|^{2(t_i - |S|)}$ messages $\mathbf{m} \in \mathbb{F}^{2k_i}$ with the *maximal set* being S. By taking the union bound, so long as ℓ_i is large enough (e.g., $|\mathbb{F}|^{\ell_i} \gg 2^{n_i}$), with overwhelming probability, for every nonzero $\mathbf{m} \in \mathbb{F}^{2k_i}$, the number of zeros in $\mathsf{Enc}_{i+1}(\mathbf{m})$ is at most $2t_i + \ell_i$.

Theorem 2. *Fix any field $\mathbb{F}$ where $|\mathbb{F}| \geq 2^{10}$ and let $\lambda \in \mathbb{N}$ be the security parameter. For a vector $\mathbf{v}$ with elements in $\mathbb{F}$, denote $\mathsf{nzero}(\mathbf{v})$ as the number of zeroes in $\mathbf{v}$. For every $d \in \mathbb{N}$, let D_d be a (c, k_0)-foldable distribution and let $k_i = k_0 2^i$, $n_i = ck_i$ for every $i \leq d$. Then,*

$$\Pr_{(\mathbf{G}_0, \ldots, \mathbf{G}_d) \leftarrow D_d} \left[\exists \mathbf{m} \in \mathbb{F}^{k_d} \setminus \{\mathbf{0}\},\ \mathsf{nzero}(\mathsf{Enc}_d(\mathbf{m})) \geq t_d \right] \leq d \cdot 2^{-\lambda}\,. \tag{2}$$

Here $t_0 = k_0$ and $t_i = 2t_{i-1} + \ell_i$ for every $i \in [d]$, where

$$\ell_i := \frac{2(d-1)\log n_0 + \lambda + 2.002 t_{d-1} + 0.6 n_d}{\log |\mathbb{F}| - 1.001}\,.$$

Proof. We prove the theorem by induction.

Case $d = 0$: $\mathbf{G}_0$ is the generator matrix of a maximum distance seperable linear code (Definition 3).[6] Therefore, the Hamming weight of any non-zero codeword in C_0 is at least $ck_0 - k_0 + 1$ and thus the number of zeros in the codeword is at most $ck_0 - (ck_0 - k_0 + 1) = k_0 - 1 < k_0 = t_0$.

Case $d > 0$: Assuming that Inequality 2 holds for all $i \leq d - 1$, we prove that the inequality also holds for d. As before, we denote by $k_i := k_0 2^i$ and by $n_i := ck_i$ for all

[6]The analysis still works if $\mathbf{G}_0$ is not maximum distance separable, in which case we change the minimum Hamming weight according to the minimum distance of C_0.

$i \in \mathbb{N}$. We say that a sampled matrix $\mathbf{G}_{d-1}$ is "good" if $\mathbf{G}_{d-1}$ is a generator matrix where the encoding of any non-zero message $\mathbf{m} \in \mathbb{F}^{k_{d-1}}$ has fewer than t_{d-1} zeros. By induction hypothesis, the probability that $\mathbf{G}_{d-1}$ is not "good" (over distribution D_{d-1}) is at most $(d-1)2^{-\lambda}$. Thus, to prove that $\mathbf{G}_d$ is not "good" (over distribution D_d) with probability at most $d2^{-\lambda}$, it is sufficient to prove that conditioned on $\mathbf{G}_{d-1}$ being "good", the probability (over $\mathsf{diag}(T_{d-1})$) that exists non-zero message $\mathbf{m} \in \mathbb{F}^{k_d}$ where $\mathbf{m} \cdot \mathbf{G}_d$ has at least t_d zeros is at most $2^{-\lambda}$.

Next, we prove the above statement. Fix any "good" matrix $\mathbf{G}_{d-1}$, we start by defining set $m_d(S) \subseteq \mathbb{F}^{k_d}$ for every $S \subseteq [1, n_{d-1}]$. Namely, $m_d(S)$ is the set of *non-zero* vectors $\mathbf{m} = (\mathbf{m}_l, \mathbf{m}_r) \in \mathbb{F}^{k_d}$ such that

$$\{i \in [1, n_{d-1}] : \mathsf{Enc}_{d-1}(\mathbf{m}_l)[i] = 0 \wedge \mathsf{Enc}_{d-1}(\mathbf{m}_r)[i] = 0\} = S\,.$$

In other words, if $\mathbf{m} \in m_d(S)$ then (i) for all $i \in S$, both $\mathsf{Enc}_{d-1}(\mathbf{m}_l)[i] = 0$ and $\mathsf{Enc}_{d-1}(\mathbf{m}_r)[i] = 0$ and (ii) for any $j \notin S$, at least one of $\mathsf{Enc}_{d-1}(\mathbf{m}_l)[j]$ and $\mathsf{Enc}_{d-1}(\mathbf{m}_r)[j]$ is non-zero. We first bound the size of $m_d(S)$.

Lemma 2. *Fix any generator matrix $\mathbf{G}_{d-1}$ where the encoding of any non-zero message $\mathbf{m} \in \mathbb{F}^{k_{d-1}}$ has fewer than t_{d-1} zeros. For any subset $S \subseteq [1, n_{d-1}]$, if $|S| < t_{d-1}$ then $|m_d(S)| \leq |\mathbb{F}|^{2t_{d-1}-2|S|}$ and if $|S| \geq t_{d-1}$ then $|m_d(S)| = 1$.*

Proof. For brevity we denote $\mathbf{G} := \mathbf{G}_{d-1}$ and let $\mathbf{G}^\top$ denote the transpose of $\mathbf{G}$. Let $\mathbf{G}^\top[S] := \{\mathbf{G}^\top[i][\cdot] : i \in S\}$ denote the submatrix of $\mathbf{G}^\top$ with rows being the subset of rows in $\mathbf{G}^\top$ according to the index set $S \subseteq [1, n_{d-1}]$. For a matrix $M \in \mathbb{F}^{k \times n}$, we define $\mathsf{kernel}(M^\top)$ as the set of vectors $\mathbf{m} \in \mathbb{F}^k$ such that $(\mathbf{m}M)^\top = M^\top \mathbf{m}^\top = \mathbf{0}^n$. Observe that by definition of linear codes (i.e., $\mathsf{Enc}_{d-1}(\mathbf{m}) := \mathbf{m}\mathbf{G}$), $m_d(S)$ is a *subset* of

$$\{\mathbf{m} \in \mathbb{F}^{k_d} : \mathbf{m}_l \in \mathsf{kernel}(\mathbf{G}^\top[S]) \wedge \mathbf{m}_r \in \mathsf{kernel}(\mathbf{G}^\top[S])\} \tag{3}$$

where $\mathbf{m} = (\mathbf{m}_l, \mathbf{m}_r)$. Therefore,

$$|m_d(S)| \leq |\mathsf{kernel}(\mathbf{G}^\top[S])|^2\,. \tag{4}$$

Next, we will show that if $|S| < t_{d-1}$ then $|\mathsf{kernel}(\mathbf{G}^\top[S])| \leq |\mathbb{F}|^{t_{d-1}-|S|}$ and if $|S| \geq t_{d-1}$ then $|\mathsf{kernel}(\mathbf{G}^\top[S])| = 1$, from which the lemma statement follows.

Case $|S| < t_{d-1}$: Pick an index subset $T \subset [1, n_{d-1}]$ such that $|T| = t_{d-1} - |S|$ and $T \cap S = \emptyset$. Consider the matrix $\mathbf{G}^\top[T \cup S]$. Then $\mathsf{rank}(\mathbf{G}^\top[T \cup S]) \leq \mathsf{rank}(\mathbf{G}^\top[S]) + \mathsf{rank}(\mathbf{G}^\top[T])$.[7] The rank of a matrix is at most the number of its rows. Therefore, $\mathsf{rank}(\mathbf{G}^\top[T]) \leq t_{d-1} - |S|$ and thus

$$\mathsf{rank}(\mathbf{G}^\top[T \cup S]) \leq \mathsf{rank}(\mathbf{G}^\top[S]) + (t_{d-1} - |S|)\,. \tag{5}$$

[7] This follows directly from the fact that the dimension of the sum of two finite dimensional subspaces is less than or equal to the sum of the dimensions of those subspaces.

Recall that if $\mathbf{m} \in \mathsf{kernel}(\mathbf{G}^\top[T \cup S])$, then $\mathsf{Enc}_d(\mathbf{m})$ has at least $|T \cup S| = |T| + |S| = t_{d-1}$ zeroes. But by the assumption of the lemma, the number of zeroes in the encoding of any non-zero $\mathbf{v} \in \mathbb{F}^{k_{d-1}}$ is less than t_{d-1} and therefore $\mathsf{kernel}(\mathbf{G}^\top[T \cup S]) = \{\mathbf{0}^{k_{d-1}}\}$ and thus $\mathsf{nullity}(\mathbf{G}^\top[T \cup S]) = 0$. Therefore, because the number of columns in $\mathbf{G}^\top[T \cup S]$ is k_{d-1}, the rank-nullity theorem implies that $k_{d-1} = \mathsf{rank}(\mathbf{G}^\top[T \cup S]) + 0$. Therefore, by Eqn 5,

$$\mathsf{rank}(\mathbf{G}^\top[S]) \geq k_{d-1} - (t_{d-1} - |S|)\,.$$

Invoking the rank-nullity theorem on $\mathbf{G}^\top[S]$ again (which also has k_{d-1} columns), we have that

$$\mathsf{nullity}(\mathbf{G}^\top[S]) = k_{d-1} - \mathsf{rank}(\mathbf{G}^\top[S]) \leq k_{d-1} - (k_{d-1} - (t_{d-1} - |S|)) = t_{d-1} - |S|$$

and thus $|\mathsf{kernel}(\mathbf{G}^\top[S])| \leq |\mathbb{F}|^{t_{d-1}-|S|}$.

Case $|S| \geq t_{d-1}$: As mentioned in the previous case, the kernel of $\mathbf{G}^\top[S]$ must be $\{\mathbf{0}^{k_{d-1}}\}$ because otherwise there exists a non-zero message $\mathbf{m} \in \mathbb{F}^{k_{d-1}}$ such that the encoding of $\mathbf{m}$ has more than t_{d-1} zeroes, which contradicts the premise of the lemma. Therefore $|\mathsf{kernel}(\mathbf{G}^\top[S])| = 1$. □

Next, we prove that for every non-zero $\mathbf{m} \in m_d(S)$, the probability that $\mathbf{m} \cdot \mathbf{G}_d$ has at least t_d zeros is small.

Lemma 3. *Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| \geq 2^{10}$. Fix any generator matrix $\mathbf{G}_{d-1} \in \mathbb{F}^{k_{d-1} \times n_{d-1}}$ where the encoding of any non-zero message $\mathbf{m} \in \mathbb{F}^{k_{d-1}}$ has fewer than t_{d-1} zeros. Define matrix*

$$\mathbf{G}_d := \begin{bmatrix} \mathbf{G}_{d-1} & \mathbf{G}_{d-1} \\ \mathbf{G}_{d-1} \cdot \mathbf{T}_{d-1} & \mathbf{G}_{d-1} \cdot -\mathbf{T}_{d-1} \end{bmatrix}$$

where $\mathsf{diag}(\mathbf{T}_{d-1}) \leftarrow\!\$\ (\mathbb{F}^\times)^{n_{d-1}}$. For every $S \subseteq [1, n_{d-1}]$ and any non-zero message $\mathbf{m} \in m_d(S)$,

$$\Pr_{\mathsf{diag}(\mathbf{T}_{d-1}) \leftarrow\$(\mathbb{F}^\times)^{n_{d-1}}} [\mathsf{nzero}(\mathbf{m} \cdot \mathbf{G}_d) \geq t_d] < n_{d-1} \cdot 2^{n_{d-1}-|S|} \cdot \left(\frac{2.002}{|\mathbb{F}|}\right)^{t_d - 2|S|}. \qquad (6)$$

Proof. The lemma follows from a probability analysis for about $n_{d-1} - |S|$ independent Bernoulli trials. See Appendix B.1 for the full proof. □

Note that $\cup_{S \subseteq [1, n_{d-1}]} m_d(S)$ covers the entire set of messages in $\mathbb{F}^{k_d}$. From Lemma 2, Lemma 3 and by taking union bound over sets $S \subseteq [1, n_{d-1}]$ and messages in $m_d(S)$, we obtain the following lemma which completes the proof.

Lemma 4. *Fix any generator matrix $\mathbf{G}_{d-1}$ where the encoding of any non-zero message $\mathbf{m} \in \mathbb{F}^{k_{d-1}}$ has fewer than t_{d-1} zeros. Define matrix $\mathbf{G}_d$ as in Lemma 3. The probability (over $\mathsf{diag}(T_{d-1})$) that exists non-zero message $\mathbf{m} \in \mathbb{F}^{k_d}$ where $\mathbf{m} \cdot \mathbf{G}_d$ has at least t_d zeros is at most $2^{-\lambda}$.*

Proof. See Appendix B.2 for the proof. □

□

Concrete bounds. We list the relative minimum distances of the `BaseFold` codes for typical instantiations of parameters. The calculation of the relative minimum distances is explained in Appendix C.

Minimum Relative Distance of a random foldable code				
k_0	k_d	c	$\lvert\mathbb{F}\rvert$	Δ_{C_d}
2^5	2^{20}	16	2^{31}	.5044
1	2^{20}	16	2^{61}	.484
1	2^{25}	8	2^{128}	.557
1	2^{25}	8	2^{256}	.728

Table 1: The relative minimum distances of random foldable codes.

Remark 1 (Encoding Messages in Extension Fields)**.** *For soundness bootstrapping in PCS and IOPP, sometimes it is useful to lift a random foldable code over message space* $(\mathbb{F}_p)^{k_d}$ *to a code over message space* $(\mathbb{F}_{p^m})^{k_d}$ *where* $\mathbb{F}_{p^m}$ *is the degree-*$(m-1)$ *extension field of a prime field* $\mathbb{F}_p$*. In this case, we can understand the encoding* $\mathsf{Enc}_d(\mathbf{m})$ *of* $\mathbf{m} \in (\mathbb{F}_{p^m})^{k_d}$ *as* $\mathsf{Enc}_d(\mathbf{m}) := \sum_{j=0}^{r-1} \mathsf{Enc}_d(\mathbf{m}_j)X^j$ *where* $\mathbf{m} = \sum_{j=0}^{r-1} \mathbf{m}_j X^j$ *and* $\mathbf{m}_j \in \mathbb{F}_p^{k_d}$ *for every* $j \in [0, r-1]$*. Hence for any* $t \in \mathbb{N}$*,* $\mathsf{Enc}_d(\mathbf{m})$ *has* t *zeros implies that* $\mathsf{Enc}_d(\mathbf{m}_j)$ *has at least* t *zeros for every* $j \in [0, r-1]$*. Since* $\mathbf{m}$ *is non-zero implies that at least one of* $\mathbf{m}_j$ *is non-zero, the relative minimum distance of the encoding over message space* $(\mathbb{F}_{p^m})^{k_d}$ *is at least that of the encoding over* $\mathbb{F}_p^{k_d}$*. Moreover, for a message* $\mathbf{m} \in (\mathbb{F}_p)^{k_d}$*, the encoding of* $\mathbf{m}$ *over the small field* $\mathbb{F}_p$ *is exactly the encoding of* $\mathbf{m}$ *over the extension field* $\mathbb{F}_{p^m}$*.*

4 `BaseFold` : IOPPs for Foldable Codes

In this section, we present the `BaseFold` IOPP, which generalizes the `FRI` IOPP to any foldable linear code. Recall Definition 5 that for some $d \in \mathbb{Z}$, a foldable linear code C_d is specified by a list of generator matrices $(\mathbf{G}_0, \ldots, \mathbf{G}_d)$ where for every $i \in [1, d]$, $\mathbf{G}_i \in \mathbb{F}^{k_i \times n_i}$ satisfies that

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{G}_{i-1} & \mathbf{G}_{i-1} \\ \mathbf{G}_{i-1} \cdot T_{i-1} & \mathbf{G}_{i-1} \cdot T'_{i-1} \end{bmatrix}$$

where $T_{i-1}, T'_{i-1} \in \mathbb{F}^{n_{i-1} \times n_{i-1}}$ are some diagonal matrices. We also require that $\mathsf{diag}(T_{i-1})[j] \neq \mathsf{diag}(T'_{i-1})[j])$ for every $j \in [n_{i-1}]$. For example, in the random foldable code described in Sect. 3, we sample the diagonal of T_{i-1} as $\mathsf{diag}(T_{i-1}) \leftarrow\!\$\; (\mathbb{F}^\times)^{n_{i-1}}$ and set $T'_{i-1} = -T_{i-1}$.

In the `BaseFold` IOPP, the goal of the verifier is to check that an oracle $\pi_d \in \mathbb{F}^{n_d}$ sent by the prover is close to a codeword in C_d. As shown in Fig. 2, the protocol is split into two phases where in the first phase, commit, the prover generates a list of oracles $(\pi_d, \ldots, \pi_0)$ given the verifier's folding challenges $\alpha_i \in \mathbb{F}$ $(0 \le i < d)$; in the second phase, query, the verifier samples a query index $\mu \in [1, n_{d-1}]$ to check the consistency between oracles. We use $\mathsf{interpolate}((x_1, y_1), (x_2, y_2))$ to denote the unique degree-1 polynomial $Q(X)$ such that $Q(x_1) = y_1$ and $Q(x_2) = y_2$. Recall that for each $i \in [0, d]$, k_i is the message length and n_i is the blocklength of the code with generator matrix $\mathbf{G}_i$.

Protocol 2 $\mathsf{IOPP.commit}$

Input oracle: $\pi_d \in \mathbb{F}^{n_d}$
Output oracles: $(\pi_{d-1}, \ldots, \pi_0) \in \mathbb{F}^{n_{d-1}} \times \cdots \times \mathbb{F}^{n_0}$

- For i from $d-1$ downto 0:
  1. The verifier samples and sends $\alpha_i \leftarrow\!\$\ \mathbb{F}$ to the prover
  2. For each index $j \in [1, n_i]$, the prover
     (a) sets $f(X) := \mathsf{interpolate}((\mathsf{diag}(T_i)[j], \pi_{i+1}[j]), (\mathsf{diag}(T'_i)[j], \pi_{i+1}[j + n_i]))$
     (b) sets $\pi_i[j] = f(\alpha_i)$
  3. The prover outputs oracle $\pi_i \in \mathbb{F}^{n_i}$.

Protocol 3 $\mathsf{IOPP.query}$

Oracles: $(\pi_d, .., \pi_0)$

- The verifier samples an index $\mu \leftarrow\!\$\ [1, n_{d-1}]$
- For i from $d-1$ downto 0, the verifier
  1. queries oracle entries $\pi_{i+1}[\mu]$, $\pi_{i+1}[\mu + n_i]$
  2. computes $p(X) := \mathsf{interpolate}((\mathsf{diag}(T_i)[\mu], \pi_{i+1}[\mu]), (\mathsf{diag}(T'_i)[\mu], \pi_{i+1}[\mu + n_i]))$
  3. checks that $p(\alpha_i) = \pi_i[\mu]$
  4. if $i > 0$ and $\mu > n_{i-1}$, update $\mu \leftarrow \mu - n_{i-1}$.
- If π_0 is a valid codeword w.r.t. generator matrix $\mathbf{G}_0$, output accept, otherwise output reject

Figure 2: The IOPP protocol for foldable codes.

Lemma 5 (Completeness)**.** *If π_d is a codeword in C_d, then the verifier always outputs* accept *in the* query *phase given the oracles $(\pi_d, \ldots, \pi_0)$ output by the honest prover in the* commit *phase.*

Proof. Note that the checks $p(\alpha_i) = \pi_i[\mu]$ at step 3 of query always pass as p is computed in the same way as the polynomial f is computed by the honest prover in the commit phase. It remains to argue that π_0 is a valid codeword. Let $\mathbf{m}_d \in \mathbb{F}^{k_d}$ denote the decoded word underlying π_d. It suffices to show that for i from $d-1$ downto 0, π_i is the encoding (w.r.t. generator matrix $\mathbf{G}_i$) of message $\mathbf{m}_i := \mathbf{m}_{i+1,l} + \alpha_i \mathbf{m}_{i+1,r}$ (where $(\mathbf{m}_{i+1,l} || \mathbf{m}_{i+1,r}) = \mathbf{m}_{i+1}$). This implies that π_0 is a codeword and thus the verifier outputs accept.

To show π_i is the encoding of $\mathbf{m}_i := \mathbf{m}_{i+1,l} + \alpha_i \mathbf{m}_{i+1,r}$, we first note that

$$\begin{aligned}
\pi_{i+1} &= \mathbf{m}_{i+1} \cdot \mathbf{G}_{i+1} \\
&= \mathbf{m}_{i+1} \cdot \begin{bmatrix} \mathbf{G}_i & \mathbf{G}_i \\ \mathbf{G}_i \cdot T_i & \mathbf{G}_i \cdot T_i' \end{bmatrix} & \text{(definition of } \mathbf{G}_{i+1}) \\
&= \left[\mathsf{Enc}_i(\mathbf{m}_{i+1,l}) + \mathsf{Enc}_i(\mathbf{m}_{i+1,r}) \circ \mathsf{diag}(T_{i-1}) \,||\, \mathsf{Enc}_i(\mathbf{m}_{i+1,l}) + \mathsf{Enc}_i(\mathbf{m}_{i+1,r}) \circ \mathsf{diag}(T'_{i-1}) \right]
\end{aligned}$$

where the last equality holds given that T_i and T_i' are diagonal and $\mathbf{G}_i$ is the generator matrix of encoding Enc_i. Hence for every index $j \in [1, n_i]$,

$$\begin{aligned}
\pi_{i+1}[j] &= \mathsf{Enc}_i(\mathbf{m}_{i+1,l})[j] + \mathsf{Enc}_i(\mathbf{m}_{i+1,r})[j] \cdot \mathsf{diag}(T_{i-1})[j] \,, \\
\pi_{i+1}[j + n_i] &= \mathsf{Enc}_i(\mathbf{m}_{i+1,l})[j] + \mathsf{Enc}_i(\mathbf{m}_{i+1,r})[j] \cdot \mathsf{diag}(T'_{i-1})[j] \,.
\end{aligned}$$

Thus $\pi_{i+1}[j], \pi_{i+1}[j+n_i]$ are the evaluation of $f_j(X) = \mathsf{Enc}_i(\mathbf{m}_{i+1,l})[j] + \mathsf{Enc}_i(\mathbf{m}_{i+1,r})[j] \cdot X$ at points $\mathsf{diag}(T_{i-1})[j], \mathsf{diag}(T'_{i-1})[j]$. By step 2.(b) of the commit phase, for every index $j \in [1, n_i]$,

$$\pi_i[j] = \mathsf{Enc}_i(\mathbf{m}_{i+1,l})[j] + \alpha_i \cdot \mathsf{Enc}_i(\mathbf{m}_{i+1,r})[j] = \mathsf{Enc}_i(\mathbf{m}_{i+1,l} + \alpha_i \cdot \mathbf{m}_{i+1,r})[j]$$

where the last equality holds by linearity of the code. Thus π_i is the encoding of $\mathbf{m}_i := \mathbf{m}_{i+1,l} + \alpha_i \cdot \mathbf{m}_{i+1,r}$ which completes the proof. □

Next, we analyze the proximity error of the IOPP protocol. We adapt the statement and the proof of improved soundness of `FRI` stated in [23] to the case of general foldable linear codes.

Theorem 3 (IOPP Soundness For Foldable Linear Codes)**.** *Let C_d be a (c, k_0, d)-foldable linear code with generator matrices $(\mathbf{G}_0, \ldots, \mathbf{G}_d)$. We use C_i $(0 \le i < d)$ to denote the code with generator matrix $\mathbf{G}_i$ and assume the relative minimum distance $\Delta_{C_i} \ge \Delta_{C_{i+1}}$ for all $i \in [0, d-1]$. Let $\gamma > 0$ and set $\delta := \min(\Delta^*(\pi_d, C_d), J_\gamma(J_\gamma(\Delta_{C_d})))$ where $\Delta^*(\pi_d, C_d)$ is the relative coset minimum distance between $\mathbf{v}$ and C_d, (Definition 2). Then with probability at least $1 - \frac{2d}{\gamma^3|\mathbb{F}|}$ (over the challenges $(\alpha_0, \ldots, \alpha_{d-1})$ in* $\mathsf{IOPP.commit}$*), for any (adaptively chosen) prover oracles $\pi_{d-1}, .., \pi_0$, the verifier outputs* accept *in all of the ℓ repetitions of* $\mathsf{IOPP.query}$ *with probability at most $(1 - \delta + \gamma d)^\ell$.*

Proof. The proof is similar to the proof of Theorem 7.2 in [23]. For completeness, we present the proof in Appendix B.3. □

5 Multilinear Polynomial Commitments From `BaseFold`

In this section, we present a commitment scheme for multilinear polynomials with fast provers and polylogarithmic proof size and verification time. Moreover, the scheme works for polynomials over any (sufficiently large) fields. The scheme combines the Basefold IOPP with the sum-check protocol [55].

Notation. Let $d \in \mathbb{N}$, for every $v \in [0, 2^d)$, we use $\mathsf{bit}(v) \in \{0,1\}^d$ to denote the d-bit-decomposition of v, that is, $v = \sum_{j=1}^{d} \mathsf{bit}(v)[j] \cdot 2^{j-1}$. For a multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$, we use vector $\mathbf{f} \in \mathbb{F}^{2^d}$ to denote the coefficients of f, that is, $f(X_1, \ldots, X_d) = \sum_{v \in [0,2^d)} \mathbf{f}[v+1] \prod_{j=1}^{d} X_j^{\mathsf{bit}(v)[j]}$. For a vector $\mathbf{z} \in \mathbb{F}^d$, we use $f(\mathbf{z})$ to denote the evaluation of f at point $\mathbf{z} = (z_1, \ldots, z_d)$. Let $d' \in \mathbb{N}$ and let $C_{d'}$ be a (c, k_0, d')-foldable linear code. We use $\mathsf{Enc}_{d'}$ to denote the encoding algorithm for $C_{d'}$ with message length $k_{d'} = k_0 2^{d'}$ and blocklength $n_{d'} = ck_{d'}$.

Remark 2 (Field Choices)**.** *In the following context, we assume that $\mathbb{F}$ is a large field such that $d/|\mathbb{F}| = \mathsf{negl}(\lambda)$. This is without loss of generality: given some polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_d]$ over a small field $\mathbb{F}_p$, as explained in Remark 1, we can lift the encoding of the coefficient vector $\mathbf{f} \in \mathbb{F}_p^{2^d}$ to an encoding over the message space $\mathbb{F}_{p^m}^{2^d}$ where $\mathbb{F} := \mathbb{F}_{p^m}$ is the extension field of $\mathbb{F}_p$. Looking ahead, for any evaluation claim $f(\mathbf{z}) = y$ where $f \in \mathbb{F}_p[X_1, \ldots, X_d]$, $\mathbf{z} \in \mathbb{F}_p^d$ and $y \in \mathbb{F}_p$, we can understand it as a claim over the extension field $\mathbb{F}_{p^m}$, and run IOPP and sum-check protocols over the extension field and reduces the claim to a random evaluation claim over the extension field.*

Commitment phase. Given multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ with coefficients $\mathbf{f} \in \mathbb{F}^{2^d}$, let $\pi_f := \mathsf{Enc}_d(\mathbf{f})$ be the encoding of $\mathbf{f}$. In the IOP setting, the commitment to f is simply the oracle π_f that the verifier can make point queries. The derived commitment in the random oracle model is the root of the Merkle tree with leaves being the vector $\pi_f \in \mathbb{F}^{c2^d}$.

Opening phase. The prover opens a polynomial commitment C by sending a multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ and a word π_f to the verifier. The verifier checks that (i) the Merkle commitment of π_f equals C, and (ii) the relative distance between π_f and $\mathsf{Enc}_d(\mathbf{f})$ is less than $\Delta_{C_d}/2$, where Δ_{C_d} is the relative minimum distance of C_d.

The sum-check protocol. Before presenting the PCS evaluation protocol, we briefly review the famous sum-check protocol [55]. Given a multivariate polynomial oracle $f \in$

$\mathbb{F}^{\leq c}[X_1, \ldots, X_d]$ where the individual degree of each variable is $c \in \mathbb{N}$, the verifier wants to check that $\sum_{\mathbf{b} \in \{0,1\}^d} f(\mathbf{b}) = y$ for some value y. A naive approach is for the verifier to query f at every point in the Boolean hypercube $\{0,1\}^d$ and sum the evaluation values, which involves 2^d polynomial queries. Lund et. al. [55] introduced an elegant sum-check protocol that reduces the verifier's work to a single polynomial query at a random point.

The protocol runs in d rounds where in each round, the prover sends a univariate polynomial of degree c and the verifier replies with a random field element as a challenge. At the end of the protocol, the verifier makes a single query to f at a random point to decide whether the sumcheck claim holds. Specifically,

- The prover sends a univariate polynomial $g_d(X) := \sum_{\mathbf{b} \in \{0,1\}^{d-1}} f(\mathbf{b}, X)$, the verifier checks that $g_d(0) + g_d(1) = y$, and samples $r_{d-1} \leftarrow\$ \mathbb{F}$. Then the sumcheck claim is reduced to $\sum_{\mathbf{b} \in \{0,1\}^{d-1}} f(\mathbf{b}, r_{d-1}) = g_d(r_{d-1})$.
- For round i from $d-1$ to 1, the prover sends a univariate polynomial

$$g_i(X) := \sum_{\mathbf{b} \in \{0,1\}^{i-1}} f(\mathbf{b}, X, r_i, \ldots, r_{d-1}),$$

  the verifier checks that $g_i(0) + g_i(1) = g_{i+1}(r_i)$, and samples $r_{i-1} \leftarrow\$ \mathbb{F}$. Then the sumcheck claim is reduced to $\sum_{\mathbf{b} \in \{0,1\}^{i-1}} f(\mathbf{b}, r_{i-1}, \ldots, r_{d-1}) = g_i(r_{i-1})$.
- The verifier queries $f(r_0, \ldots, r_{d-1})$ and `accept` if $f(r_0, \ldots, r_{d-1}) = g_1(r_0)$.

It is easy to see that the protocol is perfectly complete. Lund et. al. also showed that if $\sum_{\mathbf{b} \in \{0,1\}^d} f(\mathbf{b}) \neq y$, for any unbounded malicious prover, the verifier outputs `accept` with probability at most $cd/|\mathbb{F}|$.

Evaluation protocol. Next, we describe the evaluation protocol in Fig. 3. Given a commitment C, a point $\mathbf{z} \in \mathbb{F}^d$ and value $y \in \mathbb{F}$, the prover wants to convince the verifier that it knows an opening $f \in \mathbb{F}[X_1, \ldots, X_d]$ of C such that $f(\mathbf{z}) = y$. Our scheme interleaves sumcheck with `BaseFold` IOPP. Before describing the protocol, recall that by the multilinear extension, a multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ can be uniquely expressed as the following sum

$$f(X_1, \ldots, X_d) = \sum_{\mathbf{b} \in \{0,1\}^d} f(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{b}}(X_1, \ldots, X_d)$$

where the polynomial $\tilde{eq}_{\mathbf{b}}(X_1, \ldots, X_d) := \prod_{i=1}^{d} [\mathbf{b}[i] X_i + (1 - \mathbf{b}[i])(1 - X_i)]$, thus checking $f(\mathbf{z}) = y$ is equivalent to checking the sum-check claim $y = \sum_{\mathbf{b} \in \{0,1\}^d} f(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{b}}(\mathbf{z})$. In Fig. 3, we describe a protocol for this claim in the language of IOPs. We require that the multilinear extension, $h_d(X)$ orders the coefficients of $\mathbf{f}$ so that they are consistent with the ordering used in the Encoding of $\mathbf{f}$ (Protocol 1). (Note that Ben-Sasson et. al. [20]

Protocol 4 $\mathsf{PC.Eval}$

Public input: oracle $\pi_f := \mathsf{Enc}_d(\mathbf{f}) \in \mathbb{F}^{n_d}$, point $\mathbf{z} \in \mathbb{F}^d$, claimed evaluation $y \in \mathbb{F}$
Prover witness: the polynomial f with coefficients $\mathbf{f} \in \mathbb{F}^{2^d}$
Code parameters: $\mathbf{G}_0$ and diagonal matrices $(T_0, \ldots, T_{d-1})$ and $(T'_0, \ldots, T'_{d-1})$

1. The prover sends $h_d(X) := \sum_{\mathbf{b} \in \{0,1\}^{d-1}} f(\mathbf{b}, X) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{b}, X)$ to the verifier
2. For i from $d-1$ to 0
   (a) Verifier samples and sends $r_i \leftarrow\$ \mathbb{F}$ to the prover
   (b) For each $j \in [1, n_i]$, the prover
      i. sets $g_j(X) := \mathsf{interpolate}(\mathsf{diag}(T_i[j], \pi_{i+1}[j])), (\mathsf{diag}(T'_i)[j], \pi_{i+1}[j + n_i]))$
      ii. sets $\pi_i[j] := g_j(r_i)$
   (c) The prover outputs oracle $\pi_i \in \mathbb{F}^{n_i}$
   (d) If $i > 0$, the prover sends verifier

$$h_i(X) = \sum_{\mathbf{b} \in \{0,1\}^{i-1}} f(\mathbf{b}, X, r_i, \ldots, r_{d-1}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{b}, X, r_i, \ldots, r_{d-1})$$

3. The verifier checks that
   - $\mathsf{IOPP.query}^{(\pi_d, \ldots, \pi_0)}$ outputs accept
   - $h_d(0) + h_d(1) = y$ and for every $i \in [1, d-1]$, $h_i(0) + h_i(1) = h_{i+1}(r_i)$
   - $\mathsf{Enc}_0(h_1(r_0)/\tilde{eq}_{\mathbf{z}}(r_0, \ldots, r_{d-1})) = \pi_0$

Figure 3: The evaluation protocol for the `BaseFold` PCS.

showed how to generically transform an IOP that satisfies round-by-round soundness into a non-interactive argument of knowledge in the random oracle model). For simplicity, we assume that the scheme is instantiated with a (c, k_0, d)-foldable code where $k_0 = 1$. In Remark 3 we will explain how to adapt the protocol to the case where $k_0 > 1$. Again, for every $i \in [0, d]$, we set $k_i = k_0 \cdot 2^i$ and $n_i = ck_i$.

Remark 3 (The case of $k_0 > 1$)**.** *The* `BaseFold` *IOPP can also be instantiated with a (c, k_0, d')-foldable code where $k_0 = 2^\kappa$ for some integer $\kappa \geq 1$ and $d' := d - \kappa$. In Protocol 4,* *we replace d with d', and after d' rounds, the sum-check claim is reduced to*

$$h_1(r_0) = \sum_{\mathbf{b}\in\{0,1\}^\kappa} f(\mathbf{b}, r_0, \ldots, r_{d'-1}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{b}, r_0, \ldots, r_{d'-1})\,. \tag{7}$$

To check this, the prover additionally sends a vector $\mathbf{m} \in \mathbb{F}^{2^\kappa}$ which is the coefficient vector for polynomial $f(X_1, \ldots, X_\kappa, r_0, \ldots, r_{d'-1})$, then the verifier checks that $\mathsf{Enc}_0(\mathbf{m}) = \pi_0$ and Equation 7 holds. Note that the evaluations of $f(X_1, \ldots, X_\kappa, r_0, \ldots, r_{d'-1})$ on set $\{0,1\}^\kappa$ can be computed in time $O(\kappa 2^\kappa)$ given the coefficients $\mathbf{m}$.

Next, we analyze the running time of the prover and verifier when executing Protocol 4.

Prover time. The prover runs the prover algorithms of $\mathsf{IOPP.commit}$ and the sum-check protocol. The sumcheck prover cost is dominated by $5k_d$ finite field multiplications and the cost of running $\mathsf{IOPP.commit}$ is $O(n_d)$ field operations and hashes. In sum, the prover complexity is $O(n_d)$ field operations and hashes.

Verifier time. Note that the evaluation $\tilde{eq}_{\mathbf{z}}(r_0, \ldots, r_{d-1})$ can be computed in $O(d)$ field operations, thus the verifier time is dominated by ℓ runs of $\mathsf{IOPP.query}$, with total cost of $O(\ell d)$ field operations and Merkle path checkings (where each Merkle path checking takes at most $O(d)$ hash computations).

Remark 4 (Efficiency comparison to the FRI-based univariate PCS.)**.** *Recall that in the evaluation proof of the FRI-based univariate PCS [51], two polynomials are committed, $f(x) \in \mathbb{F}[X]$ and $\frac{f(x)-f(v)}{x-v}$. `FRI` IOPP is only executed over the latter one, but deriving the second polynomial takes $4ck_d$ finite field multlications using Montgomery's trick for batch inversion. For example, when $c = 8$, the `FRI` PCS additionally performs $32k_d$ finite field multiplications, while in the BaseFold PCS the extra overhead is $5k_d$ field multiplications for running the sumcheck protocol.*

5.1 Security Proofs

Completeness and binding. Completeness holds as an honest prover can always pass the evaluation check. To argue binding, suppose for contradiction that the adversary can output valid openings for two different polynomials $f_1, f_2 \in \mathbb{F}[X_1, \ldots, X_d]$, that is, there exists π_1, π_2 where $\mathsf{Merkle.commit}(\pi_1) = \mathsf{Merkle.commit}(\pi_2)$ and $\Delta^*(\mathsf{Enc}_d(\mathbf{f}_i), \pi_i) < \Delta_{C_d}/2$ for every $i \in \{1,2\}$. Then either $\pi_1 = \pi_2$ or the adversary finds a hash collision. The first case (i.e., $\pi_1 = \pi_2$) never happens as otherwise by definition of minimum distance ,triangle inequality, and Lemma 6,

$$\begin{aligned}\Delta_{C_d} &\le \Delta(\mathsf{Enc}_d(\mathbf{f}_1), \mathsf{Enc}_d(\mathbf{f}_2)) \\ &\le \Delta(\mathsf{Enc}_d(\mathbf{f}_1), \pi_1) + \Delta(\mathsf{Enc}_d(\mathbf{f}_2), \pi_2) \\ &< \Delta^*(\mathsf{Enc}_d(\mathbf{f}_1), \pi_1) + \Delta^*(\mathsf{Enc}_d(\mathbf{f}_2), \pi_2) \\ &< 2(\Delta_{C_d}/2)\,,\end{aligned}$$

which is a contradiction. The second case happens with negligible probability by the collision resistance of the hash function.

Soundness. Before showing knowledge soundness, we prove a useful lemma arguing the soundness of the PCS, that is, if a prover can pass the check in Protocol 4 with non-negligible probability, then the oracle π_f is close to a unique codeword $\mathsf{Enc}_d(\mathbf{f})$ in C_d and the corresponding polynomial f satisfies $f(\mathbf{z}) = y$. Note that in order to securely instantiate the Fiat-Shamir transform from [18] to `BaseFold`, we need to prove `BaseFold` satisfies round-by-round soundness. In this section, we prove that `BaseFold` satisfies the traditional notion of soundness (Definition 6) but we believe it will be straightforward to prove using techniques from [24]. Before proving the PCS soundness, we state a useful fact about the *coset relative distance* (Definition 2).

Lemma 6. *Let C be an $[n, k, d]$ code and let $\mathbf{v} \in \mathbb{F}^n$. Then*

$$\Delta(\mathbf{v}, C) \le \Delta^*(\mathbf{v}, C)$$

Proof. Let $c, c' \in C$ satisfy $\Delta(\mathbf{v}, C) = \Delta(\mathbf{v}, c')$ and $\Delta^*(\mathbf{v}, C) = \Delta^*(\mathbf{v}, c)$. It suffices to show that $\Delta(\mathbf{v}, c') \le \Delta^*(\mathbf{v}, c)$. By definition of relative minimum distance, $\Delta(\mathbf{v}, c') \le \Delta(\mathbf{v}, c)$. Moreover, the absolute distance between c and $\mathbf{v}$ is at most twice the number of pairs included in the coset distance between c and $\mathbf{v}$. Thus, $\Delta(\mathbf{v}, c') \le \Delta(\mathbf{v}, c) \le (2 \cdot \Delta^*(\mathbf{v}, c) \cdot n/2)/n = \Delta^*(\mathbf{v}, c)$, which completes the proof. □

Lemma 7 (Soundness)**.** *Let $\gamma, \delta \in (0, 1)$ satisfy $\delta < J_\gamma(J_\gamma(\Delta_{C_d}))$ and $\Delta_{C_{i+1}} \le \Delta_{C_i}$ for every $i \in [0, d-1]$. Assume that $3\delta - d\gamma < \Delta_{C_d}$ and $\frac{2d}{\gamma^3|\mathbb{F}|} + (1 - \delta + \gamma d)^\ell \le \mathsf{negl}(\lambda)$. In the evaluation protocol 4 with instance $(\pi_f, \mathbf{z}, y)$ (π_f is the input oracle), if the prover $\mathcal{P}^*$ passes the verification with non-negligible probability, then exists unique polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ (with coefficients $\mathbf{f} \in \mathbb{F}^{2^d}$) such that $\Delta^*(\pi_f, \mathsf{Enc}_d(\mathbf{f})) \le \delta$ (Definition 2) and $f(\mathbf{z}) = y$. Moreover, the probability that $\mathcal{P}^*$ passes the verification but $f(r_0, \ldots, r_{d-1}) \neq h_1(r_0)/\tilde{eq}_{\mathbf{z}}(r_0, \ldots, r_{d-1})$ is negligible (where $h_1(r_0)/\tilde{eq}_{\mathbf{z}}(r_0, \ldots, r_{d-1})$ is the claimed evaluation in sumcheck).*

Proof. By soundness of IOPP (Theorem 3) and by Lemma 6, it holds that $\Delta(\pi_f, C_d) \le \Delta^*(\pi_f, C_d) \le \delta < \Delta_{C_d}/2$. Thus there exists a unique multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$

such that $\Delta(\pi_f, \mathsf{Enc}_d(\mathbf{f})) \leq \delta$. It remains to argue that $f(\mathbf{z}) = y$. Assume for contradiction that $f(\mathbf{z}) \neq y$, that is,

$$f(\mathbf{z}) = \sum_{\mathbf{b} \in \{0,1\}^d} f(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{b}}(\mathbf{z}) = \sum_{\mathbf{b} \in \{0,1\}^d} f(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{b}) \neq y\,.$$

By soundness of the sum-check protocol, $f(\mathbf{r}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{r}) = h_1(r_0)$ with probability at most $2d/|F|$ over the random choice of $\mathbf{r} \in \mathbb{F}^d$ (as $f'(\mathbf{X}) := f(\mathbf{X}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{X})$ is a polynomial with total degree $2d$). Here $h_1(X)$ is the last univariate polynomial sent by the prover in the sumcheck protocol. Next, we argue that with probability more than $2d/|F|$ over $\mathbf{r}$, we actually have $f(\mathbf{r}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{r}) = h_1(r_0)$, which leads to a contradiction and completes the proof. WLOG we assume that $\mathcal{P}^*$ is deterministic. We start by defining two bad events.

- Event B_1: $\mathsf{IOPP.query}^{(\pi_d,\ldots,\pi_0)}$ outputs accept with probability less than $(1 - \delta + \gamma d)$ (over the choice of sampled index $\mu \in [n_{d-1}]$).
- Event B_2: There exists $i \in [0, d]$, such that the success probability of the IOPP partial execution on transcript prefix $(\pi_d, r_{d-1}, \ldots, \pi_{i+1}, r_i, \pi_i)$ (instead of π_d) is negligible, where the randomness is over freshly sampled challenges $(r'_{i-1}, \ldots, r'_0)$ in the commit phase and the sampled indices in the query phase. For brevity, we use $\mathsf{IOPP}^{\pi_i}\langle \mathcal{P}^*, \mathcal{V} \rangle$ to denote the execution where $\mathcal{P}^*$ is the malicious prover.

Claim 1. *The probability that $\mathcal{P}^*$ succeeds while at least one of the events B_1, B_2 happens is negligible.*

Proof. Note that if event B_1 happens, $\mathcal{P}^*$ succeeds with probability at most $(1 - \delta + d\gamma)^\ell = \mathsf{negl}(\lambda)$ (over the sampled index in $\mathsf{IOPP.query}$). On the other hand, by definition of event B_2, only a negligible portion of $\mathcal{P}^*$'s success probability will fall into the case where B_2 happens. By union bound, the probability that $\mathcal{P}^*$ succeeds and B_1 or B_2 happens is negligible, and the claim holds. $\square$

Next, we show that $f(\mathbf{r}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{r}) = h_1(r_0)$ for "good" $\mathbf{r}$ (i.e., bad events do not happen).

Lemma 8. *Let $\gamma, \delta \in (0, 1)$ satisfy the conditions as in Lemma 7. Let $f \in \mathbb{F}[X_1, \ldots, X_d]$ be the unique multilinear polynomial such that $\Delta^*(\pi_f, \mathsf{Enc}_d(\mathbf{f})) \leq \delta$. For any challenge set $\mathbf{r} = (r_0, \ldots, r_{d-1})$ and corresponding oracles $(\pi_d, \ldots, \pi_0)$ output by $\mathcal{P}^*$ such that events B_1, B_2 do not happen and $\mathsf{Enc}_0(h_1(r_0)/\tilde{eq}_{\mathbf{z}}(\mathbf{r})) = \pi_0$, it holds that $f(\mathbf{r}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{r}) = h_1(r_0)$ where $h_1(X)$ is the last univariate polynomial sent by the prover in the sumcheck protocol.*

Proof. For every $i \in [0, d]$, we use $\mathbf{f}_i \in \mathbb{F}^{2^i}$ to denote the coefficient vector for the i-variate multilinear polynomial $f(X_1, \ldots, X_i, r_i, \ldots, r_{d-1})$. For every $i \in [0, d-1]$, denote by $\mathbf{f}_{i+1} = (\mathbf{f}_{i+1,l}, \mathbf{f}_{i+1,r})$, we have $\mathbf{f}_i = \mathbf{f}_{i+1,l} + r_i \cdot \mathbf{f}_{i+1,r}$ by definition of multilinear polynomials.

Assume for contradiction that $h_1(r_0)/\tilde{eq}_{\mathbf{z}}(\mathbf{r}) \neq f(\mathbf{r})$. By the premise of the lemma that event B_2 does not happen, for every $i \in [0, d]$, the probability that $\mathsf{IOPP}^{\pi_i}\langle \mathcal{P}^*, \mathcal{V} \rangle = 1$ is non-negligible (i.e., larger than $\frac{2i}{\gamma^3|\mathbb{F}|} + (1-\delta+\gamma i)^\ell = \mathsf{negl}(\lambda)$). Thus by the IOPP soundness (Theorem 3), it holds that $\Delta^*(\pi_i, C_i) \le \delta$ for every $i \in [0, d]$. Since $\Delta^*(\pi_d, \mathsf{Enc}_d(\mathbf{f}) \le \delta$ but $\pi_0 = \mathsf{Enc}_0(h_1(r_0)/\tilde{eq}_{\mathbf{z}}(\mathbf{r})) \ne \mathsf{Enc}_0(\mathbf{f}_0)$, there exists a round $k \in [0, d-1]$, such that $\Delta^*(\pi_k, \mathsf{Enc}_k(\mathbf{g}_k)) \le \delta$ for some coefficient vector $\mathbf{g}_k \ne \mathbf{f}_k$; while in the previous round, it holds that $\Delta^*(\pi_{k+1}, \mathsf{Enc}_{k+1}(\mathbf{f}_{k+1})) \le \delta$. Denote $\delta' := \delta - \gamma d$. Next, we argue that

$$\Delta(\pi_k, \mathsf{Enc}_k(\mathbf{f}_k)) \le \delta' + \Delta^*(\pi_{k+1}, \mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))\,. \tag{8}$$

Recall that by assumption, both $\Delta^*(\pi_{k+1}, \mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))$ and $\Delta^*(\pi_k, \mathsf{Enc}_k(\mathbf{g}_k))$ are no more than δ. If Eqn. 8 holds, by triangle inequality and by definition of minimum relative distance, it implies

$$\begin{aligned}
\Delta_{C_k} &\le \Delta(\mathsf{Enc}_k(\mathbf{g}_k), \mathsf{Enc}_k(\mathbf{f}_k)) \\
&\le \Delta(\mathsf{Enc}_k(\mathbf{g}_k), \pi_k) + \Delta(\pi_k, \mathsf{Enc}_k(\mathbf{f}_k)) \\
&\le \Delta^*(\mathsf{Enc}_k(\mathbf{g}_k), \pi_k) + \Delta(\pi_k, \mathsf{Enc}_k(\mathbf{f}_k)) && \text{(Lemma 6)} \\
&\le \delta + \delta' + \Delta^*(\pi_{k+1}, \mathsf{Enc}_{k+1}(\mathbf{f}_{k+1})) && \text{(Eqn 8)} \\
&\le \delta + \delta' + \delta = 3\delta - \gamma d\,,
\end{aligned}$$

Note that we have $3\delta - \gamma d < \Delta_{C_k}$ by the premise of the statement, which leads to a contradiction and completes the proof.

Next, we prove Eqn. 8. Let $\delta' := \delta - \gamma d$ and denote $\pi_d := \pi_f$. Recall that $\mathsf{IOPP.query}$ outputs accept for more than $(1-\delta')n_{d-1}$ (out of n_{d-1}) sampled indices $\mu \in [1, n_{d-1}]$. Our goal is to show that for every round $i \in [0, d-1]$, there are at least $(1-\delta')n_i$ indices $\mu \in [n_i]$ such that $\pi_i[\mu]$ is consistent with $\pi_{i+1}[\mu]$ and $\pi_{i+1}[\mu + n_i]$ in terms of the folding operation (defined in Eqn. 27). Actually, we prove an even stronger statement: For every $i \in [0, d-1]$, we show that there are at least $(1-\delta')n_i$ entries $\mu \in [n_i]$ such that in $\mathsf{IOPP.query}$, if the verifier's query position for π_i is μ, then the verifier's checks for oracles $\pi_0, \ldots, \pi_i$ are all passing. We prove by induction. It holds when $i = d-1$ as event B_1 does not happen by the claim statement. Suppose by induction hypothesis that the number of bad query positions for π_i is at most $\delta' n_i$. For $i-1$, we note that for every $\mu \in [n_{i-1}]$, $\pi_{i-1}[\mu]$ is a good query position so long as *one* of the query entries $\pi_i[\mu]$ and $\pi_i[\mu + n_{i-1}]$ for π_i is good (i.e. the verifier's checks to $\pi_0, \ldots, \pi_i$ are all passing). Therefore, $\pi_{i-1}[\mu]$ is a bad query entry only if *both* μ and $\mu + n_{i-1}$ are bad query positions for π_i. Hence the number of bad query positions $\mu \in [n_{i-1}]$ for π_{i-1} is at most $\delta' n_i/2 = \delta' n_{i-1}$.

Given above, it follows that $\Delta(\mathsf{fold}_{r_k}(\pi_{k+1}), \pi_k) < \delta'$ as we've proved that $\pi_k[\mu]$ is consistent with $\pi_{k+1}[\mu]$ and $\pi_{k+1}[\mu + n_k]$ for more than $(1-\delta')n_k$ entries of μ. Moreover, recall $\mathbf{f}_k = \mathbf{f}_{k+1,l} + r_k \cdot \mathbf{f}_{k+1,r}$ and we've shown in Lemma 5 that

$$\mathsf{Enc}_k(\mathbf{f}_k) = \mathsf{Enc}_k(\mathbf{f}_{k+1,l} + r_k \cdot \mathbf{f}_{k+1,r}) = \mathsf{fold}_{r_k}(\mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))\,, \tag{9}$$

where the fold operation is defined in Eqn. 27. Thus we have

$$
\begin{aligned}
&\Delta(\pi_k, \mathsf{Enc}_k(\mathbf{f}_k)) \\
=&\Delta(\pi_k, \mathsf{fold}_{r_k}(\mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))) && \text{(Eqn. 9)} \\
\leq&\Delta(\pi_k, \mathsf{fold}_{r_k}(\pi_{k+1})) + \Delta(\mathsf{fold}_{r_k}(\pi_{k+1}), \mathsf{fold}_{r_k}(\mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))) && \text{(triangle inequality)} \\
\leq&\delta' + \Delta(\mathsf{fold}_{r_k}(\pi_{k+1}), \mathsf{fold}_{r_k}(\mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))) && (\Delta(\pi_k, \mathsf{fold}_{r_k}(\pi_{k+1})) < \delta') \\
\leq&\delta' + \Delta^*(\pi_{k+1}, \mathsf{Enc}_{k+1}(\mathbf{f}_{k+1}))
\end{aligned}
$$

where the last inequality holds because each element in $\mathsf{fold}_{r_k}(\pi_{k+1})$ that is inconsistent with $\mathsf{fold}_{r_k}(\mathsf{Enc}_{k+1}(\mathbf{f}_{k+1})$ maps to at least one element in π_{k+1} that is inconsistent with $\mathsf{Enc}_{k+1}(\mathbf{f}_{k+1})$. Thus Eqn 8 holds and we complete the proof. $\square$

In sum, since $\mathcal{P}^*$ succeeds with non-negligible probability, from Lemma 8 and Claim 1, with non-negligible probability (that is more than $2d/|F|$) over $\mathbf{r}$, we have $f'(\mathbf{r}) := f(\mathbf{r}) \cdot \tilde{eq}_{\mathbf{z}}(\mathbf{r}) = h_1(r_0)$, thus $f(\mathbf{z}) = y$ by the soundness of the sumcheck protocol. $\square$

Knowledge soundness. Next, we prove knowledge soundness of the PCS evaluation protocol. By the IOP-to-NARK transformation of [20], given any PCS evaluation prover that convinces the verifier with non-negligible probability, there is an efficient extractor that outputs the IOP oracle string that opens the Merkle commitment sent by the prover (intuitively by querying Merkle paths from the prover). Thus it is sufficient to prove the following theorem in the language of IOP.

Theorem 4 (Knowledge soundness.). *Let $\gamma, \delta \in (0,1)$ satisfy the conditions as in Lemma 7. Fix finite field $\mathbb{F}$ and set $\ell \in \mathbb{N}$ such that $(2d/\gamma^3|\mathbb{F}|) + (1-\delta+d\gamma)^\ell \leq \mathsf{negl}(\lambda)$. For any PCS evaluation instance $(\pi_f, \mathbf{z}, y)$ (where π_f is the input oracle), and any malicious prover $\mathcal{P}^*$ that suceeds in Protocol 4 with non-negligible probability, there is a polynomial-time extractor $\mathsf{Ext}_{\mathcal{P}^*}$ such that with overwhelming probability, $\mathsf{Ext}_{\mathcal{P}^*}$ outputs a polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ where $\Delta^*(\mathsf{Enc}_d(\mathbf{f}), \pi_f) \leq \delta$ and $f(\mathbf{z}) = y$, where Δ^* is the coset minimum relative distance (Definition 2).*

Before proving the theorem, we state a useful "predicate forking lemma", which is a special case of Lemma 3 from [29]. Loosely speaking, the lemma says that if A is an algorithm that on uniform random input $m \leftarrow\$ \mathcal{M}$ returns $A(m) = 1$ with probability ϵ, and $\Phi(m_1, ...m_k)$ is any predicate that holds with overwhelmingly high probability for independent random $m_i \leftarrow\$ \mathcal{M}$, then it is possible to efficiently "extract" k inputs to A that satisfy the predicate Φ and for which $A(m_i) = 1$ for all $i \in [k]$. The runtime of this "extractor" algorithm is proportional to $1/\epsilon$ and the success is overwhelmingly high. A bit more precisely, it isn't enough for Φ to hold true with high probability over random vectors in $\mathcal{M}^k$, but a sufficient condition is that for any $i \leq k$ the conditional probability $Pr[\Phi(m_1, ...m_i) \neq 1 | \Phi(m_1, .., m_{i-1}) = 1]$ over $m_i \leftarrow\$ \mathcal{M}$ is negligible.

Lemma 9 (Variant of Lemma 3 in [29]). *Let $\Phi : \mathcal{M}^* \to \{0,1\}$ be any predicate such that for any $(m_1, \ldots, m_i) \in \mathcal{M}^i$ where $\Phi(m_1, \ldots, m_i) = 1$,*

$$\Pr_{m_{i+1} \leftarrow \$ \mathcal{M}} [\Phi(m_1, \ldots, m_{i+1}) = 1] \geq 1 - \mathsf{negl}(\lambda) .$$

Let $N = \mathsf{poly}(\lambda)$. For any $\epsilon > 0$ there exists an extractor Ext which runs in time $T \in O(\lambda/\epsilon)$ and, given oracle access to any algorithm A where $\Pr_{m \leftarrow \$ \mathcal{M}}[A(m) = 1] \geq \epsilon$, the following holds:

$$\Pr\left[\begin{array}{c} \Phi(m_1, \ldots, m_N) = 1 \wedge \\ A(m_i) = 1 \forall i \in [N] \end{array} \middle| \; (m_1, \ldots, m_N) \leftarrow \mathsf{Ext}^A \right] \geq 1 - T \cdot \mathsf{negl}(\lambda)$$

Proof. For completeness, we present the proof in Appendix B.4. □

Equipped with Lemma 9, we are ready to prove knowledge soundness of the PCS evaluation protocol.

Proof of Theorem 4. By Lemma 7 (for soundness), we know that there exists a unique multilinear polynomial $f \in \mathbb{F}[X_1, \ldots, X_d]$ such that the *coset relative distance* $\Delta^*(\pi_f, \mathsf{Enc}_d(\mathbf{f})) \leq \delta$ and $f(\mathbf{z}) = y$. The remaining task is to construct an algorithm that recovers the polynomial f. Note that this is trivial if π_f is efficiently decodable. However, we do not have any guarantee that the code C_d is efficiently decodable. Fortunately, we can recover f by building an extractor Ext that outputs a list of 2^d points $S := \{\mathbf{v}_i \in \mathbb{F}^d\}_{i \in [2^d]}$ and their corresponding evaluations $\{f(\mathbf{v}_i)\}_{i \in [2^d]}$, such that the expansion vectors $\{\mathsf{expd}(\mathbf{v}_1), \ldots, \mathsf{expd}(\mathbf{v}_{2^d})\}$ are *linearly independent*. Here for a vector $\mathbf{v} \in \mathbb{F}^d$, the expansion vector $\mathsf{expd}(\mathbf{v}) \in \mathbb{F}^{2^d}$ is defined so that for every $i \in [0, 2^d)$,

$$\mathsf{expd}(\mathbf{v})[i+1] := \prod_{j=1}^{d} \mathbf{v}[j]^{\mathsf{bit}(i)[j]} . \tag{10}$$

E.g., for a vector $(x, y) \in \mathbb{F}^2$, the expansion vector is $\mathsf{expd}([x, y]) := (1, x, y, xy)$. Importantly, we note that the evaluation $f(\mathbf{v})$ satisfies that $f(\mathbf{v}) = \sum_{i=1}^{2^d} \mathbf{f}[i] \cdot \mathsf{expd}(\mathbf{v})[i]$ where $\mathbf{f} \in \mathbb{F}^{2^d}$ is the coefficient vector of the polynomial f. Hence given the set of 2^d linearly independent expansion vectors $\{\mathsf{expd}(\mathbf{v}_1), \ldots, \mathsf{expd}(\mathbf{v}_{2^d})\})$ and the set of evaluations $f(S) := \{f(\mathbf{v}_i)\}_{i \in [2^d]}$, we can solve the system of equations using Gaussian elimination to recover vector $\mathbf{f} \in \mathbb{F}^{2^d}$ and thus recover the polynomial f.

Next we show how to build the extractor Ext using the predicate forking lemma (Lemma 9). There are two major steps: the first step is to define the predicate to be used in Lemma 9; the second step is defining the algorithm A in Lemma 9 that outputs 1 with non-negligible probability.

The predicate. We define the predicate Φ as follows. Let $\mathcal{M} := \mathbb{F}^d$ be the message space that consists of a length-d vector. For input $(\mathbf{v}_1, \ldots, \mathbf{v}_i) \in \mathcal{M}^i$ which consists of i length-d vectors[8], we say $\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_i) = 1$ $\mathsf{expd}(\mathbf{V}) := \{\mathsf{expd}(\mathbf{v}_1), \ldots, \mathsf{expd}(\mathbf{v}_i)\}$ are *linearly independent*.

Lemma 10. *For any $i \in [0, 2^d)$ and any $(\mathbf{v}_1, \ldots, \mathbf{v}_i) \in (\mathbb{F}^d)^i$ where $\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_i) = 1$, it holds that*

$$\Pr_{\mathbf{v}_{i+1} \leftarrow\$ \mathbb{F}^d} [\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_{i+1}) = 1] \geq 1 - (d/|\mathbb{F}|) \geq 1 - \mathsf{negl}(\lambda) .$$

Proof. Let $M \in \mathbb{F}^{i \times 2^d}$ be the matrix such that the jth $(1 \leq j \leq i)$ row of M is $\mathsf{expd}(\mathbf{v}_j)$. Note that $\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_i) = 1$ implies that the rank of M is i. Let M' be the matrix obtained by adding the row vector $\mathsf{expd}(\mathbf{v}_{i+1})$ to M. Again $\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_{i+1}) = 1$ if and only if $\mathsf{rank}(M') = i+1 = \mathsf{rank}(M)+1$. Meanwhile we have $\mathsf{rank}(M') \geq \mathsf{rank}(M)$ and $\mathsf{kernel}(M') \subseteq \mathsf{kernel}(M)$ (as any vector $\mathbf{f}$ with $M' \cdot \mathbf{f} = 0^{i+1}$ also satisfies $M \cdot \mathbf{f} = 0^i$). Therefore, $\mathsf{rank}(M') = i+1 > \mathsf{rank}(M)$ if and only if $\mathsf{kernel}(M') \subsetneq \mathsf{kernel}(M)$ and it suffices to analyze the probability that exist a non-zero element in $\mathsf{kernel}(M)$ that's not in $\mathsf{kernel}(M')$. Note that $|\mathsf{kernel}(M)| = |\mathbb{F}|^{2^d - i} > 1$, thus we can pick a non-zero vector $\mathbf{f} \in \mathbb{F}^{2^d}$ from $\mathsf{kernel}(M)$. Now we check the probability that $\mathbf{f}$ is also in $\mathsf{kernel}(M')$ (i.e. $M' \cdot \mathbf{f} = \mathbf{0}^{i+1}$). This is the probability that $\langle \mathsf{expd}(\mathbf{v}_{i+1}), \mathbf{f} \rangle = f(\mathbf{v}_{i+1}) = 0$ for a uniformly random $\mathbf{v}_{i+1}$, where f is the d-variate multilinear polynomial with coefficients being $\mathbf{f}$. By the Schwartz-Zippel Lemma, this happens with probability at most $d/|\mathbb{F}|$. Thus $\mathsf{kernel}(M') \subsetneq \mathsf{kernel}(M)$ with probability at least $1 - d/|\mathbb{F}|$ and the lemma holds. $\square$

The algorithm A: Next, we construct a PPT algorithm A to be used in Lemma 9. To help with extraction of the polynomial f, we first define an algorithm A' that will additionally output some evaluation values. Given input challenges $\mathbf{v} \in \mathbb{F}^d$, the algorithm A' runs Protocol 4 with the prover $\mathcal{P}^*$ where the folding challenge vector is set as $\mathbf{v}$. After $\mathcal{P}^*$ outputs all of the oracles (from the sumcheck and the IOPP commit phase), A' simulates the verifier of Protocol 4 by sampling the query positions in $\mathsf{IOPP.query}$. The algorithm A' outputs 1 plus the claimed evaluations $y \in \mathbb{F}$ (from the sumcheck) if the PCS verifier accepts; otherwise A' outputs 0. After describing algorithm A', the algorithm $A(\mathbf{v})$ on input $\mathbf{v}$ simply runs $A'(\mathbf{v})$ and outputs 1 if and only if $A'(\mathbf{v})$ also outputs 1.

A' (and thus A) runs in polynomial time as the interaction with $\mathcal{P}^*$ in Protocol 4 runs in polynomial time. Moreover, the probability that A outputs 1 is exactly the probability that $\mathcal{P}^*$ pass the verification, which is non-negligible.

The polynomial extractor. By setting $N := 2^d$ and applying Lemma 9 given the above predicate Φ and algorithm A, we can obtain an extractor E that runs in time $T \in O(\lambda/\epsilon)$, and with probability at least $1 - T \cdot \mathsf{negl}(\lambda)$, E outputs 2^d vectors $[\mathbf{v}_1, \ldots, \mathbf{v}_{2^d}]$

[8]Intuitively, $\mathbf{v}_i$ will be the vector of folding challenges used in the execution of Protocol 4.

such that (i) $A(\mathbf{v}_i) = 1 \forall i \in [2^d]$, and (ii) $\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_{2^d}) = 1$, that is, the expansion vectors $\{\mathsf{expd}(\mathbf{v}_i)\}_{i \in [2^d]}$ are *linearly independent*.

Recall that $A(\mathbf{v}_i) = 1 \forall i \in [2^d]$ implies $A'(\mathbf{v}_i) = (1; y_i) \forall i \in [2^d]$ where $y_i \in \mathbb{F}^m$ is the additional output (i.e. the claimed evaluation) of $A'(\mathbf{v}_i)$. Given $(\mathbf{v}_1, \ldots, \mathbf{v}_{2^d})$ output by E, the polynomial extractor simply runs A' on inputs $\{\mathbf{v}_i\}_{i \in [2^d]}$ to obtain the evaluations[9] $(y_1, \ldots, y_{2^d})$, and attempts to solve the system of equations using Gaussian elimination to recover the coefficients of the polynomial f.

However, at this point, we do not have a guarantee that $(y_1, \ldots, y_{2^d})$ are the *correct* evaluations of f, thus it's possible that the polynomial extractor cannot recover f (as the system of equations might have no solution) even if the extractor E succeeds. Fortunately, by Lemma 7, the probability that $\mathcal{P}^*$ succeeds but outputs an incorrect evaluation is negligible (over a random input). Thus the probability that A' outputs 1 plus an incorrect evaluation is negligible. Recall that the extractor E only invokes A (and thus A') (on uniformly random $\mathbf{v} \in \mathbb{F}^d$) for polynomial number of times. Let T_A denote the number of invocations. From the above claim and by taking union bounds, we have that with probability at least $1 - T \cdot \mathsf{negl}(\lambda) - T_A \cdot \mathsf{negl}(\lambda)$, E outputs 2^d vectors $[\mathbf{v}_1, \ldots, \mathbf{v}_{2^d}]$ such that $\Phi(\mathbf{v}_1, \ldots, \mathbf{v}_{2^d}) = 1$; and for all $i \in [2^d]$, $A'(\mathbf{v}_i)$ outputs *correct* evaluation $f(\mathbf{v}_i)$ for the point $\mathbf{v}_i \in \mathbb{F}^d$. This implies that the polynomial extractor will successfully extract the coefficients of f using Gaussian elimination and the theorem holds. □

6 Experiments

In this section, we compare BaseFold PCS with state-of-the-art multilinear polynomial commitment schemes, including Brakedown [43], Zeromorph-`FRI` [53].[10] and Multilinear-KZG [58]. We instantiate Basefold PCS over both the random foldable code defined in Section 3 and over FFT-Friendly Reed-Solomon codes. We refer to the former as Basefold and the latter as BasefoldFri, as it is a generalization of `FRI` to the multilinear setting.

We measure the performance of these protocols both as stand-alone PCS as well as components of the SNARK from Hyperplonk [33]. We also compare Basefold to ECFFT2 [14], which is another field-agnostic error-correcting code that can be used with the FRI IOPP.

Methodology and setup. Our testbed is an AWS r6i.8xlarge EC2 instance, which has 16 cores and 256 GiB of RAM using Ubuntu 22. All experiments are run over 16 cores except for the ECFFT2 comparions, which we run over just 1 core to achieve a direct comparison with the existing implementation. We use the hash function Blake2s256 across all schemes. We test polynomial commitment schemes on both 256-bit fields and 64-bit

[9] Note that the claimed evaluations $(y_1, \ldots, y_{2^d})$ (from sumcheck) are fully deterministic given the input challenges $\{\mathbf{v}_i\}_{i \in [2^d]}$, as they are independent of the challenges from the IOPP query phase.

[10] Zeromorph is a `FRI`-based multilinear PCS constructed via the generic univariate-to-multilinear PCS compiler from [53]

fields. For the 64-bit field $\mathbb{F}_p$, as mentioned in Remark 2, we use an extension field of $\mathbb{F}_p$ in the IOPP and sum-check for soundness bootstrapping. The metrics we consider are prover time, verifier time, and proof size. The choices of parameters for Brakedown, `BaseFold`, and Zeromorph-FRI all achieve at least 100 bits of security.

Comparison to Field-Agnostic SNARKs

Field-agnosticity can enable extremely efficient SNARKs for certain applications. For instance, according to [62], a circom circuit encoding ECDSA signature verification uses 2^{20} gates when it is encoded over a field that is not native to the underlying SNARK, while a field-agnostic SNARK only needs circuits with 2^{14} multiplication gates to compute the same function. This discrepency stems from the fact that each non-native field operation incurs an overhead of approximately 2^6 constraints, according to standard techniques from [54]. For instance, as shown in Figure 2, Hyperplonk[Basefold] can prove (mock) ECDSA signature verification in only 122ms, while ZeromorphFri takes 2.9 *seconds*, 23 times slower.

To our knowledge, the only other field-agnostic SNARKs are derived from Brakedown PCS and ECFFT2([14]). Next, we compare Hyperplonk[Basefold] with Hyperplonk[Brakedown] and compare our random foldable code with ECFFT2, respectively.

Comparison with Brakedown. Hyperplonk[Brakedown] has a comparable prover speed to HyperPlonk[Basefold], but has a larger proof size and slower verifier time. For circuits with 2^{20} gates, Hyperplonk[Brakedown]'s prover time is 3.691s while Hyperplonk[Basefold]'s is 3.862s. Hyperplonk[Brakedown]'s proof size is 254MB while Hyperplonk[Basefold]'s is only 23MB, more than 10 times smaller. Finally, Hyperplonk[Brakedown]'s verifier time is 2.725s while Hyperplonk[Basefold]'s is 87ms, 31 times faster.

Remark 5. *As shown in Figures 5 and 6, Brakedown's prover is up to* 10 *times faster than Basefold as a PCS, but the SNARK prover times are comparable. This is largely due to the more expensive single PCS open protocol from BaseFold/FRI: in Brakedown, it is a one-round protocol that requires no additional hashing; while in* `BaseFold` *(and* `FRI`*), the prover runs in* $\log(n)$ *rounds of interaction and require* $\log n$ *additional Merkle Trees. For instance, we find that for* 25 *variables, the Brakedown opening phase is* 9.65 *times faster than* `BaseFold` *while the commit protocol is only* 3.2 *times faster. Fortunately, this gap is significantly reduced in the batch opening setting as BaseFold/FRI enables efficient batching techniques from [33]. We therefore expect Brakedown to be about* 3 *times faster than Basefold in the batched setting, as the commit protocol dominates the prover time in that case. In addition to batching, the Hyperplonk PIOP prover contributes some additional overhead (See Remark 8), so the SNARK prover times are ultimately comparable.*

Remark 6. *Recent work ([39]) halves the prover time and verifier time of the Brakedown PCS and slightly reduces the proof size (by a factor of 1.4). Similarly, [46] reduces*

ECDSA Circuit			
Protocol	Prover Time (ms)	Proof Size (KB)	Verifier Time (ms)
Hyperplonk[Basefold]	122	6258	24
Hyperplonk[Brakedown]	168	32271	797
Hyperplonk[ZeromorphFri]	2888	7739	47
HyperPlonk[MKZG]	71027	7.74	107

Table 2: Prover time, proof size and verifier time for an ECDSA signature verification circuit. We use the lowest benchmarked rates for Basefold and Brakedown, which minimizes their verifier work, and we use the highest benchmarked rate for ZeromorphFri, which minimizes its prover time.

the encoding time of the Brakedown PCS by 25 *percent. This may slightly reduce Hyperplonk[Brakedown]'s proving time and we leave rigorous benchmarking to future work.*

Comparison with ECFFT2. Next, we compare our random foldable code with ECFFT2 [14]. In order for ECFFT2 to be integrated into a multilinear PCS and used with HyperPlonk, the prover needs to first convert a polynomial from the standard coefficient basis to a special basis over an elliptic curve. This is referred to as the ENTER protocol, which has complexity $O(n \log^2(n))$. For $n = 2^{20}$, it takes 26.894s to run the ENTER algorithm, which is 16 times slower than the Basefold's encoding algorithm, which takes only 1.67s.

Evaluation	
Protocol	Time(ms)
ECFFT Enter	26894
ECFFT Extend	2726
Basefold Encode	1670

Table 3: Evaluation

Interpolation	
Protocol	Time(ms)
ECFFT Reverse Extend	2339
Interpolate BH	168

Table 4: Interpolation

Remark 7. *When ECFFT2 is integrated into STARK, as described in [13], the witness can be directly encoded into the appropriate basis, using protocols EXTEND and Reverse EXTEND, which run in $O(n \log n)$ and together take 5s. This is comparable to the analgous operations in Hyperplonk[Basefold] (overall about* 2.5 *times slower). However, as noted in the introduction, the STARK proof system has considerable additional overhead when proving a large circuit with high-degree custom-gates.*

Comparison to Non-Field Agnostic SNARKs

In this section, we compare HyperPlonk[BaseFold] with HyperPlonk[Zeromorph] and HyperPlonk[MultilinearKZG], respectively. Here HyperPlonk[MultilinearKZG] refers to plugging HyperPlonk with the PCS from [57].

We find that *over the same 256-bit finite field*, BasefoldPCS's prover is approximately 2.7 times faster than ZeromorphFri's for polynomials with more than 20 variables. This difference is due to the fact that the opening phase of ZeromorphFri requires the equivalent of two more commitment computations (of complexity $O(n \log n)$) while the Basefold `open` protocol runs in strictly linear time. The Basefold verifier is about 1.2 times slower than ZeromorphFri's verifier and its proofs are approximately 2.8 times bigger. This is both because the relative minimum distance of the underlying code of Basefold is lower than that of RS-codes and because the soundness proof for BasefoldPCS requires more queries from the verifier than the soundness proof from `FRI` does (using techniques from Deep-FRI [21]). We leave it to future work to determine whether similar techniques can be applied to BasefoldPCS.

Over *the same 64-bit finite field*, BasefoldPCS's prover is only about 1.7 times faster than ZeromorphFri's for polynomials with more than 21 variables. The reason for this is that over smaller fields, random foldable codes require lower rates to maintain a sufficiently high relative minimum distance. If we instead instantiate Basefold with a Reed-Solomon code, we can still use a rate of $\frac{1}{2}$ and in that case, BasefoldFri's prover time is approximately 3 times faster than ZeromorphFri's for polynomials with more than 20 variables. We compare prover time, verifier time, and proof size of PCS and SNARKs over a 64-bit field in Figures 6 and 8.

Next, we compare Basefold to MultilinearKZG [57], which requires a stronger cryptographic assumption, a trusted setup, and is not field-agnostic. As a SNARK, we found that for circuits with 2^{20} gates, Hyperplonk[MultilinearKZG] [57] took 71 seconds, whereas Hyperplonk[Basefold]'s prover (over a code of rate $\frac{1}{8}$) takes about 5.5 seconds. Hyperplonk[MultilinearKZG]'s verifier time was 107ms whereas Hyperplonk[Basefold]'s is 46ms but its proof size was only 7.3 *kilobytes*, whereas Hyperplonk[Basefold]'s is 9.5MB.

6.0.1 Analyzing the Tradeoff Between Prover and Verifier

Finally, we compare the performance of Hyperplonk[ZeromorphFri], Hyperplonk[Brakedown], and Hyperplonk[Basefold] across different rates. As demonstrated in Figure 4, Hyperplonk[Brakedown] has slightly faster proving times across rates, but at the cost of considerably larger proof sizes. On the other hand, we find that Hyperplonk[ZeromorphFri] has the slower proving times across rates, but with the advantage of having the smallest proof sizes. Hyperplonk[Basefold] strikes a balance between these two protocols. It has (slightly) faster prover for rates (1/2) and (1/4) at the cost of a (slightly) larger proof size.

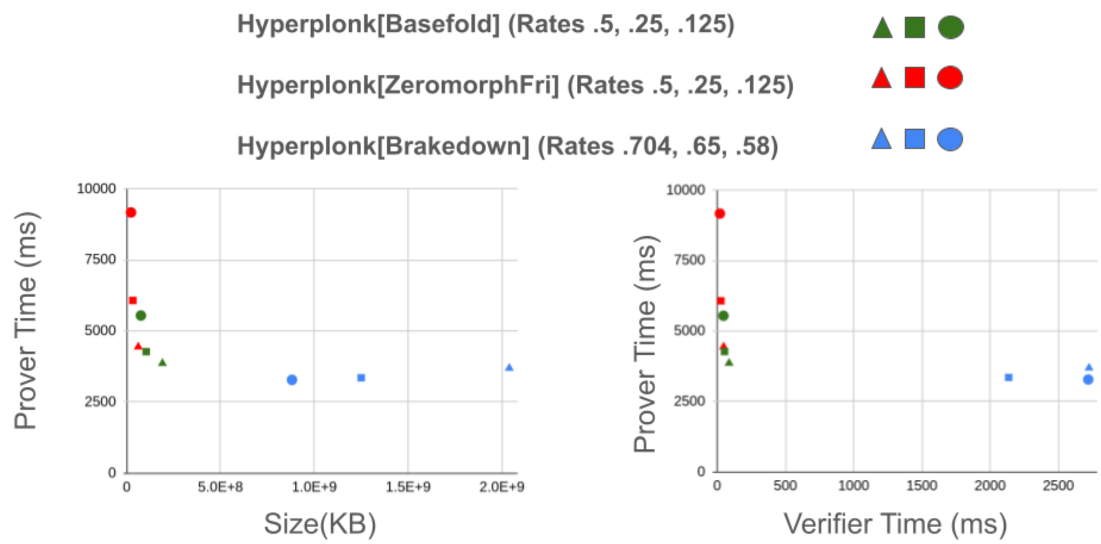

Figure 4: Hyperplonk[Basefold/ZeromorphFRI/Brakedown] over the same 256-bit field for circuits with 2^{20} gates with different encoding rates (i.e. the ratio $\frac{k}{n}$, for block length n, message length k.). Recall that Brakedown and Basefold are field-agnostic while ZeromorphFri is not.

Remark 8. *In the batch opening setting, Brakedown PCS is approximately 3 times faster than BaseFold PCS, while BaseFold PCS is 2-3 times faster than Zeromorph-FRI. However, when compiled into Hyperplonk, the prover times of the three schemes are comparable. This is largely due to the overhead imposed by the high-degree multivariate sum-check protocol from Hyperplonk PIOP [33]. We believe that the implementation of multivariate sum-checks can be further optimized and we leave it to future work.*

Acknowledgements

We want to thank Justin Thaler for providing helpful feedback to an earlier draft. We would also like to thank Han Jian for answering many questions about his plonkish repo[11], which we used for our benchmarks. Finally, we thank Daniel Lubarov for helpful discussions about the concrete performance of existing SNARKs.

[11] https://github.com/han0110/plonkish

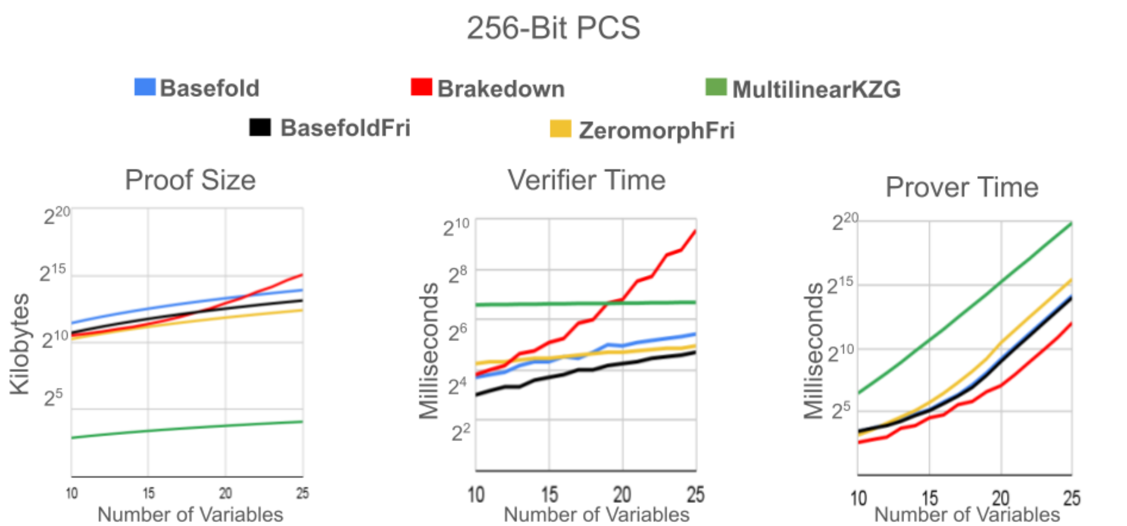

Figure 5: Performance of different PCS over 256-bit fields. Recall that Brakedown and Basefold are field-agnostic while Multilinear-KZG, ZeromorphFri, and BasefoldFri are not.

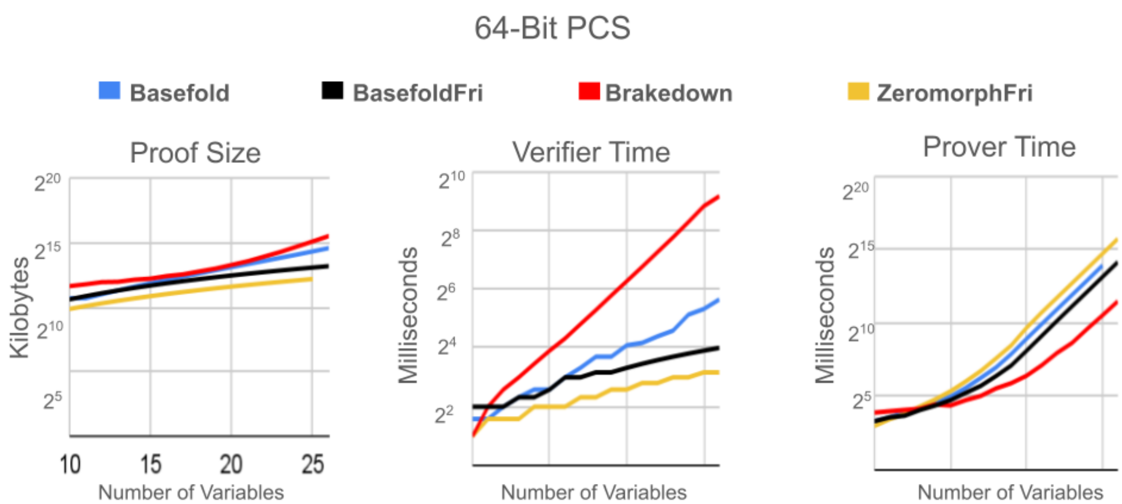

Figure 6: Performance of different PCS over (the extensions of) 64-bit fields. Recall that Basfold is field-agnostic while BasefoldFri and ZeromorphFri are not.

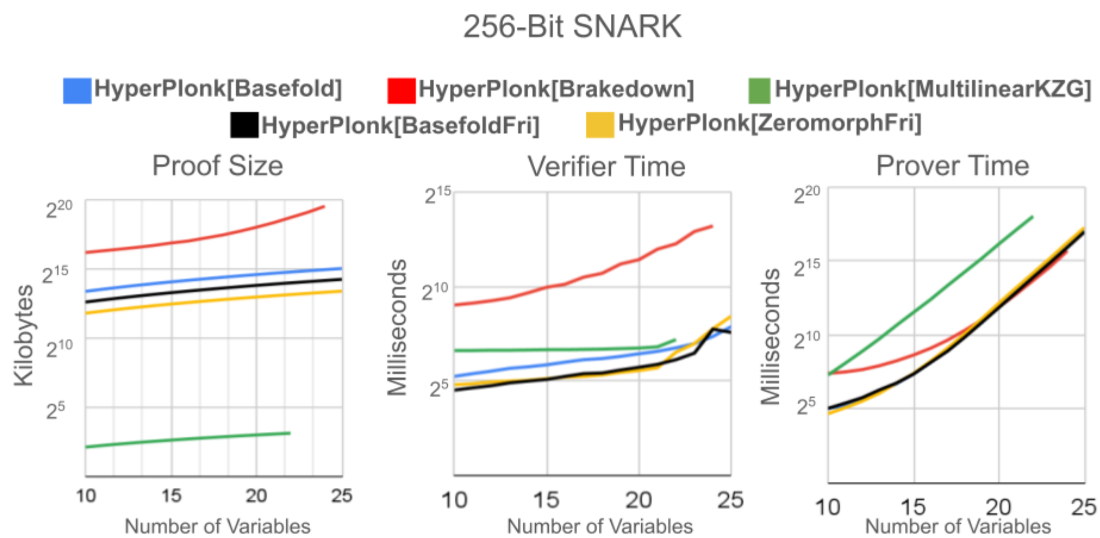

Figure 7: Performance of different SNARKs over the same 256-bit field. Recall that Brakedown and Basefold are field-agnostic while ZeromorphFri, MultilinearKZG, and BasefoldFri are not.

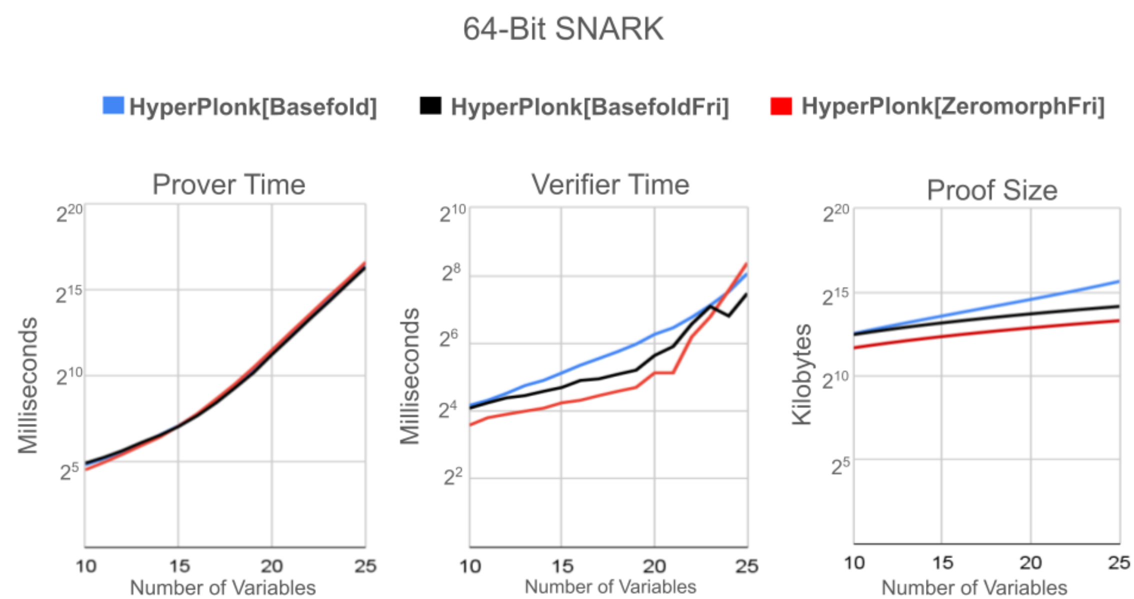

Figure 8: Hyperplonk[Basefold] and Hyperplonk[ZeromorphFRI] over *the same* 64-bit field. Note that Basefold is field-agnostic, while ZeromorphFri and BasefoldFri are not.

References

[1] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. *Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols*. Cryptology ePrint Archive, Report 2019/426. `https://eprint.iacr.org/2019/426`. 2019.

[2] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *ACM CCS 2017*. Ed. by Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM Press, 2017, pp. 2087–2104. DOI: `10.1145/3133956.3134104`.

[3] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. *Ligero: Lightweight Sublinear Arguments Without a Trusted Setup*. Cryptology ePrint Archive, Report 2022/1608. `https://eprint.iacr.org/2022/1608`. 2022.

[4] Gal Arnon, Alessandro Chiesa, and Eylon Yogev. "IOPs with Inverse Polynomial Soundness Error". In: *Cryptology ePrint Archive* 2023 (2023), p. 1062.

[5] Thomas Attema, Serge Fehr, and Nicolas Resch. *Generalized Special-Sound Interactive Proofs and their Knowledge Soundness*. Cryptology ePrint Archive, Paper 2023/818. `https://eprint.iacr.org/2023/818`. 2023. URL: `https://eprint.iacr.org/2023/818`.

[6] Daniel Augot, Sarah Bordage, and Jade Nardi. "Efficient multivariate low-degree tests via interactive oracle proofs of proximity for polynomial codes". In: *Designs, Codes and Cryptography* (2022). DOI: `10.1007/s10623-022-01134-z`. URL: `https://inria.hal.science/hal-03454113`.

[7] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking Computations in Polylogarithmic Time". In: *23rd ACM STOC*. ACM Press, May 1991, pp. 21–31. DOI: `10.1145/103418.103428`.

[8] Eli Ben-Sasson, Iddo Bentov, Ariel Gabizon, and Michael Riabzev. "A security analysis of Probabilistically Checkable Proofs". In: *Electron. Colloquium Comput. Complex.* TR16 (2016). URL: `https://api.semanticscholar.org/CorpusID:41937281`.

[9] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity". In: *Electron. Colloquium Comput. Complex.* 2017.

[10] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Fast Reed-Solomon Interactive Oracle Proofs of Proximity". In: *ICALP 2018*. Ed. by Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella. Vol. 107. LIPIcs. Schloss Dagstuhl, July 2018, 14:1–14:17. DOI: `10.4230/LIPIcs.ICALP.2018.14`.

[11] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Report 2018/046. https://eprint.iacr.org/2018/046. 2018.

[12] Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. "Scalable and Transparent Proofs over All Large Fields, via Elliptic Curves - (ECFFT Part II)". In: *TCC 2022, Part I*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13747. LNCS. Springer, Heidelberg, Nov. 2022, pp. 467–496. DOI: 10.1007/978-3-031-22318-1_17.

[13] Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. "Scalable and Transparent Proofs over All Large Fields, via Elliptic Curves - (ECFFT Part II)". In: *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13747. Lecture Notes in Computer Science. Springer, 2022, pp. 467–496. DOI: 10.1007/978-3-031-22318-1_17. URL: https://doi.org/10.1007/978-3-031-22318-1_17.

[14] Eli Ben-sasson, Dan Carmon, Swastik Kopparty, and David Levit. "Scalable and Transparent Proofs over All Large Fields, via Elliptic Curves: (ECFFT Part II)". In: Jan. 2023, pp. 467–496. ISBN: 978-3-031-22317-4. DOI: 10.1007/978-3-031-22318-1_17.

[15] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. *Short Interactive Oracle Proofs with Constant Query Complexity, via Composition and Sumcheck*. Cryptology ePrint Archive, Report 2016/324. https://eprint.iacr.org/2016/324. 2016.

[16] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. *Aurora: Transparent Succinct Arguments for R1CS*. Cryptology ePrint Archive, Report 2018/828. https://eprint.iacr.org/2018/828. 2018.

[17] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *EUROCRYPT 2019, Part I*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. LNCS. Springer, Heidelberg, May 2019, pp. 103–128. DOI: 10.1007/978-3-030-17653-2_4.

[18] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Theory of Cryptography Conference*. 2016. URL: https://api.semanticscholar.org/CorpusID:8363041.

[19] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. *Interactive Oracle Proofs*. Cryptology ePrint Archive, Report 2016/116. https://eprint.iacr.org/2016/116. 2016.

[20] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. “Interactive Oracle Proofs”. In: *TCC 2016-B, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. LNCS. Springer, Heidelberg, 2016, pp. 31–60. DOI: 10.1007/978-3-662-53644-5_2.

[21] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. “DEEP-FRI: Sampling Outside the Box Improves Soundness”. In: *ITCS 2020*. Ed. by Thomas Vidick. Vol. 151. LIPIcs, Jan. 2020, 5:1–5:32. DOI: 10.4230/LIPIcs.ITCS.2020.5.

[22] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. “Robust PCPs of proximity, shorter PCPs and applications to coding”. In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. 2004, pp. 1–10.

[23] Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. “Worst-Case to Average Case Reductions for the Distance to a Code”. In: *Proceedings of the 33rd Computational Complexity Conference*. CCC ’18. San Diego, California: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. ISBN: 9783959770699.

[24] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michal Zajac. *Fiat-Shamir Security of FRI and Related SNARKs*. Cryptology ePrint Archive, Paper 2023/1071. 2023. URL: https://eprint.iacr.org/2023/1071.

[25] Alexander R Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Micha Zajc. “Fiat-Shamir Security of FRI and Related SNARKs”. In: *Cryptology ePrint Archive* (2023).

[26] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. “Linear-Time Arguments with Sublinear Verification from Tensor Codes”. In: *TCC 2020, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Heidelberg, Nov. 2020, pp. 19–46. DOI: 10.1007/978-3-030-64378-2_2.

[27] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. “Gemini: Elastic SNARKs for Diverse Environments”. In: *EUROCRYPT 2022, Part II*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13276. LNCS. Springer, Heidelberg, 2022, pp. 427–457. DOI: 10.1007/978-3-031-07085-3_15.

[28] Sarah Bordage, Mathieu Lhotel, Jade Nardi, and Hugues Randriam. “Interactive Oracle Proofs of Proximity to Algebraic Geometry Codes”. In: *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*. Ed. by Shachar Lovett. Vol. 234. LIPIcs. 2022, 30:1–30:45. DOI: 10.4230/LIPIcs.CCC.2022.30. URL: https://doi.org/10.4230/LIPIcs.CCC.2022.30.

[29] Benedikt Bünz and Ben Fisch. *Schwartz-Zippel for multilinear polynomials mod N*. Cryptology ePrint Archive, Report 2022/458. https://eprint.iacr.org/2022/458. 2022.

[30] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. “Transparent SNARKs from DARK Compilers”. In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 677–706. DOI: 10.1007/978-3-030-45721-1_24.

[31] Matteo Campanelli, Nicolas Gailly, Rosario Gennaro, Philipp Jovanovic, Mara Mihali, and Justin Thaler. *Testudo: Linear Time Prover SNARKs with Constant Size Proofs and Square Root Size Universal Setup*. Cryptology ePrint Archive, Paper 2023/961. https://eprint.iacr.org/2023/961. 2023. URL: https://eprint.iacr.org/2023/961.

[32] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. *HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates*. Cryptology ePrint Archive, Report 2022/1355. https://eprint.iacr.org/2022/1355. 2022.

[33] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. “HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates”. In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Heidelberg, Apr. 2023, pp. 499–530. DOI: 10.1007/978-3-031-30617-4_17.

[34] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas Ward. *Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS*. Cryptology ePrint Archive, Report 2019/1047. https://eprint.iacr.org/2019/1047. 2019.

[35] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. “Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS”. In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 738–768. DOI: 10.1007/978-3-030-45721-1_26.

[36] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. *Fractal: Post-Quantum and Transparent Recursive Proofs from Holography*. Cryptology ePrint Archive, Report 2019/1076. https://eprint.iacr.org/2019/1076. 2019.

[37] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. “Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract)”. In: *26th FOCS*. IEEE Computer Society Press, Oct. 1985, pp. 383–395. DOI: 10.1109/SFCS.1985.64.

[38] Cas Cremers, Jaiden Fairoze, Benjamin Kiesl, and Aurora Naska. “Clone Detection in Secure Messaging: Improving Post-Compromise Security in Practice”. In: *ACM CCS 2020*. Ed. by Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna. ACM Press, Nov. 2020, pp. 1481–1495. DOI: 10.1145/3372297.3423354.

[39] Benjamin E. Diamond and Jim Posen. *Proximity Testing with Logarithmic Randomness*. Cryptology ePrint Archive, Paper 2023/630. https://eprint.iacr.org/2023/630. 2023. URL: https://eprint.iacr.org/2023/630.

[40] Irit Dinur and Omer Reingold. "Assignment testers: Towards a combinatorial proof of the PCP theorem". In: *SIAM Journal on Computing* 36.4 (2006), pp. 975–1024.

[41] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. *PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge*. Cryptology ePrint Archive, Report 2019/953. https://eprint.iacr.org/2019/953. 2019.

[42] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *J. ACM* 62.4 (2015). ISSN: 0004-5411. DOI: 10.1145/2699436. URL: https://doi.org/10.1145/2699436.

[43] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. *Brakedown: Linear-time and post-quantum SNARKs for R1CS*. Cryptology ePrint Archive, Report 2021/1043. https://eprint.iacr.org/2021/1043. 2021.

[44] Venkatesan Guruswami, Lingfei Jin, and Chaoping Xing. "Efficiently List-Decodable Punctured Reed-Muller Codes". In: *IEEE Transactions on Information Theory* 63 (2015), pp. 4317–4324. URL: https://api.semanticscholar.org/CorpusID:14176561.

[45] Venkatesan Guruswami and Jonathan Mosheiff. *Punctured Low-Bias Codes Behave Like Random Linear Codes*. 2022. arXiv: 2109.11725 [cs.CC].

[46] Ulrich Haböck. *Brakedown's expander code*. Cryptology ePrint Archive, Paper 2023/769. https://eprint.iacr.org/2023/769. 2023. URL: https://eprint.iacr.org/2023/769.

[47] Ulrich Haböck, Daniel Lubarov, and Jacqueline Nabaglo. "Reed-Solomon Codes over the Circle Group". In: https://eprint.iacr.org/2023/824. 2023. URL: https://eprint.iacr.org/2023/824.

[48] Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. *Foundations of Data Availability Sampling*. Cryptology ePrint Archive, Paper 2023/1079. https://eprint.iacr.org/2023/1079. 2023. URL: https://eprint.iacr.org/2023/1079.

[49] Ari Juels and Burton S. Kaliski Jr. "Pors: proofs of retrievability for large files". In: *ACM CCS 2007*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM Press, Oct. 2007, pp. 584–597. DOI: 10.1145/1315245.1315317.

[50] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. "Constant-Size Commitments to Polynomials and Their Applications". In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[51] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. *RedShift: Transparent SNARKs from List Polynomial Commitment IOPs*. Cryptology ePrint Archive, Report 2019/1400. `https://eprint.iacr.org/2019/1400`. 2019.

[52] Assimakis A. Kattis, Konstantin Panarin, and Alexander Vlasov. "RedShift: Transparent SNARKs from List Polynomial Commitments". In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 1725–1737. DOI: `10.1145/3548606.3560657`.

[53] Tohru Kohrita and Patrick Towa. "Zeromorph: Zero-Knowledge Multilinear-Evaluation Proofs from Homomorphic Univariate Commitments". In: *Cryptology ePrint Archive* 2023 (2023), p. 917.

[54] Ahmed E. Kosba, Charalampos Papamanthou, and Elaine Shi. "xJsnark: A Framework for Efficient Verifiable Computation". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 944–961. DOI: `10.1109/SP.2018.00018`.

[55] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic methods for interactive proof systems". In: *Journal of the ACM (JACM)* 39.4 (1992), pp. 859–868.

[56] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. "Algebraic Methods for Interactive Proof Systems". In: *Journal of the ACM* 39 (Apr. 1999). DOI: `10.1145/146585.146605`.

[57] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. *Signatures of Correct Computation*. Cryptology ePrint Archive, Report 2011/587. `https://eprint.iacr.org/2011/587`. 2011.

[58] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. "Signatures of Correct Computation". In: *TCC 2013*. Ed. by Amit Sahai. Vol. 7785. LNCS. Springer, Heidelberg, Mar. 2013, pp. 222–242. DOI: `10.1007/978-3-642-36594-2_13`.

[59] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. "Constant-round interactive proofs for delegating computation". In: *48th ACM STOC*. Ed. by Daniel Wichs and Yishay Mansour. ACM Press, June 2016, pp. 49–62. DOI: `10.1145/2897518.2897652`.

[60] Srinath Setty. *Spartan: Efficient and general-purpose zkSNARKs without trusted setup*. Cryptology ePrint Archive, Report 2019/550. `https://eprint.iacr.org/2019/550`. 2019.

[61] Srinath Setty, Justin Thaler, and Riad Wahby. *Customizable constraint systems for succinct arguments*. Cryptology ePrint Archive, Paper 2023/552. `https://eprint.iacr.org/2023/552`. 2023. URL: `https://eprint.iacr.org/2023/552`.

[62] Yi Sun, Tony L, Wen-Ding L, and gubsheep. *zk-ECDSA: zkSNARKs for ECDSA (Part 1)*. URL: `https://0xparc.org/blog/zk-ecdsa-1`.

[63] Alexander Vlasov and Konstantin Panarin. *Transparent Polynomial Commitment Scheme with Polylogarithmic Communication Complexity*. Cryptology ePrint Archive, Report 2019/1020. https://eprint.iacr.org/2019/1020. 2019.

[64] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. "Doubly-Efficient zkSNARKs Without Trusted Setup". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 926–943. DOI: 10.1109/SP.2018.00060.

[65] Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Orion: Zero Knowledge Proof with Linear Prover Time". In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Heidelberg, Aug. 2022, pp. 299–328. DOI: 10.1007/978-3-031-15985-5_11.

[66] Alex Luoyuan Xiong, Binyi Chen, Zhenfei Zhang, Benedikt Bünz, Ben Fisch, Fernando Krell, and Philippe Camacho. *VERI-ZEXE: Decentralized Private Computation with Universal Setup*. Cryptology ePrint Archive, Report 2022/802. https://eprint.iacr.org/2022/802. 2022.

[67] Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time". In: *ACM CCS 2021*. Ed. by Giovanni Vigna and Elaine Shi. ACM Press, Nov. 2021, pp. 159–177. DOI: 10.1145/3460120.3484767.

[68] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof". In: *2020 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2020, pp. 859–876. DOI: 10.1109/SP40000.2020.00052.

[69] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. "vRAM: Faster Verifiable RAM with Program-Independent Preprocessing". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2018, pp. 908–925. DOI: 10.1109/SP.2018.00013.

A Other Related Work

Polynomial Commitment Schemes. The notion of polynomial commitment schemes was first introduced in 2010 by Kate, Zaverucha, and Goldberg [50], who construct a univariate polynomial commitment scheme using bilinear groups. Polynomial commitment schemes with knowledge soundness (that are suitable to be used in SNARKs) were introduced in Marlin [34]. Several works extended the notion of polynomial commitment schemes to the multivariate setting (e.g. [57, 69, 31]). Since then, many practical multilinear polynomial commitment schemes with fast provers have been introduced, including Hyrax [64], Brakedown [43], Orion [65], and Orion+ [33].

Orion [65] presents a SNARK based on multilier PCS that has a linear-time prover and polylogarithmic proof size, through the use of *proof recursion*, which encodes the verifier computation into a SNARK circuit so that the prover can non-interactively prove that the verifier will accept. However, as mentioned in [39], the Orion verifier ends up running in linear time when compiled into a non-interactive proof via the Fiat-Shamir transform. Furthermore, Orion is not field-agnostic - as it only achieves polylogarithmic communication costs by using a proof system that is not field-agnostic in its recursive step.

In ECFFT2 [12], the authors present a method for using `FRI` over any finite field, however it is only efficient when used within a univariate PIOP, such as STARK. To use it within a *multilinear* PIOP, the runtime of their prover is asymptotically $O(n \log^2(n))$. In a similar vein, the authors of [47] present a method for encoding a Reed-Solomon code over the super efficient Mersenne prime finite-field $GF(2^{31} - 1)$, which is progress towards eventually enabling `FRI` over this field. However, it has a constant overhead that cancels out the gains of the more efficient finite field. Additionally, this solution only enables `FRI` over one additional field, while our goal is to enable `FRI`-like IOPPs over all finite fields while maintaining similar prover and verifier costs.

Interactive Oracle Proofs of Proximity (IOPP). The notion of IOPP was introduced by [15] and [59] independently. IOPPs are analogous to PCPPs [22, 40] but with multiple rounds: in each round, the verifier sends random challenges and the prover replies with oracle messages; in the last round, the verifier makes oracles queries to prover messages. Both IOPPs and PCPPs can be used to test the proximity of a vector to an error-correcting code, but interaction has the benefit of reducing the proof length and prover complexity without compromising soundness. Two works [8, 15] present IOPPs that are linear in the proof length but are still $O(n\mathsf{poly}(\log(n)))$ in prover complexity. `FRI` [10] improved this result and presented a linear-time IOPP for Reed-Solomon codes. The introduction of `FRI` [10] leads to extensive study of IOPPs in the context of SNARKs. For example, `FRI` was used to construct proof systems in [11, 16, 68, 67] and was used to build polynomial commitment schemes in [17, 63, 32]. Several works study the security of the Fiat-Shamir heuristic when applied to `FRI`, including [25, 5]. Additionally, there have been works generalizing the `FRI` IOPP to other codes [12, 28, 6]. E.g., Ligero [3] presented an IOPP that can be used with tensor codes, and Brakedown [43] extends this to codes that are additionally linear-time encodable.

Interactive Oracle Proofs. Interactive Oracle Proofs are analogous to PCPs [7] but with multiple rounds. They were introduced and formalized by Ben-Sasson, Chiesa, and Spooner in [19], which also presented a generic compilation from IOP to Non-interactive Argument of Knowledge in the random oracle model. The transformation can be useful in compiling IOPs for polynomial evaluation relations into polynomial commitment schemes [38, 2, 36]. Polynomial IOP [35, 30, 59] is a variant of IOP where the prover messages are oracles to evaluations of polynomials. PIOPs can also be compiled into SNARKs [33, 41, 34] from polynomial commitments or through the transformation from [19].

B Deferred Proofs

B.1 Proof of Lemma 3

Proof. In this proof, we will show that the probability that $\mathsf{Enc}_d(\mathbf{m})$ has more than t zeroes is $\leq (\frac{1}{2})^\lambda$ for non-zero $\mathbf{m}$. To accomplish this, we consider the subset $S \subseteq [1, n_{d-1}]$ such that $\mathbf{m} \in m_{d-1}(S)$ and then consider positions outside of S on which $\mathsf{Enc}_d(\mathbf{m})$ is zero. We will show if $j \notin S$, then the event that $\mathsf{Enc}_d(\mathbf{m})[j] = 0$ is either an independent Bernulli trial with a very small probability of success or it is an event that happens with a probability of 0.

Let $S^+ = S \cup \{j + n_{d-1} : j \in S\}$ and let $\neg S^+ = [1, n_d] \setminus S^+$. For all $j \in S^+$, $\mathsf{Enc}_d(\mathbf{m})[j] = 0$ with a probability of 1 (by definition of $m_{d-1}(S)$). Therefore, we know that $\mathsf{Enc}_d(\mathbf{m})$ is zero everywhere on S^+. To complete this proof, we need to show that the probability that $\mathsf{Enc}_d(\mathbf{m})$ has more than $t_d - |S^+|$ zeroes at positions outside of S^+ is as stated in Equation 6. Since S^+ is a set of pairs $(j, j + n_{d-1})$ for $j \in S$, it follows that $\neg S^+$ is also a set of pairs. Therefore, without loss of generality, we can reason over the set of representatives, $\neg S = [1, n_{d-1}] \setminus S$, and for any position $j \in \neg S$, we can refer to position $j + n_{d-1}$ when needed.

We consider the subset $\neg S^* \subseteq \neg S$ such that,

$$\neg S^* = \{j \in [1, n_{d-1}] \setminus S : \mathsf{Enc}_{d-1}(\mathbf{m}_r)[j] \neq 0\}\,.$$

Let $\mathbf{t} = \mathsf{diag}(T)$. For each $j \in \neg S^*$, set $A_j = \mathsf{Enc}_{d-1}(\mathbf{m}_l)[j]$ and $B_j = \mathsf{Enc}_{d-1}(\mathbf{m}_r)[j]$ and define $f_j(x) = A_j + xB_j$. Note that by only considering $j \in \neg S^*$, we guarantee that $f_j(x)$ is a non-zero polynomial since B_j is non-zero. We now consider the event that $f_j(x)$ evaluates to 0 at *either* $\mathbf{t}[j]$ or $-\mathbf{t}[j]$. Without loss of generality, we are able to confine our analysis to $\neg S^*$ because for every j in $\neg S$ but not in $\neg S^*$, with full certainty that $\mathsf{Enc}_d(\mathbf{m})[j] \neq 0$, because then A_j is not zero and so $f_j(x)$ is a non-zero constant polynomial.

For each $j \in \neg S^*$, define the random variable

$$X_j = 1\{f_j(\mathbf{t}[j]) = 0\} + 1\{f_j(-\mathbf{t}[j]) = 0\}$$

First, we observe that X_j is an independent Bernulli Trial since $\mathbf{t}[j]$ is an independent sample from $\mathbb{F}^\times$. We now evaluate the probability mass function of the random variable X_j. Let $z_j \in \mathbb{F}^\times$ be the unique non-zero root of f_j such that $f_j(z_j) = 0$. $1\{f_j(\mathbf{t}[j]) = 0\}$ is equal to 1 when $\mathbf{t}[j] = z_j$ and $1\{f_j(-\mathbf{t}[j]) = 0\}$ when $\mathbf{t}[j] = -z_j$. Therefore, $X_j = 2$ corresponds with the event that $\mathbf{t}[j] = z_j = -z_j$, which is impossible for non-zero z_j. $X_j = 1$ corresponds with the event that t_j is equal to either z_j or $-z_j$, which happens with probability $\frac{2}{|\mathbb{F}|-1}$ and $X_j = 0$ corresponds with the event that t_j is not equal to z_j or $-z_j$. Therefore, we can write the probability mass function as follows.

$$PMF(X_j) = \left\{ \begin{array}{l} \Pr[X_j = 2] = 0 \\ \Pr[X_j = 1] = \frac{2}{|\mathbb{F}|-1} \\ \Pr[X_j = 0] = 1 - \frac{2}{|\mathbb{F}|-1} \end{array} \right\}$$

Define the random variable $X = \sum_{j \in \neg S^*} X_j$. X has a binomial distribution with $\neg S^*$ trials and success probability of $\frac{2}{|\mathbb{F}|-1}$. We need to compute the probability that $X \geq t_d - 2|S|$, ie that there are more than $t_d - 2|S|$ successess out of $|\neg S^*|$ trials, where each trial is an independent Bernulli trial with probability $\frac{2}{|\mathbb{F}|-1}$. This can be computed using the cumulative distribution function as follows.

$$\begin{aligned} & \Pr_{T \leftarrow \mathbb{F}^{n_{d-1}}} [X \geq t_d - 2|S|] && (11) \\ & \leq \sum_{i=t_d-2|S|}^{|\neg S^*|} \binom{|\neg S^*|}{i} \cdot (\frac{2}{|\mathbb{F}|-1})^i \cdot (1 - \frac{2}{|\mathbb{F}|-1})^{|\neg S^*|-i} && (12) \\ & \leq |\neg S^*| \cdot 2^{|\neg S^*|} \cdot (\frac{2}{|\mathbb{F}|-1})^{t_d-2|S|} && (13) \\ & \leq |\neg S| \cdot 2^{|\neg S|} \cdot (\frac{2}{|\mathbb{F}|-1})^{t_d-2|S|} \qquad (\neg S^* \subseteq \neg S) && (14) \\ & = |[1, n_{d-1}] \setminus S| \cdot 2^{|[1,n_{d-1}] \setminus S|} \cdot (\frac{2}{|\mathbb{F}|-1})^{t_d-2|S|} && (15) \\ & = (n_{d-1} - |S|) \cdot 2^{n_{d-1}-|S|} \cdot (\frac{2}{|\mathbb{F}|-1})^{t_d-2|S|} && (16) \\ & \leq n_{d-1} \cdot 2^{n_{d-1}-|S|} \cdot (\frac{2}{|\mathbb{F}|-1})^{t_d-2|S|} && (17) \end{aligned}$$

By definition of $m_{d-1}(S)$, $\mathsf{Enc}_d(\mathbf{m})$ is 0 at every position in S^+. Therefore $\mathsf{nzero}(\mathsf{Enc}_d(\mathbf{m})) = X + |S^+| = X + 2|S|$ and so $\Pr[\mathsf{nzero}(\mathsf{Enc}_d(\mathbf{m})) \geq t_d] = \Pr[X \geq t_d - 2|S|]$ which is less than or equal to $n_{d-1} \cdot 2^{n_{d-1}-|S|} \cdot (\frac{2}{|\mathbb{F}|-1})^{t_d-2|S|}$ by Equation 17.

Finally, we show that for $|\mathbb{F}| \geq 2^{10}$, $\frac{2}{|\mathbb{F}|-1} \leq \frac{2.002}{|\mathbb{F}|}$. We solve the following for x, where $\tau = \log(|\mathbb{F}|)$

$$\frac{2}{|\mathbb{F}|-1} = \frac{x}{|\mathbb{F}|} \implies x = \frac{2|\mathbb{F}|}{|\mathbb{F}|-1} = \frac{2^{\tau+1}}{2^\tau - 1}$$

The function $f(\tau) = \frac{2^{\tau+1}}{2^\tau-1}$ is decreasing. Therefore for $\tau \geq 10$, $f(\tau) \leq f(10) = \frac{2^{11}}{2^{10}-1} = 2.002$, which completes the proof. $\square$

B.2 Proof of Lemma 4

Proof. Recall that

$$\mathbf{G}_d := \begin{bmatrix} \mathbf{G}_{d-1} & \mathbf{G}_{d-1} \\ \mathbf{G}_{d-1} \cdot T & \mathbf{G}_{d-1} \cdot -T \end{bmatrix} .$$

The statement of the lemma assumes that $\mathbf{G}_{d-1}$ is the generator matrix of a code such that the encoding of any non-zero messages $\mathbf{m} \in \mathbb{F}^{k_d-1}$ has fewer than t_{d-1} zeroes. By Lemma 2 and Lemma 3 and by definition of $t_d := 2t_{d-1} + \ell_d$, we obtain that,

$$\begin{aligned}
&\Pr_{\mathsf{diag}(\mathbf{T}_d) \leftarrow \$ (\mathbb{F}^\times)^{n_{d-1}}} \left[\exists \mathbf{m} \in \mathbb{F}^{k_d} \setminus \{\mathbf{0}\} : \mathsf{nzero}(\mathsf{Enc}_d(\mathbf{m})) \geq t_d \right] \\
&\leq \sum_{\mathbf{m} \in \mathbb{F}^{k_d} \setminus \{\mathbf{0}\}} \Pr_{\mathsf{diag}(\mathbf{T}_d) \leftarrow \$ (\mathbb{F}^\times)^{n_{d-1}}} [\mathsf{nzero}(\mathsf{Enc}(\mathbf{m})) \geq t_d] && \text{(Union Bound)} \\
&\leq \sum_{S \subseteq [1, n_{d-1}]} \sum_{\mathbf{m} \in m_{d-1}(S)} \Pr_{\mathsf{diag}(\mathbf{T}_d) \leftarrow \$ (\mathbb{F}^\times)^{n_{d-1}}} [\mathsf{nzero}(\mathsf{Enc}_d(\mathbf{m})) \geq t_d] && (\textstyle\bigcup m_{d-1}(S) \text{ covers } \mathbb{F}^{k_d}) \\
&\leq \sum_{S \subseteq [1, n_{d-1}]} |\mathbb{F}|^{2t_{d-1} - 2|S|} \cdot n_{d-1} \cdot 2^{n_{d-1} - |S|} \cdot \left(\frac{2.002}{|\mathbb{F}|} \right)^{2t_{d-1} + \ell_d - 2|S|} && \text{(Lemma 2 and 3)} \\
&= \sum_{S \subseteq [1, n_{d-1}]} (\frac{|\mathbb{F}|}{|\mathbb{F}|})^{2t_{d-1} - 2|S|} \cdot n_{d-1} \cdot 2^{n_{d-1} - |S|} \cdot 2.002^{2t_{d-1} - 2|S|} \left(\frac{2.002}{|\mathbb{F}|} \right)^{\ell_d} && \text{(Rearranging terms)}
\end{aligned} \tag{18}$$

Simplifying and moving all terms that are independent of S outside the sum, Equation 18 is equal to

$$\begin{aligned}
&n_{d-1} \cdot 2^{n_{d-1}} \cdot (2.002)^{2t_{d-1}} \left(\frac{2.002}{|\mathbb{F}|} \right)^{\ell_d} \left(\sum_{S \subseteq [1, n_{d-1}]} 2^{-|S|} \cdot (2.002)^{-2|S|} \right) \\
&= n_{d-1} \cdot 2^{n_{d-1}} \cdot (2.002)^{2t_{d-1}} \left(\frac{2.002}{|\mathbb{F}|} \right)^{\ell_d} \left(\sum_{x \in [0, n_{d-1}]} \binom{n_{d-1}}{x} \cdot 2^{-x} \cdot (2.002)^{-2x} \right)
\end{aligned} \tag{19}$$

Next, we evaluate the sum in Equation 19.

$$\sum_{x\in[0,n_{d-1}]} \binom{n_{d-1}}{x} \cdot 2^{-x} \cdot (2.002)^{-2x} \tag{20}$$

$$\leq \sum_{x\in[0,n_{d-1}]} \left(\frac{n_{d-1}\exp(1)}{x}\right)^x \cdot (2^{-3x}) \tag{21}$$

$$= \sum_{x\in[0,n_{d-1}]} \left(\frac{2^{\log(n_{d-1})+\log(\exp(1))}}{2^{\log(x)}}\right)^x \cdot (2^{-3x}) \tag{22}$$

$$= \sum_{x\in[0,n_{d-1}]} 2^{x(\log(\frac{n_{d-1}}{x})+\log(\exp(1))-3)} \tag{23}$$

$$= \sum_{x\in[0,n_{d-1}]} 2^{x(\log(\frac{n_{d-1}}{x})-1.55)} \tag{24}$$

The function $f(x) = x(\log(\frac{n_{d-1}}{x}) - 1.55)$ has a maximum at $x = 0.126 \cdot n_{d-1}$. Therefore Equation 24 is less than or equal to

$$n_{d-1} \cdot 2^{0.126 \cdot n_{d-1}(\log(\frac{n_{d-1}}{0.126 n_{d-1}})-1.55)} = n_{d-1} \cdot 2^{1.43 \cdot 0.126 \cdot n_{d-1}} \leq n_{d-1} 2^{\frac{n_{d-1}}{5.55}} \tag{25}$$

Therefore,

$$\begin{aligned}
&\Pr_{\mathsf{diag}(\mathbf{T}_d) \leftarrow\$ (\mathbb{F}^*)^{n_{d-1}}}[\exists \mathbf{m} \in \mathbb{F}^{k_d} \setminus \{\mathbf{0}\} : \mathsf{nzero}(\mathsf{Enc}_d(\mathbf{m})) \geq t_d] \\
&\leq n_{d-1}^2 \cdot 2^{n_{d-1}} \cdot (2.002)^{2t_{d-1}} \left(\frac{2.002}{|\mathbb{F}|}\right)^{\ell_d} \cdot 2^{\frac{n_{d-1}}{5.55}} \\
&\leq n_{d-1}^2 \cdot 2^{n_d(\frac{1}{2}+\frac{1}{2\cdot 5})} \cdot (2.002)^{n_d(1-\Delta_{C_d})} \cdot \left(\frac{2.002}{|\mathbb{F}|}\right)^{\ell_d}
\end{aligned} \tag{26}$$

Finally, by plugging in the value of ℓ_d in the statement of Theorem 2, the Formula 26 is no more than $2^{-\lambda}$, which completes the proof. □

B.3 Proof of Lemma 3 (IOPP Soundness)

Proof. We first define a bad event B in the commit phase: Let $(\alpha_{d-1}, \ldots, \alpha_0)$ be the folding challenges output by the verifier and let $(\pi_d, \ldots, \pi_0)$ be the prover oracles. Intuitively, the bad event happens if for some $i \in [0, d-1]$, the "folding" of the oracle π_{i+1} with challenge α_i has significantly smaller relative Hamming distance to C_i compared to the distance between π_{i+1} and C_{i+1}. More formally, the bad event B happens if there exists $i \in [0, d-1]$ such that

$$\Delta(\mathsf{fold}_{\alpha_i}(\pi_{i+1}), C_i) \leq \min(\Delta^*(\pi_{i+1}, C_{i+1}), J_\gamma(J_\gamma(\Delta_{C_d}))) - \gamma\,,$$

where Δ^* is defined in Definition 2 and $\mathsf{fold}_{\alpha_i}(\pi_{i+1})$ is defined as follows: let $\mathbf{u}, \mathbf{u}' \in \mathbb{F}^{n_i}$ be the unque interpolated vectors such that

$$\pi_{i+1} = (\mathbf{u} + \mathsf{diag}(T_i) \circ \mathbf{u}', \mathbf{u} + \mathsf{diag}(T_i') \circ \mathbf{u}') \,,$$

then $\mathsf{fold}_{\alpha_i}(\pi_{i+1})$ is set to

$$\mathsf{fold}_{\alpha_i}(\pi_{i+1}) := \mathbf{u} + \alpha_i \cdot \mathbf{u}' \,. \tag{27}$$

Next, we prove that the bad event B happens with probability at most $\frac{2d}{\gamma^3|\mathbb{F}|}$, which is implied by the following corollary that adapts Corollory 7.3 from [23] to general foldable linear codes.

Corollary 1 (Adapted from Corollory 7.3 from [23])**.** *Fix any $i \in [0, d-1]$ and any $\gamma, \delta > 0$ such that $\delta \leq J_\gamma(J_\gamma(\Delta_{C_d})))$. Then if $\Delta^*(\mathbf{v}, C_{i+1}) > \delta$ then*

$$\Pr_{\alpha_i \leftarrow \$ \mathbb{F}}[\Delta(\mathsf{fold}_{\alpha_i}(\mathbf{v}), C_i) \leq \delta - \gamma] \leq \frac{2}{\gamma^3|\mathbb{F}|} \tag{28}$$

where $\mathsf{fold}_{\alpha_i}(\mathbf{v})$ is defined as in Eqn. 27.

Proof. Let $\mathbf{u}, \mathbf{u}' \in \mathbb{F}^{n_i}$ be the two unique vectors such that $\mathsf{fold}_{\alpha_i}(\mathbf{v}) = \mathbf{u} + \alpha_i \mathbf{u}'$. Let $U = \{\mathbf{u} + x\mathbf{u}' : x \in \mathbb{F}\}$ and let $\hat{U}$ be the set of elements in U that have distance less than $\delta - \gamma$ from C_i. Assume for contradiction that $|\hat{U}| > \frac{2}{\gamma^3}$. Then Theorem 4.4 from [23] implies the existence of $\mathbf{w}', \mathbf{w} \in C_i$ and a subset $T \subseteq [1, n_i]$, $|T| \geq (1-\delta)n_i$, such that $\mathbf{w}'[t] = \mathbf{u}'[t]$ and $\mathbf{w}[t] = \mathbf{u}[t]$ for all $t \in T$. Since $\mathbf{w}, \mathbf{w}'$ are codewords in C_i, by definition of C_{i+1}, the following is a codeword in C_{i+1}

$$c_w = \left(\mathbf{w} + \mathsf{diag}(T_{i-1}) \circ \mathbf{w}', \mathbf{w} + \mathsf{diag}(T_{i-1}') \circ \mathbf{w}'\right) \,. \tag{29}$$

Therefore, for each $t \in T$, c_w agrees with $\mathbf{v}$ at positions t and $t+n_i$. Therefore $\Delta^*(\mathbf{v}, C_{i+1}) \leq \delta$, which contradicts with our assumption that $\Delta^*(\mathbf{v}, C_{i+1}) > \delta$. $\square$

From Corollary 1 and by taking union bound over d folding rounds, the bad event B happens with probability at most $\frac{2d}{\gamma^3|\mathbb{F}|}$.

Next, conditioned on the bad event B doesn't happen in the commit phase, we argue that $\mathsf{IOPP.query}$ outputs reject with probability at least $\delta - d\gamma$, which implies that the verifier outputs accept in all of the ℓ independent query executions with probability at most $(1 - \delta + d\gamma)^\ell$.

Fix any folding challenges $(\alpha_{d-1}, \ldots, \alpha_0)$ such that the bad event B doesn't happen. Let $(\pi_d, \ldots, \pi_0)$ be the prover oracles. Without loss of generality we can slightly modify the oracle strings $(\pi_{d-1}, \ldots, \pi_1)$ (but not π_d or π_0) without increasing its rejecting probability in $\mathsf{IOPP.query}$. We can understand the oracle entries of $(\pi_d, \ldots, \pi_0)$ as the nodes of binary trees. For every $i \in [0, d-1]$ and every $\mu \in [n_i]$, node (i, μ) has two children $(i+1, \mu)$, $(i+1, \mu + n_i)$, and we say (i, μ) is a *bad node* if $\pi_i[\mu]$ is inconsistent with $\pi_{i+1}[\mu]$ and

$\pi_{i+1}[\mu + n_i]$ in terms of the folding operation (Eqn. 27). Note that in $\mathsf{IOPP.query}$, given a challenge query index $\mu \in [n_{d-1}]$, the verifier outputs reject if and only if there is at least one bad node in the path Q_μ queried by the verifier. We modify the oracle strings as follows: scan the tree top-down with $i = 0, \dots, d-2$ and left-right with $\mu = 1, \dots, n_i$. Whenever there is a bad node (i, μ), we reset the node values in the subtree of (i, μ) as follows: we go from layer $j = d-1$ to $i+1$ and for each node in the subtree, we set the node's oracle string value to be consistent with their children. Note that the modification doesn't change oracles π_0, π_d and we never turn a good node into a bad node. Thus the rejecting probability of $\mathsf{IOPP.query}$ never increase. Hence without loss of generality we can assume that the oracles $(\pi_d, \dots, \pi_0)$ has the form abvove. It remains to argue that the rejecting probability of $\mathsf{IOPP.query}$ is at least $\delta - \gamma d$.

It is easy to see that after the modification, the rejecting probability of $\mathsf{IOPP.query}$ is precisely $\sum_{i=0}^{d-1} \beta_i$, where $\beta_i := \Delta(\pi_i, \mathsf{fold}_{\alpha_i}(\pi_{i+1}))$ is the ratio of bad nodes in layer i.

Claim 2. *For every $i \in [0, d]$, define $\delta^{(i)} := \min(J_\gamma(J_\gamma(\Delta_{C_d})), \Delta^*(\pi_i, C_i))$. For all $i \in [0, d-1]$, we have*

$$\beta_i \geq \delta^{(i+1)} - \delta^{(i)} - \gamma\,.$$

Proof. By the condition that the bad event B doesn't happen, we have that for every $i \in [0, d-1]$,

$$\Delta(\mathsf{fold}_{\alpha_i}(\pi_{i+1}), C_i) > \delta^{(i+1)} - \gamma$$

On the other hand, by triangle inequality,

$$\Delta(\mathsf{fold}_{\alpha_i}(\pi_{i+1}), C_i) \leq \Delta(\mathsf{fold}(\pi_{i+1}), \pi_i) + \Delta(\pi_i, C_i) \leq \beta_i + \Delta^*(\pi_i, C_i)$$

where the last inequality follows by Lemma 6. Rearranging the terms we have

$$\beta_i > \delta^{(i+1)} - \Delta^*(\pi_i, C_i) - \gamma\,. \tag{30}$$

WLOG we can assume that $\delta^{(i)} < \delta^{(i+1)} - \gamma$ as otherwise the claim trivially holds. This implies that $\delta^{(i)} < \delta^{(i+1)} \leq J_\gamma(J_\gamma(\Delta_{C_d}))$ and thus $\delta^{(i)} = \Delta^*(\pi_i, C_i)$. From Eqn. 30, the claim holds. □

Recall that $\delta = \delta^{(d)}$, and $\Delta^*(\pi_0, C_0) = \Delta(\pi_0, C_0) = 0$ as otherwise the IOPP verifier will never accept, thus $\delta^{(0)} = 0$. By the claim above, we have

$$\delta = \delta^{(d)} - \delta^{(0)} = \sum_{i=0}^{d-1} \delta^{(i+1)} - \delta^{(i)} \leq \sum_{i=0}^{d-1} \beta_i + \gamma d\,,$$

which implies that $\sum_{i=0}^{d-1} \beta_i \geq \delta - \gamma d$ as desired and the theorem holds. □

B.4 Proof of Lemma 9 (Path Predicate Forking Lemma)

Proof. We will first construct an expected time extractor E that repeatedly samples $m \leftarrow\$ \mathcal{M}$ and checks that $A(m) = 1$. It repeats this process, sampling with replacement, until its gets N inputs $m_1, ...m_N \in \mathcal{M}$ such that $A(m_i) = 1$ for all $i \in [N]$. If the probability that $A(m) = 1$ over random m is exactly ϵ then E runs in expected time N/ϵ. Next, we use E to build a new algorithm E' that runs for time $T \in O(\lambda \cdot N/\epsilon)$ and succeeds with probability $(1 - \texttt{negl}(\lambda))$. This algorithm E' will start by running λ copies of E each for $2N/\epsilon$ steps. Technically, E' will not run them in parallel, but will iterate over the copies running each for one step at a time. By Markov's inequality, each copy terminates with probability at least 1/2. The probability no copy terminates is less than $2^{-\lambda}$. So at least one copy terminates with probability $1 - 2^{-\lambda}$. In other words, in each step E' is sampling a new message m_j, interpreted as the $\lfloor j/N \rfloor$th message in the $(j \bmod N)$th copy of E. Suppose towards contradictions that $\Phi(m_1, ..., m_N) \neq 1$. Let j be the smallest index for which $\Phi(m_1, ...m_j) \neq 1$. This means there occurred an event where $\Phi(m_1, ..., m_{j-1}) = 1$ and m_j was sampled randomly resulting in $\Phi(m_1, ..., m_j) \neq 1$, an event which happens with probability at most $\texttt{negl}(\lambda)$. The probability it occurred in any of the T steps is at most $T \cdot \texttt{negl}(\lambda)$ by a union bound. □

C Minimum Relative Distance Calculation for BaseFold

Let $d, k_0, c \in \mathbb{N}$ and let C_d be a (c, k_0, d) random foldable linear code. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| \geq 2^{10}$. The maximum number of 0s of C_d is given by

$$t_d = 2t_{d-1} + \ell_d$$

Therefore, the maximum *relative* number of 0s in C_d, Z_{C_d} is equal to the following reccurence relation

$$\begin{aligned}
Z_{C_0} &= \frac{1}{c}, \\
Z_{C_d} &= \frac{t_d}{n_d} = \frac{2t_{d-1}}{n_d} + \frac{\ell_d}{n_d} \\
&= Z_{C_{d-1}} + \frac{1}{\log(|\mathbb{F}|) - 1.001}\left(\frac{2\log(n_{d-1}) + \lambda}{n_d} + 1.001 Z_{C_{d-1}} + 0.6\right) \\
&= Z_{C_{d-1}} \cdot \left(\frac{\log(|\mathbb{F}|) - 1.001}{\log|\mathbb{F}| - 1.001}\right) + \frac{1.001 Z_{C_{d-1}}}{\log(|\mathbb{F}|) - 1.001} + (\frac{1}{\log(|\mathbb{F}|) - 1.001}) \cdot \left(\frac{2\log(n_{d-1}) + \lambda}{n_d} + 0.6\right) \\
&= Z_{C_{d-1}} \cdot \left(\frac{\log(\mathbb{F})}{\log(|\mathbb{F}|) - 1.001}\right) + \left(\frac{1}{\log(|\mathbb{F}|) - 1.001}\right) \cdot \left(\frac{2\log(n_{d-1}) + \lambda}{n_d} + 0.6\right)
\end{aligned}$$

Finally, Z_{C_d} resolves to

$$= \left(\frac{1}{c}\right)\left(\frac{\log(|\mathbb{F}|)}{\log(|\mathbb{F}|)-1.001}\right)^d + \sum_{i=1}^{d}\left(\frac{\log(|\mathbb{F}|)}{\log(|\mathbb{F}|)-1.001}\right)^{d-i}\left(\frac{0.6}{\log(|\mathbb{F}|)-1.001}+\frac{2\log(n_{i-1})+\lambda}{n_i(\log(|\mathbb{F}|)-1.001)}\right) \tag{31}$$

The sum in Equation 31 only has d terms and therefore can be computed very efficiently. To obtain the relative minimum distance, Δ_{C_d}, we simply subtract Z_{C_d} from 1. The Table 1 refers to (c, k_0, d)-random foldable linear codes that achieve the relative minimum distance with probability at least $(1 - 2^{-128})$.

D Foldable Linear Codes as Punctured Reed-Muller Codes

Lemma 11 (Foldable Punctured Reed-Muller Codes)**.** *Let C_d be a foldable linear code with generator matrices $(\mathbf{G}_0, .., \mathbf{G}_{d-1})$ and diagonal matrices $(T_0, .., T_{d-1}), (T'_0, .., T'_{d-1})$. Then there exists a subset $D \subset \mathbb{F}^d$ such that $C_d = \{(P(\mathbf{x}) : \mathbf{x} \in D) : P \in \mathbb{F}[X_1, .., X_d]\}$, i.e. each codeword in C_d is a vector whose elements are evaluations of a multilinear polynomial P at each point in D.*

Proof. We proceed by induction and for simplicity, we consider the case where C_0 is the repetition code. The statement is true for the base case as $\mathsf{Enc}_0(m) = m||..||m$ is the evaluation of a degree-0 polynomial m at c distinct points. For the inductive hypothesis, suppose that for $i < d$, there exists a set, D_i such that $C_i = \{(P(\mathbf{x}) : \mathbf{x} \in D_i) : P \in \mathbb{F}[X_1, .., X_i]\}$. Without loss of generality, we index elements of D_i by arbitrarily assigning an integer $j \in [1, c \cdot 2^i]$ to each element in D_i. We will denote x_j as the jth element of D_i according to this ordering.

Let $t = \mathsf{diag}(T_i), t' = \mathsf{diag}(T'_i), n_i = c \cdot 2^i, \mathbf{v} \in \mathbb{F}^{2^{i+1}}$ and let $P \in \mathbb{F}[X_1, .., X_{i+1}]$ be the polynomial with the elements of $\mathbf{v}$ as coefficients. Finally, let $P_l, P_r \in \mathbb{F}[X_1, .., X_i]$ such that $P(X_1, .., X_{i+1}) = P_l(X_1, .., X_i) + X_{i+1} \cdot P_r(X_1, .., X_i)$. Then,

$$\begin{aligned}
&\mathsf{Enc}_{i+1}(\mathbf{v}) \\
&= \mathsf{Enc}_i(\mathbf{v}_l) + \mathsf{diag}(T_i) \circ \mathsf{Enc}_i(\mathbf{v}_r) \quad || \quad \mathsf{Enc}_i(\mathbf{v}_l) + \mathsf{diag}(T'_i) \circ \mathsf{Enc}_i(\mathbf{v}_r) && (32) \\
&= (P_l(\mathbf{x}_1), .., P_r(\mathbf{x}_n)) + \mathsf{diag}(T_i) \circ (P_r(\mathbf{x}_1), .., P_r(\mathbf{x}_n)) && (33) \\
&|| \quad (P_l(\mathbf{x}_1), .., P_r(\mathbf{x}_n)) + \mathsf{diag}(T'_i) \circ (P_r(\mathbf{x}_1), .., P_r(\mathbf{x}_n)) \\
&= (P_l(\mathbf{x}_1) + t_1 P_r(\mathbf{x}_1), \cdots, P_l(\mathbf{x}_n) + t_n P_r(\mathbf{x}_n), P_l(\mathbf{x}_1) + t'_1 P_r(\mathbf{x}_1), \cdots, P_l(\mathbf{x}_n) + t'_n P_r(\mathbf{x}_n) && (34) \\
&= (P(\mathbf{x}_1, t_1), .., P(\mathbf{x}_n, t_n), P(\mathbf{x}_1, t'_1), .., P(\mathbf{x}_n, t'_n)) && (35)
\end{aligned}$$

where Equation (11) follows from Protocol 1, Equation (12) follows from the inductive hypothesis, Equation (13) follows from the definition of hadamard product and Equation (14) follows from definition of P. Thus, we set
$D_{i+1} = \{(\mathbf{x}_1, t_1), .., (\mathbf{x}_n, t_n), (\mathbf{x}_1, t'_1), .., (\mathbf{x}_n, t'_n)\}$, which completes the proof. □

E FRI as a foldable linear code

Let D be a multiplicative cyclic group of order n with generator g, where n is a power of 2. Let M be a Vandermonde matrix over D for degree-d polynomials. By definition of Vandermonde matrix, for each $j \leq n, i \leq d$, $M[i,j] = (g^j)^i = g^{j \cdot i}$. We consider $i \leq d$ such that $i = 2i'$ for $i' \leq \lfloor d/2 \rfloor$. Then $M[i,j] = M[i, j + n/2]$ because

$$M[i,j] = g^{ij} = g^{2i'j}$$
$$M[i, j+n/2] = g^{2i'(j+n/2)} = g^{2i'j} \cdot g^{2i' \cdot n/2} = g^{2i'j} \cdot g^{ni'} = g^{2i'j}$$

Furthermore, looking at each individual column $j \leq n$, consider *odd* i such that $i = 2i' + 1$ for $i' \leq \lfloor d/2 \rfloor$. Then

$$M[i,j] = g^{(2i'+1)j} = g^{2i'j+j} = g^{2i'j} \cdot g^j$$

Thus, each column vector $\mathbf{M}[\cdot, \mathbf{j}]$ is equal to $(g^0, g^j g^0, g^2, g^j g^2 ... g^{\lfloor d/2 \rfloor} g^j)$.

We create a matrix, M' as follows:

$$M' = \begin{bmatrix} M_{even} \\ M_{odd} \end{bmatrix}$$

where M_{even} are the even rows of M and M_{odd} are the odd rows of M. Then each column vector of $\mathbf{M}[\cdot, \mathbf{j}]$ is equal to $(M_{even}[\cdot, j], g^j \cdot M_{even}[\cdot, j])$. Since for all $j \leq n/2$, $M_{even}[j] = M_{even}[j + n/2]$, we set M^{d-1} to be the first $n/2$ columns of M_{even} and so

$$M' = \begin{bmatrix} M^{d-1} & M^{d-1} \\ M^{d-1} \cdot T_1 & M^{d-1} \cdot T_2 \end{bmatrix}$$

where T_1, T_2 are two diagonal matrices such that $\mathsf{diag}(T_1) = (g^0, .., g^{n/2})$ and $\mathsf{diag}(T_2) = (g^{n/2+1}, .., g^{n-1})$.