FRIttata: Distributed Proof Generation of FRI-based SNARKs

Hua Xu[1], Mariana Gama[1], Emad Heydari Beni[1,2] and Jiayi Kang[1]

[1] COSIC, KU Leuven, Leuven, Belgium
[2] Nokia Bell Labs, Antwerp, Belgium

Abstract. We present the first horizontally scalable SNARK for general circuits that is both transparent and plausibly post-quantum (PQ) secure. This system adapts the distributed proof generation technique introduced in Pianist (IEEE S&P 2024), which achieves linear scalability by encoding witnesses using bivariate polynomials and committing to them using the KZG polynomial commitment scheme. While Pianist and other scalable SNARK systems offer strong performance profiles, they rely on trusted setup ceremonies and cryptographic assumptions that are not PQ secure, e.g., pairing-based primitives. In contrast, we present a bivariate polynomial commitment scheme based on FRI, achieving a transparent and plausibly PQ alternative. Distributed FRI has a high communication cost. Therefore, we introduce *Fold-and-Batch*, a customizable technique that applies partial folding locally before performing batched FRI centrally. We formally prove the security of our constructions and provide an implementation for three variants of distributed FRI with thorough performance evaluations. Our results show that Fold-and-Batch reduces communication overhead compared to existing distributed FRI approaches while preserving scalability and keeping proof sizes moderate. To our knowledge, this is the first horizontally scalable SNARK for general circuits that at the same time achieves transparency, plausible PQ security, with a tunable tradeoff between efficiency, verifier cost and communication.

Keywords: Zero-knowledge proofs · Horizontal scalability · FRI

1 Introduction

A *Succinct Non-Interactive Argument of Knowledge*, also called a SNARK, is a short proof that a statement is true, and any prover who can generate a valid SNARK must know a corresponding "witness". SNARKs are *succinct*, meaning that the proof length and the time needed to verify it are poly-logarithmic in the size of the statement. Additionally, the proving process is *non-interactive*, i.e., the prover can generate the proof without communicating with the verifier. If the proof does not reveal any information about the witness beyond the statement itself, we call it a *zero-knowledge SNARK* (zk-SNARK). Zk-SNARKs have enabled many real-world applications, from concealing cryptocurrency transaction details [BSCG+14] to proving the existence of software vulnerabilities without disclosing the details of the bug [GHAH+23, CHP+23]. However, proof generation in existing SNARKs remains computationally expensive, and proving large statements on a single machine requires a significant amount of memory. *Plonk* [GWC19], one of the most popular zk-SNARK proof systems, requires 121 seconds and 200 GB of memory to prove a statement expressed using 2^{25} gates [LXZ+24]. This severely limits the applications of SNARKs.

E-mail: hua.xu@student.kuleuven.be (Hua Xu), mariana.botelhodagama@esat.kuleuven.be (Mariana Gama), emad.heydaribeni@esat.kuleuven.be (Emad Heydari Beni), jiayi.kang@esat.kuleuven.be (Jiayi Kang)

One solution to this problem is to generate SNARK proofs distributively by dividing the data and computational tasks among a cluster of machines. Yet, most SNARK proof systems were not designed to be distribution-friendly, and oftentimes the SNARK protocol must be adapted for efficient distributed computation. DIZK [WZC+18], based on the Groth16 [Gro16] proof system, was the first effort to build a distributed zk-SNARK. Using DIZK, one can prove statements that are 100 times larger than when using a monolithic prover; however, it requires a linear amount of communication between the prover nodes due to the interleaving pattern of the FFT algorithm.

Pianist [LXZ+24] is another distributed proof system based on *Plonk*. Instead of using univariate polynomials to encode the prover's witness, *Pianist* employs a novel use of bivariate polynomials to encode the witnesses. This allows the cluster of provers to compute FFTs only on their local polynomials, thereby avoiding the linear communication cost associated with distributing the FFT. *Pianist* is linearly scalable in both the proving time and the memory usage; that is, if we double the number of prover nodes, the proving time and memory usage of each node are reduced by half.

Plonk and *Pianist* both use the KZG polynomial commitment scheme (PCS) [KZG10], which requires a trusted setup procedure where a secret value is used to generate parameters for the proof computation. This secret value must be destroyed at the end of the setup stage; otherwise, it can be used to render the proof system completely insecure. It is therefore desirable to have *transparent* proof systems that do not require such trusted setups. Among these are SNARKs based on FRI [BBHR17], the ***F****ast* ***R****eed-Solomon* ***I****nteractive Oracle Proofs of Proximity.* In addition to being transparent, FRI is also plausibly post-quantum secure and is used in most of the post-quantum zk-SNARK systems deployed today.

This paper explores the possibility of adapting *Pianist*'s distribution technique to the setting of FRI-based SNARKs. The goal is to obtain *the first SNARK for general circuits that is simultaneously transparent, plausibly post-quantum, and horizontally scalable.*

1.1 Related Work

Besides DIZK and *Pianist*, there have been many other constructions for distributed SNARKs. We review them below and summarize their properties in Table 1.

HyperPianist [LZL+25a] is a distributed SNARK based on Pianist and HyperPlonk [CBBZ23]. Plonk encodes the witness using a univariate polynomial evaluated on a cyclic subgroup of a finite field $\mathbb{F}$. On the other hand, HyperPlonk encodes the witness using a multilinear polynomial evaluated on a Boolean hypercube. This encoding eliminates the need for FFT computations, enabling HyperPlonk to have a linear-time prover. Similarly, HyperPianist is able to avoid FFT computations and has a linearly scalable prover. The communication cost for each machine is logarithmic in the overall circuit size. Moreover, instantiating the HyperPianist PIOP with a distributed variant of the Dory PCS [Lee21] results in a transparent (but not post-quantum secure) SNARK system.

Cirrus [WSVZ24] is another distributed SNARK based on HyperPlonk [CBBZ23]. It features *accountability*, which means the master node can identify malicious worker nodes. Cirrus has linearly scalable provers and a per-machine communication cost that is logarithmic in the size of the sub-circuits.

Apart from Plonk-based SNARKs, there is also a line of work distributing SNARKs using the Rank-1 constraint system (R1CS). DIZK [WZC+18] is the first distributed SNARK for R1CS. Hekaton [RMH+24] is a distributed SNARK based on Mirage [KPPS20], which is a variant of Groth16 [Gro16]. Hekaton uses a *divide-and-aggregate* framework, where each prover node generates a separate SNARK proof for its sub-circuit. Then, these proofs are aggregated into a single succinct proof. The wires connecting different sub-circuits are replaced with a global memory bank, and the correctness of memory access is proven using an additional memory-check circuit. Hekaton has a horizontally scalable prover, a constant amount of communication, and a logarithmic proof size.

Soloist [LZL$^+$25b] is the latest distributed SNARK for R1CS. It has a horizontally scalable prover that runs in time quasi-linear in the size of the sub-circuit, with a constant amount of communication per machine and a constant proof size.

Note that only one of the mentioned schemes is transparent and none is plausibly post-quantum; to the best of our knowledge, there is no horizontally scalable SNARK for *general circuits* that satisfies both properties. There is, however, such a proof system known as DeVirgo [XZC$^+$22] with the restriction that it can only prove statements represented by *data-parallel circuits*, meaning that all the sub-circuits handled by prover nodes are identical, with no connecting wires among them. DeVirgo is a distributed SNARK using a FRI-based PCS, based on the transparent and plausibly post-quantum SNARK Virgo [ZXZS20]. It has a linearly scalable prover, but the communication cost per machine is relatively high due to the need for prover nodes to exchange witness data. As a result of using FRI, deVirgo also produces relatively large proofs, which can be significantly reduced using recursive verification with Groth16.

1.2 Our Setting and Contribution

Starting point: the *Pianist* proving system. *Pianist* consists of two main building blocks that are combined to obtain a distributively computable SNARK with perfect scalability:

1. A distributed version of the Plonk polynomial IOP (Interactive Oracle Proof).
2. A distributed version of the KZG polynomial commitment scheme ([LXZ$^+$24], Protocol 2 DKZG), which allows a prover to
   (a) Commit to a bivariate polynomial of the form
   $$F(X,Y)=\sum_{i=0}^{M-1} F_i(X)R_i(Y)$$
   where deg $F_i < T$ for some degree bound T and R_i's are Lagrange polynomials defined on the M-th roots of unity.
   (b) Open $F(X,Y)$ at a point $(\alpha,\beta)\in\mathbb{F}^2$ while providing an evaluation proof.

In *Pianist* (as well as in this work), a master node is responsible for coordinating computations and communication among the worker nodes. In addition, the master node also acts as one of the worker nodes. The master and worker nodes form a star topology, with the master node sitting in the center, so there is no communication between two non-master nodes. The entire circuit is evenly distributed among the worker nodes, meaning that every node will compute a proof for a subcircuit of the same size.

Our Contribution. The contributions of this work are as follows:

- We present a bivariate polynomial commitment scheme based on FRI (denoted as `PCS`) built upon a distributed FRI subprotocol (denoted as `DFRI`). As an extension of the univariate FRI-based PCS [VP19], our `PCS` scheme retains key features such as transparency and plausible post-quantum security. We formally establish the security of `PCS`, including its completeness, (polynomial and evaluation) binding properties, and knowledge soundness that is inherited from knowledge soundness of `DFRI`.
- We construct a distributively computable SNARK for general circuits that is transparent and plausibly post-quantum secure. This is built by compiling *Pianist*'s bivariate IOP with our `PCS` scheme. We also provide a security analysis of this resulting protocol. A summary of our results in comparison with other distributed SNARKs is presented in Table 1.

- We describe three instantiations for `DFRI`: Parallel FRI, Distributed Batched FRI [Che24], and Fold-and-Batch. While Parallel FRI gives a large proof size and Distributed Batched FRI incurs a large communication linear in the overall circuit size, our newly-proposed method, Fold-and-Batch, takes advantage of both to reduce communication costs while maintaining a relatively small proof size. Furthermore, we prove all three `DFRI` instantiations have knowledge soundness as interactive arguments.

- We implement the three `DFRI` instantiations in Rust based on the implementation of FRI in winterfell[1]. We present the measurements on the prover cost, the communication cost, and the verifier cost, and demonstrate that Fold-and-Batch has an efficient prover-verifier tradeoff.

Table 1: A comparison of distributed SNARKs. N: size of the overall circuit. M: number of worker machines. $T = \frac{N}{M}$: size of each sub-circuit held by a worker node. $K \in [1, T]$: Fold-and-Batch folding parameter. "Gen.": general circuits. "Trans.": transparent setup. "PQ": plausibly post-quantum. "Our work" shows the complexities of our FRI-based SNARK instantiated with Fold-and-Batch.

System	Gen.	Trans.	PQ	$\mathcal{P}_i$ time	Overall runtime	Total comm.	$\mathcal{V}$ time
DIZK	✓	✗	✗	$O(T\log^2 T)$	$O(T\log^2 T)$	$O(N)$	$O(1)$
deVirgo	✗	✓	✓	$O(T\log T)$	$O(T\log T)$	$O(N)$	$O(\log^2 N)$
Pianist	✓	✗	✗	$O(T\log T)$	$O(T\log T)$	$O(M)$	$O(1)$
Hekaton	✓	✗	✗	$O(T\log T)$	$O(T\log T)$	$O(M)$	$O(\log M)$
HyperPianist^K	✓	✗	✗	$O(T)$	$O(T + M\log T)$	$O(M\log N)$	$O(\log N)$
HyperPianist^D	✓	✓	✗	$O(T)$	$O(T + M\log T)$	$O(M\log N)$	$O(\log N)$
Cirrus	✓	✗	✗	$O(T)$	$O(T)$	$O(M\log T)$	$O(\log N)$
Soloist	✓	✗	✗	$O(T\log T)$	$O(T\log T)$	$O(M)$	$O(1)$
Our work	✓	✓	✓	$O(T\log T)$	$O(MK + M\log\frac{T}{K}\log T + T\log T)$	$O(MK + M\log\frac{T}{K}\log T)$	$O(M\log\frac{T}{K}\log T + \log^2 K)$

2 Preliminaries

2.1 Notations

For any positive integer k, $[k]$ denotes $\{1, \ldots, k\}$. Let $\mathbb{F} := \mathbb{F}_p$ be a prime field and N be the size of the arithmetic circuit over $\mathbb{F}$ for which we would like to compute a SNARK. Let M be the number of machines participating in the protocol. We use P_0 for the master node, and $\{P_i\}_{1\le i\le M-1}$ for worker nodes. Each node computes a proof for a subcircuit of size $T := \frac{N}{M}$.

Let $\mathtt{RS}[\mathbb{F}, D, T]$ denote the following set of Reed-Solomon codewords:

$$\mathtt{RS}[\mathbb{F}, D, T] := \{(F(x))_{x\in D} \mid F(X) \in \mathbb{F}[X]_{<T}\},$$

where D is the *evaluation domain* of size $|D| = n > T$. The *rate* of the Reed-Solomon code is $\rho := \frac{T}{n}$. This paper considers $D = \mu_n$, the n-th roots of unity with $n = 2^k$ being a power of 2. We also use the shorthand $\mathtt{RS}$ to denote $\mathtt{RS}[\mathbb{F}, D, T]$. In FRI, the inverse of

[1]https://github.com/facebook/winterfell

the code rate, ρ^{-1}, is referred to as the *blow-up factor*. In practice, the blow-up factor is always a power of 2, commonly set to 4 or 8 [0xP21]. For functions $f, g : D \to \mathbb{F}$, we use $\Delta(f, g)$ to denote the *relative Hamming distance* between f and g, i.e.,

$$\Delta(f, g) := \frac{1}{n} \cdot |\{x \in D \mid f(x) \neq g(x)\}|.$$

Let $\delta \in [0, 1)$. A function $f : D \to \mathbb{F}$ is *δ-close* to $\mathtt{RS}$ if there is a polynomial $F \in \mathbb{F}[X]_{<T}$ such that $\Delta(f, F) < \delta$. In this work, we assume $\delta < \frac{1-\rho}{2}$, the unique decoding radius of $\mathtt{RS}$. This means, if f is δ-close to $\mathtt{RS}$, then there is a unique polynomial $F \in \mathbb{F}[X]_{<T}$ in the ball of radius δ around f. For any function $f : D \to \mathbb{F}$, define $\mathtt{RS.Decode}(f)$ to be this unique polynomial $F \in \mathbb{F}[X]_{<T}$ if f is δ-close to $\mathtt{RS}$. Define $\mathtt{RS.Decode}(f)$ to be f if it is δ-far.

2.2 Interactive Proofs and Arguments

Definition 1 (**Relations**). A *relation* $\mathcal{R}$ is a subset of $\{0, 1\}^* \times \{0, 1\}^*$. If $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, then $\mathbb{x}$ is an *instance* of $\mathcal{R}$ and $\mathbb{w}$ is a *witness* for the instance $\mathbb{x}$.

Definition 2 (**Languages**). The *language* $\mathcal{L}_{\mathcal{R}}$ associated to a relation $\mathcal{R}$ is the set of strings $\mathbb{x} \in \{0, 1\}^*$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ for some $\mathbb{w} \in \{0, 1\}^*$.

Definition 3 (**Interactive Arguments of Knowledge**). Let $\mathcal{R}$ be a relation. An *interactive argument of knowledge* for $\mathcal{R}$ is a pair of probabilistic polynomial time interactive algorithms (P, V) satisfying the following conditions:

- **Completeness** Let $\langle P(\mathbb{x}, \mathbb{w}), V(\mathbb{x})\rangle$ denote the output by V after the interaction between P and V where P has input $\mathbb{x}, \mathbb{w}$ and V has input $\mathbb{x}$. If $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, then $\langle P(\mathbb{x}, \mathbb{w}), V(\mathbb{x})\rangle = \mathtt{accept}$ with probability 1.

- **Knowledge Soundness** There is a polynomial time algorithm E with the following property: Let $\mathbb{x} \in \{0, 1\}^*$ and let P^* be any polynomial-time interactive algorithm. If given input $\mathbb{x}$ and access to messages from P^* during the protocol, E outputs a $\mathbb{w} \in \{0, 1\}^*$ satisfying

$$\Pr[\langle P^*(\mathbb{x}), V(\mathbb{x})\rangle = \mathtt{accept} \wedge (\mathbb{x}, \mathbb{w}) \notin \mathcal{R}] = \mathtt{negl}(\lambda).$$

Definition 4 (**Public-coin**). An interactive protocol is *public-coin* if all the messages sent by the verifier are uniformly random and independent of the prover's messages. Intuitively, all verifier's messages are uniformly random coin tosses that could have been sampled by a trusted third party.

Definition 5 (**Interactive Oracle Proofs**). An *interactive oracle proof* for a relation $\mathcal{R}$ with δ-completeness and ϵ-soundness is a pair of interactive algorithms $\Pi = (\mathcal{P}, \mathcal{V})$ such that:

- For $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, $\mathcal{P}$ receives $(\mathbb{x}, \mathbb{w})$ as input and $\mathcal{V}$ receives $\mathbb{x}$ as input before the start of the protocol.

- During each round of interaction, $\mathcal{P}$ sends $\mathcal{V}$ an oracle message m which $\mathcal{V}$ can query at any position. Then, $\mathcal{V}$ responds with a message c. At the end of the protocol, $\mathcal{V}$ outputs either $\mathtt{accept}$ or $\mathtt{reject}$.

- (δ-completeness) For all $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, we have

$$\Pr\big[\langle \mathcal{P}(\mathbb{x}, \mathbb{w}), \mathcal{V}(\mathbb{x})\rangle = \mathtt{accept}\big] \geq \delta,$$

where the probability is taken over the random coins of $\mathcal{V}$.

- (ϵ-soundness) If $\mathbb{x} \notin \mathcal{L}_{\mathcal{R}}$, then for any unbounded interactive algorithm $\mathcal{P}^*$ we have

$$\Pr\left[\langle \mathcal{P}^*(\mathbb{x}), \mathcal{V}(\mathbb{x})\rangle = \texttt{accept}\right] \leq \epsilon,$$

where the probability is taken over the random coins of $\mathcal{V}$.

Definition 6 ([BGK$^+$23], Definition 3.13)**.** A k-round (public-coin) IOP for a relation $\mathcal{R}$ has *round-by-round knowledge soundness* with knowledge error $\epsilon_{\mathtt{k}}$ if there exists a polynomial-time extractor $\texttt{Ext}$ and a (not necessarily efficiently computable) "doomed set" $\mathcal{D}$ containing pairs $(\mathbb{x}, \tau)$ where $\mathbb{x}$ is an input and τ is a partial or complete transcript such that:

1. $(\mathbb{x}, \emptyset) \in \mathcal{D}$ for all possible input $\mathbb{x}$, regardless of whether $\mathbb{x} \in \mathcal{L}_{\mathcal{R}}$ or not.
2. For any possible input $\mathbb{x}$ and any complete transcript τ, if $(\mathbb{x}, \tau) \in \mathcal{D}$ then $\mathcal{V}(\mathbb{x}, \tau) = \texttt{reject}$.
3. If $i \in [k]$ and τ is a $(i-1)$-round partial transcript such that $(\mathbb{x}, \tau) \in \mathcal{D}$, then for every potential prover next message m, if

$$\Pr_{c \xleftarrow{\$} C_i}\left[(\mathbb{x}, \tau_{i-1} \mid\mid m \mid\mid c) \notin \mathcal{D}\right] > \epsilon_{\mathtt{k}},$$

then $\texttt{Ext}(\mathbb{x}, \tau, m)$ outputs a valid witness for $\mathbb{x}$.

2.3 The FRI low-degree test and Batched FRI

The FRI low-degree test was introduced in [BBHR17] to prove that a Merkle-committed function $f : D \to \mathbb{F}$ is δ-close to a polynomial of degree $< T$, i.e. δ-close to $\texttt{RS}[\mathbb{F}, D, T]$. The FRI protocol consists of a *folding phase* and a *query phase*. In the folding phase, $\mathcal{P}$ folds f into a smaller instance so that it takes less work for $\mathcal{V}$ to check that f is close to a polynomial of low degree. In the query phase, $\mathcal{V}$ sends several challenges to $\mathcal{P}$, which $\mathcal{P}$ must answer correctly for $\mathcal{V}$ to accept. These queries check that $\mathcal{P}$ has folded f honestly. We include the full FRI protocol in Appendix A for completeness.

Batched FRI. The batched FRI protocol [BCI$^+$20] proves proximity to $\texttt{RS}$ for multiple polynomials $f_1, \ldots, f_k \in \mathbb{F}[X]$ by running a single instance of FRI. The batched FRI procedure consists of the following steps:

1. $\mathcal{P}$ sends the Merkle commitments $\{\texttt{MT.Commit}(f_i)\}_{i=1}^k$ to $\mathcal{V}$.
2. $\mathcal{V}$ samples a random challenge $\theta \xleftarrow{\$} \mathbb{F}$ and sends it to $\mathcal{P}$.
3. $\mathcal{P}$ computes the random linear combination $f(X) := \sum_{i=1}^k \theta^i f_i(X)$. $\mathcal{P}$ and $\mathcal{V}$ run FRI on $f(X)$. During the query phase, $\mathcal{V}$ performs some additional checks for each FRI query g:
   (a) $\mathcal{V}$ queries $f_i(X)$ at g for all i.
   (b) $\mathcal{V}$ checks that the opening proof (Merkle authentication path) for each $f_i(g)$ is valid and that $f(g) = \sum_{i=1}^k \theta^i f_i(g)$.
4. If $\mathcal{V}$ accepts the FRI proof for f and the additional checks pass, $\mathcal{V}$ outputs $\texttt{accept}$. Otherwise, $\mathcal{V}$ outputs $\texttt{reject}$.

The soundness of batched FRI is proven in [BCI$^+$20], Theorem 8.3. Informally, it says that if any of the functions f_i is δ-far from $\texttt{RS}$, then with high probability, the batched FRI verifier will reject.

2.4 A FRI-based Univariate PCS

Vlasov and Panarin [VP19] proposed a univariate polynomial commitment scheme based on FRI, avoiding the trusted setup required in KZG-based schemes. It consists of the following algorithms:

- $\texttt{Setup}(1^\lambda) \to \mathbf{pp}$: Generates the public parameters $\mathbf{pp} = (\mathbb{F}, T, l, \rho, \delta)$ containing the field $\mathbb{F}$, degree bound T, number of query rounds l, code rate ρ and the proximity parameter δ for FRI.
- $\texttt{Commit}(f, \mathbf{pp}) \to \mathcal{C}$: Given any function $f : D \to \mathbb{F}$, outputs the Merkle commitment $\mathcal{C} = \texttt{MT.Commit}(f)$.
- $\texttt{Open}(\mathcal{C}, \alpha, \mathbf{pp}) \to \texttt{accept}$ or $\texttt{reject}$:
  1. It is assumed that $\alpha \notin D$. $\mathcal{P}$ computes $z := f(\alpha)$ and sends it to $\mathcal{V}$.
  2. $\mathcal{P}$ and $\mathcal{V}$ run the FRI low-degree test on the polynomial $q(X) := \frac{f(X)-z}{X-\alpha}$. The oracle queries for $q(X)$ are provided through the oracle of $f(X)$. In other words, if $\mathcal{V}$ would like to query $q(x)$ for some $x \in D$, then $\mathcal{V}$ queries $f(x)$ and computes $\frac{f(x)-z}{x-\alpha}$. If $\mathcal{V}$ accepts the FRI proof, then it outputs $\texttt{accept}$. Otherwise, $\mathcal{V}$ outputs $\texttt{reject}$.

The FRI low-degree test in $\texttt{Open}$ proves that $q(X) = \frac{f(X)-z}{X-\alpha}$ is δ-close to $\texttt{RS}$. In fact, this implies that $f(X)$ must also be δ-close to $\texttt{RS}$. In the end, the verifier is convinced that $f(X)$ is δ-close to a polynomial $F(X) \in \mathbb{F}[X]_{<T}$ and $F(\alpha) = z$.

3 Bivariate FRI-based PCS

We start by defining the algorithms and functionalities for $\texttt{DFRI}$ (Distributed FRI) in Section 3.1. $\texttt{DFRI}$ is an important sub-protocol of the bivariate FRI-based polynomial commitment scheme. In Section 3.2, we describe the construction of the bivariate FRI-based PCS, define its desired security properties, and provide security proofs for them. Towards the end of the section, we discuss how the security of the PCS implies the security of the compiled SNARK under appropriate conditions.

3.1 Distributed FRI

First, we define the abstract functionality $\texttt{DFRI}$ (short for **D**istributed **FRI**) consisting of the following algorithms:

- $\texttt{Commit}(f, \mathbf{pp}) \to \mathcal{C}$: given any function $f : D \to \mathbb{F}$, outputs the Merkle commitment $\mathcal{C} = \texttt{MT.Commit}(\texttt{RS.Decode}(f))$.
- $\texttt{Prove}((q_i)_{i=0}^{M-1}, \mathbf{pp}) \to \pi$: given functions $q_i : D \to \mathbb{F}$, outputs a proof π proving that each $\texttt{RS.Decode}(q_i)$ is δ-close to $\texttt{RS}$.
- $\texttt{Verify}((\mathcal{C}_i)_{i=0}^{M-1}, (z_i)_{i=0}^{M-1}, \alpha, \pi, \mathbf{pp}) \to 0$ or 1.

such that the pair of algorithms $\Pi_{\texttt{DFRI}} = (\texttt{Prove}, \texttt{Verify})$ is a public-coin interactive argument for the following relation satisfying **completeness** and **knowledge soundness**:

$$\mathcal{R}_{\texttt{DFRI}} = \left\{ \left(\mathbb{x} = ((\mathcal{C}_i)_{i=0}^{M-1}, (z_i)_{i=0}^{M-1}, \alpha), \mathbb{w} = (q_i)_{i=0}^{M-1} \right) \middle| \begin{array}{c} q_i : D \to \mathbb{F} \text{ is } \delta\text{-close to } \texttt{RS}, \\ \mathcal{C}_i = \texttt{DFRI.Commit}(f_i) \\ \text{where } f_i(X) := q_i(X)(X-\alpha) + z_i \end{array} \right\}$$

Intuitively, f_i is a function that is able to *explain* the commitment $\mathcal{C}_i$ and the evaluation $(\texttt{RS.Decode}(f_i))(\alpha) = z_i$ simultaneously.

The most basic instantiation of this functionality is the batched FRI protocol from [BCI+20]. However, there are other variants of the batched FRI protocol designed specifically for distributed proof generation [Che24]. In the description of the FRI-based Distributed PCS defined below, we will only refer to the abstract `DFRI` interface and leave the specific choice of instantiation open.

Informally, the guarantee we would like from `DFRI` is the following: If π is a proof generated by running `DFRI.Prove` on functions $q_0, \ldots, q_{M-1}$ and the verifier accepts, then each q_i is δ-close to a polynomial $Q_i(X) \in \mathbb{F}[X]$ of degree $< T$ with high probability.

3.2 Bivariate FRI-based PCS

3.2.1 Construction

Let $\omega_Y \in \mathbb{F}$ be a primitive M-th root of unity. Let $R_i(Y) \in \mathbb{F}[Y]_{<M}$ be the i-th Lagrange polynomial defined at the M-th roots of unity $\mu_M = \{1, \omega_Y, \ldots, \omega_Y^{M-1}\}$. Let $S_i(Y) \in \mathbb{F}[Y]_{<M-1}$ be the i-th Lagrange polynomial defined at $\mu_M \setminus \{\omega_Y^{M-1}\}$. Let $\mathcal{F}$ be the collection of bivariate polynomials $F(X,Y) \in \mathbb{F}[X,Y]$ of the form

$$F(X,Y) = \sum_{i=0}^{M-1} F_i(X) R_i(Y),$$

where $F_i(X) \in \mathbb{F}[X]_{<T}$. The bivariate FRI-based polynomial commitment scheme `PCS` for $\mathcal{F}$ consists of the following algorithms:

- `Precomputation`$(M) \to (s_i)_{i=0}^{M-2}$: The verifier computes and outputs $s_i := S_i(\omega_Y^{M-1})$ for $i = 0, \ldots, M-2$. This precomputation only depends on the number of machines M. Hence, its output can be reused for different circuits.

- `Setup`$(1^\lambda, (s_i)_{i=0}^{M-2}) \to \mathbf{pp}$: Outputs the public parameters $\mathbf{pp} = (\mathbb{F}, T, l, \rho, \delta, (s_i)_{i=0}^{M-2})$ containing the field $\mathbb{F}$, the degree bound T, the number of query rounds l, the code rate ρ, the proximity parameter δ for `DFRI` and the precomputed values $(s_i)_{i=0}^{M-2}$ for verification.

- `Commit`$(F, \mathbf{pp}) \to \mathcal{C}$: The prover computes $\mathcal{C}_i = $ `DFRI.Commit`(F_i) for each i and outputs $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_{M-1})$.

- `VerifyPoly`$(\mathcal{C}, F, \mathbf{pp}) \to 0$ or 1: Check if $F \in \mathcal{F}$ and $\mathcal{C} = $ `Commit`$(F, \mathbf{pp})$. Output 1 if both checks pass. Otherwise, output 0.

- `Open`$(F, \alpha, \beta, \mathbf{pp}) \to (z, (z_i)_{i=0}^{M-1}, \pi)$: Here we assume $\beta \notin \mu_M$. If $\beta \in \mu_M$, then the protocol is aborted. Since $|\mu_M| = M \ll |\mathbb{F}|$, the probability of this is very small.
  1. The prover computes $z_i := F_i(\alpha)$, $Q_i := \frac{F_i(X) - z_i}{X - \alpha}$ for all i and $z := F(\alpha, \beta)$.
  2. The prover produces a proof π by running `DFRI.Prove`$(Q_0, \ldots, Q_{M-1}, \mathbf{pp})$.
  3. The prover outputs $(z, (z_i)_{i=0}^{M-1}, \pi)$.

- `Verify`$(\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}, \pi, \mathbf{pp}) \to 0$ or 1:
  1. The verifier runs `DFRI.Verify`$(\mathcal{C}, (z_i)_{i=0}^{M-1}, \alpha, \pi, \mathbf{pp})$, and outputs 0 if the verification fails.
  2. The verifier computes $h_i := \frac{z_i - z}{\omega_Y^i - \beta}$ for $i = 0, \ldots, M-2$.

3. The verifier checks that
$$z_{M-1} - z - \sum_{i=0}^{M-2} h_i \cdot S_i(\omega_Y^{M-1})(\omega_Y^{M-1} - \beta) = 0 \tag{1}$$
using the precomputed values $S_i(\omega_Y^{M-1})$, and outputs 1 if the check passes.

We explain the intuition behind the construction for `Open` and `Verify`. If $f_0, \ldots, f_{M-1}$ are the functions committed by the prover in the `Commit` phase, and all f_i's are indeed δ-close to $\texttt{RS}$, then we know the bivariate polynomial committed by the prover is $F(X,Y) = \sum_{i=0}^{M-1} F_i(X)R_i(Y)$ where $F_i(X) = \texttt{RS.Decode}(f_i)$. Essentially, the prover proves to the verifier that $F(\alpha,\beta) = z$ in two steps:

1. Prove that $F_i(\alpha) = z_i$ for all i.
2. Prove that $F(\alpha, Y) - z$ has a root at $Y = \beta$.

To prove the first step, the prover runs `DFRI.Prove` on the functions $\frac{f_i(X)-z_i}{X-\alpha}$. By the same reasoning for the univariate FRI-based PCS, the verifier is convinced that $F_i(\alpha) = z_i$. At this point, the verifier can already construct $F(\alpha, Y)$ by computing $F(\alpha, Y) = \sum_{i=0}^{M-1} F_i(\alpha)R_i(Y) = \sum_{i=0}^{M-1} z_i R_i(Y)$. Directly checking that $F(\alpha, Y) - z$ vanishes at $Y = \beta$ would take time $O(M^2)$. We use an approach that takes time linear in M with a small constant by using the following fact: $F(\alpha, Y) - z$ has a root at $Y = \beta$ if and only if $F(\alpha, Y) - z = H(Y)(Y - \beta)$ for some polynomial $H(Y)$. Notice that the verifier already has all the information it needs to construct H. Since $H(Y) = \frac{F(\alpha,Y)-z}{Y-\beta}$ has degree $M-2$, it is determined by its evaluation at $M-1$ points. A convenient choice would be $M-1$ points chosen from $\mu_M = \{1, \omega_Y, \omega_Y^2, \ldots, \omega_Y^{M-1}\}$, because the verifier already knows how to evaluate $F(\alpha, Y)$ at any of these points: evaluating at ω_Y^i simply produces z_i. This is exactly what the verifier does in Step 2 of `Verify`. It constructs $H(Y)$ by computing its evaluation at $1, \omega_Y, \omega_Y^2, \ldots, \omega_Y^{M-2}$. To conclude that $F(\alpha, Y) - z = H(Y)(Y - \beta)$, it suffices to show that both sides are equal when evaluated at M distinct points. The verifier already knows that both sides are equal when evaluated at $1, \omega_Y, \omega_Y^2, \ldots, \omega_Y^{M-2}$ by the construction of H. It remains to check the equality when $Y = \omega_Y^{M-1}$. This is done in Step 3 of `Verify`.

3.2.2 Security Properties

In this section, we define the security properties for a polynomial commitment scheme `PCS`.

Completeness. `PCS` has *completeness* if for any polynomial $F \in \mathcal{F}$ and field elements $\alpha, \beta \in \mathbb{F}$, the following probability is at least $1 - \texttt{negl}(\lambda)$:
$$\Pr\left[\begin{array}{c} (s_i)_{i=0}^{M-2} \leftarrow \texttt{Precomputation}(M) \\ \mathbf{pp} \leftarrow \texttt{Setup}(1^\lambda, (s_i)_{i=0}^{M-2}) \\ \mathcal{C} \leftarrow \texttt{Commit}(F, \mathbf{pp}) \\ (z, (z_i)_{i=0}^{M-1}, \pi) \leftarrow \texttt{Open}(F, \alpha, \beta, \mathbf{pp}) \end{array} : \texttt{Verify}(\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}, \pi, \mathbf{pp}) = 1\right]$$
Informally, if the prover followed the protocol to commit and open F, then the verifier always accepts except with negligible probability.

Computationally Polynomial Binding. `PCS` is *computationally polynomial binding* if for any polynomial-time adversary $\mathcal{A}$ and any tuple $(\mathcal{C}, F, G)$ output by $\mathcal{A}$ given properly generated public parameters $\mathbf{pp}$, we have
$$\Pr\left[\begin{array}{c} \texttt{VerifyPoly}(\mathcal{C}, F, \mathbf{pp}) = 1, \\ \texttt{VerifyPoly}(\mathcal{C}, G, \mathbf{pp}) = 1, \\ F \neq G \end{array}\right] = \texttt{negl}(\lambda).$$

This property ensures that, with high probability, a commitment cannot be successfully opened to two different polynomials by a polynomial-time prover.

Computationally Evaluation Binding. $\texttt{PCS}$ is *computationally evaluation binding* if for any polynomial-time adversary $\mathcal{A}$, any tuple $(\mathcal{C}, \alpha, \beta, (z, (z_i)_{i=0}^{M-1}, \pi), (z', (z'_i)_{i=0}^{M-1}, \pi'))$ output by $\mathcal{A}$ given properly generated public parameters $\mathbf{pp}$, we have

$$\Pr\left[\begin{array}{c} \texttt{Verify}(\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}, \pi, \mathbf{pp}) = 1, \\ \texttt{Verify}(\mathcal{C}, \alpha, \beta, z', (z'_i)_{i=0}^{M-1}, \pi', \mathbf{pp}) = 1, \\ z \neq z' \end{array}\right] = \texttt{negl}(\lambda).$$

Knowledge Soundness. $\texttt{PCS}$ has *knowledge soundness* if any probabilistic polynomial-time prover $\mathcal{P}^*$ who successfully convinces the verifier the validity of an opening $(\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1})$ also *knows* a polynomial $F(X, Y)$ that explains this opening, i.e. $\mathcal{C}$ is the commitment of F, $F_i(\alpha) = z_i$ and $F(\alpha, \beta) = z$.

Formally, we view the pair of algorithms $\Pi_{\texttt{PCS}} = (\mathcal{P} = \texttt{Open}, \mathcal{V} = \texttt{Verify})$ as a public-coin interactive argument for the following relation:

$$\mathcal{R}_{\texttt{PCS}} := \left\{ \left(\begin{array}{c} \mathbb{x} = (\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}), \\ \mathbb{w} = (f_0, \dots, f_{M-1}) \end{array} \right) \middle| \begin{array}{c} f_i \text{ is } \delta\text{-close to } \texttt{RS} \text{ for all } i \wedge \\ \mathcal{C}_i = \texttt{DFRI.Commit}(f_i) \text{ for all } i \wedge \\ F_i(\alpha) = z_i \text{ where} \\ F_i(X) = \texttt{RS.Decode}(f_i) \wedge \\ \sum_i F_i(\alpha) R_i(\beta) = z \end{array} \right\}$$

The polynomial commitment scheme $\texttt{PCS}$ has **knowledge soundness** if $\Pi_{\texttt{PCS}}$ has knowledge soundness as an interactive argument.

3.2.3 Security Analysis

In this section, we prove that the polynomial commitment scheme $\texttt{PCS}$ defined in Section 3.2.1 has all the security properties defined in Section 3.2.2 when using a valid instantiation of $\texttt{DFRI}$, i.e., when $\Pi_{\texttt{DFRI}}$ is an interactive argument of knowledge. Then, we prove that this is enough to establish the security of the interactive protocol obtained by compiling *Pianist*'s bivariate polynomial IOP with our $\texttt{PCS}$.

Security of the polynomial commitment scheme. If $\Pi_{\texttt{DFRI}}$ has completeness, then $\texttt{PCS}$ also has completeness. This follows directly from the construction of $\texttt{PCS}$. $\texttt{PCS}$ also satisfies polynomial binding and evaluation binding. Polynomial binding follows from the fact that two polynomials of degree $< T$ cannot agree on every point in the evaluation domain D of size $\rho^{-1}T > T$ and the tree-collision property of Merkle trees. We prove these three security properties in Appendix B.

We will prove that $\texttt{PCS}$ has knowledge soundness if $\Pi_{\texttt{DFRI}}$ has knowledge soundness after proving a useful lemma:

Lemma 1. *Suppose that for functions $f_i : D \to \mathbb{F}$, elements $\alpha, \beta, z_i, z \in \mathbb{F}$ and proof π, the following conditions are satisfied:*

1. $\frac{f_i(X) - z_i}{X - \alpha}$ *is δ-close to $\texttt{RS}$ for $i = 0, \dots, M-1$.*

2. $\texttt{PCS.Verify}(\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}, \pi, \mathbf{pp}) = 1$.

Let $F_i(X) := \texttt{RS.Decode}(f_i)$ and $F(X, Y) := \sum_{i=0}^{M-1} F_i(X) R_i(Y)$. Then, $F(\alpha, \beta) = z$.

Proof. Since $\frac{f_i(X)-z_i}{X-\alpha}$ is δ-close to $\mathtt{RS}$, there is a polynomial $P_i(X) \in \mathbb{F}[X]_{<T}$ that is δ-close to it. This means $f_i(X)$ is δ-close to the polynomial $G_i(X) := P_i(X)(X-\alpha)+z_i$ for each i. It is clear that $G_i(\alpha) = z_i$ for all i.

Let $h_i := \frac{z_i - z}{\omega_Y^i - \beta}$ for $i = 0, \dots, M-2$, $H(Y) := \sum_{i=0}^{M-2} h_i \cdot S_i(Y)$, and $G(X,Y) := \sum_{i=0}^{M-1} G_i(X) R_i(Y)$. Then $G(\alpha, \omega_Y^i) = G_i(\alpha) = z_i$ for $i = 0, \dots, M-1$. Therefore,

$$\begin{aligned} G(\alpha, \omega_Y^i) - z - H(\omega_Y^i)(\omega_Y^i - \beta) &= G(\alpha, \omega_Y^i) - z - h_i \cdot (\omega_Y^i - \beta) \\ &= z_i - z - h_i \cdot (\omega_Y^i - \beta) = 0, \end{aligned}$$

for $i = 0, \dots, M-2$. Since the last check of Equation 1 passes, we also have

$$\begin{aligned} G(\alpha, \omega_Y^{M-1}) - z - H(\omega_Y^{M-1})(\omega_Y^{M-1} - \beta) &= z_{M-1} - z - \sum_{i=0}^{M-2} h_i \cdot S_i(\omega_Y^{M-1})(\omega_Y^{M-1} - \beta) \\ &= 0\,. \end{aligned}$$

We have shown that the polynomial $G(\alpha, Y) - z - H(Y)(Y-\beta) \in \mathbb{F}[Y]$ of degree $< M$ has M distinct roots, therefore it must be the zero polynomial. Taking $Y = \beta$, we get $G(\alpha, \beta) = z$. Since we are in the unique decoding regime, there is only one polynomial of degree $< T$ in the ball of radius δ around $f_i(X)$. Thus, $G_i(X) = \mathtt{RS.Decode}(f_i) = F_i(X)$, which means $F(\alpha, \beta) = G(\alpha, \beta) = z$. □

Proposition 1. *If* $\Pi_{\mathtt{DFRI}}$ *has knowledge soundness, then* $\mathtt{PCS}$ *also has knowledge soundness.*

Proof. We describe $\mathcal{E}_{\mathtt{PCS}}$, the extractor for the polynomial commitment scheme. Let $\mathcal{P}^*_{\mathtt{PCS}}$ denote the $\mathtt{PCS}$ prover. Since in $\mathtt{PCS}$, the prover does not send any message before running $\mathtt{DFRI.Prove}$, we can set $\mathcal{P}^*_{\mathtt{DFRI}} := \mathcal{P}^*_{\mathtt{PCS}}$. Since $\mathcal{E}_{\mathtt{PCS}}$ has access to $\mathcal{P}^*_{\mathtt{DFRI}}$'s messages, it can run $\mathcal{E}_{\mathtt{DFRI}}$ and output the witness $(q_0, \dots, q_{M-1})$ for $\mathcal{R}_{\mathtt{DFRI}}$. Then the witness $(f_0, \dots, f_{M-1})$ for $\mathcal{R}_{\mathtt{PCS}}$ is obtained by computing $f_i(X) = q_i(X)(X-\alpha) + z_i$.

Let A be the event that $\mathcal{V}_{\mathtt{DFRI}}$ accepts, and

$$(\mathbb{x} = (\mathcal{C}, (z_i)_{i=0}^{M-1}, \alpha), \mathbb{w} = (q_0, \dots, q_{M-1})) \notin \mathcal{R}_{\mathtt{DFRI}}$$

Let B be the event that $\mathcal{V}_{\mathtt{PCS}}$ accepts, and

$$(\mathbb{x} = (\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}), \mathbb{w} = (f_0, \dots, f_{M-1})) \notin \mathcal{R}_{\mathtt{PCS}}$$

By the knowledge soundness of $\Pi_{\mathtt{DFRI}}$, $\Pr[A] = \mathtt{negl}(\lambda)$. To show knowledge soundness of $\mathtt{PCS}$, we need to show $\Pr[B] = \mathtt{negl}(\lambda)$. We have $\Pr[B] = \Pr[A \cap B] + \Pr[A^c \cap B]$, and $\Pr[A \cap B] = \mathtt{negl}(\lambda)$. We will show that $\Pr[A^c \cap B] = 0$. In the event $A^c \cap B$, since B happened, we have that $\mathcal{V}_{\mathtt{PCS}}$ accepts. This means $\mathcal{V}_{\mathtt{DFRI}}$ accepted as well. Since A did not happen, we must have $((\mathcal{C}, (z_i)_{i=0}^{M-1}, \alpha), (q_0, \dots, q_{M-1})) \in \mathcal{R}_{\mathtt{DFRI}}$. Explicitly, this means

1. Each $\frac{f_i(X)-z_i}{X-\alpha}$ is δ-close to $\mathtt{RS}$, therefore each f_i is δ-close to $\mathtt{RS}$.
2. $\mathcal{C}_i = \mathtt{DFRI.Commit}(f_i)$ for all i.

By Lemma 1, $((\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}), (f_0, \dots, f_{M-1})) \in \mathcal{R}_{\mathtt{PCS}}$, contradicting B. □

Security of the compiled protocol. In *Pianist* [LXZ+24], they built an interactive argument of knowledge ([LXZ+24], Protocol 3) by compiling the bivariate polynomial IOP ([LXZ+24], Protocol 1) with the $\mathtt{DKZG}$ polynomial commitment scheme ([LXZ+24], Protocol 2). The security of the final interactive protocol comes from Theorem 4 of [BFS20], which says if the polynomial commitment scheme Γ has witness-extended emulation and the polynomial IOP for the relation $\mathcal{R}$ has negligible knowledge error, then the compiled protocol is a public-coin interactive argument for $\mathcal{R}$ with witness-extended emulation. It was proven ([Lin01], Lemma 3.1) that any interactive argument of knowledge has witness-extended emulation. Therefore, it suffices to show that the polynomial commitment scheme Γ has knowledge soundness. This is exactly what is done in *Pianist*. In this work, we would like to compile *Pianist*'s bivariate polynomial IOP with our bivariate FRI-based polynomial commitment scheme.

Proposition 2. *Let $\Pi_{\texttt{Compiled}}$ be the interactive protocol compiled from Pianist's bivariate PIOP using the bivariate FRI-based polynomial commitment scheme* `PCS`*. If the distributed FRI subprotocol $\Pi_{\texttt{DFRI}}$ used in* `PCS` *is an interactive argument with negligible knowledge error, then $\Pi_{\texttt{Compiled}}$ is a public-coin interactive argument with witness-extended emulation.*

Proof. We have proven that if $\Pi_{\texttt{DFRI}}$ is an interactive argument with completeness and knowledge soundness, then the polynomial commitment scheme `PCS` has completeness (Proposition 5) and knowledge soundness (Proposition 1). Since *Pianist*'s bivariate polynomial IOP has negligible knowledge error ([LXZ+24], Theorem 1), the conclusion follows from Theorem 4 of [BFS20]. □

4 Instantiations for `DFRI`

In this section, we first describe three methods for instantiating `DFRI`: Parallel FRI, Distributed Batched FRI, and Fold-and-Batch. Then, in Section 4.1, we compare their asymptotic costs and in Section 4.2 we establish their security.

Parallel FRI. The simplest method to run `DFRI` is to let each worker node P_i run a separate instance of FRI on the input function $q_i : D \to \mathbb{F}$. Then, the verifier checks each one of the M FRI proofs. In the non-interactive version of the protocol, since each worker node derives its folding challenges using the Fiat-Shamir transformation, no communication is needed. Therefore, we do not need a master node in this setting. The overall runtime for Parallel FRI is $O(T)$, the runtime for FRI with input size T. The proof size is $O(M \log^2 T)$.

Distributed Batched FRI. In [Che24], the author described a distributed protocol for running batched FRI. For convenience, we will refer to this method as Distributed Batched FRI. In Table 3, we describe this protocol adapted to the setting of `DFRI` with a step-by-step cost analysis. In [Che24], each prover node operates on multiple polynomials. Here, each prover node operates on a single polynomial. Furthermore, we assume that a prover node already has the evaluation vector of its polynomial at the beginning of the protocol. This is because the bivariate FRI-based PCS (Section 3.2.1) requires the computation of evaluation vectors at the `Commit` phase. Since `DFRI` is invoked at a later stage, each P_i already has its evaluation vector by then. Due to the absence of FFT computations, the runtime for each worker node becomes $O(T)$ (as opposed to $O(T \log T)$). However, the runtime of P_0 and the total communication remains $O(N)$. The communication bottleneck occurs when P_0 receives an evaluation vector of $\rho^{-1}T$ field elements from each worker node.

Fold-and-Batch. Fold-and-Batch is a `DFRI` instantiation that improves the communication cost of Distributed Batched FRI [Che24]. The idea is to make P_i fold its local polynomial to a smaller size before sending it to P_0. This reduces P_0's runtime and the overall communication. However, it slightly increases the proof size and verification time.

Fold-and-Batch can be viewed as an "interpolation" between Distributed Batched FRI and Parallel FRI. A folding parameter $K \in [1, T]$ (K a power of 2) is used to adjust how much folding is done locally by the worker nodes versus how much folding is done by the master node using batched FRI. When $K = T$, Fold-and-Batch is the same as Distributed Batched FRI. When $K = 1$, Fold-and-Batch is essentially Parallel FRI (with adjustments). By setting $K = \frac{T}{M}$, Fold-and-Batch can achieve a runtime of $O(T + M \log M \log T)$ for all provers and the same for total communication, while Distributed Batched FRI costs $O(N)$ for both. Below is a description of the Fold-and-Batch protocol:

1. P_i computes the Merkle commitment of $F_i(X)$ and sends it to P_0. P_0 sends these commitments to V.
2. P_i folds its evaluation vector $(F_i(x))_{x \in D}$ for $\log \frac{T}{K}$ times. The result is a vector $(G_i(y))_{y \in D'}$ where $G_i(X) = F_i^{(\log \frac{T}{K})}(X)$ is a polynomial of degree $< K$ and $D' = \mu_{\rho^{-1}K}$.
3. P_i sends P_0 the Merkle commitments for every intermediate folded polynomial. This includes the Merkle commitments of $(G_i(y))_{y \in D'}$. P_0 sends these commitments to V.
4. P_0 receives a random challenge θ from V and sends it to P_i.
5. P_i computes the vector $(\theta^i G_i(y))_{y \in D'}$ and sends it to P_0.
6. P_0 computes the vector $(G(y))_{y \in D'}$ where $G(X) = \sum_{i=0}^{M-1} \theta^i G_i(X)$ has degree $< K$.
7. P_0 and V perform the FRI folding phase for $G(X)$.
8. During the query phase, for each query point $g \in D = \mu_{\rho^{-1}T}$, P_0 sends it to all P_i's. Each P_i sends back $F_i^{(0)}(g)$, $F_i^{(0)}(-g)$, $F_i^{(1)}(g^2)$, $F_i^{(1)}(-g^2)$, $\ldots$, $F_i^{(\log \frac{T}{K})}(g^{\frac{T}{K}})$, $F_i^{(\log \frac{T}{K})}(-g^{\frac{T}{K}})$ and their Merkle authentication paths which P_0 sends to V. (This is to check that each P_i folded $F_i(X)$ correctly)
9. For each query g, let $h := g^{\frac{T}{K}}$. P_0 in addition sends the values $G^{(0)}(h)$, $G^{(0)}(-h)$, $G^{(1)}(h^2)$, $G^{(1)}(-h^2)$, $\ldots$, $G^{\log K}(h^K)$ and their Merkle authentication paths to V. (This is to check that P_0 folded $G(X)$ correctly)
10. For each g and the corresponding $h = g^{\frac{T}{K}}$, V checks that:
    (a) Each P_i folded $F_i(X)$ correctly using Equation 2.
    (b) P_0 folded $G(X)$ correctly using Equation 2.
    (c) $G(h) = \sum_{i=0}^{M-1} \theta^i G_i(h)$. V indeed has all the necessary values to perform this check, since $G(h) = G^{(0)}(h)$ and $G_i(h) = F_i^{(\log \frac{T}{K})}(h) = F_i^{(\log \frac{T}{K})}(g^{\frac{T}{K}})$ were sent to V during Step 8 and 9.

4.1 Asymptotic Costs

Table 2 shows the asymptotic complexities of the three DFRI methods and compares them with monolithic FRI. The cost of Distributed Batched FRI and Fold-and-Batch is analyzed step by step in Appendix C, Tables 3 and 4.

4.2 Security

We now establish the security of the three DFRI instantiations: Parallel FRI, Distributed Batched FRI, and Fold-and-Batch. Specifically, we need to prove that these instantiations are interactive arguments of knowledge for $\mathcal{R}_{\texttt{DFRI}}$. Then, by Proposition 1, the bivariate FRI-based PCS has knowledge soundness when instantiated with these DFRI methods.

Table 2: A comparison of different DFRI instantiations

	Monolithic FRI	Distributed Batched FRI	Fold-and-Batch	Parallel FRI
P_0 runtime	$O(N)$	$O(N)$	$O(MK + M\log\frac{T}{K}\log T + \log^2 K)$	—
P_i runtime	—	$O(T)$	$O(T)$	$O(T)$
Total communication between P_0 and P_i	—	$O(N)$	$O(MK + M\log\frac{T}{K}\log T)$	—
Communication for P_i	—	$O(T)$	$O(K + \log\frac{T}{K}\log T)$	—
Proof size / Verifier runtime	$O(\log^2 N)$	$O(\log^2 T + M\log T)$	$O(M\log\frac{T}{K}\log T + \log^2 K)$	$O(M\log^2 T)$

In [BGK+23], the authors proved the knowledge soundness of the non-interactive version of FRI and batched FRI using the following approach: First, prove that the IOP version of FRI and batched FRI have round-by-round knowledge soundness. Then, they compile the IOP into a non-interactive argument using the BCS transformation [BSCS16]. We will follow this approach and argue the security for the three instantiations of DFRI.

The BCS transformation turns any IOP into a non-interactive argument in two steps:

1. Whenever the IOP prover sends an oracle string, the non-interactive argument prover sends a Merkle commitment of that string. Whenever the IOP verifier queries the oracle, the non-interactive argument prover sends the queried value together with a Merkle opening proof.

2. The protocol is then made non-interactive using the Fiat-Shamir transformation [FS87].

In Theorem 3 of [CMS19], the authors proved that if an IOP has round-by-round knowledge soundness, then the BCS-compiled non-interactive argument has knowledge soundness. Furthermore, it has been shown that applying only the first step of the BCS transformation to a round-by-round knowledge sound IOP produces a knowledge sound interactive argument ([CY24], Lemma 26.1.4). The transformation consisting of only the first step of the BCS transformation is called the "iBCS" (*interactive BCS*) transformation in [CY24]. Therefore, to show that a DFRI instantiation has knowledge soundness as an interactive argument, it suffices to show that the corresponding IOP has round-by-round knowledge soundness. For Distributed Batched FRI (i.e., Batched FRI), its round-by-round knowledge soundness is shown in [BGK+23], Theorem 5.11. The same theorem also proves the round-by-round knowledge soundness of FRI. We will follow this proof and establish the round-by-round knowledge soundness of Parallel FRI and Fold-and-Batch. For completeness, we formally define the parallel execution of multiple IOPs in Appendix D. Then, Parallel FRI can be viewed as the parallel execution of M different instances of FRI.

Notations. We define some additional notations that will be used in the upcoming proofs. Let D be the evaluation domain of FRI, and let l be the number of FRI query rounds. Let $G_{i,j}$ denote the polynomial obtained by the honest j-th worker node during the i-th round of Parallel FRI or Fold-and-Batch. Let $G^*_{i,j}$ denote the corresponding polynomial sent by a potentially malicious prover. We assume $G^*_{0,j} = G_{0,j}$ for all $j \in [M]$. Since we are dealing with IOPs, we simply use the same notation G to denote the oracle string of a polynomial G sent by the prover, with the understanding that the verifier does not have to read it in full. For Parallel FRI and the partial FRI folding phase of Fold-and-Batch, we use $\mathbf{G_i}$ to denote $(G_{i,1}, \ldots, G_{i,M})$, $\mathbf{G}^*_i$ to denote $(G^*_{i,1}, \ldots, G^*_{i,M})$, and $\mathbf{c}_i = (c_{i,1}, \ldots, c_{i,M})$ to denote the random folding challenges drawn by the verifier during round i. If the verifier only draws a single FRI folding challenge during round i, we denote it c_i. $\mathtt{s}$ denotes the set of FRI query positions. τ_i denotes the transcript of messages up to and including round i.

We assume the Fold-and-Batch IOP $\Pi_{\texttt{Fold-and-Batch}}$ is run with the following parameters:

- k partial FRI folding rounds, i.e., each worker node folds its local polynomial k times.
- r batched FRI folding rounds, i.e., the batched polynomial becomes a constant after being folded r times by the master node.

The predicate $\texttt{CorrectFolding}(i)$ means that all the FRI foldings done during the protocol up to and including the i-th round are honest. $\texttt{IncorrectFolding}(i)$ means the prover cheated in at least one FRI folding performed up to and including the i-th round. For Fold-and-Batch, we abuse notations and treat the batched FRI linear combination $F := \sum_{j=1}^{M} \alpha^{j-1} G_j$ as one of the FRI foldings. For example, if the prover sent $F^* \neq F = \sum_{j=1}^{M} \alpha^{j-1} G_j$ during round k, then $\texttt{IncorrectFolding}(i) = \texttt{True}$ for all $i \geq k$.

4.2.1 Round-by-round Knowledge Soundness of Parallel FRI

To prove the round-by-round knowledge soundness of Parallel FRI, we need to define a doomed set for it. We start by recalling the doomed set for FRI defined in [BGK+23]:

Definition 7 ([BGK+23], Lemma 5.2, 5.3)**.** The doomed set $\mathcal{D}_{\texttt{FRI}}$ for FRI with k folding rounds is defined as follows:

1. $(\mathbb{x}, \emptyset) \in \mathcal{D}_{\texttt{FRI}}$ for all possible $\mathbb{x}$.
2. If $\mathbb{x} \notin \mathcal{L}$, then $(\mathbb{x}, \tau_0) \in \mathcal{D}_{\texttt{FRI}}$ if and only if G_1 is δ-far from $\texttt{RS}^{(1)}$.
3. If $\mathbb{x} \notin \mathcal{L}$ and $i \in \{1, \dots, k-1\}$, then $(\mathbb{x}, \tau_i) \in \mathcal{D}_{\texttt{FRI}} \iff \texttt{IncorrectFolding}(i)$ or G_{i+1} is δ-far from $\texttt{RS}^{(i+1)}$.
4. (Query phase) If $\mathbb{x} \notin \mathcal{L}$ then for a complete transcript τ_k, $(\mathbb{x}, \tau_k) \in \mathcal{D}_{\texttt{FRI}} \iff$ one of the consistency checks performed by the verifier fails.

Using the doomed set for FRI, we can define a doomed set for Parallel FRI and prove its round-by-round knowledge soundness:

Proposition 3. *Define the language $\mathcal{L} := \mathcal{L}_{RS,\delta}^{M}$ where $\mathcal{L}_{RS,\delta} = \{w \in \mathbb{F}^D \mid \Delta(w, RS) < \delta\}$ is the language of words that are δ-close to RS. Assume the Parallel FRI IOP $\Pi_{\texttt{ParallelFRI}}$ is run with $\delta < \frac{1-\rho}{2}$ (unique decoding radius), then it has round-by-round knowledge error*

$$\epsilon = \max\left\{\frac{|D|}{|\mathbb{F}|}, (1-\delta)^l\right\}$$

Proof. Define the doomed set $\mathcal{D}$ for $\Pi_{\texttt{ParallelFRI}}$ as follows: for any input $\mathbb{x} = (\mathbb{x}_1, \dots, \mathbb{x}_M)$ and an i-th round transcript τ_i, $(\mathbb{x}, \tau_i) \in \mathcal{D}$ if and only if $(\mathbb{x}_j, p_j(\tau_i)) \in \mathcal{D}_{\texttt{FRI}}$ for some $j \in [M]$. We check each property for round-by-round knowledge soundness:

1. For any input $\mathbb{x} = (\mathbb{x}_1, \dots, \mathbb{x}_M)$, by definition $(\mathbb{x}_j, \emptyset) \in \mathcal{D}_{\texttt{FRI}}$ for all j. Hence, $(\mathbb{x}, \emptyset) \in \mathcal{D}$.
2. For any input $\mathbb{x} = (\mathbb{x}_1, \dots, \mathbb{x}_M)$ and complete transcript τ, $(\mathbb{x}_j, p_j(\tau))$ is a complete transcript for $\mathbb{x}_j$. If $(\mathbb{x}, \tau) \in \mathcal{D}$ then $(\mathbb{x}_j, p_j(\tau)) \in \mathcal{D}_{\texttt{FRI}}$ for some j. This means $\mathcal{V}_j$ rejects and so does $\mathcal{V}$.

3. We treat the cases $\mathbb{x} \in \mathcal{L}$ and $\mathbb{x} \notin \mathcal{L}$ separately. Suppose $\mathbb{x} \in \mathcal{L}$, then the conditions $(\mathbb{x}, \tau_{i-1}) \in \mathcal{D}$ and $\Pr_{\mathbf{c} \overset{\$}{\leftarrow} C_i}\left[(\mathbb{x}, \tau_{i-1} \mid\mid m \mid\mid \mathbf{c}) \notin \mathcal{D}\right] > \epsilon$ are both true only when $i = 0$, i.e. when $(\mathbb{x}, \tau_{i-1}) = (\mathbb{x}, \emptyset)$. This is because, by definition, $(\mathbb{x}, \tau) \in \mathcal{D}$ only when $\tau = \emptyset$. In this case, the prover message m is $\mathbf{G}_0$. All $G_{0,j}$'s are δ-close to $\mathtt{RS}^{(0)}$ as $\mathbb{x} \in \mathcal{L}$. Thus, the extractor can simply output the witness $\mathbb{w} = \mathbf{G}_0$.

   Suppose $\mathbb{x} \notin \mathcal{L}$. We will show that it is not possible to have $(\mathbb{x}, \tau_{i-1}) \in \mathcal{D}$ and $\Pr_{\mathbf{c}_i \overset{\$}{\leftarrow} C_i}\left[(\mathbb{x}, \tau_{i-1} \mid\mid m \mid\mid \mathbf{c}_i) \notin \mathcal{D}\right] > \epsilon$ at the same time:

   - <u>Round -1 to Round 0</u>: Suppose $(\mathbb{x}, \emptyset) \in \mathcal{D}$ and $\Pr_{\mathbf{c}_0 \overset{\$}{\leftarrow} \mathbb{F}}\left[(\mathbb{x}, \mathbf{G}_0 \mid\mid \mathbf{c}_0) \notin \mathcal{D}\right] > \epsilon$. Rewriting using the definition of $\mathcal{D}$ we get

     $$\Pr_{\mathbf{c}_0 \overset{\$}{\leftarrow} \mathbb{F}}\left[G_{1,j} \text{ is } \delta\text{-close to } \mathtt{RS}^{(1)} \text{ for all } j \in [M]\right] > \epsilon \geq \frac{|D|}{|\mathbb{F}|}$$

     Since $\mathbb{x} \notin \mathcal{L}$, some G_{0,j_0} must be δ-far from $\mathtt{RS}^{(0)}$. This is a contradiction to Lemma 5.6 of [BGK+23].

   - <u>Round 0 to Round 1</u>: Suppose $(\mathbb{x}, \tau_0) \in \mathcal{D}$, then G_{1,j_0} is δ-far from $\mathtt{RS}^{(1)}$ for some $j_0 \in [M]$. If $\Pr_{\mathbf{c}_1 \overset{\$}{\leftarrow} \mathbb{F}}\left[(\mathbb{x}, \tau_0 \mid\mid \mathbf{G}_1^* \mid\mid \mathbf{c}_1) \notin \mathcal{D}\right] > \epsilon$, then

     $$\Pr_{\mathbf{c}_1 \overset{\$}{\leftarrow} \mathbb{F}}\left[\mathtt{CorrectFolding}(1) \ \wedge G_{2,j} \text{ is } \delta\text{-close to } \mathtt{RS}^{(2)} \text{ for all } j \in [M]\right] > \epsilon.$$

     This means

     $$\Pr_{\mathbf{c}_1 \overset{\$}{\leftarrow} \mathbb{F}}\left[G_{2,j_0} \text{ is } \delta\text{-close to } \mathtt{RS}^{(2)}\right] > \epsilon \geq \frac{|D|}{|\mathbb{F}|},$$

     contradicting Lemma 5.6 of [BGK+23].

   - <u>Round $i-1$ to Round i, $2 \leq i \leq k-1$</u>: Suppose $(\mathbb{x}, \tau_{i-1}) \in \mathcal{D}$, then we either have $\mathtt{IncorrectFolding}(i-1)$ or G_{i,j_0} is δ-far from $\mathtt{RS}^{(i)}$ for some $j_0 \in [M]$. If

     $$\Pr_{\mathbf{c}_i \overset{\$}{\leftarrow} \mathbb{F}}\left[(\mathbb{x}, \tau_{i-1} \mid\mid \mathbf{G}_i^* \mid\mid \mathbf{c}_i) \notin \mathcal{D}\right] > \epsilon$$

     then

     $$\Pr_{\mathbf{c}_i \overset{\$}{\leftarrow} \mathbb{F}}\left[\mathtt{CorrectFolding}(i) \wedge G_{i+1,j} \text{ is } \delta\text{-close to } \mathtt{RS}^{(i+1)} \text{ for all } j \in [M]\right] > \epsilon.$$

     This implies $\mathtt{CorrectFolding}(i-1)$, so G_{i,j_0} is δ-far from $\mathtt{RS}^{(i)}$ for some $j_0 \in [M]$. The last probability also implies

     $$\Pr_{\mathbf{c}_i \overset{\$}{\leftarrow} \mathbb{F}}\left[G_{i+1,j_0} \text{ is } \delta\text{-close to } \mathtt{RS}^{(i+1)}\right] > \epsilon \geq \frac{|D|}{|\mathbb{F}|},$$

     contradicting Lemma 5.6 of [BGK+23].

   - <u>Round $k-1$ to Round k</u>: Suppose $\Pr_{\mathbf{s} \overset{\$}{\leftarrow} \mathbb{F}}\left[(\mathbb{x}, \tau_{k-1} \mid\mid \mathbf{G}_k^* \mid\mid \mathbf{s}) \notin \mathcal{D}\right] > \epsilon$, then

     $$\Pr_{\mathbf{s} \overset{\$}{\leftarrow} \mathbb{F}}\left[\ \mathcal{V} \text{ accepts }\ \right] > \epsilon \geq (1-\delta)^l.$$

     Since $\mathbb{x} \notin \mathcal{L}$, G_{0,j_0} is δ-far from $\mathtt{RS}^{(0)}$ for some j_0. However, we always have $G_{k,j}^* \in \mathbb{F}$ which represents a codeword in $\mathtt{RS}^{(k)}$ for all j. This is a contradiction to Lemma 5.4 of [BGK+23]. $\square$

4.2.2 Round-by-round Knowledge Soundness of Fold-and-Batch

Before defining a doomed set for Fold-and-Batch, we give a description of its transcripts:

- (**Round** 0 **to Round** $k-1$) For these partial FRI folding rounds, the i-th round partial transcript is $\tau_i := \tau_{i-1} \mid\mid \mathbf{G}_i^* \mid\mid \mathbf{c}_i$ where $\mathbf{c}_i = (c_{i,1}, \dots, c_{i,M})$ contains the random folding challenges sent by the verifier and will be used by the provers to obtain the next folded local polynomial.
- (**Round** k) This is the last partial FRI folding round, and $\tau_k := \tau_{k-1} \mid\mid \mathbf{G}_k^* \mid\mid \alpha$. Instead of sending the next folding challenge, the verifier sends the batched FRI challenge α.
- (**Round** $k+1$ **to Round** $k+r$) During these rounds, the (honest) master prover performs the FRI folding of the batched polynomial $F_0 = \sum_{j=1}^{M} \alpha^{j-1} G_{k,j}$. For $i \in \{0, \dots, r-1\}$, the $(k+1+i)$-th round partial transcript is $\tau_{k+1+i} := \tau_{k+i} \mid\mid F_i^* \mid\mid y_i$.
- (**Round** $k+r+1$) This is the last round of the batched FRI phase and the entire protocol. We have $\tau_{k+r+1} := \tau_{k+r} \mid\mid F_r^* \mid\mid \mathbf{s}$ where $\mathbf{s}$ contains the FRI query positions sampled from the evaluation domain D by the verifier.

A complete transcript for $\Pi_{\texttt{Fold-and-Batch}}$ has the following format:

$$\mathbf{G}_0 \mid\mid \mathbf{c}_0 \mid\mid \mathbf{G}_1^* \mid\mid \mathbf{c}_1 \mid\mid \dots \mid\mid \mathbf{c}_{k-1} \mid\mid \mathbf{G}_k^* \mid\mid \alpha \mid\mid$$
$$F_0^* \mid\mid y_0 \mid\mid F_1^* \mid\mid y_1 \mid\mid \dots \mid\mid y_{r-1} \mid\mid F_r^* \mid\mid \mathbf{s}$$

Definition 8 (**Doomed Set for Fold-and-Batch**)**.** The doomed set $\mathcal{D}$ for Fold-and-Batch is defined as follows:

1. $(\mathbb{x}, \emptyset) \in \mathcal{D}$ for all possible $\mathbb{x}$.
2. If $\mathbb{x} \notin \mathcal{L}$, then $(\mathbb{x}, \tau_0) \in \mathcal{D}$ if and only if $G_{1,j}$ is δ-far from $\mathtt{RS}^{(1)}$ for some $j \in [M]$.
3. If $\mathbb{x} \notin \mathcal{L}$ and $i \in \{1, \dots, k-1\}$, then $(\mathbb{x}, \tau_i) \in \mathcal{D} \iff \mathtt{IncorrectFolding}(i)$ or $G_{i+1,j}$ is δ-far from $\mathtt{RS}^{(i+1)}$ for some $j \in [M]$.
4. If $\mathbb{x} \notin \mathcal{L}$ and $i \in \{0, \dots, r\}$, then $(\mathbb{x}, \tau_{k+i}) \in \mathcal{D} \iff \mathtt{IncorrectFolding}(k+i)$ or F_i is δ-far from $\mathtt{RS}^{(k+i)}$.
5. If $\mathbb{x} \notin \mathcal{L}$, then $(\mathbb{x}, \tau_{k+r+1}) \in \mathcal{D} \iff$ one of the following consistency checks performed by the verifier fails: (1) consistency between $G_{i,j}^*$ and $G_{i+1,j}^*$ for any $j \in [M]$; (2) consistency between F_0^* and $\{G_{k,j}^*\}_{j=1}^{M}$; and (3) consistency between F_i^* and F_{i+1}^*.

Proposition 4 (**Fold-and-Batch round-by-round knowledge soundness**)**.** *Define the language $\mathcal{L} := \mathcal{L}_{RS,\delta}^{M}$ where $\mathcal{L}_{RS,\delta} = \{w \in \mathbb{F}^D \mid \Delta(w, RS) < \delta\}$ is the language of words that are δ-close to RS. Assume the Fold-and-Batch IOP $\Pi_{\texttt{Fold-and-Batch}}$ is run with $\delta < \frac{1-\rho}{2}$ (unique decoding radius), then it is an IOP for $\mathcal{L}$ with round-by-round knowledge error*

$$\epsilon := \max\left\{\frac{|D| \cdot (M-1)}{|\mathbb{F}|}, (1-\delta)^l\right\}$$

Proof. Properties (1) and (2) for round-by-round knowledge soundness follow from the definition of the doomed set $\mathcal{D}$. For property (3), we prove it for the cases where $\mathbb{x} \in \mathcal{L}$ and $\mathbb{x} \notin \mathcal{L}$. For all the rounds when $\mathbb{x} \in \mathcal{L}$ and up to round $k-1$ for $\mathbb{x} \notin \mathcal{L}$, the proof is the same as in Proposition 3. We start the analysis from round $k-1$ for $\mathbb{x} \notin \mathcal{L}$:

- Round $k-1$ to Round k: Suppose $(\mathbb{x}, \tau_{k-1}) \in \mathcal{D}$, then $\texttt{IncorrectFolding}(k-1)$ or G_{k,j_0} is δ-far from $\texttt{RS}^{(k)}$ for some $j_0 \in [M]$. If $\Pr_{\alpha \xleftarrow{\$} \mathbb{F}}\left[(\mathbb{x}, \tau_{k-1} \mid\mid \mathbf{G}_k^* \mid\mid \alpha) \notin \mathcal{D}\right] > \epsilon$, then

$$\Pr_{\alpha \xleftarrow{\$} \mathbb{F}}\left[\texttt{CorrectFolding}(k) \wedge F_0 = \sum_{j=1}^{M} \alpha^{j-1} G_{k,j} \text{ is } \delta\text{-close to } \texttt{RS}^{(k)}\right] > \epsilon.$$

This implies $\texttt{CorrectFolding}(k)$, so G_{k,j_0} is δ-far from $\texttt{RS}^{(k)}$ for some $j_0 \in [M]$. However, the last probability implies

$$\Pr_{\alpha \xleftarrow{\$} \mathbb{F}}\left[F_0 = \sum_{j=1}^{M} \alpha^{j-1} G_{k,j} \text{ is } \delta\text{-close to } \texttt{RS}^{(k)}\right] > \epsilon \geq \frac{|D| \cdot (M-1)}{|\mathbb{F}|},$$

contradicting Lemma 5.10 of [BGK+23].

- Round k to Round $k+1$: Suppose $(\mathbb{x}, \tau_k) \in \mathcal{D}$, then $\texttt{IncorrectFolding}(k)$ or F_0 is δ-far from $\texttt{RS}^{(k)}$. If $\Pr_{y_0 \xleftarrow{\$} \mathbb{F}}\left[(\mathbb{x}, \tau_k \mid\mid F_0^* \mid\mid y_0) \notin \mathcal{D}\right] > \epsilon$, then

$$\Pr_{y_0 \xleftarrow{\$} \mathbb{F}}\left[\texttt{CorrectFolding}(k+1) \ \wedge \ F_1 \text{ is } \delta\text{-close to } \texttt{RS}^{(k+1)}\right] > \epsilon.$$

This implies $\texttt{CorrectFolding}(k)$, so F_0 is δ-far from $\texttt{RS}^{(k)}$. However, the last probability implies

$$\Pr_{y_0 \xleftarrow{\$} \mathbb{F}}\left[F_1 \text{ is } \delta\text{-close to } \texttt{RS}^{(k+1)}\right] > \epsilon \geq \frac{|D|}{|\mathbb{F}|},$$

contradicting Lemma 5.6 of [BGK+23].

- Round $i-1$ to Round i, $k+2 \leq i \leq k+r$: We omit this part as it is the same as the analysis for FRI in the proof of Theorem 5.11 in [BGK+23].

- Round $k+r$ to Round $k+r+1$: Suppose $(\mathbb{x}, \tau_{k+r}) \in \mathcal{D}$, then $\texttt{IncorrectFolding}(k+r)$ or F_r is δ-far from $\texttt{RS}^{(k+r)}$. However, we always have $F_r \in \texttt{RS}^{(k+r)}$, therefore $\texttt{IncorrectFolding}(k+r) = \texttt{True}$. If

$$\Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}}\left[(\mathbb{x}, \tau_{k+r} \mid\mid F_r^* \mid\mid \mathbf{s}) \notin \mathcal{D}\right] > \epsilon,$$

then

$$\Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}}\left[\mathcal{V} \text{ accepts the complete transcript } \tau_{k+r+1}\right] > \epsilon.$$

Suppose an incorrect folding happened during the partial FRI folding phase, so $\texttt{IncorrectFolding}(k) = \texttt{True}$. Let τ_k denote the prefix of τ_{k+r+1} that includes messages up to and including the k-th round, the last round of partial FRI folding. Let τ denote any "honest" extension of τ_k to a complete Parallel FRI transcript. That is, the Parallel FRI prover only behaves honestly after round k. If $\mathcal{V}_{\texttt{partialFRI}}$ denotes the verifier that checks the partial FRI folding proof, then

$$\Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}}\left[\mathcal{V}_{\texttt{partialFRI}} \text{ accepts } \tau_k\right] = \Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}}\left[\mathcal{V}_{\texttt{ParallelFRI}} \text{ accepts } \tau\right]$$

because any inconsistency detected by $\mathcal{V}_{\texttt{ParallelFRI}}$ can only be from the first k rounds. Since $\texttt{IncorrectFolding}(k) = \texttt{True}$, by Proposition 3 we have

$$\Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}}\left[\mathcal{V}_{\texttt{partialFRI}} \text{ accepts } \tau_k\right] = \Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}}\left[\mathcal{V}_{\texttt{ParallelFRI}} \text{ accepts } \tau\right] \leq \epsilon_{\texttt{ParallelFRI}} \leq \epsilon$$

which is a contradiction. Therefore, the incorrect folding happened during the batched FRI phase. Since $\mathbb{x} \notin \mathcal{L}$, some G_{0,j_0} is δ-far from $\mathtt{RS}^{(0)}$. By Lemma 5.4 of [BGK+23], G^*_{k,j_0} must be δ-far from $\mathtt{RS}^{(k)}$. Therefore, the input $\mathbb{x}_{\mathtt{bFRI}}$ to the batched FRI phase is not in $\mathcal{L}_{\mathtt{bFRI}}$. Let $\mathcal{V}_{\mathtt{bFRI}}$ denote the verifier that checks the batched FRI proof. Since

$$\Pr_{\mathbf{s} \xleftarrow{\$} \mathbb{F}} \left[\mathcal{V}_{\mathtt{bFRI}} \text{ accepts}\right] > \epsilon,$$

all the foldings performed up to and including the $(k+r)$-th round during the batched FRI phase must be honest by the round-by-round knowledge soundness of batched FRI. This is again a contradiction. □

5 Implementation and Evaluation

In this section, we present the implementation details of the three DFRI methods and compare their performance. We implemented all three variants of DFRI using 3700+ lines of code in Rust. Our implementation[2] is based on the STARK implementation winterfell. The experiments were run on a Rocky Linux 9 server with 48 cores and 377 GiB of RAM.

All experiments set the circuit size to $N = 2^{25}$. We ran the experiments for all DFRI methods on up to 128 machines. For Fold-and-Batch, we use a folding parameter $K = \frac{T}{4}$, i.e., each worker node folds its local polynomial twice using a folding factor of 2. The Merkle trees used in FRI are instantiated with the BLAKE3[3] hash function with 256-bit outputs. Instead of setting up multiple machines, we simulated all the machines on a single one by running them sequentially. Therefore, the measured prover runtime does not include the time spent on network communications.

We present the measurements on several performance metrics and discuss how the three DFRI methods compare on each metric.

Worker node. In Figure 1, we show the cost of a single worker node during proof generation. All three DFRI methods are perfectly scalable for worker nodes. By increasing the number of machines from 2 to 128, the proving time is at least 61× faster; the memory usage decreases by 63× for all three methods.

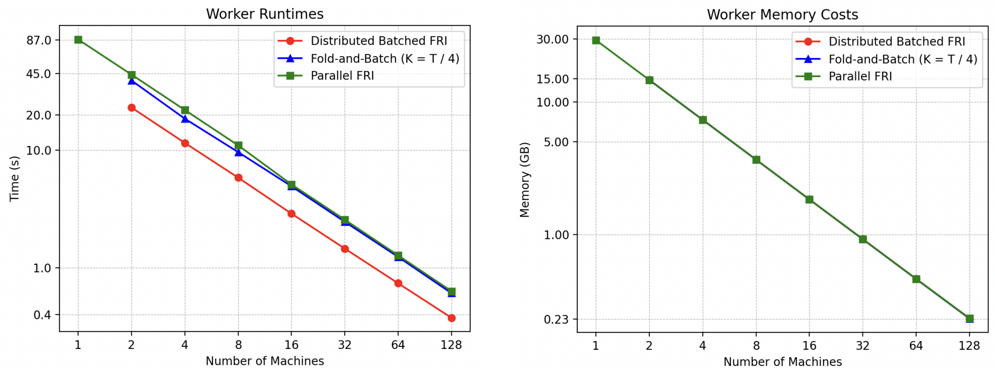


Figure 1: Cost for a single worker node

[2]https://github.com/XuHuaXH/winterfell
[3]https://github.com/BLAKE3-team/BLAKE3-specs

Overall prover cost. Figure 2 shows the cost for provers overall. The *overall memory cost* is the peak memory usage measured at the master node during proof generation. Since the master node also acts as one of the worker nodes, this number represents the maximum memory requirement for a prover node. The data points corresponding to 1 machine show the costs for monolithic FRI. For parallel FRI, there is no master node, and each prover node does the same amount of work. Hence, the overall prover cost is the same as that of a single worker node. For Distributed Batched FRI and Fold-and-Batch, the overall prover costs have limited scalability. Specifically, as we increase the number of machines from 1 to 128, the runtime is 7× faster for Distributed Batched FRI and 22× faster for Fold-and-Batch; the memory usage decreased by 7× for Distributed Batched FRI and 27× for Fold-and-Batch.

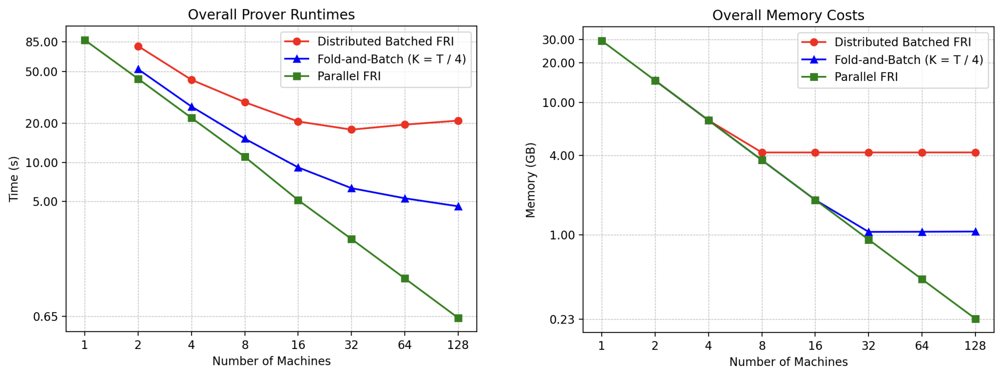
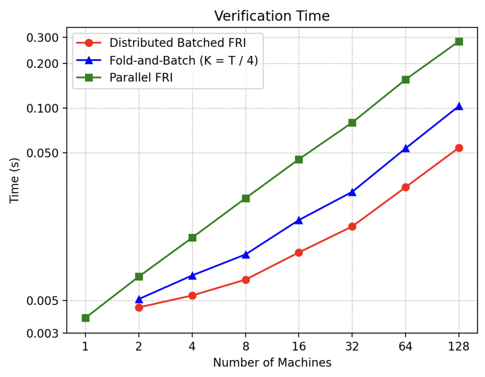


Figure 2: Overall prover cost

Communication cost. The total communication cost is 4.3 GB for Distributed Batched FRI and 1.07 GB for Fold-and-Batch, and it is constant regardless of how many machines are used. For both protocols, these costs come from all the worker nodes sending their evaluation vectors to the master node right before running batched FRI. However, the cost for Fold-and-Batch is about four times smaller than Distributed Batched FRI. This is because, by setting $K := \frac{T}{4}$, the witness data sent in Fold-and-Batch is four times smaller. Assuming a network bandwidth of 25 Gbps, the communication adds 1.38s and 0.34s to the overall proving time for Distributed Batched FRI and Fold-and-Batch, respectively.

Verification cost. The verification cost is shown in Figure 3. For all three `DFRI` methods, the verification time and proof size increase with the number of machines. As we increase from 1 node to 128 nodes, the verification time is 11× slower for Distributed Batched FRI, 22× slower for Fold-and-Batch, and 74× slower for Parallel FRI. The proof size is 7× larger for Distributed Batched FRI, 20× larger for Fold-and-Batch and 60× larger for Parallel FRI.

Among the three `DFRI` methods, Parallel FRI has the smallest and perfectly scalable prover cost while having the most expensive verifier cost. Distributed Batched FRI has the most expensive and least scalable prover while having the smallest verifier cost. For Fold-and-Batch with a folding parameter $K = \frac{T}{4}$, its prover and verifier costs are both in between the two other methods. In conclusion, there is a prover-verifier trade-off for `DFRI`, with Parallel FRI being the most prover-friendly, Distributed Batched FRI being the most verifier-friendly, and Fold-and-Batch in between.



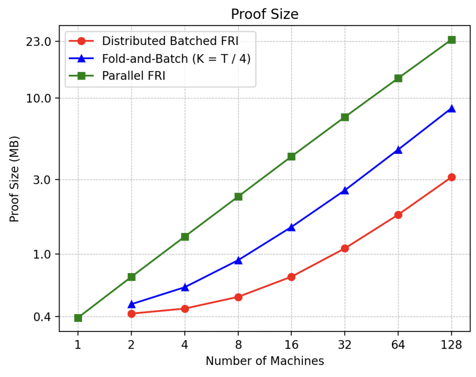


Figure 3: Verification cost

We also summarize the two performance bottlenecks observed from the experiments:

- **Master node scalability:** For Distributed Batched FRI and Fold-and-Batch, the master node has limited scalability. This is because the master node must handle witness data of total size $O(N)$ sent from all the worker nodes.

- **Verifier cost:** The verifier cost increases with the number of machines for all three `DFRI` methods. In the worst case (i.e., Parallel FRI), the cost increases linearly with M. This is in contrast to *Pianist*, which has $O(1)$ proof size and verification time. There are two reasons for *Pianist*'s efficient verification cost: (1) The KZG polynomial commitment scheme has an $O(1)$ proof size and (2) since group exponentiation is homomorphic, in the DKZG polynomial commitment scheme, the master node can combine the commitments/partial proofs sent from the worker nodes into a single group element. On the other hand, FRI has a poly-logarithmic proof size, and the Merkle tree commitment scheme used in FRI lacks homomorphism.

References

[0xP21] 0xPolygonZero. Plonky2, 2021. URL: https://github.com/0xPolygonZero/plonky2/blob/main/plonky2/plonky2.pdf.

[BBHR17] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. *Electron. Colloquium Comput. Complex.*, TR17-134, 2017. URL: https://eccc.weizmann.ac.il/report/2017/134, arXiv:TR17-134.

[BCI+20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. In *61st Annual Symposium on Foundations of Computer Science*, pages 900–909, Durham, NC, USA, November 16–19, 2020. IEEE Computer Society Press. doi:10.1109/FOCS46700.2020.00088.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 677–706, Cham, 2020. Springer International Publishing.

[BGK+23] Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michał Zając. Fiat-shamir security of FRI and related SNARKs. Cryptology ePrint Archive, Paper 2023/1071, 2023. URL: https://eprint.iacr.org/2023/1071.

[BSCG+14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014. doi:10.1109/SP.2014.36.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography - Volume 9986*, page 31–60, Berlin, Heidelberg, 2016. Springer-Verlag. doi:10.1007/978-3-662-53644-5_2.

[CBBZ23] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 499–530, Cham, 2023. Springer Nature Switzerland.

[Che24] Alisa Cherniaeva. On distributed fri-based proof generation - hackmd, 10 2024. URL: https://hackmd.io/@nil-research/rJ_NVyiRA.

[CHP+23] Santiago Cuéllar, Bill Harris, James Parker, Stuart Pernsteiner, and Eran Tromer. Cheesecloth: Zero-Knowledge proofs of real world vulnerabilities. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6525–6540, Anaheim, CA, August 2023. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/cuellar.

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In *Theory of Cryptography: 17th International Conference, TCC 2019, Nuremberg, Germany, December 1–5, 2019, Proceedings, Part II*, page 1–29, Berlin, Heidelberg, 2019. Springer-Verlag. doi:10.1007/978-3-030-36033-7_1.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 09 2024. URL: https://hash-based-snargs-book.github.io/.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[GHAH+23] Matthew Green, Mathias Hall-Andersen, Eric Hennenfent, Gabriel Kaptchuk, Benjamin Perez, and Gijs Van Laer. Efficient proofs of software exploitability for real-world processors. *Proceedings on Privacy Enhancing Technologies*, 2023(1):627–640, January 2023. doi:10.56553/popets-2023-0036.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326, Vienna, Austria, May 8–12, 2016. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_11.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019. URL: https://eprint.iacr.org/2019/953.

[KPPS20] Ahmed Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, and Dawn Song. Mirage: succinct arguments for randomized algorithms with applications to universal zk-snarks. In *Proceedings of the 29th USENIX Conference on Security Symposium*, SEC'20, USA, 2020. USENIX Association.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-17373-8_11.

[Lee21] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II*, page 1–34, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-90453-1_1.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. Cryptology ePrint Archive, Paper 2001/107, 2001. URL: https://eprint.iacr.org/2001/107.

[LXZ+24] Tianyi Liu, Tiancheng Xie, Jiaheng Zhang, Dawn Song, and Yupeng Zhang. Pianist: Scalable zkRollups via fully distributed zero-knowledge proofs. In *2024 IEEE Symposium on Security and Privacy*, pages 1777–1793, San Francisco, CA, USA, May 19–23, 2024. IEEE Computer Society Press. doi:10.1109/SP54263.2024.00035.

[LZL+25a] Chongrong Li, Pengfei Zhu, Yun Li, Cheng Hong, Wenjie Qu, and Jiaheng Zhang. HyperPianist: Pianist with Linear-Time Prover and Logarithmic Communication Cost . In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 3383–3401, Los Alamitos, CA, USA, May 2025. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00202, doi:10.1109/SP61157.2025.00202.

[LZL+25b] Weihan Li, Zongyang Zhang, Yun Li, Pengfei Zhu, Cheng Hong, and Jianwei Liu. Soloist: Distributed SNARKs for rank-one constraint system. Cryptology ePrint Archive, Paper 2025/557, 2025. URL: https://eprint.iacr.org/2025/557.

[PV07] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. An efficient parallel repetition theorem for arthur-merlin games. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 420–429, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1250790.1250853.

[RMH+24] Michael Rosenberg, Tushar Mopuri, Hossein Hafezi, Ian Miers, and Pratyush Mishra. Hekaton: Horizontally-scalable zksnarks via proof aggregation. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 929–940, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3658644.3690282.

[VP19] Alexander Vlasov and Konstantin Panarin. Transparent polynomial commitment scheme with polylogarithmic communication complexity. Cryptology ePrint Archive, Paper 2019/1020, 2019. URL: https://eprint.iacr.org/2019/1020.

[WSVZ24] Wenhao Wang, Fangyan Shi, Dani Vilardell, and Fan Zhang. Cirrus: Performant and accountable distributed SNARK. Cryptology ePrint Archive, Paper 2024/1873, 2024. URL: https://eprint.iacr.org/2024/1873.

[WZC+18] Howard Wu, Wenting Zheng, Alessandro Chiesa, Raluca Ada Popa, and Ion Stoica. DIZK: A distributed zero knowledge proof system. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 675–692, Baltimore, MD, August 2018. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/wu.

[XZC+22] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 3003–3017, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3548606.3560652.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 859–876, 2020. doi:10.1109/SP40000.2020.00052.

A Backgrounds on FRI

Let $f(X) = f_0 + f_1 X + \cdots + f_{T-1} X^{T-1} \in \mathbb{F}[X]$. The FRI folding phase consists of the following steps:

1. $\mathcal{P}$ sends $\mathcal{V}$ the commitment of f, $\texttt{MT.Commit}(f)$.

2. $\mathcal{V}$ sends $\mathcal{P}$ a random folding factor $r_1 \xleftarrow{\$} \mathbb{F}$.

3. $\mathcal{P}$ splits $f = f^{(0)}$ into an even part and an odd part by writing it as

$$\begin{aligned} f^{(0)}(X) &= \underbrace{(f_0 + f_2 X^2 + \cdots + f_{T-2} X^{T-2})}_{\text{even terms}} + \underbrace{(f_1 X + f_3 X^3 + \cdots + f_{T-1} X^{T-1})}_{\text{odd terms}} \\ &= f^{(0)}_{\text{even}}(X^2) + X \cdot f^{(0)}_{\text{odd}}(X^2) \end{aligned}$$

   so

$$\begin{aligned} f^{(0)}_{\text{even}}(X) &= f_0 + f_2 X + f_4 X^2 + \cdots + f_{T-2} X^{\frac{T-2}{2}}, \\ f^{(0)}_{\text{odd}}(X) &= f_1 + f_3 X + f_5 X^2 + \cdots + f_{T-1} X^{\frac{T-2}{2}} \end{aligned}$$

   $\mathcal{P}$ combines $f^{(0)}_{\text{even}}(X)$ and $f^{(0)}_{\text{odd}}(X)$ using r_1 and obtains

$$f^{(1)}(X) = f^{(0)}_{\text{even}}(X) + r_1 \cdot f^{(0)}_{\text{odd}}(X)$$

   $\mathcal{P}$ builds a Merkle tree on the evaluation vector $(f^{(1)}(x))_{x \in D^{(1)}}$ where $D^{(1)} := (D^{(0)})^2 = \{x^2 \mid x \in D^{(0)}\} = \mu_{\frac{n}{2}}$ and sends the Merkle root to $\mathcal{V}$. The new polynomial $f^{(1)}(X)$ has degree about half of $\deg f^{(0)}$ and $D^{(1)}$ is half the size of $D = D^{(0)}$.

4. $\mathcal{V}$ sends $\mathcal{P}$ a random folding factor $r_2 \xleftarrow{\$} \mathbb{F}$.

5. $\mathcal{P}$ repeats the same process as in Step 3: $\mathcal{P}$ write $f^{(1)}$ as

$$f^{(1)}(X) = f^{(1)}_{\text{even}}(X^2) + X \cdot f^{(1)}_{\text{odd}}(X^2)$$

   and obtains

$$f^{(2)}(X) = f^{(1)}_{\text{even}}(X) + r_2 \cdot f^{(1)}_{\text{odd}}(X)$$

   $\mathcal{P}$ sends $\mathcal{V}$ the Merkle commitment of $f^{(2)}$ evaluated on $D^{(2)} := (D^{(1)})^2 = \mu_{\frac{n}{4}}$.

6. $\cdots$

This process repeats for a total of $\log T$ rounds. In the last round, f has been folded to a constant polynomial $f^{(\log T)}(X) = c$, so $\mathcal{P}$ sends $\mathcal{V}$ the constant c instead of a Merkle commitment. This concludes the folding phase.

Alternatively, one can view the folding phase as folding the evaluation vector $(f(x))_{x \in D}$. Recall that in the i-th round of folding, $\mathcal{P}$ writes $f^{(i)}(X)$ as

$$f^{(i)}(X) = f^{(i)}_{\text{even}}(X^2) + X \cdot f^{(i)}_{\text{odd}}(X^2)$$

Thus,

$$f^{(i)}(-X) = f^{(i)}_{\text{even}}(X^2) - X \cdot f^{(i)}_{\text{odd}}(X^2)$$

Combining these two equations, we get

$$f^{(i)}_{\text{even}}(X^2) = \frac{f^{(i)}(X) + f^{(i)}(-X)}{2}$$
$$f^{(i)}_{\text{odd}}(X^2) = \frac{f^{(i)}(X) - f^{(i)}(-X)}{2X}$$

Therefore, $f^{(i+1)}(X^2)$ can be expressed as

$$\begin{aligned} f^{(i+1)}(X^2) &\overset{\text{def}}{=} f^{(i)}_{\text{even}}(X^2) + r_{i+1} \cdot f^{(i)}_{\text{odd}}(X^2) \\ &= \frac{f^{(i)}(X) + f^{(i)}(-X)}{2} + r_{i+1} \cdot \frac{f^{(i)}(X) - f^{(i)}(-X)}{2X} \\ &= \frac{X + r_{i+1}}{2X} f^{(i)}(X) + \frac{X - r_{i+1}}{2X} f^{(i)}(-X) \end{aligned}$$

This means, if $\mathcal{P}$ already has the evaluation vector $(f^{(i)}(x))_{x \in D^{(i)}}$ for the i-th round, $\mathcal{P}$ can compute the evaluation vector $(f^{(i+1)}(y))_{y \in D^{(i+1)}}$ for the $(i+1)$-th round as follows: For any $y \in D^{(i+1)} = (D^{(i)})^2$, find x such that $x^2 = y$. Then $x \in D^{(i)}$ and $f^{(i+1)}(y)$ is a linear combination of $f^{(i)}(x)$ and $f^{(i)}(-x)$:

$$f^{(i+1)}(y) = \frac{x + r_{i+1}}{2x} f^{(i)}(x) + \frac{x - r_{i+1}}{2x} f^{(i)}(-x) \tag{2}$$

After the folding phase, $\mathcal{P}$ and $\mathcal{V}$ run the query phase of FRI. The purpose of the query phase is for $\mathcal{V}$ to check whether $\mathcal{P}$ has folded f correctly by "quizzing" $\mathcal{P}$ at various points $g \in D$. The query phase proceeds as follows: For each $g \overset{\$}{\leftarrow} D$ that $\mathcal{V}$ queries,

1. $\mathcal{V}$ sends $\mathcal{P}$ the query point g.
2. $\mathcal{P}$ sends $\mathcal{V}$ the evaluations

$$\begin{aligned} &f^{(0)}(g), f^{(0)}(-g), \\ &f^{(1)}(g^2), f^{(1)}(-g^2), \\ &f^{(2)}(g^4), f^{(2)}(-g^4), \\ &\dots \\ &f^{(\log T)}(g^T), f^{(\log(T))}(-g^T) \end{aligned}$$

   together with their Merkle authentication paths.
3. $\mathcal{V}$ checks that
   (a) All the Merkle authentication paths are indeed valid opening proofs for their corresponding evaluations.
   (b) For each $1 \le i \le \log T$, $f^{(i)}(g^{2^i})$ is indeed computed correctly using Formula 2, i.e. that

$$\begin{aligned} f^{(1)}(g^2) &= \frac{g + r_1}{2g} f^{(0)}(g) + \frac{g - r_1}{2g} f^{(0)}(-g) \\ f^{(2)}(g^4) &= \frac{g^2 + r_2}{2g^2} f^{(1)}(g^2) + \frac{g^2 - r_2}{2g^2} f^{(1)}(-g^2) \\ &\dots \end{aligned}$$

If both conditions are satisfied, $\mathcal{V}$ outputs `accept`. Otherwise, $\mathcal{V}$ outputs `reject`.

Soundness of FRI One can estimate the number of queries needed in the FRI query phase in order to guarantee λ bits of security using Theorem 3.3 from [BBHR17]: If the function $f : D \to \mathbb{F}$ is δ-far from $\texttt{RS}$, i.e.

$$\frac{|\{x \in D \mid f(x) \neq F(x)\}|}{|D|} \geq \delta$$

for any $F \in \texttt{RS}$ and $\mathcal{V}$ sent l queries during the FRI query phase, then

$$\Pr[\mathcal{V} \text{ outputs } \texttt{accept}] \leq \frac{3|D|}{|\mathbb{F}|} + (1-\delta)^l$$

Since the term $\frac{3|D|}{|\mathbb{F}|}$ is negligibly small, we get about λ bits of security once we have

$$(1-\delta)^l < 2^{-\lambda}$$

This means the number of queries l should be at least $-\frac{\lambda}{\log_2(1-\delta)}$. For a fixed security parameter λ, choosing a larger δ allows us to run fewer query rounds, which in turn reduces the proof size and verification time. However, a larger δ means a larger FRI blowup factor ρ^{-1} (since $\delta < \frac{1-\rho}{2}$), which implies longer proving time because the prover needs to fold a larger polynomial during the FRI folding phase.

B Security Proofs of the bivariate FRI-based PCS

Proposition 5. *$\texttt{PCS}$ is complete if $\Pi_{\texttt{DFRI}}$ is complete.*

Proof. Let F be a bivariate polynomial of the form

$$F(X,Y) = \sum_{i=0}^{M-1} F_i(X)R_i(Y)$$

with $\deg F_i < T$ and let $(\alpha, \beta) \in \mathbb{F}^2$ with $\beta \notin \mu_M$. Suppose we have $(s_i)_{i=0}^{M-2} \leftarrow \texttt{Precomputation}(M)$, $\mathbf{pp} \leftarrow \texttt{Setup}(1^\lambda, (s_i)_{i=0}^{M-2})$, $\mathcal{C} \leftarrow \texttt{Commit}(F, \mathbf{pp})$ and $(z, (z_i)_{i=0}^{M-1}, \pi) \leftarrow \texttt{Open}(F, \alpha, \beta, \mathbf{pp})$.

The inputs to $\texttt{DFRI.Prove}$ to generate π, $(Q_i(X))_i$, are polynomials of degree $< T$. By completeness of $\Pi_{\texttt{DFRI}}$, the first check in $\texttt{Verify}$ will pass.

Since $F(\alpha, \beta) = z$, $Y - \beta \mid F(\alpha, Y) - z$. Therefore, $\widetilde{H}(Y) := \frac{F(\alpha,Y)-z}{Y-\beta}$ is a polynomial in $\mathbb{F}[Y]$ of degree $< M-1$, with

$$\widetilde{H}(\omega_Y^i) = \frac{F(\alpha, \omega_Y^i) - z}{\omega_Y^i - \beta} = \frac{z_i - z}{\omega_Y^i - \beta} = h_i$$

for $i = 0, \ldots, M-2$. Consider the polynomial

$$H(Y) := \sum_{i=0}^{M-2} h_i \cdot S_i(Y)$$

of degree $< M-1$. Clearly, $H(\omega_Y^i) = h_i$ for $i = 0, \ldots, M-2$. $\widetilde{H}$ and H are two polynomials of degree $< M-1$ and share the same evaluations at $M-1$ points, so must be equal.

This means

$$\begin{aligned}
&z_{M-1} - z - \sum_{i=0}^{M-2} h_i \cdot S_i(\omega_Y^{M-1})(\omega_Y^{M-1} - \beta) \\
&= z_{M-1} - z - H(\omega_Y^{M-1})(\omega_Y^{M-1} - \beta) \\
&= z_{M-1} - z - \widetilde{H}(\omega_Y^{M-1})(\omega_Y^{M-1} - \beta) \\
&= z_{M-1} - z - \frac{F(\alpha, \omega_Y^{M-1}) - z}{\omega_Y^{M-1} - \beta}(\omega_Y^{M-1} - \beta) \\
&= z_{M-1} - z - (F(\alpha, \omega_Y^{M-1}) - z) \\
&= 0
\end{aligned}$$

If the opening point (α, β) sampled by the verifier happens to be one such that $\beta \in \mu_M$, then the protocol aborts. This happens with probability $\frac{M}{|\mathbb{F}|}$ which is $\mathtt{negl}(\lambda)$ because $\mathbb{F}$ has size at least super-polynomial in λ. □

Proposition 6. *PCS satisfies polynomial binding.*

Proof. Suppose $\mathcal{A}$ is an efficient adversary that can output a tuple $(\mathcal{C}, F, G)$ such that $\mathtt{VerifyPoly}(\mathcal{C}, F, \mathbf{pp}) = 1$ and $\mathtt{VerifyPoly}(\mathcal{C}, G, \mathbf{pp}) = 1$ and $F \neq G$ with non-negligible probability. For any such tuple, let $\mathcal{C}_0, \ldots, \mathcal{C}_{M-1}$ be the Merkle commitments in $\mathcal{C}$. Since both openings successfully verify, we must have $F, G \in \mathcal{F}$ and $\mathtt{MT.Commit}(F_i) = \mathcal{C}_i = \mathtt{MT.Commit}(G_i)$ for all i. Since $F_i \neq G_i$ for some i, $((F_i(x))_{x \in D}) \neq ((G_i(x))_{x \in D})$ as vectors. Thus, there is an efficient algorithm $\mathcal{B}$ using $\mathcal{A}$ as a subroutine to output a hash collision for the collision-resistant hash function $\mathcal{H}$ with non-negligible probability. This is a contradiction. □

Proposition 7. *PCS satisfies evaluation binding.*

Proof. Suppose we have two sets of inputs satisfying

$$\begin{aligned}
&\mathtt{Verify}(\mathcal{C}, \alpha, \beta, z, (z_i)_{i=0}^{M-1}, \pi, \mathbf{pp}) = 1 \wedge \\
&\mathtt{Verify}(\mathcal{C}, \alpha, \beta, z', (z'_i)_{i=0}^{M-1}, \pi', \mathbf{pp}) = 1
\end{aligned}$$

Since $\mathtt{DFRI.Verify}(\pi) = 1$ and we are in the unique decoding regime, with high probability there are unique polynomials $P_i(X) \in \mathbb{F}[X]_{<T}$ that are δ-close to $q_i(X) = \frac{f_i(X) - z_i}{x - \alpha}$, where f_i's are the functions committed in $\mathcal{C}$. Let $F_i(X) := \mathtt{RS.Decode}(f_i)$ and let $F(X, Y) := \sum_{i=0}^{M-1} F_i(X) R_i(Y)$. By Lemma 1, we have $F(\alpha, \beta) = z$.

Since $\mathtt{DFRI.Verify}(\pi') = 1$, with high probability there are unique polynomials $P'_i(X) \in \mathbb{F}[X]_{<T}$ that are δ-close to $q'_i(X) = \frac{f_i(X) - z'_i}{X - \alpha}$, so the polynomials $G'_i(X) := P'_i(X)(X - \alpha) + z'_i$ are δ-close to f_i's. By Lemma 1 again, $F(\alpha, \beta) = z'$. This shows $z = z'$. □

C Cost analysis of DFRI instantiations

In Tables 3 and 4 we present a step-by-step cost analysis of Distributed Batched FRI and of Fold-and-Batch, respectively.

Table 3: Distributed Batched FRI step-by-step cost analysis.

P_0 Runtime	P_i Runtime	Total comm. between P_0 and P_i	Protocol steps
$O(M)$	$O(T)$	$O(M)$	1. P_i builds the Merkle tree from its evaluation vector $(F_i(x))_{x\in D}$ and sends the commitment to P_0. P_0 sends these commitments to V.
$O(M)$	$O(1)$	$O(M)$	2. P_0 receives a random challenge θ from V and sends it to P_i.
$O(N)$	$O(T)$	$O(N)$	3. P_i computes the new evaluation vector $(\theta^i F_i(x))_{x\in D}$ and sends it to P_0.
$O(N)$	—	—	4. P_0 computes the evaluation vector $(F(x))_{x\in D}$ where $F(X) = \sum_i \theta^i F_i(X)$.
$O(T)$ for folding and $O(\log^2 T)$ for querying	—	—	5. P_0 and V run `FRI` with $F(X)$.
$O(M \log T)$	$O(\log T)$	$O(M \log T)$	6. Extra consistency checks for batched `FRI`: For each queried $g \in D$, P_i sends $F_i(g)$ and its Merkle authentication path to P_0. P_0 sends all received data to V. V checks that the Merkle authentication paths are valid and $F(g) = \sum_{i=0}^{M-1} \theta^i F_i(g)$.
$O(N)$	$O(T)$	$O(N)$	**Total**

Table 4: Fold-and-Batch step-by-step cost analysis.

P_0	P_i	Total Communication	V	Protocol steps
$O(M)$	$O(T)$	$O(M)$	$O(M)$	Step 1
—	$O(T-K)$	—	—	Step 2
$O(M \log \frac{T}{K})$	$O(\log \frac{T}{K})$	$O(M \log \frac{T}{K})$	$O(M \log \frac{T}{K})$ from the $\log \frac{T}{K}$ Merkle commitments of intermediate folded polynomials from M parties.	Step 3
$O(M)$	$O(1)$	$O(M)$	—	Step 4
$O(MK)$	$O(K)$	$O(MK)$	—	Step 5
$O(MK)$	—	—	—	Step 6
$O(K)$	—	—	$O(\log K)$	Step 7
$O(M \log \frac{T}{K} \log T)$	$O(\log \frac{T}{K} \log T)$	$O(M \log \frac{T}{K} \log T)$	$O(M \log \frac{T}{K} \log T)$	Step 8
$O(\log^2 K)$	—	—	$O(\log^2 K)$	Step 9
—	—	—	$O(M)$ for (c)	Step 10
$O(MK + M \log \frac{T}{K} \log T + \log^2 K)$	$O(T)$	$O(MK + M \log \frac{T}{K} \log T)$	$O(M \log \frac{T}{K} \log T + \log^2 K)$	**Total**

D Parallel Execution of IOPs

We formally define the parallel execution of multiple IOPs. Then, Parallel FRI can be viewed as the parallel execution of M different instances of FRI, with input $\mathbb{x} = (\mathbb{x}_1, \ldots, \mathbb{x}_M)$ where each $\mathbb{x}_j$ is an input for FRI. Note that this is different from parallel repetition (e.g., as in [PV07]), where the inputs to all the parallel instances are the same. We use $[M]$ to denote the set $\{1, \ldots, M\}$.

Definition 9 (Parallel Execution of IOPs)**.** For each $j \in [M]$, let $\Pi_j = (\mathcal{P}_j, \mathcal{V}_j)$ be a t-round IOP for a relation $\mathcal{R}_j$ with δ_j-completeness and ϵ_j-soundness. The *parallel execution of* Π_j*'s* is the t-round interactive protocol $\Pi = (\mathcal{P}, \mathcal{V})$ where the prover $\mathcal{P}$ runs $\mathcal{P}_j$'s in parallel, i.e., if $\mathcal{P}_j$ sends $m_{i,j}$ during round i, then $\mathcal{P}$ sends $m_i := (m_{i,1}, \ldots, m_{i,M})$ during round i. Similarly, $\mathcal{V}$ runs $\mathcal{V}_j$'s in parallel. Let p_j be the j-th "projection" function that takes as input a round-i transcript τ_i for Π and outputs a round-i transcript $\tau_{i,j}$ for Π_j as follows: if

$$\tau_i = (m_{1,1}, \ldots, m_{1,M}) \mid\mid (c_{1,1}, \ldots, c_{1,M}) \mid\mid \ldots \mid\mid (m_{i,1}, \ldots, m_{i,M}) \mid\mid (c_{i,1}, \ldots, c_{i,M})$$

then

$$\tau_{i,j} := p_j(\tau_i) = m_{1,j} \mid\mid c_{1,j} \mid\mid \ldots \mid\mid m_{i,j} \mid\mid c_{i,j}$$

If τ is a complete transcript for Π, then $\mathcal{V}$ accepts τ if and only if $\mathcal{V}_j$ accepts $p_j(\tau)$ for all $j \in [M]$.

Proposition 8. *The parallel execution of* Π_j*'s,* Π*, is a* t*-round IOP for the relation*

$$\mathcal{R} := \big\{\big(\mathbb{x} = (\mathbb{x}_1, \ldots, \mathbb{x}_M), \mathbb{w} = (\mathbb{w}_1, \ldots, \mathbb{w}_M)\big)\big|(\mathbb{x}_j, \mathbb{w}_j) \in \mathcal{R}_j \text{ for } j \in [M]\big\}$$

with δ*-completeness and* ϵ*-soundness where* $\delta = \prod_{j=1}^{M} \delta_j$ *and* $\epsilon = \max_{1 \le j \le M} \epsilon_j$*.*

Proof. δ-Completeness: Let $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$. Since $\mathcal{V}$ accepts if and only if all $\mathcal{V}_j$'s accept and the $\mathcal{V}_j$'s have independent random coins,

$$\begin{aligned}
&\Pr\big[\ \langle \mathcal{P}(\mathbb{w}), \mathcal{V}\rangle(\mathbb{x}) = 1\ \big] \\
&= \Pr\Big[\ \bigwedge_{j=1}^{M} \langle \mathcal{P}_j(\mathbb{w}_j), \mathcal{V}_j\rangle(\mathbb{x}_j) = 1\ \Big] \\
&= \prod_{j=1}^{M} \Pr\big[\ \langle \mathcal{P}_j(\mathbb{w}_j), \mathcal{V}_j\rangle(\mathbb{x}_j) = 1\ \big] \\
&\ge \prod_{j=1}^{M} \delta_j = \delta
\end{aligned}$$

ϵ-Soundness: Let $\mathbb{x} = (\mathbb{x}_1, \ldots, \mathbb{x}_M) \notin \mathcal{L}_{\mathcal{R}}$, then some $\mathbb{x}_{j_0} \notin \mathcal{L}_{\mathcal{R}_{j_0}}$. Let $\mathcal{P}^*$ be any unbounded interactive algorithm and let $\mathcal{P}_j^*$ be the corresponding unbounded interactive algorithm for Π_j, i.e., the message sent by $\mathcal{P}_j^*$ during each round is the j-th entry of the message sent by $\mathcal{P}^*$ during that round. For the same reason as above, we have

$$\begin{aligned}
&\Pr\big[\ \langle \mathcal{P}^*, \mathcal{V}\rangle(\mathbb{x}) = 1\ \big] \\
&= \Pr\Big[\ \bigwedge_{j=1}^{M} \langle \mathcal{P}_j^*, \mathcal{V}_j\rangle(\mathbb{x}_j) = 1\ \Big] \\
&= \prod_{j=1}^{M} \Pr\big[\ \langle \mathcal{P}_j^*, \mathcal{V}_j\rangle(\mathbb{x}_j) = 1\ \big] \\
&\le \epsilon_{j_0} \le \max_{1 \le j \le M} \epsilon_j = \epsilon
\end{aligned}$$

□