

FRI-Style Rank Metric PCS via FFT for Gabidulin Encoding

Songsong Li¹, Zhe Li², Shu Liu³, Chaoping Xing¹, and Chen Yuan¹

¹ School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
`{songsli,xingcp,chen_yuan}@sjtu.edu.cn`

² Institute of Mathematics and Interdisciplinary Sciences, Xidian University, Xi'an, China
`lizh0048@e.ntu.edu.sg`

³ National Key Laboratory on Wireless Communications, University of Electronic Science and Technology of China, Chengdu, China
`shuliu@uestc.edu.cn`

Abstract. Polynomial commitment schemes (PCSs) are fundamental cryptographic primitives, with applications to SNARKs, data availability, and privacy-preserving protocols. Many efficient transparent PCSs are built from interactive oracle proofs of proximity (IOPPs) for codes in the Hamming metric. In constructions based on the fast Reed-Solomon IOPP (FRI), low degree polynomials are encoded as Reed-Solomon (RS) codewords over suitable evaluation domains, and FRI is used to test proximity to the RS code. The efficiency of this approach relies on fast Fourier transform (FFT) based encoding and recursive proximity testing through folding operations.

Motivated by applications of rank metric codes in post quantum cryptography, random linear network coding, and distributed storage, we develop an analogous framework based on Gabidulin codes, the rank metric counterparts of RS codes. These codes encode q -linearized polynomials of bounded q -degree over $\mathbb{F}_{q^m}$ by evaluating them at $\mathbb{F}_q$ -linearly independent points in this field. We construct a recursive FFT for these polynomials and show that its algebraic structure supports both proximity testing for Gabidulin codes and succinct verification of polynomial evaluations, yielding a FRI-style PCS for q -linearized polynomials. Our contributions are as follows.

- (1) **FFT algorithms for q -linearized polynomials.** For every $O(1)$ -smooth positive integer $n \mid m$, we construct an $\mathbb{F}_q$ -linearly independent evaluation set $\mathcal{A} \subseteq \mathbb{F}_{q^m}$ of size n . On this set, we present FFT and inverse FFT algorithms for evaluating and interpolating q -linearized polynomials of q -degree less than n over $\mathbb{F}_{q^m}$. Both algorithms require $O(n \log n)$ operations in $\mathbb{F}_{q^m}$. Consequently, Gabidulin codes defined over $\mathcal{A}$ admit encoding with the same complexity.
- (2) **A FRI-style IOPP for Gabidulin codes.** The recursive structure of our FFT induces a folding operation that reduces a Gabidulin code instance of length n to one of length $n/2$. Combining this operation with recent proximity gap results for Gabidulin codes, we construct a FRI-style IOPP, which we call the Fast Gabidulin IOPP (FGI).

The soundness analysis accounts for the Frobenius twists introduced by folding and their effect on rank metric proximity.

- (3) **A FGI based PCS for q -linearized polynomials.** The usual quotient based opening reduction in PCSs for ordinary polynomials does not directly preserve for q -linearized polynomials. We instead use the FGI folding structure to recursively reduce the proximity claim and the evaluation claim together. Combined with Merkle tree commitments, this yields a transparent PCS for q -linearized polynomials with polylogarithmic proof size and verification time.

1 Introduction

A polynomial commitment scheme (PCS) enables a prover to succinctly commit to a polynomial and later prove its evaluations at chosen points. PCSs have become fundamental components of modern cryptographic systems, with applications to succinct non-interactive arguments of knowledge (SNARKs) [15,12,16,31], verifiable secret sharing [18,32], blockchain data availability sampling [17], and secure multiparty computation [8]. Many efficient transparent PCSs are built from algebraic error correcting codes in the Hamming metric, with Reed-Solomon (RS) codes playing a central role. Over suitable evaluation domains, fast Fourier transform (FFT) algorithms efficiently encode univariate polynomials as codewords in RS codes or their interleavings, while proximity to these codes can be tested using interactive oracle proofs of proximity (IOPPs), as in FRI [6] and the proximity testing protocol in Ligero [2], respectively. Combined with Merkle commitments and protocols for verifying polynomial evaluations, these ingredients underpin transparent PCSs and succinct arguments. We refer to PCSs constructed through this approach as IOPP based PCSs. Subsequent developments include STIR and WHIR [3,4], which refine RS proximity testing in FRI. Brakedown [16] follows the Ligero approach using efficiently encodable linear codes. BaseFold [31] extends the FRI approach using foldable codes. Blaze [11] combines code switching with interleaved RAA codes.

Motivations for rank metric PCSs. Applications of rank metric codes motivate extending IOPP based PCSs beyond the Hamming metric. For a vector $\mathbf{e} = (e_1, \dots, e_n)$ over a finite field $\mathbb{F}_{q^m}$, its rank weight is defined as $\text{wt}_R(\mathbf{e}) = \dim_{\mathbb{F}_q}(\text{span}_{\mathbb{F}_q}\{e_1, \dots, e_n\})$, which measures the number of linearly independent components rather than the number of nonzero coordinates. In random linear network coding, a few injected error packets may corrupt many received packets while the resulting error matrix remains of low rank [19,29]. Similarly, in distributed storage where repair uses $\mathbb{F}_q$ -linear combinations of symbols in extension field $\mathbb{F}_{q^m}$, static errors can spread across nodes while remaining within the original error space [5]. In these applications, receivers and storage clients may need to verify that recoded packets or repair contributions are consistent with authenticated source data without reconstructing the original file. This motivates homomorphic authentication for network coding [1,10] and verification for storage repair [27].

Gabidulin codes [14], the rank metric counterparts of RS codes, encode q -linearized polynomials $f(x) = \sum_{i=0}^{k-1} a_i x^{q^i}$ over $\mathbb{F}_{q^m}$ by evaluating them at $\mathbb{F}_q$ -linearly independent points $\alpha_1, \dots, \alpha_n$ in $\mathbb{F}_{q^m}$. The $\mathbb{F}_q$ -linearity of these polynomials gives a direct connection between linear combinations of encoded data and polynomial evaluation through the identity $\sum_{i=1}^n b_i f(\alpha_i) = f(\sum_{i=1}^n b_i \alpha_i)$ for $b_1, \dots, b_n \in \mathbb{F}_q$. Thus, PCSs for q -linearized polynomials can therefore support verification of such linear combinations against an authenticated commitment.

A complementary motivation arises in code based cryptography, particularly in proofs of knowledge for rank syndrome decoding relations. RYDE [9] uses a linearized polynomial to characterize the rank constraint in its MPC-in-the-head construction. A vector $\mathbf{e} \in \mathbb{F}_{q^m}^n$ satisfying a prescribed rank bound $\text{wt}_R(\mathbf{e}) \leq t$ if and only if there exists a monic q -linearized polynomial f of q -degree at most t such that $f(e_j) = 0$ for every coordinate j . This characterization motivates investigating q -linearized PCSs as building blocks for arguments involving polynomial witnesses and rank constraints.

Related work. Directly applying a standard PCS for ordinary polynomials to a q -linearized polynomial does not necessarily yield succinct verification with complexity sublinear in the input size. This is because the efficiency of ordinary PCSs is measured in terms of the polynomial's degree, whereas a q -linearized polynomial of q -degree less than k has only k coefficients but may have ordinary degree as large as q^{k-1} . Consequently, standard PCSs can incur prohibitive overhead. Our goal is to design a PCS whose verification and communication costs are sublinear in k .

IOPP based constructions suggest combining efficient Gabidulin encoding with proximity testing and an evaluation consistency protocol. Recent work [21] establishes proximity gaps for Gabidulin codes and constructs a Liger-style PCS for q -linearized polynomials. The resulting scheme achieves square root verifier time and proof size in k under the usual arithmetic cost model, analogous to Liger based constructions in the Hamming metric.

By contrast, FRI based PCSs for ordinary polynomials achieve polylogarithmic verifier complexity and proof size. Their efficiency relies on the FFT for ordinary polynomials. The recursive decomposition underlying the FFT supports both fast RS encoding and folding for proximity testing. Constructing an FRI-style PCS for q -linearized polynomials requires a compatible recursive decomposition and proximity testing for Gabidulin codes. This motivates the following question:

Can one develop a recursive FFT for q -linearized polynomials that supports both fast Gabidulin encoding and FRI-style proximity testing, thereby yielding a q -linearized PCS with efficiency comparable to that of FRI based PCSs?

1.1 Our Contributions

We first develop a pair of recursive evaluation and interpolation algorithms for q -linearized polynomials, analogues to the classic Cooley-Tukey FFT [13] for

ordinary polynomials. We begin by defining the q -linearized discrete Fourier transform (DFT) and its inverse (iDFT).

Definition 1.1. Let $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ be an ordered $\mathbb{F}_q$ -linearly independent subset of $\mathbb{F}_{q^m}$. For every q -linearized polynomial $f(x) = \sum_{i=0}^{n-1} a_i x^{q^i}$ of q -degree less than n , its DFT with respect to $\mathcal{A}$ is defined by $\text{DFT}_{\mathcal{A}}(f) = (f(\alpha_1), \dots, f(\alpha_n))$. Conversely, for any vector $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}_{q^m}^n$, its iDFT with respect to $\mathcal{A}$ is defined by $\text{iDFT}_{\mathcal{A}}(\mathbf{y}) = f$, where f is the unique q -linearized polynomial of q -degree less than n such that $\text{DFT}_{\mathcal{A}}(f) = \mathbf{y}$.

By [Lemma 2.1](#), $\text{DFT}_{\mathcal{A}}$ is a bijection. Thus iDFT is well-defined. We call algorithms that compute DFT and its inverse using $O(n \log n)$ operations over $\mathbb{F}_{q^m}$ as the q -linearized FFT and inverse FFT (iFFT), respectively. Here and throughout, arithmetic complexity is measured in operations over $\mathbb{F}_{q^m}$, following the convention used in classical FFT literature and in prior work on linearized polynomials.

Our first main result is summarized as follows.

Theorem 1.2 (FFT and iFFT). Let n, m be positive integers such that $n \mid m$, and assume that n is B -smooth, i.e., all prime factors of n are at most B . Let $\mathcal{A} \subset \mathbb{F}_{q^m}$ be a set of n $\mathbb{F}_q$ -linearly independent points constructed as in [Section 3](#). Then there exist FFT and iFFT algorithms for q -linearized polynomials over the evaluation domain $\mathcal{A}$, each requiring $O(Bn \log n)$ operations over $\mathbb{F}_{q^m}$.

Remark 1.3. Our FFT algorithm for q -linearized polynomials relies on the condition that the field extension degree $m = [\mathbb{F}_{q^m} : \mathbb{F}_q]$ contains a B -smooth divisor n . This bears similarity to classical FFT algorithms for ordinary univariate polynomials over a finite field $\mathbb{F}_q$, where one typically assumes that $q - 1$ (in multiplicative FFTs [\[13, 24\]](#)), q (in additive FFTs [\[23\]](#)), or $q + 1$ (in the group based FFTs [\[20\]](#)) contains a smooth divisor n . Whether the condition on n , or the selection of the evaluation set $\mathcal{A}$ can be relaxed while preserving the same asymptotic complexity remains an open problem, as in the classical FFT setting.

A direct corollary of our FFT algorithm is fast encoding for Gabidulin codes. More specifically, let $1 \leq k \leq n \leq m$ be integers and $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\} \subseteq \mathbb{F}_{q^m}$ be an $\mathbb{F}_q$ -linearly independent set. An $\mathbb{F}_{q^m}$ -linear Gabidulin code defined over $\mathcal{A}$ of length n , dimension k , and rate $\rho = k/n$ consists of the vectors

$$\mathcal{G}(\mathcal{A}, k) := \left\{ \text{ev}_{\mathcal{A}}(f) := (f(\alpha_1), \dots, f(\alpha_n)) \mid f(X) = \sum_{i=0}^{k-1} a_i X^{q^i} \in \mathbb{F}_{q^m}[X] \right\}.$$

Corollary 1.4. Assume $n \mid m$ and n is B -smooth. Let $\mathcal{A}$ be the evaluation set from [Theorem 1.2](#) and $k \leq n$. There exists an encoding algorithm for the Gabidulin code $\mathcal{G}(\mathcal{A}, k)$ using $O(Bn \log n)$ operations over $\mathbb{F}_{q^m}$.

Remark 1.5 (Comparison with existing algorithms). Fast evaluation and interpolation of linearized polynomials have been studied extensively in connection

with Gabidulin codes [28,25,26]. Over a general $\mathbb{F}_q$ -linearly independent set $\mathcal{A}$ of size n , Puchinger and Wachter-Zeh [26] give evaluation and interpolation algorithms using $\tilde{O}(n^{1.635})$ operations over $\mathbb{F}_{q^m}$. For the special case where $n \mid m$ and $\mathcal{A}$ is a normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, their q -transform and its inverse perform multipoint evaluation and interpolation in $O(n \log^2 n \log \log n)$ operations over $\mathbb{F}_{q^m}$.

Our contribution is an $O(n \log n)$ algorithm for $O(1)$ -smooth n , together with a recursive decomposition of q -linearized polynomials. In the radix-two case, the decomposition reduces a transform of length n to two transforms of length $n/2$, with the q -degree bound halved. The q -transform achieve quasilinear complexity through Toeplitz matrix multiplication, reducible to ordinary polynomial multiplication, but it does not provide the recursive decomposition required by the Gabidulin IOPP construction.

The recursive structure developed for our FFT algorithm naturally induces a folding operation for q -linearized polynomials. We use this operation to construct a fast Gabidulin interactive oracle proof of proximity, which we call FGI.

Informally, a public coin IOPP for a Gabidulin code $\mathcal{C} = \mathcal{G}(\mathcal{A}, k) \subseteq \mathbb{F}_{q^m}^n$ is an interactive protocol between a prover $\mathcal{P}$ and a randomized verifier $\mathcal{V}$. The verifier has oracle access to a fixed input $\mathbf{w} = \text{ev}_{\mathcal{A}}(f)$ and interacts with the prover to distinguish, by querying $\mathbf{w}$ at few locations, whether $\mathbf{w} \in \mathcal{C}$ or is far from every codeword in $\mathcal{C}$ under the rank distance. More precisely, an IOPP with perfect completeness, proximity threshold δ , and soundness error ε accepts every codeword $\mathbf{w}$ with probability 1 when interacting with the honest prover, whereas any input whose relative rank distance from $\mathcal{C}$ exceeds δ is accepted with probability at most ε , regardless of the prover's strategy. Please refer to Definition ?? for formal definition and its efficiency measures.

Theorem 1.6 (Fast Gabidulin IOPP). *Assume $n = 2^r \mid m$, and let $\mathcal{G}(\mathcal{A}, k)$ be an $[n, k]$ Gabidulin code over $\mathbb{F}_{q^m}$, where $\mathcal{A}$ is supported by our FFT construction. For any $0 < \delta \leq (1 - k/n)/2$, there exists an IOPP **FGI** for $\mathcal{G}(\mathcal{A}, k)$ with perfect completeness, proximity threshold δ , and soundness error at most $(1 - \delta)^\ell + \frac{10q^{n-1}}{q^m}$, where ℓ is the number of query repetitions. The protocol has $O(\log n)$ rounds, prover time and proof length $O(n)$, and verifier time and query complexity $O(\ell \log n)$.*

Finally, we construct a polynomial commitment scheme for q -linearized polynomials based on our Fast Gabidulin IOPP. Informally, the scheme allows a prover to commit to a polynomial $f(x) \in \mathcal{L}(q, m, k)$ and later provide a succinct proof that a claimed value v equals $f(\alpha)$ at an evaluation point $\alpha \in \mathbb{F}_{q^m}$. Please refer to Definition 2.7 for the formal definition of q -linearized PCS.

Theorem 1.7 (FGI based PCS). *Let κ be a security parameter. Assume $n = 2^r \mid m$, and let $k = pn$ be an integer for a constant rate $p \in (0, 1)$. There exists a transparent and extractable PCS over field $\mathbb{F}_{q^m}$ for q -linearized polynomials of degree less than k with perfect completeness and the following costs:*

- *Proof size:* $O(\kappa \log n)$ field elements and $O(\kappa \log^2 n)$ hash values.
- *Verifier time:* $O(\kappa \log n)$ field operations and $O(\kappa \log^2 n)$ hash computations.
- *Prover time* $O(n \log n)$ field operations and $O(n)$ hash computations.

1.2 Technical Overview

The main idea of q -linearized FFT. We begin by recalling the Cooley-Tukey’s FFT [13] for univariate polynomials and identifying the key obstacles that arise in extending these techniques to q -linearized polynomials. We then outline our approach to overcoming these challenges. For simplicity, assume that $n = 2^r$.

Let $\mathbb{F}$ be a finite field such that $n \mid |\mathbb{F}^*|$, and let $\mu_n \subseteq \mathbb{F}^*$ denote the set of n -th roots of unity. Every polynomial $f(x) \in \mathbb{F}[x]$ of degree less than n can be decomposed into its even and odd parts

$$f(x) = f_0(x^2) + x \cdot f_1(x^2), \quad (1)$$

where f_0, f_1 as polynomials of x^2 have degree less than $n/2$. Since the squaring map sends μ_n onto $\mu_{n/2}$, evaluating f on μ_n reduces to evaluating f_0 and f_1 on $\mu_{n/2}$, after which all evaluations of f can be recovered in $O(n)$ field operations using the relations $f(\pm x) = f_0(x^2) \pm x f_1(x^2)$. Hence the number of operations $C(n)$ in $\mathbb{F}$ required to compute the evaluations of $f(x)$ on μ_n satisfies

$$C(n) = 2C(n/2) + O(n), \quad C(1) = 1$$

which gives $C(n) = O(n \log n)$.

First, a direct application of the classical FFT is inefficient for q -linearized polynomials. Since a polynomial $f(x) = \sum_{i=0}^{n-1} a_i x^{q^i}$ of q -degree less than n has ordinary degree as large as $q^n - 1$. Treating f as an ordinary polynomial would therefore require a transform of length $\Omega(q^n - 1)$, resulting in complexity exponential in n . Second, q -linearized polynomials form a ring under addition and symbolic multiplication “ $\circ$ ” (i.e., composition, see [Section 2.1](#)), rather than ordinary polynomial multiplication. For a polynomial $f(x) = \sum_{i=0}^{n-1} a_i x^{q^i}$, a similar decomposition according to its even and odd q -degree is given as follows

$$f(x) = f_0(x) + f_1(x) \circ x^q,$$

where $f_0(x) = \sum_{i=0}^{n/2-1} a_{2i} x^{q^{2i}}$, $f_1(x) = \sum_{i=0}^{n/2-1} a_{2i+1} x^{q^{2i}}$ are q^2 -linearized polynomials of q^2 -degree less than $n/2$. However, evaluating this decomposition at a point $\alpha \in \mathbb{F}_{q^m}$ gives

$$f(\alpha) = f_0(\alpha) + f_1(\alpha^q).$$

Thus, if f is evaluated on a set $\mathcal{A} \subseteq \mathbb{F}_{q^m}$ of size $n = 2^r$, then f_0 is evaluated on $\mathcal{A}$, while f_1 is evaluated on its Frobenius image $\mathcal{A}^q = \{\alpha^q : \alpha \in \mathcal{A}\}$. Since the Frobenius map is bijective, both $\mathcal{A}$ and $\mathcal{A}^q$ have size n . Consequently, the DFTs of f_0 and f_1 still have length n and this naive decomposition reduces the polynomial degrees but not the sizes of the evaluation domain.

Our construction overcomes this obstruction through two ingredients. First, to keep the same evaluation domain of f_0 and f_1 , we adopt another decomposition via the left composition of Frobenius map and write

$$f(x) = \sum_{j=0}^{\frac{n}{2}-1} a_{2j} x^{q^{2j}} + x^q \circ \left(\sum_{j=0}^{\frac{n}{2}-1} a_{2j+1}^{q^{-1}} x^{q^{2j}} \right) := f_0(x) + f_1(x)^q. \quad (2)$$

Here $a^{q^{-1}} = a^{q^{m-1}}$ denotes the inverse Frobenius image of a . Crucially, the Frobenius now acts on the output $f_1(\alpha)$, rather than on the evaluation point α , so both f_0 and f_1 are evaluated on the same domain.

Second, we construct an evaluation set whose recursive structure is compatible with this decomposition. Consider the quadratic field extension tower

$$\mathbb{F}_q \subset \mathbb{F}_{q^2} \subset \cdots \subset \mathbb{F}_{q^{2^r}} := \mathbb{F}_{q^n} \subset \mathbb{F}_{q^m}$$

For $i = 1, \dots, r$, let $\lambda_i \in \mathbb{F}_{q^{2^i}} \setminus \mathbb{F}_{q^{2^{i-1}}}$. Define the product basis

$$\mathcal{A}_r = \{\lambda_1^{e_1} \lambda_2^{e_2} \cdots \lambda_r^{e_r} \mid e_i \in \{0, 1\}\}.$$

Since $\{1, \lambda_i\}$ is a basis of $\mathbb{F}_{q^{2^i}}$ over $\mathbb{F}_{q^{2^{i-1}}}$, $\mathcal{A}_r$ is an $\mathbb{F}_q$ -basis of $\mathbb{F}_{q^n}$ and $|\mathcal{A}_r| = n$. This set satisfies the recursive decomposition

$$\mathcal{A}_r^{(0)} := \mathcal{A}_r, \quad \mathcal{A}_r^{(i-1)} = \mathcal{A}_r^{(i)} \sqcup \lambda_i \mathcal{A}_r^{(i)}, \quad \text{for } i \geq 1.$$

We then have that $\mathcal{A}_r^{(i)}$ is $\mathbb{F}_{q^{2^i}}$ -linearly independent and $|\mathcal{A}_r^{(i)}| = n/2^i$. At recursion level i , the points in evaluation domain $\mathcal{A}_r^{(i-1)}$ can be paired as $\{\alpha, \lambda_i \alpha\}$ with $\alpha \in \mathcal{A}_r^{(i)}$. This structure, together with $\lambda_i^{q^{2^i}} = \lambda_i$ allows a DFT of length $n/2^{i-1}$ to be reduced recursively to two DFTs of length $n/2^i$.

More precisely, at the first recursion level, for every $\alpha \in \mathcal{A}_r^{(1)}$, the evaluations of f on the pair $\{\alpha, \lambda_1 \alpha\} \subset \mathcal{A}_r^{(0)}$ satisfy

$$(f(\alpha), f(\lambda_1 \alpha)) = (f_0(\alpha), f_1(\alpha)^q) \cdot M_{\lambda_1}, \quad M_{\lambda_1} := \begin{pmatrix} 1 & \lambda_1 \\ 1 & \lambda_1^q \end{pmatrix}. \quad (3)$$

Since M_{λ_1} is invertible, the DFT of f over $\mathcal{A}_r^{(0)}$ is reduced to two DFTs of q^2 -linearized polynomials f_0, f_1 over $\mathcal{A}_r^{(1)}$. Once these two DFTs are computed, each pair of evaluations of f is recovered by one application of the fixed matrix M_{λ_1} . This step will cost $O(n)$ operations in $\mathbb{F}_{q^m}$. Thus the number of operations $C(n)$ in $\mathbb{F}_{q^m}$ required to compute the evaluations of $f(x)$ at $\mathcal{A}_r$ satisfies

$$C(n) = 2C(n/2) + O(n).$$

The recursion continues until the final level, where we will evaluate n q^{2^r} -linearized polynomials of degree less than one at the singleton set $\mathcal{A}_r^{(r)} = \{1\}$. This base case requires only constant work. Since the recursion depth is $\log n$, we obtain $C(n) = O(n \log n)$.

FRI-style IOPP for Gabidulin codes and rank metric PCS. We first recall the FRI protocol for testing proximity to Reed-Solomon codes and its application to ordinary PCS, and then identify the challenges in extending this framework to the rank metric setting.

FRI and FRI based PCS. Given oracle access to a function $f : \mu_n \rightarrow \mathbb{F}$, the FRI protocol enables a verifier to test its proximity to the Reed-Solomon code consisting of evaluations of polynomials of degree less than k . On multiplicative domains μ_n , its folding operation is based on the classical even and odd decomposition [Eq.\(1\)](#). After receiving a random challenge $z \in \mathbb{F}$, the prover forms

$$\text{Fold}(f, z)(y) = f_0(y) + z \cdot f_1(y).$$

The evaluations of f on μ_n naturally induces the evaluations of $\text{Fold}(f, z)$ on $\mu_{n/2}$. For each $y = x^2 \in \mu_{n/2}$, the pair $f(x), f(-x)$ recovers $f_0(y)$ and $f_1(y)$, and hence determines $\text{Fold}(f, z)(y)$. Each folding round halves the domain size and reduces the claimed degree bound from k to $\lfloor k/2 \rfloor$. Iterating this procedure eventually yields a constant size instance, which the verifier checks directly. The verifier also checks consistency between successive folding layers by querying a small number of paths through the layers.

To commit to a polynomial $f \in \mathbb{F}[x]$ of degree less than k , the prover evaluates f on μ_n and commits to the resulting RS codeword using a Merkle tree. For a claimed evaluation $f(\alpha) = v$ with $\alpha \notin \mu_n$, define the quotient oracle $g : \mu_n \rightarrow \mathbb{F}$ by $g(x) := \frac{f(x)-v}{x-\alpha}$. If the claim is correct, then $x - \alpha$ divides $f(x) - v$, so g is the evaluation of a polynomial of degree less than $k - 1$. For each queried point $\omega \in \mu_n$, the verifier queries $f(\omega)$ against the original Merkle root and computes $g(\omega)$ locally. FRI can therefore test the derived quotient oracle for proximity to the RS code with degree bound $k - 1$.

The analogous folding for Gabidulin codes. Our q -linearized FFT provides the algebraic folding operation needed to extend the FRI paradigm to Gabidulin codes. According [Eq.\(2\)](#), for a random challenge $z \in \mathbb{F}_{q^m}$, define

$$\text{Fold}(f, z)(x) := f_0(x) + z \cdot f_1(x).$$

Then $\text{Fold}(f, z)$ is q^2 -linearized and satisfies $\deg_{q^2}(\text{Fold}(f, z)) \leq \left\lfloor \frac{\deg_q(f)}{2} \right\rfloor$. The product evaluation set $\mathcal{A}_r$ has a natural 2-to-1 structure

$$\mathcal{A}_r^{(0)} := \mathcal{A}_r = \bigsqcup_{\alpha \in \mathcal{A}_r^{(1)}} \{\alpha, \lambda_1 \alpha\}.$$

Thus, as in the FRI folding step, the pair of evaluations $f(\alpha)$ and $f(\lambda_1 \alpha)$ locally determines $f_0(\alpha)$ and $f_1(\alpha)^q$ by [Eq.\(3\)](#). Applying the inverse Frobenius map recovers $f_1(\alpha)$, so each value of the folded oracle over $\mathcal{A}_r^{(1)}$ can be computed from two values of the previous oracle over $\mathcal{A}_r^{(0)}$. This locality allows the verifier to check consistency between successive folding layers using only a few queries.

The same procedure is applied recursively. At level i , a $q^{2^{i-1}}$ -linearized polynomial is folded into a q^{2^i} -linearized polynomial, while the evaluation domain

shrinks from $\mathcal{A}_r^{(i-1)}$ to $\mathcal{A}_r^{(i)}$. Hence, as in FRI, each round simultaneously halves the degree bound and the oracle length.

Challenges for rank metric IOPP soundness. The key difference in the soundness analysis arises from the Frobenius twist in the q -linearized decomposition. While folding forms the random linear combination $f_0 + zf_1$, unfolding reconstructs $f = f_0 + f_1^q$.

For Reed-Solomon codes, FRI's soundness relies on correlated agreement. If a random fold $f_0 + zf_1$ is close to the target code with sufficiently high probability, then there exist codewords c_0, c_1 that agree with f_0, f_1 , respectively, on a common large set of coordinates. Through the even and odd decomposition, this common agreement set yields a codeword close to the original word of $f_0 + xf_1$.

For Gabidulin codes, proximity is measured in the rank metric. The proximity gap result of [21] yields codewords c_0, c_1 such that $f_0 - c_0$ and $f_1 - c_1$ vanish on a common large $\mathbb{F}_q$ -subspace W . Here, we identify codewords with their defining q -linearized polynomials. To obtain proximity of the original word, this correlated agreement must be transferred through the Frobenius twisted unfolding. Specifically, the original word and the candidate unfolded codeword are $f = f_0 + f_1^q$ and $c = c_0 + c_1^q$, where the error associated with the second component is transformed by the Frobenius map. One must additionally control the interaction between W and its Frobenius image W^q . We establish the required Frobenius compatibility, which transfers the correlated proximity guarantee to the unfolded word and completes the soundness analysis.

Challenges for an FRI-style PCS. A second difficulty arises when compiling the Gabidulin IOPP into a PCS for q -linearized polynomials. For an opening point $\alpha \neq 0$, the natural linearized analogue of the ordinary quotient identity is

$$f(x) - \frac{f(\alpha)}{\alpha}x = g(x) \circ (x^q - \alpha^{q-1}x), \quad (4)$$

where g has q -degree one less than that of f . The polynomial $L_\alpha(x) = x^q - \alpha^{q-1}x$ is the subspace polynomial whose kernel is one dimensional $\mathbb{F}_q$ -space $\mathbb{F}_q\alpha$.

Unlike division by $x - \alpha$, the symbolic quotient in Eq.(4) changes the evaluation domain. Indeed, the identity provides evaluations of g on $L_\alpha(\mathcal{A}_r) = \{L_\alpha(\beta) : \beta \in \mathcal{A}_r\}$, which depends on the verifier's opening point α and generally does not have the recursive structure of $\mathcal{A}_r$. Therefore, oracle access to f on $\mathcal{A}_r$ does not induce oracle access to g on a fixed domain suitable for the fast Gabidulin IOPP. This prevents a direct adaptation of the FRI based commitment scheme.

Simultaneous reduction of low degree test and evaluation proof. To avoid this challenge, we do not follow the way via symbolic quotient. Instead, we fold the committed evaluation oracle and the claimed evaluation in simultaneously. At each round, the recursive folding reduces the q -degree bound while transforming the claim $f(\alpha) = v$ into a corresponding evaluation claim for the folded polynomial. The resulting procedure therefore verifies both proximity to the appropriate Gabidulin code and consistency with the claimed evaluation.

More specific, at the i -th recursion level, the prover provides oracle access to the folded polynomial $f^{(i)}$, together with two auxiliary evaluations $f_0^{(i)}(\alpha)$ and $f_1^{(i)}(\alpha)$ satisfying the folding relation

$$f^{(i)}(\alpha) = f_0^{(i)}(\alpha) + \left(f_1^{(i)}(\alpha)\right)^{q^{2^i}}.$$

These two values also determine the next folded polynomial's evaluation at α :

$$f^{(i+1)}(\alpha) = f_0^{(i)}(\alpha) + z_{i+1} \cdot f_1^{(i)}(\alpha),$$

where $z_{i+1} \in \mathbb{F}_{q^m}$ is a random challenge sampled by the verifier. Thus, the evaluation claim is recursively transformed together with the degree reduction.

We next analyze the soundness of this evaluation propagation. Suppose that the claimed evaluation value of $f^{(i)}$ at α is incorrect, and denote it by $v^{(i)}$. Nevertheless, the prover may attempt to provide two values $v_0^{(i)}$ and $v_1^{(i)}$ satisfying the claimed folding relation $v^{(i)} = v_0^{(i)} + \left(v_1^{(i)}\right)^{q^{2^i}}$. The verifier then implicitly binds the next layer evaluation to $v_0^{(i)} + z_{i+1}v_1^{(i)}$. Because z_{i+1} is sampled uniformly at random from $\mathbb{F}_{q^m}$, the probability that this evaluation agrees with the true next layer evaluation $f_0^{(i)}(\alpha) + z_{i+1}f_1^{(i)}(\alpha)$ is at most $\frac{1}{q^m}$. In other words, it remains false except with probability $\frac{1}{q^m}$.

Applying this argument independently to all r' recursive layers and taking a union bound, the probability that an incorrect initial evaluation claim passes all recursive consistency checks is bounded by $\frac{r'}{q^m}$. Therefore, the recursive folding procedure simultaneously reduces the degree testing problem and preserves evaluation binding, yielding a sound evaluation argument for q -linearized polynomial commitment schemes.

2 Preliminaries

In this section, we briefly introduce some notations and basic results about linearized polynomials, Gabidulin codes and its proximity gaps, interactive oracle proofs and polynomial commitment schemes.

Throughout this paper, q is always a power of a prime and $\mathbb{F}_q$ denotes the finite field of q elements. Let $\mathbb{F}_{q^m}$ be an extension field of $\mathbb{F}_q$ of degree m . For an integer $n \geq 1$, we denote by $[n]$ the set $\{1, 2, \dots, n\}$.

2.1 Linearized Polynomials

A q -linearized polynomial over $\mathbb{F}_{q^m}$ is a polynomial of the form $f(x) = \sum_{i=0}^d a_i x^{q^i}$, $a_i \in \mathbb{F}_{q^m}$. Note that f induces an $\mathbb{F}_q$ -linear map on $\mathbb{F}_{q^m}$. For all $a, b \in \mathbb{F}_q$ and $\alpha, \beta \in \mathbb{F}_{q^m}$, $f(a\alpha + b\beta) = af(\alpha) + bf(\beta)$. If $f \neq 0$, its q -degree is defined by $\deg_q(f) := \max\{i \in \{0, \dots, d\} : a_i \neq 0\}$. For the zero polynomial, we set $\deg_q(0) := -\infty$.

For an integer $k \geq 1$, let $\mathcal{L}(q, m, k)$ denote the set of all q -linearized polynomials over $\mathbb{F}_{q^m}$ of q -degree less than k , including the zero polynomial, i.e.,

$$\mathcal{L}(q, m, k) := \left\{ \sum_{i=0}^{k-1} a_i x^{q^i} \mid a_i \in \mathbb{F}_{q^m} \right\}.$$

It is an $\mathbb{F}_{q^m}$ -vector space of dimension k .

Let $\mathcal{L}(q, m) := \bigcup_{k \geq 1} \mathcal{L}(q, m, k)$ be the set of all q -linearized polynomials. For every $f = \sum_{i=0}^{\ell_1} a_i x^{q^i}$ and $g(x) = \sum_{j=0}^{\ell_2} b_j x^{q^j}$ in $\mathcal{L}(q, m)$, define their symbolic multiplication by

$$(f \circ g)(x) := f(g(x)) = \sum_{k=0}^{\ell_1 + \ell_2} \left(\sum_{i+j=k} a_i b_j^{q^i} \right) x^{q^k}.$$

Then $(\mathcal{L}(q, m), +, \circ)$ forms a ring under polynomial addition and symbolic multiplication with multiplicative identity is x [22].

The following lemma establishes unique interpolation over any ordered evaluation set $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\}$ of n $\mathbb{F}_q$ -linearly independent elements in $\mathbb{F}_{q^m}$. This implies that the DFT over $\mathcal{A}$ is a bijection between $\mathcal{L}(q, m, n)$ and $\mathbb{F}_{q^m}^n$.

Lemma 2.1. ([21, Lemma 2.2]) *Assume $\mathcal{A} = \{\alpha_1, \dots, \alpha_n\} \subset \mathbb{F}_{q^m}$ is $\mathbb{F}_q$ -linearly independent. Then, for any $y_1, \dots, y_n \in \mathbb{F}_{q^m}$, there exists a unique polynomial $f \in \mathcal{L}(q, m, n)$ such that $f(\alpha_j) = y_j$, $j = 1, \dots, n$.*

Proof. Under the q -linearized standard monomial basis $\{x, x^q, \dots, x^{q^{n-1}}\}$, the interpolation conditions are equivalent to the linear system

$$\begin{pmatrix} \alpha_1 & \alpha_1^q & \dots & \alpha_1^{q^{n-1}} \\ \alpha_2 & \alpha_2^q & \dots & \alpha_2^{q^{n-1}} \\ \vdots & \vdots & & \vdots \\ \alpha_n & \alpha_n^q & \dots & \alpha_n^{q^{n-1}} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

where $(a_0, \dots, a_{n-1})$ is the coefficients of the desired polynomial $f(x)$. The coefficient matrix is a Moore matrix, which is nonsingular whenever $\alpha_1, \dots, \alpha_n$ are $\mathbb{F}_q$ -linearly independent. Hence the system has a unique solution, giving the a unique interpolating polynomial $f(x)$.

2.2 Rank Metric Codes and Gabidulin Codes

Fix an $\mathbb{F}_q$ -basis $\beta = (\beta_1, \dots, \beta_m)$ of $\mathbb{F}_{q^m}$. For any $\mathbf{c} = (c_1, \dots, c_n) \in \mathbb{F}_{q^m}^n$, let $M_\beta(\mathbf{c}) \in \mathbb{F}_q^{m \times n}$ be the matrix whose j -th column is the coordinate vector of c_j with respect to β . The rank weight of $\mathbf{c}$ is defined as

$$\text{wt}_R(\mathbf{c}) := \text{rank}_{\mathbb{F}_q}(M_\beta(\mathbf{c})) = \dim_{\mathbb{F}_q}(\text{span}_{\mathbb{F}_q}\{c_1, \dots, c_n\}).$$

This quantity is independent of the choice of β . The rank distance between two vectors $\mathbf{c}, \mathbf{c}' \in \mathbb{F}_{q^m}^n$ is defined as

$$d_{\text{rk}}(\mathbf{c}, \mathbf{c}') := \text{wt}_R(\mathbf{c} - \mathbf{c}').$$

An $[n, k]$ linear rank metric code $\mathcal{C}$ over $\mathbb{F}_{q^m}$ is a k dimensional $\mathbb{F}_{q^m}$ -linear subspace of $\mathbb{F}_{q^m}^n$. Its minimum rank distance is defined as

$$d(\mathcal{C}) := \min\{d_{\text{rk}}(\mathbf{c}, \mathbf{c}') \mid \mathbf{c}, \mathbf{c}' \in \mathcal{C}, \mathbf{c} \neq \mathbf{c}'\}.$$

For every $\mathbf{v} \in \mathbb{F}_{q^m}^n$, define its distance and relative distance to $\mathcal{C}$ by

$$d_{\text{rk}}(\mathbf{v}, \mathcal{C}) := \min_{\mathbf{c} \in \mathcal{C}} d_{\text{rk}}(\mathbf{v}, \mathbf{c}), \quad \Delta_{\text{rk}}(\mathbf{v}, \mathcal{C}) := \frac{d_{\text{rk}}(\mathbf{v}, \mathcal{C})}{n}.$$

We say that $\mathbf{v}$ is δ -close to $\mathcal{C}$ if $\Delta_{\text{rk}}(\mathbf{v}, \mathcal{C}) \leq \delta$, and δ -far from $\mathcal{C}$ otherwise.

Gabidulin codes [14] are often viewed as the rank metric analogues of Reed-Solomon codes.

Definition 2.2 (Gabidulin code). Let $\mathcal{A} = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_{q^m}^n$ be an ordered tuple of $\mathbb{F}_q$ -linearly independent elements, and let $1 \leq k \leq n \leq m$. The Gabidulin code of dimension k evaluated at $\mathcal{A}$ is

$$\mathcal{G}(\mathcal{A}, k) := \{(f(\alpha_1), \dots, f(\alpha_n)) \mid f \in \mathcal{L}(q, m, k)\}.$$

The code $\mathcal{G}(\mathcal{A}, k)$ is an $\mathbb{F}_{q^m}$ -linear $[n, k]$ code with minimum rank distance

$$d_{\text{rk}}(\mathcal{G}(\mathcal{A}, k)) = n - k + 1.$$

2.3 Proximity Gaps for Rank Metric Codes

Proximity gaps are a key ingredient in the soundness analysis of IOPPs and PCS based on codes. Informally, a proximity gap states that if more than an ε fraction of the vectors on an affine line are within distance δ of a code, then the two vectors defining the line must be jointly close to two codewords. In the Hamming metric, this correlated closeness requires the corresponding error vectors to be supported on a common set of at most δn coordinates. Recent work [21] extends this notion to rank metric codes by replacing common Hamming support with a common error space over $\mathbb{F}_q$.

For $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{F}_{q^m}^n$, define their interleaved rank by

$$\text{rank}_{\mathbb{F}_q} \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \end{pmatrix} := \text{rank}_{\mathbb{F}_q} \begin{pmatrix} M_{\beta}(\mathbf{u}_0) \\ M_{\beta}(\mathbf{u}_1) \end{pmatrix},$$

where $M_{\beta}(\mathbf{u}_i) \in \mathbb{F}_q^{m \times n}$ is the matrix expansion with respect to a fixed $\mathbb{F}_q$ -basis $\beta = \{\beta_1, \dots, \beta_m\}$ of $\mathbb{F}_{q^m}$. This quantity is independent of the choice of basis.

Definition 2.3 (Rank metric proximity gap for lines). Let $\mathcal{C} \subseteq \mathbb{F}_{q^m}^n$ be an $\mathbb{F}_{q^m}$ -linear rank metric code. We say that $\mathcal{C}$ has a (δ, ε) proximity gap for lines if, for every $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{F}_{q^m}^n$,

$$\Pr_{\lambda \in \mathbb{F}_{q^m}} [\Delta_{\text{rk}}(\mathbf{u}_0 + \lambda \mathbf{u}_1, \mathcal{C}) \leq \delta] > \varepsilon$$

implies that there exist $\mathbf{c}_0, \mathbf{c}_1 \in \mathcal{C}$ such that

$$\text{rank}_{\mathbb{F}_q} \begin{pmatrix} \mathbf{u}_0 - \mathbf{c}_0 \\ \mathbf{u}_1 - \mathbf{c}_1 \end{pmatrix} \leq \delta n.$$

We call δ the proximity gap parameter and ε the soundness error.

Since $\mathcal{C}$ is linear, $\mathbf{c}_0 + \lambda \mathbf{c}_1 \in \mathcal{C}$ for every $\lambda \in \mathbb{F}_{q^m}$. Moreover,

$$\text{wt}_R((\mathbf{u}_0 - \mathbf{c}_0) + \lambda(\mathbf{u}_1 - \mathbf{c}_1)) \leq \text{rank}_{\mathbb{F}_q} \begin{pmatrix} \mathbf{u}_0 - \mathbf{c}_0 \\ \mathbf{u}_1 - \mathbf{c}_1 \end{pmatrix}.$$

Consequently, the conclusion of the definition implies that every point on the affine line

$$\ell_{\mathbf{u}_0, \mathbf{u}_1} = \{\mathbf{u}_0 + \lambda \mathbf{u}_1 : \lambda \in \mathbb{F}_{q^m}\}$$

is δ -close to $\mathcal{C}$. Thus, either all points on the line are δ -close to $\mathcal{C}$, or at most an ε fraction of its points are δ -close to the code under the relative rank metric.

Theorem 2.4 (Rank metric proximity gaps [21]). Let $\mathcal{C} \subseteq \mathbb{F}_{q^m}^n$ be an $\mathbb{F}_{q^m}$ -linear $[n, k, d]$ rank metric code. For $\delta \in (0, 1)$, set $e = \lfloor \delta n \rfloor$.

- (1) If $0 < \delta \leq \frac{d-1}{3n}$, then $\mathcal{C}$ has a (δ, ε_e) proximity gap with $\varepsilon_e = \frac{q^{e+1}}{q^m} = q^{e+1-m}$.
- (2) If $\mathcal{C}$ is a Gabidulin code and $\delta \leq \frac{d-1}{2n}$, then $\mathcal{C}$ has a (δ, ε_e) proximity gap with $\varepsilon_e = \frac{\tau_e}{q^m} \leq \frac{10q^{n-1}}{q^m}$, where $\tau_e := (\sigma_e - 1) + \sigma_e((\sigma_e - 1)\sigma_{k-1} + 1) \leq 10q^{n-1}$, $\sigma_r := \sum_{i=0}^r q^i$.

2.4 Interactive Oracle Proofs of Proximity

We briefly recall the definitions of interactive oracle proofs (IOPs) and interactive oracle proofs of proximity for error-correction codes [31].

Interactive oracle proofs (IOPs). An r -round public coin IOP $= (\mathcal{P}, \mathcal{V})$, for a relation $\mathcal{R}$ runs as follows. For a public instance x , initially, $\mathcal{P}$ sends an oracle string π_0 . In each round $i \in [1, r]$, the verifier samples and sends a random challenge z_i , and the prover replies with an oracle string π_i . After r rounds of communications, the verifier $\mathcal{V}$ queries some entries of the oracle strings $\pi_0, \pi_1, \dots, \pi_r$ and outputs a bit b .

Definition 2.5. Let $\text{IOP} = (\mathcal{P}, \mathcal{V})$ be a r -round public coin IOP protocol for a relation $\mathcal{R}$. We say that IOP has perfect completeness if for every $(x, w) \in \mathcal{R}$,

$$\Pr_{z_1, \dots, z_r} \left[\mathcal{V}^{\pi_0, \pi_1, \dots, \pi_r}(x, z_1, \dots, z_r) = 1 \mid \begin{array}{l} \pi_0 \leftarrow \mathcal{P}(x, w) \\ \pi_1 \leftarrow \mathcal{P}(x, w, z_1) \\ \dots \\ \pi_r \leftarrow \mathcal{P}(x, w, z_1, \dots, z_r) \end{array} \right] = 1$$

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

$$\Pr_{z_1, \dots, z_r} \left[\mathcal{V}^{\pi_0, \pi_1, \dots, \pi_r}(x, z_1, \dots, z_r) = 1 \mid \begin{array}{l} \pi_0 \leftarrow \mathcal{A}(x) \\ \pi_1 \leftarrow \mathcal{A}(x, z_1) \\ \dots \\ \pi_r \leftarrow \mathcal{A}(x, z_1, \dots, z_r) \end{array} \right] < \varepsilon$$

IOPs of proximity. An 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 $q_w \ll |w|$ entries of w . For a proximity threshold $\delta \in (0, 1)$, the soundness states that if w is δ -far from any valid witness, then the verifier rejects with high probability $1 - \varepsilon(\delta)$.

Definition 2.6. A public coin IOPP $= (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ is an IOP of proximity if it satisfies IOP perfect completeness, and the $\varepsilon(\cdot)$ -IOPP soundness holds for some function $\varepsilon(\cdot)$. For every (x, w) where w is δ -far (in relative Rank distance) from any w' such that $(x, w') \in \mathcal{R}$, it holds that for any adversary $\mathcal{A}$,

$$\Pr_{z_1, \dots, z_r} \left[\mathcal{V}^{w, \pi_0, \dots, \pi_r}(x, z_1, \dots, z_r) = 1 \mid \begin{array}{l} \pi_0 \leftarrow \mathcal{A}(x, w) \\ \pi_1 \leftarrow \mathcal{A}(x, w, z_1) \\ \dots \\ \pi_r \leftarrow \mathcal{A}(x, w, z_1, \dots, z_r) \end{array} \right] < \varepsilon(\delta)$$

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

Efficiency Measures . All of these complexity measures are implicitly functions of the input size of instance x .

- The round complexity $r(n)$ is the number of interaction rounds.
- Proof length $l(n)$ is the sum of the lengths of all prover's messages, excluding the input function w .
- The query complexity $q(n)$ is the total number of entries read by the verifier from w and all prover oracle messages,
- The prover complexity $t_p(n)$ is the time required to generate these messages. It excludes the cost of constructing w .
- The verifier complexity $t_v(n)$ is the time spent by the verifier to make her decision when queries and query-answers are given as inputs.

2.5 Polynomial Commitment Schemes

We adapt the definition from [18,31] to define a rank-metric polynomial commitment scheme for q -linearized polynomials.

Definition 2.7. Let $\mathcal{L}(q, m, k)$ be the space of q -linearized polynomials of q -degree less than k . A tuple of algorithms $(\text{Gen}, \text{Commit}, \text{Open}, \text{EvalOpen}, \text{EvalVrfy})$ is an extractable polynomial commitment scheme for $\mathcal{L}(q, m, k)$ if the following properties hold.

- $\text{Gen}(1^\kappa, q, m, k) \rightarrow \text{pp}$ takes security parameter κ , prime power q , positive integers $k \leq m$ and outputs public parameters pp .
- $\text{Commit}(\text{pp}, f) \rightarrow \text{com}$ takes a q -linearized polynomial $f \in \mathcal{L}(q, m, k)$ and outputs a commitment com .
- $\text{Open}(\text{pp}, \text{com}, f) \rightarrow b \in \{0, 1\}$ takes a commitment com and a q -linearized polynomial f and outputs a bit $b \in \{0, 1\}$.
- $\text{EvalOpen}(\text{pp}, \text{com}, f, \alpha) \rightarrow (\beta, \pi)$ the Prover $\mathcal{P}$ is asked to open evaluation of $f(x)$ at point α . $\mathcal{P}$ returns the opening $\beta = f(\alpha)$ alongside with some proof of correctness π .
- $\text{EvalVrfy}(\text{pp}, \text{com}, \alpha, \beta, \pi) \rightarrow b \in \{0, 1\}$: given polynomial commitment com , evaluation point α , purported value $\beta = f(\alpha)$ of committed polynomial on this point, the verifier $\mathcal{V}$ checks correctness of proof and either accept or reject it.

Completeness. An honest prover can successfully convince the verifier of any evaluation. Specifically, for every $f \in \mathcal{L}(q, m, k)$ and every $\alpha \in \mathbb{F}_{q^m}$, if $\text{com} \leftarrow \text{Commit}(\text{pp}, f)$ and $(\beta, \pi) \leftarrow \text{EvalOpen}(\text{pp}, \text{com}, f, \alpha)$, then

$$\text{EvalVrfy}(\text{pp}, \text{com}, \alpha, \beta, \pi) = 1.$$

Polynomial binding. If for every PPT adversary $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\kappa, q, m, k); \\ f_0 \neq f_1 \wedge b_0 = b_1 = 1 : \\ \quad (\text{com}, f_0, f_1) \leftarrow \mathcal{A}(\text{pp}); \\ \quad \text{Open}(\text{pp}, f_0, \text{com}) = b_0, \\ \quad \text{Open}(\text{pp}, f_1, \text{com}) = b_1; \end{array} \right] \leq \text{negl}(\kappa).$$

Evaluation binding. If for every PPT adversary $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\kappa, q, m, k); \\ \beta \neq \beta' \wedge b_0 = b_1 = 1 : \\ \quad (\text{com}, \alpha, \beta, \pi, \beta', \pi') \leftarrow \mathcal{A}(\text{pp}); \\ \quad \text{EvalVrfy}(\text{pp}, \text{com}, \alpha, \beta, \pi) = b_0; \\ \quad \text{EvalVrfy}(\text{pp}, \text{com}, \alpha, \beta', \pi') = b_1 \end{array} \right] \leq \text{negl}(\kappa).$$

Knowledge Soundness. If for every PPT adversary $\mathcal{P}^*$, there exists an expected PPT extractor $\mathcal{E}$ such that

$$\Pr \left[\begin{array}{l} f(\alpha) \neq y \vee \\ \text{Open}(\text{pp}, f, \text{com}) \neq 1 \end{array} \mid \begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\kappa, q, m, k); \\ (\text{com}, \alpha, y, \pi) \leftarrow \mathcal{P}^*(\text{pp}); \\ 1 \leftarrow \text{EvalVrfy}(\text{com}, \alpha, y, \pi) \\ f \leftarrow \mathcal{E}^{\mathcal{P}^*}(\text{pp}, \text{com}, \alpha, y, \pi) \end{array} \right] \leq \text{negl}(\kappa).$$

3 Fast Fourier Transforms for q -Linearized Polynomials

In this section, we propose algorithms for multipoint evaluation and interpolation of q -linearized polynomials. We first construct a structured evaluation domain from a tower of quadratic field extensions. The recursive structure of this domain supports both the forward FFT and the inverse FFT.

Throughout this section, we consider transform length $n = 2^r$ over a finite field $\mathbb{F}_{q^m}$ and assume that $n \mid m$. The divisibility condition ensures that $\mathbb{F}_{q^m}$ contains the tower of subfields used in our construction. The FFT and iFFT of $O(1)$ -smooth transform length are presented in [Appendix A](#).

Construction of the evaluation domain. For $0 \leq i \leq r$, let $F_i = \mathbb{F}_{q^{2^i}}$. Since $n = 2^r \mid m$, these fields forms a tower of quadratic field extension

$$\mathbb{F}_{q^n} = F_r \supset F_{r-1} \supset \cdots \supset F_1 \supset F_0 = \mathbb{F}_q, \quad [F_i : F_{i-1}] = 2.$$

For each $1 \leq i \leq r$, choose $\lambda_i \in F_i \setminus F_{i-1}$. Then $\{1, \lambda_i\}$ is a basis of F_i over F_{i-1} . Define

$$\mathcal{A}_r := \{\alpha_e = \lambda_1^{e_1} \lambda_2^{e_2} \cdots \lambda_r^{e_r} \mid e_1, e_2, \dots, e_r \in \{0, 1\}\}. \quad (5)$$

We order these elements by the binary index $e = \sum_{j=1}^r e_j 2^{j-1}$ in $[0, n-1]$.

The evaluation domain $\mathcal{A}_r$ has a recursive structure. For $0 \leq i \leq r$, define

$$\mathcal{A}_r^{(i)} := \left\{ \alpha_e^{(i)} = \lambda_{i+1}^{e_{i+1}} \cdots \lambda_r^{e_r} \mid e_{i+1}, \dots, e_r \in \{0, 1\} \right\}, \quad (6)$$

where the empty product gives $\mathcal{A}_r^{(r)} = \{1\}$. We also order elements by the binary index $e = \sum_{j=i+1}^r e_j 2^{j-i-1}$ in $[0, n_i - 1]$. In particular, $\mathcal{A}_r^{(0)} = \mathcal{A}_r$.

Lemma 3.1. *For every $0 \leq i \leq r$, the set $\mathcal{A}_r^{(i)}$ is linearly independent over F_i . Moreover, for every $0 \leq i < r$,*

$$\mathcal{A}_r^{(i)} = \mathcal{A}_r^{(i+1)} \sqcup \lambda_{i+1} \mathcal{A}_r^{(i+1)}. \quad (7)$$

Proof. For each $i < r$, the pair $\{1, \lambda_{i+1}\}$ is a basis of F_{i+1} over F_i . Iterating these quadratic bases shows that the products in [Eq.\(6\)](#) form a basis of F_r over F_i . The decomposition [Eq.\(7\)](#) follows by partitioning the elements of $\mathcal{A}_r^{(i)}$ according to whether the exponent e_{i+1} of λ_{i+1} is 0 or 1. $\square$

Therefore, for each $0 \leq i < r$, there exists a 2-to-1 map

$$\pi_{i,i+1} : \mathcal{A}_r^{(i)} \longrightarrow \mathcal{A}_r^{(i+1)}, \quad \lambda_{i+1}^{e_{i+1}} \lambda_{i+2}^{e_{i+2}} \cdots \lambda_r^{e_r} \mapsto \lambda_{i+2}^{e_{i+2}} \cdots \lambda_r^{e_r}.$$

Every $\beta \in \mathcal{A}_r^{(i+1)}$ has two preimages β and $\lambda_{i+1}\beta$ in $\mathcal{A}_r^{(i)}$.

The significance of [Lemma 3.1](#) is that the evaluation domain is halved at every level $0 \leq i < r$ while remaining linearly independent over the corresponding base field. This is the domain reduction needed for an FFT-style recursion.

3.1 Fast Fourier Transform

For $0 \leq i \leq r$, set

$$q_i := q^{2^i} = |F_i|, \quad n_i := 2^{r-i} = [\mathbb{F}_{q^n} : F_i], \quad m_i := m/2^i = [\mathbb{F}_{q^m} : F_i].$$

Then $\mathbb{F}_{q^m} = \mathbb{F}_{q_i^{m_i}}$. In particular $q_0 = q$, $n_0 = n$, $m_0 = m$ and $q_r = q^n$, $n_r = 1$, $m_r = m/n$. For every q_i -linearized polynomial $g(x) = \sum_{j=0}^{n_i-1} b_j x^{q_i^j}$ in $\mathcal{L}(q_i, m_i, n_i)$ of q_i -degree less than n_i . Separating the even and odd coefficients gives

$$g(x) = g_0(x) + g_1(x)^{q_i}, \quad (8)$$

where

$$g_0(x) = \sum_{j=0}^{n_i/2-1} b_{2j} x^{q_i^{2j}}, \quad g_1(x) = \sum_{j=0}^{n_i/2-1} b_{2j+1}^{q_i^{-1}} x^{q_i^{2j+1}}.$$

Here $(\cdot)^{q_i^{-1}}$ denotes the inverse of the Frobenius automorphism $(\cdot)^{q_i} : a \mapsto a^{q_i}$. Then both g_0 and g_1 are q_{i+1} -linearized polynomials of q_{i+1} -degree less than $n_{i+1} = n_i/2$.

Since $\lambda_{i+1} \in F_{i+1}$, $\lambda_{i+1}^{q_i^{2j+1}} = \lambda_{i+1}$ for $j \geq 0$. For every $\beta \in \mathcal{A}_r^{(i+1)}$, since g_0, g_1 are q_{i+1} -linearized, we have $g_s(\lambda_{i+1}\beta) = \lambda_{i+1}g_s(\beta)$ for $s \in \{0, 1\}$. Therefore,

$$(g(\beta), g(\lambda_{i+1}\beta)) = (g_0(\beta), g_1(\beta)^{q_i}) \cdot M_i, \quad M_i := \begin{pmatrix} 1 & \lambda_{i+1} \\ 1 & \lambda_{i+1}^{q_i} \end{pmatrix}. \quad (9)$$

The matrix M_i is invertible because $\lambda_{i+1}^{q_i} \neq \lambda_{i+1}$. Thus, a transform of length n_i reduces to two transforms of length $n_i/2$. After the evaluations of g_0 and g_1 over $\mathcal{A}_r^{(i+1)}$ are computed, the evaluation pair $(g(\beta), g(\lambda_{i+1}\beta))$ of g can be recovered by applying the fixed M_i according to Eq.(9).

Theorem 3.2 (FFT). *Let $n = 2^r \mid m$, and let $\mathcal{A}_r^{(0)} = \mathcal{A}_r$ be the evaluation domain defined in Eq.(5). For every $f = \sum_{j=0}^{n-1} b_j x^{q^j} \in \mathcal{L}(q, m, n)$, Algorithm 1 with input $\mathbf{b} = (b_0, \dots, b_{n-1})$ and $i = 0$ computes $\text{DFT}_{\mathcal{A}_r}(f) = (f(\alpha_0), \dots, f(\alpha_{n-1}))$ using $O(n \log n)$ operations over $\mathbb{F}_{q^m}$.*

Proof. Its correctness follows directly from the polynomial decomposition Eq.(8), the recursive structure of $\mathcal{A}_r$ by Lemma 3.1, and the evaluation relation Eq.(9).

For the complexity analysis, let $C(n)$ denote the cost of a recursive transform of length n . We count each addition, multiplication, and application of a Frobenius automorphism or its inverse as one operation over $\mathbb{F}_{q^m}$. Actually, with a normal basis representation of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, Frobenius maps and their inverses can be implemented as cyclic coordinate shifts by Lemma B.1.

At the top level call FFT_0 , the transform of length n is reduced to two transforms of length $n/2$. Constructing the coefficient vector in Step 4 requires $\frac{n}{2}$ times of inverse q -Frobenius map, which is actually a q^{m-1} -th power. After the two recursive transforms $\text{FFT}_1(\mathbf{b}_0)$ and $\text{FFT}_1(\mathbf{b}_1)$ have been computed,

Algorithm 1: FFT_{*i*}(**b**) over $\mathcal{A}_r^{(i)}$

Input: $\mathbf{b} = (b_0, \dots, b_{n_i-1})$, the coefficients of $g(x) = \sum_{j=0}^{n_i-1} b_j x^{q_i^j}$
Output: $(g(\alpha_0^{(i)}), \dots, g(\alpha_{n_i-1}^{(i)}))$, where $\mathcal{A}_r^{(i)} = \{1 = \alpha_0^{(i)}, \dots, \alpha_{n_i-1}^{(i)}\}$
1 if $n_i = 1$ then
2 return (b_0)
3 $\mathbf{b}_0 \leftarrow (b_0, b_2, \dots, b_{n_i-2})$
4 $\mathbf{b}_1 \leftarrow (b_1^{q_i^{-1}}, b_3^{q_i^{-1}}, \dots, b_{n_i-1}^{q_i^{-1}})$
5 $\mathbf{z}_0 \leftarrow \text{FFT}_{i+1}(\mathbf{b}_0)$ over $\mathcal{A}_r^{(i+1)}$
6 $\mathbf{z}_1 \leftarrow \text{FFT}_{i+1}(\mathbf{b}_1)$ over $\mathcal{A}_r^{(i+1)}$
7 for $0 \leq j < n_i/2$ do
8 $y_{2j} \leftarrow z_{0,j} + z_{1,j}^{q_i}$
9 $y_{2j+1} \leftarrow \lambda_{i+1} z_{0,j} + \lambda_{i+1}^{q_i} z_{1,j}^{q_i}$
10 return ($y_0, \dots, y_{n_i-1}$)

each of the $n/2$ output pairs in Steps 8-9 requires one Frobenius application, two multiplications, and two additions. Hence the total complexity satisfies

$$C(n) = 2C(n/2) + \frac{n}{2} + 5 \cdot \frac{n}{2} = 2C(n/2) + 3n.$$

After r iterations, $\text{DFT}_{\mathcal{A}_r}(f)$ will finally reduced to n $\text{DFT}_{\mathcal{A}_r^{(r)}}$ of q_r -linearized polynomials of q_r -degree less than $n_r = 1$ on $\mathcal{A}_r^{(r)} = \{1\}$, i.e., evaluating polynomials in the form bx at $x = 1$. Thus $C(1) = 1$ and $C(n) = 3n \log n + n = O(n \log n)$. $\square$

Remark 3.3. We implemented [Algorithm 1](#) and validate its correctness. We tested it over $\mathbb{F}_{2^{128}}$ with $n = 128$. Compared with the naive point-by-point evaluation, which costs $O(n^2)$ operations, our algorithm achieves a speedup about $n/\log n$. The code is available at <https://anonymous.4open.science/r/qFFT/>.

Corollary 3.4 (Fast Gabidulin encoding). *Let $1 \leq k \leq n$, and let $\mathcal{G}(\mathcal{A}_r, k)$ be the Gabidulin code defined over $\mathcal{A}_r$. After padding the message coefficient vector $(a_0, \dots, a_{k-1})$ with $n-k$ zeros, the corresponding Gabidulin codeword can be computed using $O(n \log n)$ arithmetic operations over $\mathbb{F}_{q^m}$.*

3.2 Inverse Fast Fourier Transform

We use the notation introduced in [Section 3.1](#). The inverse transform follows by reversing the recursive steps in FFT. From [Eq.\(9\)](#), for every $\beta = \alpha_j^{(i+1)} \in \mathcal{A}_r^{(i+1)}$,

$$(g_0(\beta), g_1(\beta)^{q_i}) = (g(\beta), g(\lambda_{i+1}\beta))M_i^{-1}, \quad (10)$$

where

$$M_i^{-1} = \frac{1}{\lambda_{i+1}^{q_i} - \lambda_{i+1}} \begin{pmatrix} \lambda_{i+1}^{q_i} & -\lambda_{i+1} \\ -1 & 1 \end{pmatrix}.$$

After applying M_i^{-1} to every pair of evaluations $(g(\beta), g(\lambda_{i+1}\beta))$, and applying the inverse q_i -Frobenius map to the second component gives the evaluation vectors of g_0 and g_1 over $\mathcal{A}_r^{(i+1)}$. Two recursive inverse transforms recover g_0 and g_1 , after which $g = g_0 + g_1^{q_i}$ reconstructs the coefficient vector of g .

Algorithm 2: iFFT $_i(\mathbf{y})$ over $\mathcal{A}_r^{(i)}$

Input: $\mathbf{y} = (y_0, \dots, y_{n_i-1})$, the evaluations of an unknown q_i -linearized polynomial of q_i -degree less than n_i
Output: Its coefficient vector $(b_0, \dots, b_{n_i-1})$

```

1 if  $n_i = 1$  then
2   return  $(y_0)$ 
3 for  $0 \leq j < n_i/2$  do
4    $(u_j, v_j) \leftarrow (y_{2j}, y_{2j+1}) \cdot M_i^{-1}$ 
5    $z_{0,j} \leftarrow u_j$ 
6    $z_{1,j} \leftarrow v_j^{q_i^{-1}}$ 
7  $\mathbf{c}_0 \leftarrow \text{iFFT}_{i+1}(\mathbf{z}_0)$  over  $\mathcal{A}_r^{(i+1)}$ 
8  $\mathbf{c}_1 \leftarrow \text{iFFT}_{i+1}(\mathbf{z}_1)$  over  $\mathcal{A}_r^{(i+1)}$ 
9 for  $0 \leq j < n_i/2$  do
10   $b_{2j} \leftarrow c_{0,j}$ 
11   $b_{2j+1} \leftarrow c_{1,j}^{q_i}$ 
12 return  $(b_0, \dots, b_{n_i-1})$ 

```

Theorem 3.5 (iFFT). *Let $n = 2^r \mid m$, and let $\mathcal{A}_r$ be defined as in Eq.(5). For every $\mathbf{y} \in \mathbb{F}_{q^m}^n$, Algorithm 2 with input $\mathbf{y}$ and $i = 0$ returns the unique $f \in \mathcal{L}(q, m, n)$ satisfying $\text{DFT}_{\mathcal{A}_r}(f) = \mathbf{y}$ using $O(n \log n)$ arithmetic operations over $\mathbb{F}_{q^m}$.*

Proof. By Eq.(10), Steps 3-6 reduce the iDFT of length n_i over $\mathcal{A}_r^{(i)}$ to two iDFTs of length n_{i+1} over $\mathcal{A}_r^{(i+1)}$. (8) shows that Steps 9-12 give the coefficients of $\text{iDFT}_{\mathcal{A}_r^{(i)}}(\mathbf{y})$. Correctness therefore follows by induction.

The running time satisfies the same recurrence as that of the FFT. Precisely, Steps 3-6 require $5n_i/2$ operations and Steps 9-12 requires $n_i/2$ Frobenius maps. Thus the complexity $I(n_i)$ for an iDFT of length n_i over $\mathcal{A}_r^{(i)}$ satisfies

$$I(n_i) = 2I(n_i/2) + 3n_i, \quad \text{and } I(1) = 1,$$

which gives $I(n) = 3n \log n + n = O(n \log n)$. $\square$

4 A FRI-Style IOPP for Gabidulin Codes

In this section, we construct a FRI-style interactive oracle proof of proximity for Gabidulin codes, which we call the *Fast Gabidulin IOPP* (FGI). Its folding

operation is induced by the same recursive decomposition that underlies the q -linearized FFT of [Section 3](#). At every round, both the evaluation domain size and the relevant linearized degree bound decrease by a factor of two.

Let $n = 2^r \mid m$, and let $\mathcal{A}_r^{(i)}$ be the nested evaluation domains defined in [Eq.\(6\)](#). Fix a number of folding rounds $r' \leq r$ and an integer k such that $2^{r'} \mid k$. For $0 \leq i \leq r'$, set

$$q_i := q^{2^i}, \quad F_i = \mathbb{F}_{q_i}, \quad m_i = \frac{m}{2^i}, \quad n_i := \frac{n}{2^i}, \quad k_i := \frac{k}{2^i}.$$

Define

$$\mathcal{C}_i := \left\{ \text{ev}_{\mathcal{A}_r^{(i)}}(f) : f \in \mathcal{L}(q_i, m_i, k_i) \right\} \subseteq \mathbb{F}_{q^m}^{n_i}. \quad (11)$$

$\mathcal{C}_i$ is an $[n_i, k_i]$ Gabidulin code. The rate $\rho := k/n = k_i/n_i$ remains unchanged throughout the recursion. The rank distance of $\mathcal{C}_i$ is taken over $\mathbb{F}_{q_i}$ -rank.

In what follows, when the base field of the rank metric needs to be made explicit, we write $\text{rk}_{\mathbb{F}_Q}$ for rank over $\mathbb{F}_Q$, and use d_{rk_Q} and Δ_{rk_Q} for the corresponding rank distance and relative rank distance, respectively.

4.1 The Folding Operator

We first define folding of linearized polynomials and then express it as a local operation on evaluation oracles. Generally, for $0 \leq i \leq r'$, let $f(x) = \sum_{j=0}^{n_i-1} b_j x^{q_i^j}$ be a q_i -linearized polynomial of q_i -degree less than n_i . As in [Eq.\(8\)](#), it has the unique decomposition

$$f(x) = f_0(x) + f_1(x)^{q_i}, \quad (12)$$

where f_0 and f_1 are q_{i+1} -linearized polynomials of q_{i+1} -degree less than $n_{i+1} = n_i/2$. For a challenge $z \in \mathbb{F}_{q^m}$, define

$$\text{Fold}_{i+1}(f, z)(x) := f_0(x) + z f_1(x).$$

If f has q_i -degree less than k_i , then f_0 and f_1 have q_{i+1} -degree less than k_{i+1} , and hence

$$f \in \mathcal{L}(q_i, m_i, k_i) \implies \text{Fold}_{i+1}(f, z) \in \mathcal{L}(q_{i+1}, m_{i+1}, k_{i+1}). \quad (13)$$

By the q -linearized interpolation as in [Lemma 2.1](#), the folding operator can be extended to arbitrary words. Let $\mathbf{w} = (w_0, \dots, w_{n_i-1}) \in \mathbb{F}_{q^m}^{n_i}$, and let $f_{\mathbf{w}}(x)$ be the unique q_i -linearized interpolation polynomial of q_i -degree less than n_i such that $\text{ev}_{\mathcal{A}_r^{(i)}} = \mathbf{w}$. Define

$$\text{Fold}_{i+1}(\mathbf{w}, z) := \text{ev}_{\mathcal{A}_r^{(i+1)}}(\text{Fold}_{i+1}(f_{\mathbf{w}}, z)).$$

This can be computed locally. For every position $j \in [n_{i+1}]$, let $\beta = \alpha_j^{(i+1)} \in \mathcal{A}_r^{(i+1)}$. By [Eq.\(9\)](#), define

$$(a_j, b_j) := (w_{2j}, w_{2j+1}) \cdot M_i^{-1} = (f_0(\beta), f_1(\beta)^{q_i}), \quad M_i = \begin{pmatrix} 1 & \lambda_{i+1} \\ 1 & \lambda_{i+1}^{q_i} \end{pmatrix}.$$

Then

$$\text{Fold}_{i+1}(\mathbf{w}, z)_j = a_j + z \cdot b_j^{q_i^{-1}}, \quad (14)$$

where $(\cdot)^{q_i^{-1}}$ denotes the inverse of the Frobenius automorphism $a \mapsto a^{q_i}$. Thus, each position of the folded oracle is determined by two positions of the preceding oracle. This local relation is the analogue of the FRI protocol for RS codes.

At the Gabidulin code level, Equation Eq.(13) immediately implies that

$$\mathbf{w} \in \mathcal{C}_i \implies \text{Fold}_{i+1}(\mathbf{w}, z) \in \mathcal{C}_{i+1}, \text{ for every } z \in \mathbb{F}_{q^m}. \quad (15)$$

Based on the above, given oracle access to a word $\mathbf{w}^{(0)} : \mathcal{A}_r^{(0)} \rightarrow \mathbb{F}_{q^m}$, one can recursively reduce the claim $\mathbf{w}^{(0)} \stackrel{?}{\in} \mathcal{C}_0$ to analogous claims $\mathbf{w}^{(i)} \stackrel{?}{\in} \mathcal{C}_i$ for $1 \leq i \leq r'$ over domains of decreasing size, where each $\mathbf{w}^{(i)} = \text{Fold}_i(\mathbf{w}^{(i-1)}, z_i)$ under a random challenge z_i . At each round, a constant number of local queries checks consistency between the two consecutive oracles. After r' rounds, the final oracle has length $2^{r-r'}$. Choosing r' so that this length is $O(1)$ allows the verifier to read the entire final oracle and check its code membership directly.

If $\mathbf{w}^{(0)} \in \mathcal{C}_0$, its completeness follows from Eq.(15). For soundness, suppose that $\mathbf{w}^{(0)}$ is δ -far from $\mathcal{C}_0$. We analyze the probability, over the folding challenges, that an intermediate fold becomes δ -close to the corresponding code.

4.2 Rank Distance Preservation

The one round soundness error rests on the proximity gap for Gabidulin codes and the following unfolding bound.

Lemma 4.1 (Rank bound under unfolding). *Let Q be a prime power such that $\mathbb{F}_{Q^2} \subseteq \mathbb{F}_{q^m}$, and let $\mathbf{u} = (u_1, \dots, u_N), \mathbf{v} = (v_1, \dots, v_N) \in \mathbb{F}_{q^m}^N$. For any $\lambda \in \mathbb{F}_{Q^2} \setminus \mathbb{F}_Q$, define*

$$\text{Unfold}_\lambda(\mathbf{u}, \mathbf{v}) = (\mathbf{u} + \mathbf{v}^Q, \lambda \mathbf{u} + \lambda^Q \mathbf{v}^Q) \in \mathbb{F}_{q^m}^{2N},$$

where the Q -th power of $\mathbf{v}$ is applied coordinatewise. Then

$$\text{rk}_{\mathbb{F}_Q}(\text{Unfold}_\lambda(\mathbf{u}, \mathbf{v})) \leq 2 \text{rk}_{\mathbb{F}_{Q^2}} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}.$$

Proof. Let r be the interleaved $\mathbb{F}_{Q^2}$ -rank of $\mathbf{u}$ and $\mathbf{v}$ and $s = [\mathbb{F}_{q^m} : \mathbb{F}_{Q^2}]$. Fix an $\mathbb{F}_{Q^2}$ -basis $\beta_1, \dots, \beta_s$ of $\mathbb{F}_{q^m}$. There exist two matrices $M_u, M_v \in \mathbb{F}_{Q^2}^{s \times N}$ such that

$$\mathbf{u} = (\beta_1, \dots, \beta_s) M_u, \quad \mathbf{v} = (\beta_1, \dots, \beta_s) M_v.$$

By the definition of interleaved rank, the joint row space generated by M_u and M_v has $\mathbb{F}_{Q^2}$ -rank r . Let $B = (B_{\ell,j}) \in \mathbb{F}_{Q^2}^{r \times N}$ be a full row rank matrix whose rows form a basis of this joint row space. Then there exist $C_u, C_v \in \mathbb{F}_{Q^2}^{s \times r}$ such that $M_u = C_u B, M_v = C_v B$. Define

$$(a_1, \dots, a_r) = (\beta_1, \dots, \beta_s) C_u, \quad (b_1, \dots, b_r) = (\beta_1, \dots, \beta_s) C_v.$$

It follows that, for every coordinate $j \in [N]$, $u_j = \sum_{\ell=1}^r a_\ell B_{\ell,j}$, and $v_j = \sum_{\ell=1}^r b_\ell B_{\ell,j}$. Define an $\mathbb{F}_Q$ -linear map

$$\Phi : \mathbb{F}_{Q^2}^r \longrightarrow \mathbb{F}_{q^m}^m, \quad \Phi(x_1, \dots, x_r) = \sum_{\ell=1}^r (a_\ell x_\ell + b_\ell^Q x_\ell^Q).$$

Let $B[j] = (B_{1,j}, \dots, B_{r,j})^\top$ be the j -th column of B . Then

$$u_j + v_j^Q = \Phi(B[j]), \quad \text{and} \quad \lambda u_j + \lambda^Q v_j^Q = \Phi(\lambda B_j).$$

Therefore, every coordinate of $\text{Unfold}_\lambda(u, v)$ belongs to $\text{Im}(\Phi)$. Consequently,

$$\text{rk}_{\mathbb{F}_Q}(\text{Unfold}_\lambda(\mathbf{u}, \mathbf{v})) \leq \dim_{\mathbb{F}_Q}(\text{Im}(\Phi)) \leq \dim_{\mathbb{F}_Q}(\mathbb{F}_{Q^2}^r) = 2r.$$

□

Lemma 4.2 (Distance preservation). *Let $0 < \delta < (1-\rho)/2$. If $\Delta_{\text{rk}_{\text{qi}}}(\mathbf{w}, \mathcal{C}_i) > \delta$, then*

$$\Pr_{z \leftarrow \mathbb{F}_{q^m}} \left[\Delta_{\text{rk}_{\text{qi}+1}}(\text{Fold}_{i+1}(\mathbf{w}, z), \mathcal{C}_{i+1}) \leq \delta \right] \leq \varepsilon_{e_{i+1}},$$

where $e_{i+1} = \lfloor n_{i+1} \delta \rfloor$ and $\varepsilon_{e_{i+1}}$ is the soundness error in [Theorem 2.4](#) for $\mathcal{C}_{i+1}$ at radius δ .

Proof. Let $f_{\mathbf{w}} = f_0 + f_1^{q_i}$ be the decomposition in [Eq.\(12\)](#), and set

$$\mathbf{f}_0 := \text{ev}_{\mathcal{A}_r^{(i+1)}}(f_0), \quad \mathbf{f}_1 := \text{ev}_{\mathcal{A}_r^{(i+1)}}(f_1).$$

Then $\text{Fold}_{i+1}(\mathbf{w}, z) = \mathbf{f}_0 + z\mathbf{f}_1$. Suppose that this word is δ -close to $\mathcal{C}_{i+1}$ for more than an ε_{i+1} fraction of the challenges z . By the proximity gap property, there exist $\mathbf{c}_0, \mathbf{c}_1 \in \mathcal{C}_{i+1}$ such that

$$\text{rk}_{\mathbb{F}_{q_{i+1}}} \begin{pmatrix} \mathbf{f}_0 - \mathbf{c}_0 \\ \mathbf{f}_1 - \mathbf{c}_1 \end{pmatrix} \leq \delta n_{i+1}.$$

Let $g_0, g_1 \in \mathcal{L}(q_{i+1}, m_{i+1}, k_{i+1})$ be the interpolation polynomials of $\mathbf{c}_0$ and $\mathbf{c}_1$, respectively, and define

$$g := g_0 + g_1^{q_i}.$$

Then $g \in \mathcal{L}(q_i, m_i, k_i)$, so $\text{ev}_{\mathcal{A}_r^{(i)}}(g) \in \mathcal{C}_i$. Applying [Lemma 4.1](#) with $Q = q_i$ and $\lambda = \lambda_{i+1}$ gives

$$\text{rk}_{\mathbb{F}_{q_i}} \left(\mathbf{w} - \text{ev}_{\mathcal{A}_r^{(i)}}(g) \right) \leq 2\delta n_{i+1} = \delta n_i.$$

This contradicts $\Delta_{\text{rk}_{\text{qi}}}(\mathbf{w}, \mathcal{C}_i) > \delta$. □

4.3 The FGI Protocol

The protocol enables a verifier with oracle access to a word $\mathbf{w}^{(0)} : \mathcal{A}_r^{(0)} \rightarrow \mathbb{F}_{q^m}$ to test its proximity to $\mathcal{C}_0$. The prover supplies a sequence of claimed folded oracles, while the verifier checks the local relation [Eq.\(14\)](#) along a small number of random paths.

FGI: an IOPP for Gabidulin codes

- **Input oracle.** The prover provides oracle access to $\mathbf{w}^{(0)} : \mathcal{A}_r^{(0)} \rightarrow \mathbb{F}_{q^m}$.
- **Folding phase.** $\mathcal{P}(\mathbf{w}^{(0)}, z_1, \dots, z_i) \rightarrow \mathbf{w}^{(i)}$ for $i = 1, \dots, r'$:
 1. The verifier samples $z_i \xleftarrow{\$} \mathbb{F}_{q^m}$ and sends it to the prover.
 2. The prover provides oracle access to a claimed folded word $\mathbf{w}^{(i)} : \mathcal{A}_r^{(i-1)} \rightarrow \mathbb{F}_{q^m}$. An honest prover sets

$$\mathbf{w}^{(i)} = \text{Fold}_i(\mathbf{w}^{(i-1)}, z_i).$$

- **Query phase.** Repeat ℓ times:
 1. The verifier samples $\alpha^{(0)} \xleftarrow{\$} \mathcal{A}_r^{(0)}$
 2. For $i = 1, 2, \dots, r'$
 - (i) Define $\alpha^{(i)} := \pi_{i-1,i}(\alpha^{(i-1)}) \in \mathcal{A}_r^{(i)}$
 - (ii) let $\beta := \alpha^{(i)}$ and query

$$\mathbf{w}^{(i-1)}(\beta), \mathbf{w}^{(i-1)}(\lambda_i \beta), \mathbf{w}^{(i)}(\beta).$$

- (iii) The verifier computes

$$(a, b) = (\mathbf{w}^{(i-1)}(\beta), \mathbf{w}^{(i-1)}(\lambda_i \beta)) \cdot \frac{1}{\lambda_i^{q_{i-1}} - \lambda_i} \begin{pmatrix} \lambda_i^{q_{i-1}} - \lambda_i \\ -1 & 1 \end{pmatrix}$$

and accepts this local check if $\mathbf{w}^{(i)}(\beta) = a + z_i b^{q_{i-1}^{-1}}$.

- 3. **Final check.** The verifier reads the entire final oracle $\mathbf{w}^{(r')}$, interpolates it over $\mathcal{A}_r^{(r')}$, and checks that the resulting $q_{r'}$ -linearized polynomial has $q_{r'}$ -degree less than $k_{r'}$. The verifier accepts and output 1 if all local checks and the final degree check pass. Otherwise the verifier rejects and output 0.

The perfect completeness of **FGI** follows from [Eq.\(15\)](#): if $\mathbf{w}^{(0)} \in \mathcal{C}_0$ and the prover follows the prescribed folding procedure, then $\mathbf{w}^{(i)} \in \mathcal{C}_i$ at every level, all local checks hold, and the final degree check accepts. Thus we have

Lemma 4.3 (Completeness). *Let $\mathcal{C}_0 = \mathcal{G}(\mathcal{A}_r^{(0)}, \rho n)$. If **FGI** is applied to an honest prover with a codeword $\mathbf{w}^{(0)} \in \mathcal{C}_0$, then the verifier always accepts.*

Lemma 4.4 (Soundness). *Let $\mathcal{C}_0 = \mathcal{G}(\mathcal{A}_r^{(0)}, \rho n)$ be the Gabidulin code. Assume $\delta \in (0, (1 - \rho)/2)$. Let $e = \lfloor \delta n \rfloor$ and ε_e be defined as in [Theorem 2.4](#). If **FGI** is applied to a oracle access $\mathbf{w}^{(0)} : \mathcal{A}_r^{(0)} \rightarrow \mathbb{F}_{q^m}$ that has relative rank distance more than δ from $\mathcal{C}_0$, then for any prover oracles $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \dots, \mathbf{w}^{(r')}$, the verifier accepts with probability at most $\varepsilon_e + (1 - \delta)^\ell$.*

Proof. We analyze the soundness by considering two possible cases in each round.

First, suppose that in some round i , the randomness chosen by the verifier happens to make the folded message $\mathbf{w}^{(i)}$ be δ -close to the corresponding code $\mathcal{C}_i$. In this case, the verifier would accept. However, by the proximity gap property, the probability of this event is at most $\varepsilon_{e_i} \leq \varepsilon_e$, where $e_i = \lfloor \delta n_i \rfloor$ and $\varepsilon_{e_i}, \varepsilon_e$ are defined as in [Theorem 2.4](#).

Otherwise, assume that in all rounds the folded function $\mathbf{w}^{(i)}$ remains δ -far from $\mathcal{C}_i$. Since the Hamming distance is at least the rank distance, it follows that for every codeword $\mathbf{c} \in \mathcal{C}_i$, the function $\mathbf{w}^{(i)}$ over $\mathcal{A}_r^{(i)}$ differs from $\mathbf{c}$ on more than a δ -fraction of positions, i.e., there are more than δn_i positions on which $\mathbf{w}^{(i)}$ and $\mathbf{c}$ disagree. Since the verifier performs ℓ random queries in each round, and hence the probability that none of these queries hits a disagreement position is at most $(1 - \delta)^\ell$. Consequently, the verifier fails to detect a mismatch and accepts with probability at most $(1 - \delta)^\ell$.

Combining the two cases, we conclude that a cheating prover can cause the verifier to accept a δ -far word $\mathbf{w}^{(0)}$ only with probability at most $\varepsilon_e + (1 - \delta)^\ell$. $\square$

Efficiency. Let r' denote the number of folding rounds and let $n_i = n/2^i$ be the oracle length at level i . We choose $r' = \Theta(\log n)$ so that the final oracle has constant length. Let ℓ be the number of independent query repetitions.

Round complexity. $r(n) = r' = O(\log(n))$.

Prover time. At round i , each entry of the folded oracle is computed using a constant number of field operations and Frobenius map by the butterfly relation Eq.(9). By Lemma B.1, we regard the Frobenius map as a constant operation over $\mathbb{F}_{q^m}$. Therefore, the total cost of constructing all folded oracles is

$$\sum_{i=1}^{r'} O(n_i) = \sum_{i=1}^{r'} O\left(\frac{n}{2^i}\right) = O(n).$$

Thus, if the initial evaluation oracle is given, the IOPP prover runs in $t_p(n) = O(n)$ field operations.

Proof size. The proof consists of the folded oracles $\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(r')}$. Their total length satisfies $\sum_{i=1}^{r'} n_i < n$, and hence the proof oracle length is $l(n) = O(n)$ field elements.

Verifier time and query complexity Each query repetition follows one path through the $r' = O(\log n)$ folding layers. At every layer, the verifier reads 3 oracle positions and performs a constant number of field operations. The final oracle has constant length, so its interpolation and degree check require only $O(1)$ additional operations. Thus, the verifier makes $q(n) = O(\ell \log n)$ oracle queries and performs $t_v(n) = O(\ell \log n)$ field operations.

5 The PCS for q -linearized Polynomials

In this section, we present a commitment scheme for q -linearized polynomials with polylogarithmic proof size and verification time. The scheme augments the FGI-IOPP with a recursive verification of polynomial evaluations, enabling simultaneous verification of low degree and the claimed polynomial evaluation.

Public parameters. Assume $n = 2^r \mid m$, let $k = \rho n$ for some constant $\rho \in (0, 1)$. For $0 \leq i \leq r$, let $\mathcal{A}_r^{(i)}$ be defined as in Eq.(6). For convenience, define $q_i = q^{2^i}$, $n_i = 2^{r-i}$, $k_i = \rho n_i$. Let $\mathcal{C}_i = \mathcal{G}(\mathcal{A}_r^{(i)}, k_i)$ be the $[n_i, q_i, k_i]$ Gabidulin code with minimum $\mathbb{F}_{q_i}$ -rank distance $d_i = n_i - k_i + 1$.

Commitment phase. Let $f^{(0)}(x) = f(x) \in \mathcal{L}(q, m, k)$. The prover $\mathcal{P}$ computes the initial evaluation

$$\mathbf{w}^{(0)} := \text{ev}_{\mathcal{A}_r^{(0)}}(f^{(0)}).$$

The prover constructs a Merkle tree whose leaves are the entries of $\mathbf{w}^{(0)}$ and publishes $\text{cm}_0 := \text{MerkleCommit}(\mathbf{w}^{(0)})$. In the oracle model, this corresponds to giving the verifier $\mathcal{V}$ oracle access to $\mathbf{w}^{(0)}$: every queried entry of $\mathbf{w}^{(0)}$ is accompanied by its Merkle authentication path.

Open phase. The prover opens a polynomial commitment cm_0 by sending a q -linearized polynomial $f \in \mathcal{L}(q, m, k)$ and a vector $\mathbf{w}$ to the verifier. The verifier checks that (i) the Merkle commitment of $\mathbf{w}$ equals cm_0 , and (ii) the rank distance between $\mathbf{w}$ and $\text{ev}_{\mathcal{A}_r^{(0)}}(f)$ is at most $(d-1)/2$, where d is the minimum rank distance of $\mathcal{C}_0$.

Evaluation protocol. The evaluation protocol presented in **PCS.Eval** consists of two phases. In the evaluation open phase, given a commitment cm_0 , a point $\alpha \in \mathbb{F}_{q^m}$, the prover $\mathcal{P}$ will compute $y = f(\alpha)$ and generate a proof π for an opening $f \in \mathcal{L}(q, m, k)$ of cm_0 such that $y = f(\alpha)$. A verifier $\mathcal{V}$ in evaluation verify phase will check the correctness of π . If all check passed, the verifier will accept such a proof and outputs 1.

Evaluation protocol

- **Public parameters:** For $0 \leq i \leq r$, let $\mathcal{A}_r^{(i)}$ be defined as in **Eq.(6)**, $q_i = q^{2^i}$, $n_i = 2^{r-i}$, $k_i = \rho n_i$. Assume that $2^{r'} \mid k$, where $r' \leq r$ is the number of folding rounds.
- **Public input:** A commitment cm_0 , a point $\alpha \in \mathbb{F}_{q^m}$.
- The private witness of the Prover: $f^{(0)} \in \mathcal{L}(q, m, k)$.

– **Evaluation open.** Output a value $y \in \mathbb{F}_{q^m}$ and a proof $\pi = (\{\text{cm}_i, f_0^{(i-1)}(\alpha), f_1^{(i-1)}(\alpha) : i \in [r']\}, \mathbf{w}^{(r')})$ as follows:

1. $\mathcal{P}$ computes and sends the claimed evaluation $y := f^{(0)}(\alpha)$.
2. For $i = 1, \dots, r'$
 - (i) Since $f^{(i-1)}$ is q_{i-1} -linearized, $\mathcal{P}$ decomposes it uniquely as

$$f^{(i-1)}(x) = f_0^{(i-1)}(x) + f_1^{(i-1)}(x)^{q_{i-1}}.$$

Then $\mathcal{P}$ sends $f_0^{(i-1)}(\alpha), f_1^{(i-1)}(\alpha)$ to $\mathcal{V}$.

- (ii) $\mathcal{V}$ samples a folding challenge $z_i \leftarrow_{\$} \mathbb{F}_{q^m}$ and sends z_i to $\mathcal{P}$.
- (iii) $\mathcal{P}$ defines the next folded polynomial by

$$f^{(i)} := \text{Fold}_i(f^{(i-1)}, z_i) = f_0^{(i-1)} + z_i f_1^{(i-1)}.$$

$\mathcal{P}$ computes the folded evaluation $\mathbf{w}^{(i)} := \text{ev}_{\mathcal{A}_r^{(i)}}(f^{(i)})$ and sends its Merkle root $\text{cm}_i := \text{MerkleCommit}(\mathbf{w}^{(i)})$ to $\mathcal{V}$. //In the oracle model, this corresponds to giving $\mathcal{V}$ oracle access to $\mathbf{w}^{(i)}$.

- (iv) If $i = r'$, $\mathcal{P}$ sends the final evaluation $\mathbf{w}^{(r')}$.

– **Evaluation verify.**

1. $\mathcal{V}$ performs checks in **FGI** with oracle access $\mathbf{w}^{(0)}, \dots, \mathbf{w}^{(r')}$.

2. $\mathcal{V}$ checks $y \stackrel{?}{=} f_0^{(0)}(\alpha) + \left(f_1^{(0)}(\alpha)\right)^q$.
3. For every $i = 1, \dots, r' - 1$, the verifier computes $y_i := f_0^{(i-1)}(\alpha) + z_i f_1^{(i-1)}(\alpha)$ and checks $y_i \stackrel{?}{=} f_0^{(i)}(\alpha) + \left(f_1^{(i)}(\alpha)\right)^{q_i}$. // These identities ensure that $y_i = f^{(i)}(\alpha)$ is consistent with the decomposition of the polynomial in the next folding round.
4. $\mathcal{V}$ evaluates $f^{(r')}$ at α and checks $f_0^{(r'-1)}(\alpha) + z_{r'} f_1^{(r'-1)}(\alpha) \stackrel{?}{=} f^{(r')}(\alpha)$, where $f^{(r')}$ is interpolated from $\mathbf{w}^{(r')}$.

If all checks pass, $\mathcal{V}$ outputs 1. Otherwise, it outputs 0.

Remark 5.1. To reduce the proof size in evaluation open, the two evaluations $f_0^{(i-1)}(\alpha)$ and $f_1^{(i-1)}(\alpha)$ in **Step 2** can be replaced by $f_1^{(i-1)}(\alpha)$ alone. Accordingly, in each round the verifier computes only

$$f_0^{(i-1)}(\alpha) := y_{i-1} - \left(f_1^{(i-1)}(\alpha)\right)^{q_i},$$

which is then used in the linear combination for the next round

$$y_i = f_0^{(i-1)}(\alpha) + z_i f_1^{(i-1)}(\alpha).$$

The consistency check $y_i \stackrel{?}{=} f_0^{(i)}(\alpha) + \left(f_1^{(i)}(\alpha)\right)^{q_i}$ is therefore omitted, as it is implicitly enforced by the computation of $f_0^{(i)}(\alpha)$.

Completeness. Suppose that the prover follows the protocol honestly and that $\mathbf{w}^{(0)} = \text{ev}_{\mathcal{A}_r^{(0)}}(f^{(0)})$ for some $f^{(0)} \in \mathcal{L}(q, m, k)$. At every round i , the oracle $\mathbf{w}^{(i)} = \text{ev}_{\mathcal{A}_r^{(i)}}(f^{(i)})$ is exactly the fold of $\mathbf{w}^{(i-1)}$ with challenge z_i . Consequently, all local folding checks and all Merkle authentication checks performed by the FGI verifier succeed.

Furthermore, for all $1 \leq i \leq r'$,

$$f^{(i)}(\alpha) = f_0^{(i-1)}(\alpha) + z_i f_1^{(i-1)}(\alpha) = f_0^{(i)}(\alpha) + f_1^{(i)}(\alpha)^{q_i}.$$

Thus all recursive evaluation consistency checks hold. In particular, $y = f^{(0)}(\alpha)$ and $f_0^{(r'-1)}(\alpha) + z_{r'} f_1^{(r'-1)}(\alpha) = f^{(r')}(\alpha)$. Finally, the terminal oracle $\mathbf{w}^{(r')}$ agrees with the evaluation of the terminal polynomial $f^{(r')}$. Hence an honest prover is always accepted. Therefore, the protocol has perfect completeness.

Binding. According to collision resistance of the Merkle hash function, the commitment cm_0 binds the prover to a unique oracle $\mathbf{w}^{(0)}$.

Lemma 5.2 (Polynomial and evaluation binding). *Let cm_0, α be the public input of **PCS.Eval**. Let r' be number of folding rounds and ℓ be the repetition queries in **FGI**. Assume $\text{cm}_0 = \text{MerkleCommit}(\mathbf{w}^{(0)})$ for some $\mathbf{w}^{(0)} \in \mathbb{F}_{q^m}^n$. Let*

$0 < \delta \leq \frac{d-1}{n}$, $e = \lfloor \delta n \rfloor$ and ε_e be defined as in [Theorem 2.4](#). For a claimed evaluation y and its proof π generated by the evaluation open phase, if all checks in the evaluation verify phase passed with probability at least $(1-\delta)^\ell + \varepsilon_e + \frac{r'}{q^m}$, then there exists a unique polynomial $g \in \mathcal{L}(q, m, k)$ such that the $\Delta_{\text{rk}}(\mathbf{w}^{(0)}, \text{ev}_{\mathcal{A}_r^{(0)}}(g)) \leq \delta$ and $y = g(\alpha)$.

Proof. By the soundness of [FGI](#), the verifier accepts with probability at least $\varepsilon_e + (1-\delta)^\ell$ implies that $\mathbf{w}^{(0)}$ is δ -close to $\mathcal{C}_0$. Therefore, there exists a polynomial $g \in \mathcal{L}(q, m, k)$ such that

$$\Delta_{\text{rk}}(\mathbf{w}^{(0)}, \text{ev}_{\mathcal{A}_r^{(0)}}(g)) \leq \delta.$$

Since $\delta \leq \frac{d-1}{2n}$, we have that such a polynomial $g \in \mathcal{L}(q, m, k)$ is unique. Hence cm_0 is bound to a unique polynomial $g \in \mathcal{L}(q, m, k)$.

It remains to show that $y = g(\alpha)$. Let $g^{(0)} = g$ and recursively define the folded polynomials

$$g^{(i)} = g_0^{(i-1)} + z_i g_1^{(i-1)}, \quad i = 1, 2, \dots, r',$$

where $g_0^{(i-1)}$ and $g_1^{(i-1)}$ are from the decomposition of $g^{(i-1)} = g_0^{(i-1)} + (g_1^{(i-1)})^{q^{i-1}}$. The true evaluations therefore satisfy

$$g^{(i)}(\alpha) = g_0^{(i-1)}(\alpha) + z_i g_1^{(i-1)}(\alpha) = g_0^{(i)}(\alpha) + (g_1^{(i)}(\alpha))^{q^i}.$$

Let $\{f_0^{(i-1)}(\alpha), f_1^{(i-1)}(\alpha) : i \in [r']\}$ be the component evaluations claimed by the prover such that $y_0 := y = f_0^{(0)}(\alpha) + f_1^{(0)}(\alpha)^q$ and

$$y_i := f_0^{(i-1)}(\alpha) + z_i f_1^{(i-1)}(\alpha) = f_0^{(i)}(\alpha) + (f_1^{(i)}(\alpha))^{q^i}, \quad i = 1, 2, \dots, r' - 1.$$

We claim that, if $y \neq g(\alpha)$, the inequality $y_1 \neq g^{(1)}(\alpha)$ holds with probability at least $1 - 1/q^m$ over the random choice of z_1 . More generally, for each $i \in [r']$,

$$\Pr_{z_i} [y_i = g^{(i)}(\alpha) \mid y_{i-1} \neq g^{(i-1)}(\alpha)] \leq \frac{1}{q^m}.$$

This follows because $y_{i-1} \neq g^{(i-1)}(\alpha)$ implying

$$(f_0^{(i-1)}(\alpha), f_1^{(i-1)}(\alpha)) \neq (g_0^{(i-1)}(\alpha), g_1^{(i-1)}(\alpha)).$$

Since $f_0^{(i-1)}(\alpha)$ and $f_1^{(i-1)}(\alpha)$ are sent before z_i is sampled, there are at most one choice of $z_i \in \mathbb{F}_{q^m}$ such that $f_0^{(i-1)}(\alpha) + z_i f_1^{(i-1)}(\alpha) = g_0^{(i-1)}(\alpha) + z_i g_1^{(i-1)}(\alpha)$.

By a union bound over the r' folding rounds, if $y \neq g(\alpha)$, the inequality $y_{r'} \neq g^{(r')}(\alpha)$ holds with probability at least $1 - r'/q^m$. However, at the final round in FGI, the terminal evaluation sent by the prover is $\text{ev}_{\mathcal{A}_{r'}^{(r')}}(g^{(r')})$. Hence the terminal check pass only if $y_{r'-1} = g^{(r')}(\alpha)$. Consequently, if the verifier accept with probability $\geq (1-\delta)^\ell + \varepsilon_e + r'/q^m$, there must be $y = g(\alpha)$. $\square$

Extractability. We follow the knowledge soundness from [31]. By the IOP to NARK transformation of [7], given any PCS evaluation prover that convinces the verifier with nonnegligible probability, there is an efficient extractor that outputs the IOP oracle string that opens the Merkle commitment sent by the prover. Thus it is sufficient to prove the following theorem in the language of IOP.

Lemma 5.3 (Knowledge soundness). *Let the notation be as in Lemma 5.2. Let $n = 2^r$ with $n \mid m$, and choose ℓ and r' such that $(1 - \delta)^\ell + \varepsilon_e + \frac{r'}{q^m} \leq \text{negl}(\kappa)$. For any instance $(\mathbf{w}^{(0)}, \alpha, y)$, and any malicious prover $\mathcal{P}^*$ that succeeds in *PCS.Eval* with nonnegligible probability, there is a polynomial time extractor $\text{Ext}^{\mathcal{P}^*}$ such that with overwhelming probability, $\text{Ext}^{\mathcal{P}^*}$ outputs a polynomial $g \in \mathcal{L}(q, m, k)$ such that $\Delta_{\text{rk}}(\mathbf{w}^{(0)}, \text{ev}_{\mathcal{A}_r^{(0)}}(g)) \leq \delta$ and $g(\alpha) = y$.*

Proof. The extractor first reads the complete initial oracle $\mathbf{w}^{(0)} \in \mathbb{F}_{q^m}^n$. By the polynomial binding analysis as above, there exists a unique $g \in \mathcal{L}(q, m, k)$ satisfying

$$\Delta_{\text{rk}}(\mathbf{w}^{(0)}, \text{ev}_{\mathcal{A}_r^{(0)}}(g)) \leq \delta.$$

The extractor therefore applies the polynomial time decoding algorithm for the Gabidulin code [30] to $\mathbf{w}^{(0)}$ and outputs the corresponding polynomial g . Moreover, the open (α, y) satisfies $y = g(\alpha)$. $\square$

Efficiency. Following the convention in [16, 31], we measure prover and verifier complexity by the number of operations over $\mathbb{F}_{q^m}$ and Merkle tree hash computations, and proof size by the number of $\mathbb{F}_{q^m}$ elements and hash values.

Assume the underlying field $\mathbb{F}_{q^m}$ is sufficiently large such that $\varepsilon_e \leq \frac{10q^{n-1}}{q^m} \leq 2^{-\kappa}$. Then $\frac{r'}{q^m} \ll \frac{n}{q^m} < 2^{-\lambda}$. To ensure $(1 - \delta)^\ell \leq 2^{-\kappa}$, since $\delta \leq \frac{d-1}{2n} = (1 - \frac{k}{n})/2$ is constant, we set $\ell = \kappa / \log_2(1/(1 - \delta)) = \Theta(\kappa)$.

Prover complexity. The prover first computes $\mathbf{w}^{(0)} = \text{ev}_{\mathcal{A}_r^{(0)}}(f^{(0)})$. Using the FFT algorithm, this costs $O(n \log n)$ operations in $\mathbb{F}_{q^m}$. Constructing the Merkle tree for $\mathbf{w}^{(0)}$ requires $O(n)$ hashes.

For $i \in [r']$, at round i , the prover computes the evaluations $\mathbf{w}^{(i)}$ of $f^{(i)} = f_0^{(i-1)} + z^{(i)} f_1^{(i-1)}$ over $\mathcal{A}_r^{(i)}$. Since $f_0^{(i-1)}$ and $f_1^{(i-1)}$ can be directly derived from the evaluations $\mathbf{w}^{(i-1)}$ of $f^{(i-1)} : \mathcal{A}_r^{(i-1)} \rightarrow \mathbb{F}_{q^m}$, the computation of $\mathbf{w}^{(i)}$ requires $O(n_i)$ operations. Constructing the Merkle trees for $\mathbf{w}^{(i)}$ requires $O(n_i)$ hashes. Therefore, these r' steps require $\sum_{i=1}^{r'} O(n_i) = O(n)$ field operations and hashes.

Moreover, the prover must also evaluate polynomials $f_0^{(i-1)}$ and $f_1^{(i-1)}$ at α . Using direct evaluation, the total cost is bounded by $O\left(\sum_{i=0}^{r'-1} k_i\right) = O(k)$.

In sum, the prover complexity is $O(n \log n)$ field operations in $\mathbb{F}_{q^m}$ and $O(n)$ hash computations.

Verifier complexity. For each of the ℓ query paths, the verifier performs 3 folding checks per round. Hence the number of field operations used for the FGI checks is $O(\ell r')$.

The verifier additionally evaluates the terminal polynomial $f^{(r')}$, which has $q_{r'}$ -degree less than $k_{r'} = \frac{k}{2^{r'}}$. This requires $O(k_{r'}) = O(1)$ field operations.

A Merkle authentication path for the i -th oracle contains $O(\log n_i)$ hash values. The number of hash computations performed by the verifier is $O(\ell r' \log n)$. Consequently, the total verifier time is $O(\ell r' \log n) = O(\kappa \log^2 n)$.

In sum, the verifier complexity is $O(\kappa \log n)$ field operations in $\mathbb{F}_{q^m}$ and $O(\kappa \log^2 n)$ hash computations.

Proof size. The proof contains

- r' Merkle roots
- $2r'$ component evaluations $f_0^{(i-1)}(\alpha), f_1^{(i-1)}(\alpha) \in \mathbb{F}_{q^m}$, or r' evaluations $f_1^{(i-1)}(\alpha)$ by [Remark 5.1](#)
- the terminal evaluations, consisting of $n/2^{r'}$ field elements
- $O(\ell r')$ queried field elements
- $O(\ell r' \log n)$ Merkle authentication hashes

Thus the total proof size is dominated by $O(\kappa \log n)$ field elements in $\mathbb{F}_{q^m}$ and $O(\kappa \log^2 n)$ hash values.

References

1. S. Agrawal and D. Boneh. Homomorphic MACs: MAC-based integrity for network coding. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS 2009*, volume 5536 of *LNCS*, pages 292–305. Springer, Berlin, Heidelberg, June 2009. [2](#)
2. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Liger: Lightweight sublinear arguments without a trusted setup. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, Oct. / Nov. 2017. [2](#)
3. G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. STIR: Reed-solomon proximity testing with fewer queries. In L. Reyzin and D. Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 380–413. Springer, Cham, Aug. 2024. [2](#)
4. G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. WHIR: Reed-solomon proximity testing with super-fast verification. In S. Fehr and P.-A. Fouque, editors, *EURO-CRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 214–243. Springer, Cham, May 2025. [2](#)
5. H. Bartz, L. Holzbaur, H. Liu, S. Puchinger, J. Renner, and A. Wachter-Zeh. Rank-metric codes and their applications. *Found. Trends Commun. Inf. Theory*, 19(3):390–546, 2022. [2](#)
6. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, *ICALP 2018*, volume 107 of *LIPICs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018. [2](#)
7. E. Ben-Sasson, D. Carmon, S. Kopparty, and D. Levit. Scalable and transparent proofs over all large fields, via elliptic curves - (ECFFT part II). In E. Kiltz and V. Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 467–496. Springer, Cham, Nov. 2022. [28](#)

8. R. Bhadauria, C. Hazay, M. Venkatasubramanian, W. Wu, and Y. Zhang. Private polynomial commitments and applications to MPC. In A. Boldyreva and V. Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 127–158. Springer, Cham, May 2023. [2](#)
9. L. Bidoux, J. Chi-Domínguez, T. Feneuil, P. Gaborit, A. Joux, M. Rivain, and A. Vinçotte. RYDE: a digital signature scheme based on rank syndrome decoding problem with mpc-in-the-head paradigm. *Des. Codes Cryptogr.*, 93(5):1451–1486, 2025. [3](#)
10. D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In S. Jarecki and G. Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 68–87. Springer, Berlin, Heidelberg, Mar. 2009. [2](#)
11. M. Brehm, B. Chen, B. Fisch, N. Resch, R. D. Rothblum, and H. Zeilberger. Blaze: Fast SNARKs from interleaved RAA codes. In S. Fehr and P.-A. Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 123–152. Springer, Cham, May 2025. [2](#)
12. A. Chiesa, Y. Hu, M. Maller, P. Mishra, P. Vesely, and N. P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Cham, May 2020. [2](#)
13. J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19(90):297–301, 1965. [3](#), [4](#), [6](#)
14. E. M. Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1):3–16, 1985. [3](#), [12](#)
15. A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. [2](#)
16. A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, and R. S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In H. Handschuh and A. Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 193–226. Springer, Cham, Aug. 2023. [2](#), [28](#)
17. M. Hall-Andersen, M. Simkin, and B. Wagner. Foundations of data availability sampling. *CiC*, 1(4):34, 2024. [2](#)
18. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, Dec. 2010. [2](#), [15](#)
19. R. Koetter and F. R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Trans. Inf. Theory*, 54(8):3579–3591, 2008. [2](#)
20. S. Li and C. Xing. Fast fourier transform via automorphism groups of rational function fields. In D. P. Woodruff, editor, *35th SODA*, pages 3836–3859. ACM-SIAM, Jan. 2024. [4](#)
21. S. Li, C. Xing, C. Yuan, and R. Zhu. Proximity gaps for Gabidulin codes and applications. *CoRR*, abs/2609.09838, 2026. [3](#), [9](#), [11](#), [12](#), [13](#)
22. R. Lidl and H. Niederreiter. *Finite fields*. Number 20. Cambridge university press, 1997. [11](#)
23. S.-J. Lin, W.-H. Chung, and Y. S. Han. Novel polynomial basis and its application to reed-solomon erasure codes. In *55th FOCS*, pages 316–325. IEEE Computer Society Press, Oct. 2014. [4](#)
24. J. M. Pollard. The fast Fourier transform in a finite field. *Math. Comput.*, 25(114):365–374, 1971. [4](#)

25. S. Puchinger and A. Wachter-Zeh. Sub-quadratic decoding of gabidulin codes. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 2554–2558. IEEE, 2016. 5
26. S. Puchinger and A. Wachter-Zeh. Fast operations on linearized polynomials and their applications in coding theory. *J. Symb. Comput.*, 89:194–215, 2018. 5
27. N. Silberstein, A. S. Rawat, and S. Vishwanath. Error-correcting regenerating and locally repairable codes via rank-metric codes. *IEEE Trans. Inf. Theory*, 61(11):5765–5778, 2015. 2
28. D. Silva and F. R. Kschischang. Fast encoding and decoding of gabidulin codes. In *IEEE International Symposium on Information Theory, ISIT 2009, June 28 - July 3, 2009, Seoul, Korea, Proceedings*, pages 2858–2862. IEEE, 2009. 5
29. D. Silva, F. R. Kschischang, and R. Koetter. A rank-metric approach to error control in random network coding. *IEEE Trans. Inf. Theory*, 54(9):3951–3967, 2008. 2
30. A. Wachter-Zeh, V. B. Afanassiev, and V. Sidorenko. Fast decoding of gabidulin codes. *DCC*, 66(1-3):57–73, 2013. 28
31. H. Zeilberger, B. Chen, and B. Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In L. Reyzin and D. Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 138–169. Springer, Cham, Aug. 2024. 2, 13, 15, 28
32. Z. Zhang, W. Li, Y. Guo, K. Shi, S. S. M. Chow, X. Liu, and J. Dong. Fast RS-IOP multivariate polynomial commitments and verifiable secret sharing. In D. Balzarotti and W. Xu, editors, *USENIX Security 2024*. USENIX Association, Aug. 2024. 2

A Fast Fourier Transforms of $O(1)$ -Smooth Length

This section extends the radix-2 FFT of [Section 3](#) to $O(1)$ -smooth transform lengths. Fix a constant B , and suppose that the transform length n admitting a bounded radix factorization

$$n = \prod_{j=1}^r p_j, \quad 2 \leq p_j \leq B.$$

The factors p_j need not be distinct. We say that n is B -smooth. Since B is a constant independent of n , we also refer to n as $O(1)$ -smooth.

We assume throughout that $n \mid m$. For $0 \leq i \leq r$, define

$$s_0 := 1, \quad s_i := \prod_{j=1}^i p_j, \quad q_i := q^{s_i}, \quad n_i := [\mathbb{F}_{q^n} : \mathbb{F}_{q_i}] = \frac{n}{s_i}.$$

Thus, $q_0 = q$, $n_0 = n$, and $n_r = 1$. These parameters satisfy

$$s_{i+1} = p_{i+1} s_i, \quad q_{i+1} = q_i^{p_{i+1}}, \quad n_{i+1} = \frac{n_i}{p_{i+1}}.$$

Because $s_i \mid n \mid m$, these fields form a tower

$$\mathbb{F}_q = \mathbb{F}_{q_0} \subset \mathbb{F}_{q_1} \subset \cdots \subset \mathbb{F}_{q_r} \subseteq \mathbb{F}_{q^m}, \quad [\mathbb{F}_{q_i} : \mathbb{F}_{q_{i-1}}] = p_i. \quad (16)$$

For an ordered multipoint set $\mathcal{A} = \{\alpha_0, \alpha_1, \dots, \alpha_{n-1}\}$, we keep the definitions of the DFT and iDFT of a q -linearized polynomial $f(x) \in \mathcal{L}(q, m, n)$ over $\mathcal{A}$ as in [Section 3](#). The fast Fourier transform and its inverse for an $O(1)$ -smooth length n follow the same principle as in the power of two case. We first use the field tower [Eq.\(16\)](#) to construct an $\mathbb{F}_q$ -linearly independent evaluation set with a compatible recursive structure. At recursion level i , a DFT or iDFT of length n_i is reduced to p_{i+1} transforms of length $n_{i+1} = \frac{n_i}{p_{i+1}}$. Thus, both the linearized degree bound and the evaluation domain size decrease by a factor of p_{i+1} at each level.

Construction of the evaluation domain. For every $1 \leq i \leq r$, choose $\lambda_i \in \mathbb{F}_{q_i}$ such that $\mathbb{F}_{q_i} = \mathbb{F}_{q_{i-1}}(\lambda_i)$. Since $[\mathbb{F}_{q_i} : \mathbb{F}_{q_{i-1}}] = p_i$, the elements $1, \lambda_i, \dots, \lambda_i^{p_i-1}$ form a basis of $\mathbb{F}_{q_i}$ over $\mathbb{F}_{q_{i-1}}$. Define

$$\mathcal{A}_r := \{\lambda_1^{e_1} \lambda_2^{e_2} \cdots \lambda_r^{e_r} : 0 \leq e_i < p_i \text{ for all } i\}. \quad (17)$$

The points are ordered by the mixed-radix index, namely, $\mathcal{A}_r = \{\alpha_e = \lambda_1^{e_1} \cdots \lambda_r^{e_r} \mid e = 0, 1, \dots, n-1\}$, where

$$e = e_1 + p_1 e_2 + p_1 p_2 e_3 + \cdots + s_{r-1} e_r.$$

For $0 \leq i \leq r$, define the nested evaluation domains

$$\mathcal{A}_r^{(i)} := \{\lambda_{i+1}^{e_{i+1}} \cdots \lambda_r^{e_r} : 0 \leq e_j < p_j \text{ for } j > i\}. \quad (18)$$

The empty product gives $\mathcal{A}_r^{(r)} = \{1\}$, and $|\mathcal{A}_r^{(i)}| = n_i$.

Lemma A.1. For every $0 \leq i \leq r$, the set $\mathcal{A}_r^{(i)}$ is linearly independent over $\mathbb{F}_{q_i}$. Moreover, for every $0 \leq i < r$,

$$\mathcal{A}_r^{(i)} = \bigsqcup_{\ell=0}^{p_{i+1}-1} \lambda_{i+1}^\ell \mathcal{A}_r^{(i+1)}. \quad (19)$$

Under the mixed-radix ordering, each $\beta = \alpha_u^{(i+1)} \in \mathcal{A}_r^{(i+1)}$ corresponds to the p_{i+1} points $\alpha_{p_{i+1}u+\ell}^{(i)} = \lambda_{i+1}^\ell \beta$, $0 \leq \ell < p_{i+1}$, in $\mathcal{A}_r^{(i)}$.

Proof. For each i , the powers $1, \lambda_{i+1}, \dots, \lambda_{i+1}^{p_{i+1}-1}$ form a basis of $\mathbb{F}_{q_{i+1}}$ over $\mathbb{F}_{q_i}$. Iterating these bases shows that the products in Eq.(18) form a basis of $\mathbb{F}_{q_r}$ over $\mathbb{F}_{q_i}$. The decomposition Eq.(19) follows by partitioning the points according to the exponent $e_{i+1} \in \{0, \dots, p_{i+1} - 1\}$ of λ_{i+1} .

A.1 Fast Multipoint Evaluation

We now derive a mixed radix FFT for evaluating a given q -linearized polynomial $f \in \mathcal{L}(q, m, n)$ over the domain $\mathcal{A}_r$. At recursion level i , each transform of length n_i is reduced to p_{i+1} transforms of length $n_{i+1} = \frac{n_i}{p_{i+1}}$. After r recursion levels, the original transform is reduced to n transforms of length one, each of which can be computed using $O(1)$ operations. We describe a generic recursion level $0 \leq i \leq r - 1$.

For notational convenience, let $p := p_{i+1}$ denote the current radix. Define the q_i -Frobenius automorphism $(\cdot)^{q_i} : a \mapsto a^{q_i}$ for every $a \in \mathbb{F}_{q^m}$. For every $t \geq 0$, we write the t -fold iterate q_i -Frobenius automorphism and its inverse by $(\cdot)^{q_i^t}$ and $(\cdot)^{q_i^{-t}}$, respectively.

Let $g(x) = \sum_{j=0}^{n_i-1} b_j x^{q_i^j}$ be a q_i -linearized polynomial of q_i -degree less than n_i . Writing each coefficient index uniquely as $j = pu + t$, where $0 \leq t < p$, yields the decomposition

$$g(x) = \sum_{t=0}^{p-1} g_t(x)^{q_i^t}, \quad (20)$$

where

$$g_t(x) := \sum_{u=0}^{n_{i+1}-1} b_{pu+t}^{q_i^{-t}} x^{q_{i+1}^u}, \quad 0 \leq t < p. \quad (21)$$

Each g_t is a q_{i+1} -linearized polynomial of q_{i+1} -degree less than $n_{i+1} = n_i/p$.

Let $\beta \in \mathcal{A}_r^{(i+1)}$. Since $\lambda_{i+1} \in \mathbb{F}_{q_{i+1}} = \mathbb{F}_{q_i^p}$ and every g_t is q_i^p -linearized,

$$g_t(\lambda_{i+1}^\ell \beta) = \lambda_{i+1}^\ell g_t(\beta) \quad \text{for } 0 \leq \ell, t < p.$$

It follows from Eq.(20) that

$$\begin{pmatrix} g(\beta) \\ g(\lambda_{i+1}\beta) \\ \vdots \\ g(\lambda_{i+1}^{p-1}\beta) \end{pmatrix} = M_i \begin{pmatrix} g_0(\beta) \\ g_1(\beta)^{q_i} \\ \vdots \\ (g_{p-1}(\beta)^{q_i^{p-1}}) \end{pmatrix}, \quad (22)$$

where

$$M_i := \left(\lambda_{i+1}^{\ell q_i^t} \right)_{0 \leq \ell, t < p}. \quad (23)$$

The matrix M_i is the Moore matrix associated with $1, \lambda_{i+1}, \dots, \lambda_{i+1}^{p-1}$. These elements are linearly independent over $\mathbb{F}_{q_i}$, and hence M_i is invertible.

Eq.(22) reduces one transform of length n_i to p transforms of length n_i/p , followed by a constant-size Moore transform for each $\beta \in \mathcal{A}_r^{(i+1)}$.

Algorithm 3: FFT _{i} (**b**) over $\mathcal{A}_r^{(i)}$

Input: $\mathbf{b} = (b_0, \dots, b_{n_i-1})$, the coefficients of $g(x) = \sum_{j=0}^{n_i-1} b_j x^{q_i^j}$
Output: $(g(\alpha_0^{(i)}), \dots, g(\alpha_{n_i-1}^{(i)}))$, where $\mathcal{A}_r^{(i)} = \{1 = \alpha_0^{(i)}, \dots, \alpha_{n_i-1}^{(i)}\}$

```

1 if  $n_i = 1$  then
2   return  $(b_0)$ 
3  $p \leftarrow p_{i+1}$ 
4 for  $t \leftarrow 0$  to  $p-1$  do
5    $\mathbf{b}_t \leftarrow (b_{pu+t}^{q_i^{-t}})_{0 \leq u < n_i/p}$ 
6    $\mathbf{z}_t \leftarrow \text{FFT}_{i+1}(\mathbf{b}_t)$  over  $\mathcal{A}_r^{(i+1)}$ 
7 for  $u \leftarrow 0$  to  $n_i/p-1$  do
8    $\mathbf{h}_u \leftarrow (z_{t,u}^{q_i^t})_{0 \leq t < p}$ 
9    $(y_{pu}, \dots, y_{pu+p-1})^\top \leftarrow M_i \mathbf{h}_u$ , where  $M_i$  is defined as in Eq.(23)
10 return  $(y_0, \dots, y_{n_i-1})$ 

```

The desired transform is the top level call FFT₀. When $p_i = 2$ for every i , the decomposition and butterfly reduce to those of the radix-2 FFT in Section 3.1.

Theorem A.2 (FFT of $O(1)$ -smooth length). *Let $n = \prod_{j=1}^r p_j$ satisfy $n \mid m$ and $2 \leq p_j \leq B$ for a constant B . Let $\mathcal{A}_r$ be the evaluation domain defined in Eq.(17). Given $\mathbf{b} = (b_0, \dots, b_{n-1}) \in \mathbb{F}_{q^m}^n$ representing $f_{\mathbf{b}}(x) = \sum_{j=0}^{n-1} b_j x^{q^j} \in \mathcal{L}(q, m, n)$, the top level call FFT₀(**b**) computes $\text{DFT}_{\mathcal{A}_r}(f_{\mathbf{b}})$ using $O(n \log n)$ arithmetic operations over $\mathbb{F}_{q^m}$.*

Proof. At recursion level i , let $p = p_{i+1}$. Forming the p coefficient vectors in Eq.(21) requires n_i Frobenius applications. As in the radix-2 case, Frobenius maps and their inverses are cyclic coordinate shifts under a normal basis representation. For each of the n_i/p points in $\mathcal{A}_r^{(i+1)}$, the local butterfly relation Eq.(22) is a multiplication by a $p \times p$ matrix and therefore costs $O(p^2)$ operations. The total nonrecursive work at this level is

$$n_i + \frac{n_i}{p} \cdot p^2 = O(p n_i) = O(Bn).$$

Let $T_i(n_i)$ denote the number of operations computing $(g(\alpha_0^{(i)}), \dots, g(\alpha_{n_i-1}^{(i)}))$. Then

$$T_i(n_i) = pT_{i+1}(n_{i+1}) + O(Bn_i).$$

Since every $p_i \geq 2$, the recursion depth satisfies $r \leq \log_2 n$. Therefore, the above recursive formula leads to

$$\begin{aligned} T_0(n) &= p_1 T_1(n_1) + O(Bn) \\ &= s_i \cdot T_i(n_i) + i \cdot O(Bn) \\ &= s_r \cdot T_r(1) + r \cdot O(Bn) \\ &= O(Bn \log n) = O(n \log n). \end{aligned}$$

□

Corollary A.3 (Fast Gabidulin encoding). *Let $1 \leq k \leq n$, and let $\mathcal{G}(\mathcal{A}_r, k)$ be the Gabidulin code defined over the domain $\mathcal{A}_r$. After padding a message coefficient vector of length k with $n - k$ zeros, the corresponding Gabidulin codeword can be computed using $O(n \log n)$ arithmetic operations over $\mathbb{F}_{q^m}$.*

A.2 Fast Multipoint Interpolation

The inverse transform reverses each mixed radix butterfly relation as in Eq.(22). For every $\beta = \alpha_u^{(i+1)} \in \mathcal{A}_r^{(i+1)}$, Equation Eq.(22) gives

$$\begin{pmatrix} g_0(\beta) \\ g_1(\beta)^{q_i} \\ \vdots \\ g_{p-1}(\beta)^{q_i^{p-1}} \end{pmatrix} = M_i^{-1} \begin{pmatrix} g(\beta) \\ g(\lambda_{i+1}\beta) \\ \vdots \\ g(\lambda_{i+1}^{p-1}\beta) \end{pmatrix}. \quad (24)$$

Applying $(\cdot)^{q_i^{-t}}$ to the t -th component recovers the evaluation of g_t at β . The component polynomials are then interpolated recursively and recombined using Eq.(20).

Theorem A.4 (Inverse FFT of $O(1)$ -smooth length). *Under the assumptions of Theorem A.2, for every $\mathbf{y} \in \mathbb{F}_{q^m}^n$, the top level call $\text{iFFT}_0(\mathbf{y})$ returns the unique polynomial $f \in \mathcal{L}(q, m, n)$ satisfying $\text{DFT}_{\mathcal{A}_r}(f) = \mathbf{y}$. The algorithm uses $O(n \log n)$ arithmetic operations over $\mathbb{F}_{q^m}$.*

Proof. The inverse Moore transform in Eq.(24) recovers the evaluations of the p component polynomials, and the final coefficient interleaving reverses Eq.(21). Correctness follows by induction on the recursion depth. Since $p \leq B$, multiplication by M_i^{-1} costs $O(1)$ per output block. The resulting recurrence is the same as for the forward transform, and hence the interpolation complexity is $O(n \log n)$.

Algorithm 4: iFFT_{*i*}(**y**) over $\mathcal{A}_r^{(i)}$

Input: $\mathbf{y} = (y_0, \dots, y_{n_i-1})$, the evaluations of an unknown q_i -linearized polynomial of q_i -degree less than n_i
Output: Its coefficient vector $(b_0, \dots, b_{n_i-1})$

```

1 if  $n_i = 1$  then
2   return  $(y_0)$ 
3  $p \leftarrow p_{i+1}$ 
4 for  $u \leftarrow 0$  to  $n_i/p - 1$  do
5    $\mathbf{h}_u \leftarrow M_i^{-1}(y_{pu}, \dots, y_{pu+p-1})^\top$ 
6   for  $t \leftarrow 0$  to  $p - 1$  do
7      $z_{t,u} \leftarrow h_{t,u}^{q_i^{-t}}$ 
8 for  $t \leftarrow 0$  to  $p - 1$  do
9    $\mathbf{c}_t \leftarrow \text{iFFT}_{i+1}(\mathbf{z}_t)$  over  $\mathcal{A}_r^{(i+1)}$ 
10 for  $u \leftarrow 0$  to  $n_i/p - 1$  do
11   for  $t \leftarrow 0$  to  $p - 1$  do
12      $b_{pu+t} \leftarrow c_{t,u}^{q_i^t}$ 
13 return  $(b_0, \dots, b_{n_i-1})$ 

```

B Frobenius maps in a normal basis

Lemma B.1. *Let $\mathcal{B} = (\theta, \theta^q, \dots, \theta^{q^{m-1}})$ be a normal basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. For $a = \sum_{j=0}^{m-1} a_j \theta^{q^j} \in \mathbb{F}_{q^m}$ with $a_j \in \mathbb{F}_q$, and any integer $s \geq 0$, the coordinate vector of a^{q^s} with respect to $\mathcal{B}$ is obtained from the coordinate vector of a by a cyclic shift through $s \bmod m$ positions.*

Proof. Since each a_j belongs to $\mathbb{F}_q$, we have $a_j^{q^s} = a_j$. Therefore,

$$a^{q^s} = \sum_{j=0}^{m-1} a_j \theta^{q^{j+s}}.$$

Because $\theta^{q^m} = \theta$, the exponents of the basis elements are taken modulo m . Hence exponentiation by q^s cyclically permutes the normal basis coordinates by $s \bmod m$ positions.

Remark B.2. **Lemma B.1** shows that a Frobenius map and its inverse act as cyclic shift of the coordinates in a normal-basis representation. Explicitly performing either shift requires $O(m \log q)$ bit operations, which is linear in the representation size of an element of $\mathbb{F}_{q^m}$ and matches the asymptotic bit cost of field addition. Accordingly, we count each application of a Frobenius map or its inverse as one operation over $\mathbb{F}_{q^m}$.