Fast RS-IOP Multivariate Polynomial Commitments and Verifiable Secret Sharing

Zongyang Zhang, Weihan Li, Yanpei Guo, and Kexin Shi, *Beihang University;* Sherman S. M. Chow, *The Chinese University of Hong Kong;* Ximeng Liu, *Fuzhou University;* Jin Dong, *Beijing Academy of Blockchain and Edge Computing*

https://www.usenix.org/conference/usenixsecurity24/presentation/zhang-zongyang

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

Fast RS-IOP Multivariate Polynomial Commitments and Verifiable Secret Sharing

Zongyang Zhang[*†] Weihan Li[*†] Yanpei Guo[*] Kexin Shi[*]
Sherman S. M. Chow[‡] Ximeng Liu[§] Jin Dong[¶]

Abstract

Supporting proofs of evaluations, polynomial commitment schemes (PCS) are crucial in secure distributed systems. Schemes based on fast *Reed–Solomon interactive oracle proofs* (RS-IOP) of proximity have recently emerged, offering transparent setup, plausible post-quantum security, efficient operations, and, notably, sublinear proof size and verification. Manifesting a new paradigm, PCS with one-to-many proof can enhance the performance of (asynchronous) verifiable secret sharing ((A)VSS), a cornerstone in distributed computing, for proving multiple evaluations to multiple verifiers. Current RS-IOP-based multivariate PCS, including HyperPlonk (Eurocrypt '23) and Virgo (S&P '20), however, only offer quasi-linear prover complexity in the polynomial size.

We propose PolyFRIM, a fast RS-IOP-based multivariate PCS with optimal linear prover complexity, 5-25× faster than prior arts while ensuring competent proof size and verification. Heeding the challenging absence of FFT circuits for multivariate evaluation, PolyFRIM surpasses Zhang et al.'s (Usenix Sec. '22) one-to-many univariate PCS, accelerating proving by 4-7× and verification by 2-4× with 25% shorter proof. Leveraging PolyFRIM, we propose an AVSS scheme FRISS with a better efficiency tradeoff than prior arts from multivariate PCS, including Bingo (Crypto '23) and Haven (FC '21).

1 Introduction

A *polynomial commitment scheme* (PCS) enables a prover to commit to a polynomial f defined over a field $\mathbb{F}$ with degree bound d and variable number μ. When given a point $\boldsymbol{x} \in \mathbb{F}^\mu$, the prover can convince a verifier via an evaluation proof that the committed polynomial f satisfies $f(\boldsymbol{x}) = y$ for a public $y \in \mathbb{F}$. A PCS is *succinct* if the size of the proof/transcript is sublinear in the polynomial size d^μ and is *efficient* if verification time is sublinear in d^μ. Succinct and efficient PCS is integral in recent zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) [16,20,29] via integrating PCS with polynomial interactive oracle proofs (IOPs) [8].

A pioneer PCS of Kate–Zaverucha–Goldberg [24] (*KZG-PCS*) and its multivariate extension [28] feature constant communication and verifier complexity. However, their security relies on a *trusted setup* to generate a structured reference string, which poses challenges in deployment for blockchain or other distributed applications. Bulletproofs [11] only needs a *transparent* setup. Contrarily, it entails a linear number of exponentiations for both the prover and verifier. PCSs [25,32] from *fast Reed–Solomon* (RS) *IOP of proximity* (FRI) [5] stand out among succinct and efficient ones [12,27] for their efficient execution and plausible post-quantum security, gaining popularity in zk-rollup[1] and post-quantum signatures [9].

Adapting the first PCS from FRI [32] (*FRI-PC* henceforth), tailored for univariate polynomials, to multivariate PCS poses challenges without additive homomorphism [26]. HyperPlonk [16, §B] (only work for $d = 2$) and Virgo [37] represent the latest FRI-based multivariate PCSs. Their prover complexity is $O(\mu \cdot d^\mu \log d)$, in contrast to $O(d^\mu)$ FRI-PC (for $\mu = 1$). In Virgo, the prover needs to invoke the Goldwasser–Kalai–Rothblum (GKR) protocol [21] for fast Fourier transformation (FFT) with a statement size of $O(\mu \cdot d^\mu \log d)$, incurring a prover complexity quasi-linear to the polynomial size d^μ. HyperPlonk is simpler without needing the GKR composition.

A PCS with *one-to-many proofs* [31] allows a prover to open distinct evaluations of a single committed polynomial, where each evaluation corresponds to a unique verifier with a lower prover complexity than repeating single proof. Zhang *et al.* (ZXH+22) [36] propose a framework for one-to-many proof of transparent univariate polynomials, including

[*]Beihang Univ. Supported by National Key R&D Program of China (2022YFB2702702), National Natural Science Foundation of China (62372020, 72031001, 62102422), Beijing Natural Science Foundation (L222050), & Fundamental Research Funds for the Central Univ. (YWF-23-L-1032). The first three authors contribute equally. Experiments are led by Yanpei Guo.

[†]Corresponding authors. Work done while Weihan Li is at CUHK.

[‡]The Chinese University of Hong Kong. Supported by GRF 14210621. CUHK authors are grateful for the reviewers' feedback and for entrusting us with final revisions, and hold sole responsibility for any editorial errors.

[§]Fuzhou Univ. [¶]Beijing Academy of Blockchain and Edge Computing.

[1] *e.g.*, github.com/0xPolygonZero/plonky2, ia.cr/2021/582

a polynomial interactive oracle proof and a (univariate) polynomial commitment. Zhang *et al.* instantiate this framework using Virgo, obtaining a one-to-many PCS with a prover complexity of $O(n \log n)$, as well as a verifier and communication complexity of $O(\log^2 n)$, for $n = O(d^\mu)$ produced proofs.

A natural application of one-to-many proofs is *verifiable secret sharing* (VSS) [31, 36] with low dealer complexity. VSS allows a dealer to distribute secret shares to parties so that enough honest parties could either recover the secret when the dealer is honest or identify a cheating dealer. *Asynchronous VSS* (AVSS) [4] operates without timing assumptions, and its resilience [1, 2] is integral for a wide array of applications, including Byzantine agreement [13], distributed key generation [19, 33], and proactive secret sharing [23, 35].

AVSS from PCS [1–3, 34] enjoys strong security guarantees (*e.g.*, termination [18]) and low communication complexity. They roughly fall into univariate or multivariate approaches [18]. The former [34] typically adapts an *encrypt-then-disperse* [18] strategy to remedy any fault due to message delay or malice. Encryption limits plaintext visibility, causing communication redundancy for sending multiple ciphertexts, and efficiency overhead for proving the well-formedness [18]. Utilizing multivariate (precisely, bivariate) PCS to build AVSS [2, 3] can achieve secret recovery and correction almost for free. A typical framework [3] involves the dealer distributing shares $f(x_i, Y)$ of polynomial $f(X, Y)$ to party P_i. P_i then delivers $f(x_i, y_j)$ to P_j, where $i, j \in \{1, 2, \ldots, n\}$. Upon receiving sufficient $f(x_i, y_j)$, P_j computes $f(X, y_j)$ and sends $f(x_k, y_j)$ to P_k, helping it to recover or correct its share $f(x_k, Y)$. One-to-many proofs fit in nicely: a dealer proves n^2 different evaluations of a bivariate polynomial to n^2 verifiers. Each $f(x_i, y_j)$ is given to a "combined" party of P_i and P_j.

AVSS schemes from bivariate PCS [2, 3] typically use PCS directly, leading to $O(n^3)$ dealer complexity for proving n^2 point evaluations on an $O(n)$-degree polynomial, each with $O(n)$ complexity. A recent work Bingo [1] reduces the dealer complexity by overloading every party with partial proof generation. The availability of one-to-many multivariate PCSs provides a potential route to lower dealer complexity. While univariate PCSs with one-to-many proofs [31, 36] exist, multivariate one-to-many proofs remain unknown. Virgo-PCS [37] (used by ZXH$^+$22 [36]) originally supports multivariate polynomials, but it is employed in a framework solely for univariate PCS, leading to only univariate one-to-many proof.

Amidst ample advantages, we tackle two open problems for advancing FRI-based multivariate polynomial commitments:

1. Can we build an FRI-based multivariate PCS with optimal linear proving, further with non-trivial one-to-many proof?
2. How far can we push the frontier of AVSS with the advancements in multivariate polynomial commitment?

1.1 Our Contribution

We solve these questions and bring the contributions below.

• **PolyFRIM: Polynomial Commitment Scheme from Fast Reed–Solomon Interactive Oracle Proofs of Proximity in the Multivariate Setting with Linear Prover Complexity.** Proving a μ-variate d-degree polynomial at least requires its evaluation, taking $O(d^\mu)$. We propose PolyFRIM with such optimal prover complexity. Its verifier and communication take $O(\mu^2 \log^2 d)$, living up to others [16, 37] in Table 1.

• **One-to-Many PolyFRIM.** Our new PCS design supports multivariate one-to-many proof, surpassing ZXH$^+$22 [36] in functionality. For $O(n^2)$ proofs on any $O(n)$-degree bivariate polynomial, the prover time is $O(n^2 \log n)$, which is optimal.[2] Per-verifier computation and communication costs stay at $O(\log^2 n)$ as the non-amortized case. Details are in Table 2.

• **FRISS: FRI AVSS from PolyFRIM.** FRISS, our new AVSS enabled by PolyFRIM (*cf.* [3]), exhibits a dealer complexity of $O(n^2 \log n)$ for party number n, the same as Bingo [1]. eAVSS [3] and HAVEN-1/2 [2] take $O(n^3)$. See Table 3.[3]

• **Implementation and Evaluation.** In terms of performance, we fully implement our schemes[4], which set new benchmarks. Our experiments show that PolyFRIM has 5-25× faster prover time than other transparent PCSs [11, 16, 37], with at most <30% (resp., <10%) worse verifier time (resp., proof size) compared with FRI-based PCSs. As there have been no one-to-many bivariate PCSs before, we only compare the performance of univariate one-to-many PCSs [36]. Ours reduces the prover time by 4-7× and improves the verifier time and proof size by 2-4× and 25%, respectively. For AVSS, FRISS has a 4-400× advantage for dealers on the cost of PCSs over existing schemes such as eAVSS [3] and HAVEN [2], and a 10× faster party computation than Bingo [1].

1.2 Technical Overview

Contribution-I: PolyFRIM from HyperPlonk. Succinctly, the core of HyperPlonk [16, §B] is that a μ-variate multilinear polynomial $\tilde{f}(X_1, X_2, \ldots, X_\mu)$ evaluated on $(x_1, x_2, \ldots, x_\mu)$ is the inner product between the coefficient vector of $\tilde{f}$ and the tensor product $(1, x_1) \otimes \cdots \otimes (1, x_\mu)$. HyperPlonk primarily involves committing μ univariate polynomials $\{\hat{f}_i\}_{i \in [0, \mu-1]}$ and running the FRI low-degree test [5] for μ times to check whether $\hat{f}_i$ have respective upper degree bounds $2^{\mu-i}$.

Motivated by the demand for multivariate polynomial commitments in AVSS, we generalize HyperPlonk to cover multivariate polynomials by our dedicated transformation. This stems from the observation that a μ-variable polynomial $\tilde{f}(X_1, \ldots, X_\mu)$ with variable degree d evaluated on $(x_1, \ldots, x_\mu)$ is equivalent to a multilinear polynomial $\tilde{g}(X_1, \ldots, X_{\mu \log d})$

[2] The prover needs to compute $O(n)$ univariate polynomials after determining the first variable with n values, taking $O(n^2)$ in total. Then, the FFT for evaluating these $O(n)$ polynomials takes $O(n) \cdot O(n \log n) = O(n^2 \log n)$.

[3] Our paper only instantiates one-to-many PolyFRIM with eAVSS. Future work includes using our (univariate and multivariate one-to-many) schemes to improve other AVSS schemes such as hbACSS and HAVEN.

[4] github.com/gyp2847399255/PolyFRIM

Table 1: Computation and communication complexities of μ-variate degree-d or degree-2 polynomial commitments

Scheme	Trustless Setup	Assumption	Commit	Prover	Verifier	Communication
PST13 [28]	no	q-SDH	$O(d^\mu)\,\mathbb{G}_B$	$O(d^\mu)\,\mathbb{G}_B$	$O(\mu)\,\mathbb{G}_B$	$O(\mu)\,\mathbb{G}_B$
DARK [12]	yes/no	Strong RSA	$O(d^\mu)\,\mathbb{G}_U$	$O(\mu d^\mu \log d)\,\mathbb{G}_U$	$O(\mu \log d)\,\mathbb{G}_U$	$O(\mu \log d)\,\mathbb{G}_U$
Bulletproofs [11]	yes	DLog	$O(d^\mu)\,\mathbb{G}_P$	$O(d^\mu)\,\mathbb{G}_P$	$O(d^\mu)\,\mathbb{G}_P$	$O(\mu \log d)\,\mathbb{G}_P$
Virgo [37]	yes	$O(\mu d^\mu \log d)\,\mathbb{F} + O(d^\mu)\,\mathsf{H}$		$O(\mu d^\mu \log d)\,\mathbb{F}$	$O(\mu^2 \log^2 d)\,\mathsf{H}$	$O(\mu^2 \log^2 d)\,\mathsf{H}$
HyperPlonk [16, §B]	yes	$O(\mu 2^\mu \log 2)\,\mathbb{F} + O(2^\mu)\,\mathsf{H}$		$O(\mu 2^\mu \log 2)\,\mathbb{F}$	$O(\mu^2 \log^2 2)\,\mathsf{H}$	$O(\mu^2 \log^2 2)\,\mathsf{H}$
PolyFRIM	yes	$O(\mu d^\mu \log d)\,\mathbb{F} + O(d^\mu)\,\mathsf{H}$		$O(d^\mu)\,\mathsf{H} + O(d^\mu)\,\mathbb{F}$	$O(\mu^2 \log^2 d)\,\mathsf{H}$	$O(\mu^2 \log^2 d)\,\mathsf{H}$

$\mathbb{G}_i$, $i \in \{B,U,P\}$, denotes a group with a *B*ilinear map, of *U*nknown order, or of known *P*rime order; $\mathbb{F}$ is a field with a large multiplicative coset. H denotes a hash function. All these represent the size or operation time depending on the context.

Table 2: One-to-many proofs for n^2 evaluations of degree-n *Uni*variate or *Bi*variate polynomial

Scheme	Trustless	PCS	Prover	Verifier/Commun.
AMT [31]	no	Uni.	$O(n^2 \log n)$	$O(\log n)$
ZXH+22 [36]	no	Uni.	$O(n^2 \log n)$	$O(1)$
ZXH+22 [36]	yes	Uni.	$O(n^2 \log n)$	$O(\log^2 n)$
Naïve [16, 37]	yes	Bi.	$O(n^3 \log n)$	$O(\log^2 n)$
PolyFRIM	yes	Bi.	$O(n^2 \log n)$	$O(\log^2 n)$

Table 3: Existing n-party AVSS from multivariate PCS

Scheme	PCS	Dealer	Party	Commun.
eAVSS [3]	q-SDH	$O(n^3)$	$O(n)$	$O(n^2)$
HAVEN-1 [2]	q-SDH	$O(n^3)$	$O(n)$	$O(n^2)$
Bingo [1]	q-SDH	$O(n^2 \log n)$	$O(n^2)$	$O(n^2)$
HAVEN-2 [2]	DLog	$O(n^3)$	$O(n^2)$	$O(n^2 \log n)$
FRISS	Ours	$O(n^2 \log n)$	$O(n \log^2 n)$	$O(n^2 \log^2 n)$

evaluated on $(x_1^{2^0},\ldots,x_1^{2^{\log d}-1},\ldots,x_\mu^{2^0},\ldots,x_\mu^{2^{\log d}-1})$. Such a transformation features an $O(d^\mu)$ prover, but verifier and communication complexities are both $O(\mu^3 \log^3 d)$ for verifying μ independent proof, worse than univariate FRI-PC [32] when μ reduces to 1. Ideally, a multivariate PCS scheme should maintain the same complexity as its univariate counterpart.

Conceivably, batch-FRI [6] over a random linear combination of the μ polynomials could reduce verifier and communication complexities to $O(\mu^2 \log^2 d)$. However, while each independent proof can be tailored to a small domain, the above direct aggregation requires a single domain of an enlarged size of $O(d^\mu)$ to accommodate the evaluations of all polynomials $\{\hat{f}_i\}_{i\in[0,\mu-1]}$. The prover now takes $O(\mu d^\mu \log d)$.

We propose a *rolling batch* approach to resolve the above complexity tension. It enables more efficient low-degree tests for $\{\hat{f}_i\}_{i\in[0,\mu-1]}$. The first round of FRI transforms $\hat{f}_0$ into polynomial $\hat{f}_0^{(1)}$ with a degree-bound of $2^{\mu-1}$, which is also the (upper) bound for the degree of $\hat{f}_1$. Generalizing our observation above, we can invoke FRI over $\hat{f}_1'$, a random linear combination of $\hat{f}_0^{(1)}$ and $\hat{f}_1$ (versus independent invocations for $\hat{f}_0^{(1)}$ and $\hat{f}_1$). Similarly, we can combine every $\hat{f}_i'^{(i)}$ with $\hat{f}_{i+1}$, effectively aggregating μ independent FRIs into one, yielding $O(d^\mu)$ prover and $O(\mu^2 \log^2 d)$ verifier/communication.

Contribution-II: One-to-many PolyFRIM. Simplistically, producing n^2 proofs for degree-n $f(X,Y)$ takes n^2 times the overhead. Merging proofs for many evaluations can be challenging. Checking the evaluation proof for bivariate $f(X,Y)$ on point (x_i,y_j) takes $2\log n = 2\cdot t$ rounds. In the k-th round ($k < \log n$), the prover and verifier handle a multilinear polynomial $\tilde{g}_k(x_i^{2^0},\ldots,x_i^{2^{k-1}},X_k,\ldots,X_t,Y_1,\ldots,Y_t)$ determined by $f(X,Y)$. They then transform $\tilde{g}_k$ into another multilinear polynomial $\tilde{g}_{k+1}(x_i^{2^0},\ldots,x_i^{2^{k-1}},x_i^{2^k},X_{k+1},\ldots,X_t,Y_1,\ldots,Y_t)$. The procedure for $k > \log n$ is similar and omitted. As every verifier has a unique evaluation pair $\{(x_i,y_j)\}_{i,j\in[n]}$, repeating the proof for n^2 evaluations seems necessary.

Methodically, we reduce the prover complexity by making evaluation pairs "repeatable" and reusing procedures in proofs even for different verifiers. First, we swap the order of the evaluation vector for each variable. Precisely, we see every $f(x_i,y_j)$ as the evaluation of a multilinear polynomial $\tilde{g}(X_1,\ldots,X_t,Y_1,\ldots,Y_t)$ on $(x_i^{2^t-1},\ldots,x_i^{2^0},y_j^{2^t-1},\ldots,y_j^0)$ instead of on $(x_i^{2^0},\ldots,x_i^{2^t-1},y_j^{2^0},\ldots,y_j^{2^t-1})$. Such a transformation yields equivalence. Second, we set $x_1,\ldots,x_n$ as the n-th roots of unity $w_1,\ldots,w_n$, respectively, *i.e.*, for every $i\in[n]$, $w_i^n = 1$, and similarly for $y_1,\ldots,y_n$.

Completing the whole evaluation proof can be done by constructing a binary tree of depth $2t$ as above. In the i-th round, the prover and verifiers would handle 2^i multilinear polynomials, each of size $O(n^2\cdot 2^{-i+1})$. The total prover complexity is reduced to $\sum_{i\in[2t]} O(2^i)\cdot O(n^2\cdot 2^{-i+1}) = O(n^2 \log n)$.

Contribution-III: FRISS from PolyFRIM. Synthesizing our new AVSS scheme FRISS from our one-to-many PolyFRIM is a non-trivial endeavor (*i.e.*, our FRISS is not derived from merely black-box calling PolyFRIM) even given existing frameworks like eAVSS [3] (instantiated with KZG [24]).

Meshing PolyFRIM into AVSS is hindered by the absence of PCS homomorphism. In a typical AVSS scheme, after the dealer sends a univariate polynomial $f(x_i,Y)$ to P_i, P_i would send $f(x_i,y_j)$ and corresponding evaluation proof $\pi_{i,j}$ to P_j. Some honest party P_k may fail to receive valid $f(x_k,Y)$ to

carry out secret reconstruction. To resolve this issue, after receiving sufficient evaluations and proofs, P_j would compute $f(X, y_j)$ and send $\{f(x_k, y_j), \pi_{k,j}\}$ to P_k to help it construct the valid $f(x_k, Y)$. Note that for such P_k, P_j has to construct $\pi_{k,j}$ using $\{\pi_{i,j}\}_{i \neq k}$. The homomorphism of KZG allows such adaption of proofs for evaluation at new points, given valid proofs for d evaluations of a d-degree univariate polynomial.

Currently, all FRI-based PCSs only use hash functions and lack homomorphism. Our solution is to let the dealer additionally send $\{f(x_j, y_i), \pi_{j,i}\}_{j \in [n]}$ to P_i apart from $f(x_i, Y)$. In the reconstruction phase, P_i sends $(f(x_j, y_i), \pi_{j,i})$ to P_j. Now, even with a message delay or a malicious dealer, any honest P_k can still construct valid $f(x_k, Y)$. Our solution increases the dealer-party communication from $O(n^2)$ to $O(n^2 \log^2 n)$ but does not affect the total complexity (as the party-party communication is already $O(n^2 \log^2 n)$). We leave it as an open problem to construct AVSS without such additional communication using bivariate PCSs without homomorphism.

1.3 Related Work

Polynomial commitment. KZG-PCS [24] achieves $O(1)$ communication and verifier complexity. It is later extended to multivariate [28]. They rely on the q-strong Diffie–Hellman (q-SDH) assumption on pairing groups and a trusted setup.

Bulletproofs [11] removes the trusted setup requirement and supports multivariate polynomials based only on the discrete-log (DLog) assumption. It uses the folding technique to achieve a proof size of $2 \log n$ group/field elements (several KBs) for polynomial size n. However, verification takes $O(n)$ exponentiations. Another PCS DARK [12] achieves logarithmic proof size and verification but relies on strong RSA and adaptive root assumptions. It is thus transparent only if (hence yes/no in Table 1) class groups (of a quadratic number field) are used, which are known for slower operation (*e.g.*, [33]).

FRI-PC [5, 32], an $O(\log n)$-round transparent PCS, features $O(n)$ prover and $O(\log^2 n)$ communication and verifier. It is concretely efficient, relying on field operations without exponentiation/pairings. However, without help from group structures, the prover uses Merkle trees to commit a vector in each round, and the verifier needs to open dozens to hundreds of leaves, leading to a concrete proof size of 50-100 KBs.

If $f(z) = y$ for an n-degree polynomial $f(X)$, $\frac{f(X)-f(z)}{X-z}$ is a degree-$(n-1)$ polynomial. This makes FRI-PC univariate, thereby complicating the generalization of the above fact, and hence FRI-PC, to multivariate polynomials. In particular, Virgo of Zhang *et al.* [37] treats a multivariate polynomial evaluation as a univariate sum-check protocol [7]. The verifier runs FFT for an $O(n)$-size random vector challenge to invoke the sum-check, resulting in $O(n \log n)$ verification. Virgo reduces it by FFT delegation using the GKR protocol [21], and both the sum-check and GKR cost $O(n \log n)$ prover complexity. Multivariate HyperPlonk [16] uses the tensor product [10] to transform a multilinear polynomial evaluation into evaluations of multiple univariate polynomials, then invokes FRI-PC for $\log n$ times to prove them, also incurring an $O(\log n) \cdot O(n) = O(n \log n)$ prover.

One-to-many proofs. Tomescu *et al.* [31] present AMT, a one-to-many proof for KZG-PCS. It constructs an authenticated multi-point evaluation tree to compute n^2 evaluation proofs of a degree $O(n)$ univariate polynomial in $O(n^2 \log n)$ time. However, both communication and per-verifier complexities increase from $O(1)$ to $O(\log n)$. Zhang *et al.* [36] achieve a one-to-many proof for KZG with the same prover complexity without any overhead on the proof size and verifier time. Its main idea is to set the evaluation points as n^2-th root of unity, enabling multiple evaluations via FFT.

For transparent one-to-many proofs, Zhang *et al.* [36] invoke an FFT circuit using the GKR protocol [21] with $O(n^2)$ outputs such that each output is an evaluation. Notably, they propose an elegant method to unify statements for all outputs and a new Fiat–Shamir transformation so the challenges can be used for multiple verifiers. The above procedure is a polynomial interactive oracle proof and can be instantiated with any univariate polynomial commitment, but it is non-trivial to be transformed into multivariate cases due to the difficulty of constructing FFT circuits for multivariate polynomials.

AVSS. Without a synchronous network, parties cannot always receive the message from the dealer in time and may not know when all honest parties have obtained their shares to allow successful reconstruction. One recent common approach for AVSS is to rely on a bivariate polynomial to share secrets. In the sharing phase, each party receives not only its share but also pieces of shares of other parties, *i.e.*, "share-of-share." Parties who have received valid messages can help other parties obtain their shares by sending the pieces of other parties' shares. eAVSS [3], employing bivariate polynomials, is the first AVSS with $O(n^2)$ communication complexity. It is also achieved by Bingo of Abraham *et al.* [1], an adaptively secure and optimally resilient packed AVSS. HAVEN [2] does not use bivariate polynomials but still follows the above idea using a univariate polynomial to share the secret and additional univariate polynomials to generate the pieces of parties' shares.

Encrypt-then-disperse is an alternative [30, 34] with amortized quasi-linear communication complexity. Via asynchronous verifiable information dispersal [14] or Byzantine reliable broadcast [17], the dealer distributes an encryption of each share under each party's key. Parties can eventually obtain their encrypted share accessible solely with their decryption key. Recently, Das *et al.* [18] give an AVSS with complaints, using univariate polynomials but requiring the dealer to stay online until sharing concludes.

2 Preliminaries

Notations. Let $\hat{f}(X)$ and $\tilde{f}(X_1, \ldots, X_\mu)$ represent univariate and multivariate polynomials, respectively. Given size-N $\tilde{f}$

with coefficients $(f_0,\ldots,f_{N-1})$ (low left), $\hat{f}(X)=\sum_{i=0}^{N-1} f_i X^i$ denotes its corresponding (implicitly converted) univariate polynomial (*e.g.*, $\tilde{\phi}$ and $\hat{\phi}$ in Figure 1). Let $\mathbb{F}$ be a prime finite field. Lowercase denotes $\mathbb{F}$-elements, and bold denotes vectors on $\mathbb{F}$. x_i denotes the i-th entry of $\boldsymbol{x}$. $\langle \boldsymbol{a},\boldsymbol{b}\rangle$ and $\boldsymbol{a}\otimes\boldsymbol{b}$ denote the inner and tensor product, respectively. Given a code rate $\rho\in(0,1)$ and $L=\{\eta_1,\ldots,\eta_{|L|}\}\in\mathbb{F}^{|L|}$, an RS code $\mathrm{RS}[L,\rho]\in\mathbb{F}^{|L|}$ means $\{\hat{f}(\eta_1),\ldots,\hat{f}(\eta_{|L|})|\deg(\hat{f})<\rho|L|\}$. Let $\hat{f}|_L$ be the evaluations of $\hat{f}$ on L.

We denote the security parameter by λ, and PPT means probabilistic polynomial time. $\mathsf{negl}(\cdot)$ denotes a negligible function, which means for all polynomials $\hat{f}$, $\mathsf{negl}(\lambda)<1/\hat{f}(\lambda)$ for sufficiently large integer λ. $y\leftarrow_\$ S$ denotes picking y from set S uniformly at random. $[n]$ denotes $\{1,2,\ldots,n\}$ and $[n,m]$ denotes $\{n,n+1,\ldots,m\}$ for positive integers $m>n$.

Merkle trees. Merkle tree is a vector commitment with linear prover time and logarithmic verifier time and proof size, all in the vector size. It consists of three algorithms [37]. $\mathsf{rt}\leftarrow\mathsf{MT.Commit}(\boldsymbol{v})$ outputs the Merkle tree root rt for vector $\boldsymbol{v}$. $(\{v_i\}_{i\in I},\mathsf{path})\leftarrow\mathsf{MT.Open}(I,\boldsymbol{v})$ outputs the query location set I and verification path path. Verification is done via $\mathsf{MT.Verify}(\mathsf{rt},I,\{v_i\}_{i\in I},\mathsf{path})$. This paper uses the Merkle tree from collision-resistant and non-invertible hash functions.

Interactive argument of knowledge (AoK). An interactive AoK for an NP relation $\mathcal{R}$ is a tuple of algorithms $(\mathcal{G},\mathcal{P},\mathcal{V})$. $\mathcal{G}$ is for public parameter (pp) generation. $\mathcal{P}$ and $\mathcal{V}$ represent a PPT prover and verifier, respectively. $\mathcal{P}$ tries to convince $\mathcal{V}$ that $\exists\mathtt{w}$ such that $(\mathtt{x},\mathtt{w})\in\mathcal{R}$ for a public statement $\mathtt{x}$ through rounds of interaction, and $\mathtt{w}$ is efficiently extractable by an extractor. Below, we adopt an existing definition [37].

Definition 2.1. *$(\mathcal{G},\mathcal{P},\mathcal{V})$ is an interactive argument of knowledge for an NP language $\mathcal{L}_\mathcal{R}$ if the following hold:*

- ***Completeness.*** *For every $\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda)$, every $\mathtt{x}\in\mathcal{L}_\mathcal{R}$ and $(\mathtt{x},\mathtt{w})\in\mathcal{R}$, $\Pr[\langle\mathcal{P}(\mathtt{w}),\mathcal{V}\rangle(\mathsf{pp},\mathtt{x})=1]=1$.*
- ***Argument of knowledge.*** *For any PPT prover $\mathcal{P}^*$, any $\mathsf{pp}\leftarrow\mathcal{G}(1^\lambda)$ and any $\mathtt{x}$, there is a PPT extractor $\mathcal{E}$ s.t. $\Pr[\langle\mathcal{P}^*(),\mathcal{V}\rangle(\mathsf{pp},\mathtt{x})=1,(\mathtt{x},\mathtt{w})\notin\mathcal{R}|\mathtt{w}\leftarrow\mathcal{E}(\mathsf{pp},\mathtt{x})]\le\mathsf{negl}(\lambda)$.*

2.1 Polynomial Commitment

Definition 2.2. *A polynomial commitment scheme (PCS) [12, 22] for μ variables and degree bound d is defined as follows.*

- $\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda,d,\mu)$*: Takes as input security parameter λ and bounds d and μ; generates public parameter pp.*
- $C\leftarrow\mathsf{Commit}(\mathsf{pp},\tilde{f})$*: Takes pp and polynomial $\tilde{f}$ as input; generates the commitment C.*
- $b\leftarrow\mathsf{VerPoly}(\mathsf{pp},C,\tilde{f})$*: Verifies the opening of commitment C to the polynomial $\tilde{f}$; outputs $b\in\{0,1\}$.*
- $b\leftarrow\mathsf{Eval}(\mathsf{pp},C,\boldsymbol{x},y,\tilde{f})$*: An interactive argument between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$, which can be denoted as $b\leftarrow\langle\mathsf{Open}(\tilde{f}),\mathsf{Verify}\rangle(\mathsf{pp},C,\boldsymbol{x},y)$. $\mathcal{P}$ attempts to convince $\mathcal{V}$ that $\tilde{f}(\boldsymbol{x})=y$. $\mathcal{V}$ outputs $b\in\{0,1\}$ at the end.*

We call the prover, verifier, and communication complexity of PC the corresponding complexity of Eval, respectively.

Recent PCS definitions [22] consider completeness, polynomial binding, and knowledge soundness as necessary properties, but not zero knowledge. Zero-knowledge PCS is also not necessary for AVSS, as the secrecy only requires that parties less than the threshold cannot recover the secret instead of obtaining no additional information. Similar to HyperPlonk, our PCS is not zero knowledge, but we define a property called *t-bound knowledge leaking* weaker than zero knowledge (inspired by [15]), meaning that a verifier can only obtain polynomial evaluations bounded by t with no additional information. This suffices for AVSS, as shown in Section 4.

Definition 2.3. *A multivariate PCS [22] requires:*

- ***Completeness.*** *For any polynomial $\tilde{f}\in\mathbb{F}^d[X]$, every $\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda,d,\mu)$, $C\leftarrow\mathsf{Commit}(\mathsf{pp},\tilde{f})$ and $\tilde{f}(\boldsymbol{x})=y$, $\Pr[\mathsf{Eval}(\mathsf{pp},C,r,y,\boldsymbol{x},\tilde{f})=1]=1$.*
- ***Polynomial binding.*** *For all $\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda,d,\mu)$, any PPT adversary $\mathcal{A}$, the following probability is $\mathsf{negl}(\lambda)$.*

$$\Pr\left[\begin{array}{c}\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda,d,\mu),(C,\tilde{f}_0,\tilde{f}_1)\leftarrow\mathcal{A}(\mathsf{pp})\\ b_0\leftarrow\mathsf{VerPoly}(\mathsf{pp},C,\tilde{f}_0)\\ b_1\leftarrow\mathsf{VerPoly}(\mathsf{pp},C,\tilde{f}_1)\\ (b_0=b_1=1)\wedge(\tilde{f}_0\neq\tilde{f}_1)\end{array}\right].$$

- ***Knowledge soundness.*** *Eval is an argument of knowledge given $\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda,d,\mu)$.*
- ***t-bound knowledge leaking.*** *For all PPT adversaries $\mathcal{A}$, its random tape $r_\mathcal{A}$, $\mathsf{pp}\leftarrow\mathsf{Gen}(1^\lambda,d,\mu)$, and polynomial $\tilde{f}$, there exists a simulator $\mathcal{S}=(\mathcal{S}_1,\mathcal{S}_2)$ such that the experiments below are computationally indistinguishable, i.e., $|\Pr[\mathsf{Real}_{\mathcal{A},\tilde{f}}(\mathsf{pp})=1]-\Pr[\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^\mathcal{A}}(\mathsf{pp})=1]|\le\mathsf{negl}(\lambda)$.*

$\mathsf{Real}_{\mathcal{A},\tilde{f}}(\mathsf{pp})$:	$\mathsf{Ideal}_{\mathcal{A},\mathcal{S}^\mathcal{A}}(\mathsf{pp})$:
1: $C\leftarrow\mathsf{Commit}(\hat{f},\mathsf{pp})$	1: $C\leftarrow\mathcal{S}_1(1^\lambda,\mathsf{pp},r_\mathcal{A})$, with oracle access to t evaluations on $\hat{f}$
2: $\boldsymbol{x}\leftarrow\mathcal{A}(C,\mathsf{pp})$	2: $\boldsymbol{x}\leftarrow\mathcal{A}(C,\mathsf{pp})$
3: $y\leftarrow\tilde{f}(\boldsymbol{x})$	3: $y\leftarrow\tilde{f}(\boldsymbol{x})$
4: $1\leftarrow\langle\mathcal{P}(\tilde{f}),\mathcal{A}\rangle(\mathsf{pp},C,\boldsymbol{x},y)$	4: $1\leftarrow\langle\mathcal{S}_2,\mathcal{A}\rangle(\mathsf{pp},C,\boldsymbol{x},y)$, with oracle access to $y=\hat{f}(\boldsymbol{x})$
5: $b\leftarrow\mathcal{A}$ and output b	5: $b\leftarrow\mathcal{A}$ and output b

2.2 FRI and FRI-based Schemes

FRI. Given a degree bound d and a committed vector, FRI [5] allows a prover to convince a verifier that the vector is δ-close to $\mathrm{RS}[L,d/|L|]$ in the meaning of relative Hamming distance, *i.e.*, the vector corresponds to $\hat{f}|_L$ such that $\deg(\hat{f})\le d$.

Theorem 2.1 ([5, 6])**.** *FRI is an argument with soundness error $\varepsilon_{\mathsf{FRI}}=O(|L|/|\mathbb{F}|)+\mathsf{negl}(q,d/|L|)$ where the verifier queries $q=O(\lambda)$ (query repetition number) entries of every*

prover's message. The prover complexity is $O(|L|)$ other than the $O(|L|\log|L|)$ commitment. The verifier complexity and the communication complexity are $O(\log^2|L|)$.

Batch-FRI enables proving efficiently that t committed vectors correspond to polynomials $\hat{f}_1,\ldots,\hat{f}_t$ with degree bounds $d_1,\ldots,d_t$. Specifically, $\mathcal{P}$ and $\mathcal{V}$ invoke an FRI to check whether $\hat{f}'|_L \in \mathrm{RS}[L, d_{\max}/|L|]$ for $\hat{f}'(X)=\sum_{i=1}^{t}\lambda^{i-1}\cdot\hat{f}_i(X)\cdot X^{d_{\max}-d_i}$, $d_{\max}=\max\{d_1,\ldots,d_t\}$ and random λ from $\mathcal{V}$.

FRI-PC [32]. FRI can be turned into a univariate PCS with no additional asymptotic overhead. We denote it by FRI-PC. Appendix A recalls the FRI protocol and FRI-PC.

HyperPlonk. HyperPlonk [16, §B], an FRI-based multilinear PCS, treats a multilinear $\tilde{f}(x_1,\ldots,x_\mu)$ as the inner product of the coefficient vector of $\tilde{f}$ and the tensor product $(1,x_1)\otimes\cdots\otimes(1,x_\mu)$. Theorem 2.2 (proven in Appendix B) asserts its security, which is missing in the original paper.

Theorem 2.2. *HyperPlonk is a PCS. Its computation complexity for* Commit *is $O(n\log n)$ for $n=d^\mu$. Other complexities can be divided into two cases according to the concrete algorithm. Prover: $O(n)$ or $O(n\log n)$; Verifier: $O(\log^3 n)$ or $O(\log^2 n)$; Communication: $O(\log^3 n)$ or $O(\log^2 n)$.*

2.3 Asynchronous Verifiable Secret Sharing

An AVSS scheme consists of two phases. In the Share phase, a distinguished party called the dealer P_d distributes shares of a secret s to all n parties $\{P_1,P_2,\ldots,P_n\}$ via Deal. When an honest party P_i receives a share s_i of s, it completes Share. In the Reconstruction phase, any $t+1$ honest parties invoke a reconstruction function to recover the secret s. Each party outputs s' or $\perp$ to indicate either the successful reconstruction of the secret s' or to claim that the dealer P_d is malicious, respectively. Each pair of parties is connected by an asynchronous channel that provides authenticity and privacy [3]. The dealer can broadcast messages to parties by *reliable broadcast* [17].

Any PPT adversary $\mathcal{A}$ can corrupt and coordinate the actions of at most t parties. A party is honest if not corrupted. AVSS assumes no upper bound on message delivery times. $\mathcal{A}$ could delay messages between any two honest parties, but these messages would eventually be delivered.

Definition 2.4 ([2, 3])**.** *(n,t)-AVSS with $n\geq 3t+1$ achieves:*

- ***Liveness.*** *If the dealer P_d is honest, all the honest parties complete the* Share *phase.*
- ***Agreement.*** *If some honest party completes the* Share *phase, then all honest parties eventually complete the* Share *phase. If all honest parties subsequently start the* Reconstruction *phase, all honest parties complete the* Reconstruction *phase.*
- ***Correctness.*** *If the dealer P_d is honest, then the secret s' reconstructed by all honest parties equals s.*
- ***Secrecy.*** *If the dealer P_d is honest, any PPT adversary $\mathcal{A}$ has no extra knowledge about s.*

3 PolyFRIM with a One-to-many Prover

We first propose an FRI variant called rolling batch FRI. Section 3.2 presents an improved multilinear PCS using the rolling batch FRI, which is asymptotically better than HyperPlonk. It is then generalized to our FRI-based multivariate PCS PolyFRIM in Section 3.3. Section 3.4 presents the one-to-many proofs for PolyFRIM, a key component of FRISS.

3.1 Rolling Batch FRI

A main overhead in HyperPlonk comes from running FRI to check every $\hat{f}_{i-1}|_{L_{i-1}}\in\mathrm{RS}[L_{i-1},n/|L_0|]$ for $i\in[\mu]$, where L_0 is an $O(n)$-size multiplicative coset and $L_i=\{x^2|x\in L_{i-1}\}$. One could run FRI for μ times separately to complete these low-degree tests. The prover complexity is $\sum_{i\in[\mu]}O(n/2^{i-1}\cdot\log(n/2^{i-1}))=O(n\log n)$. The verifier and communication complexities are $\sum_{i\in[\mu]}O(\log^2(n/2^{i-1}))=O(\log^3 n)$. Alternatively, a batch FRI evaluates all $\{\hat{f}_i\}_{i\in[\mu]}$ on L_0. It reduces verifier and communication complexities to $O(\log^2 n)$ but leads to a prover complexity of $\sum_{i\in[\mu]}O(n\cdot\log(n/2^{i-1}))=O(n^2\log n)$ due to polynomial evaluations using FFT.

We propose rolling batch FRI (Protocol 1) to perform these μ low-degree tests with $O(n\log n)$ prover complexity and $O(\log^2 n)$ verifier and communication complexities. Our high-level idea is to combine transformed polynomials during the FRI and new polynomials with the same degree to be tested. Specifically, in the i-th round of the rolling batch FRI, the current-round polynomial $\hat{f}^{(i-1)}$ is transformed to a new polynomial $\hat{p}^{(i)}$, which has both the same degree bound $n/2^i$ and evaluation domain L_i as $\hat{f}_i$. So, the prover could combine $\hat{p}^{(i)}$ and $\hat{f}_i$ using a random linear combination and run the FRI on a new combined polynomial. Recursively, for every $i\in[\mu-1]$, $\hat{p}^{(i)}$ can be combined with $\hat{f}_i$, and μ independent FRIs are aggregated into one. Theorem 3.1 asserts its security.

Theorem 3.1. *Protocol 1 is an AoK with a soundness error $\varepsilon_{\mathsf{FRI}}=|L_0|/|\mathbb{F}|+((1+\rho)/2)^q+\mathsf{negl}(\lambda)$ for $\delta\in(0,(1-\rho)/2]$.*

The soundness of FRI comes from the correlated agreement over lines of RS code [6]: if $\boldsymbol{u_0}+\lambda\boldsymbol{u_1}$ is close to an RS codeword $\boldsymbol{v}\in\mathbb{F}^{|L|}$, $\boldsymbol{u_0},\boldsymbol{u_1}\in\mathbb{F}^{|L|}$ are respectively close to two codewords with high probability for random λ. As for rolling batch FRI, the tested vector is a random linear combination of three vectors instead of two (Steps 5 and 9, Protocol 1). Now we rely on the generalized correlated agreement over span spaces [6]: if $\boldsymbol{u_0}+\lambda\boldsymbol{u_1}+\lambda^2\boldsymbol{u_2}$ is close to $\boldsymbol{v}$, then still with high probability $\boldsymbol{u_0},\boldsymbol{u_1},\boldsymbol{u_2}\in\mathbb{F}^{|L|}$ are close to three codewords, respectively. Appendix C gives the detailed proof.

Complexity. The prover complexity is $O(n\log n)$ due to the FFT and Merkle tree construction for $\hat{f}_0|_{L_0}$. The computation overhead other than this is caused by FRI, where in every round, the prover computation takes $\sum_{i\in[0,\log n-1]}O(n/2^i)=O(n)$. The proof size in the i-th round is $O(q)$ field elements and $O(\log(n/2^i))$ hashes for $q=O(\lambda)$. The consis-

Protocol 1 Rolling batch FRI

Inputs: $\big((\mathbb{F}, L_0, d_0); (\{\hat{f}_i\}_{i\in[0,\mu-1]})\big)$. For $i \in [0,\mu-1]$, let $d_i = d_0 \cdot 2^{-i}$ and $L_{i+1} = \{x^2 | x \in L_i\}$. Also, set $n = d_0 = 2^\mu$. The prover shows $\hat{f}_i|_{L_i} \in \mathrm{RS}[L_i, n/|L_0|]$.

1: For every $i \in [0,\mu-1]$, $\mathcal{P}$ computes $\hat{f}_i|_{L_i}$, executes $C_i \leftarrow \mathsf{MT.Commit}(\hat{f}_i|_{L_i})$, and sends C_i to $\mathcal{V}$. Set $\hat{f}^{(0)} \leftarrow \hat{f}_0$.
2: **for** $i = 1$ to μ **do**
3: $\quad\mathcal{P}$ decomposes $\hat{f}^{(i-1)}(X)$ into $\hat{g}_i(X^2) + X \cdot \hat{h}_i(X^2)$.
4: $\quad\mathcal{V}$ sends $\alpha_i \leftarrow_\$ \mathbb{F}$ to $\mathcal{P}$.
5: $\quad\mathcal{P}$ computes $\hat{p}^{(i)}(X) \leftarrow \hat{g}_i(X) + \alpha_i \cdot \hat{h}_i(X)$.
6: $\quad$**if** $i = \mu$ **then** $\mathcal{P}$ directly sends the coefficients of $\hat{p}^{(i)}$.
7: $\quad$**else** $\mathcal{P}$ computes $\hat{p}^{(i)}|_{L_i}$.
8: $\quad\quad\mathcal{P}$ sends $\mathsf{MT.Commit}(\hat{p}^{(i)}|_{L_i})$ to $\mathcal{V}$.
9: $\quad\quad\mathcal{P}$ computes $\hat{f}^{(i)}(X) \leftarrow \hat{p}^{(i)}(X) + \alpha_i^2 \cdot \hat{f}_i(X)$.
10: **for** $j = 1$ to q **do**
11: $\quad\mathcal{V}$ sends $\beta \leftarrow_\$ L_0$ to $\mathcal{P}$.
12: $\quad\mathcal{P}$ invokes $\mathsf{MT.Open}$ to open $\{\hat{f}_i(\pm\beta^{2^i})\}_{i\in[0,\mu-1]}$ and $\{\hat{p}^{(i)}(\pm\beta^{2^i})\}_{i\in[\mu-1]}$. $\mathcal{V}$ checks them by $\mathsf{MT.Verify}$.
13: $\quad$For each $i \in [\mu]$, $\mathcal{V}$ checks whether the three pairs of evaluations $(\pm\beta^{2^{i-1}}, \hat{f}^{(i-1)}(\pm\beta^{2^{i-1}}))$, $(\alpha_i, \hat{p}^{(i)}(\beta^{2^i}))$ are on a common line. Note that $\hat{f}^{(i)}(\pm\beta^{2^i})$ can be computed by $\hat{p}^{(i)}(\pm\beta^{2^i})$ and $\hat{f}_i(\pm\beta^{2^i})$.
14: Iff all the above checks pass, $\mathcal{V}$ outputs 1; 0 otherwise.

tency of $O(q)$ elements and Merkle trees can be checked in one go. Similarly, the verifier computation in the i-th round mainly comes from verifying the Merkle tree validity, which is $O(\log(n/2^i))$. Thus, the verifier and communication complexities are both $\sum_{i\in[0,\log n-1]} O(\log(n/2^i)) = O(\log^2 n)$.

3.2 An Improved Multilinear PCS

We rely on the rolling batch FRI to improve HyperPlonk. The main procedure of HyperPlonk includes: for every $i \in [\mu]$, (1) check $\hat{f}_{i-1}|_{L_{i-1}} \in \mathrm{RS}[L_{i-1}, n/|L_0|]$; (2) prove the validity of $\hat{f}_{i-1}(\pm\beta), \hat{f}_i(\beta^2)$ by FRI-PC for $\beta \leftarrow_\$ \mathbb{F}\setminus\{0\}$ from the verifier. Specifically, for each $i \in [\mu]$, the prover $\mathcal{P}$ proves that

$$\hat{f}_{i-1}\big|_{L_{i-1}} \in \mathrm{RS}[L_{i-1}, d_{i-1}/|L_{i-1}|], \tag{1}$$

$$\frac{\hat{f}_{i-1}(X) - \hat{f}_{i-1}(\pm\beta)}{X \mp \beta}\Big|_{L_{i-1}} \in \mathrm{RS}\left[L_{i-1}, (d_{i-1}-1)/|L_{i-1}|\right],$$

$$\frac{\hat{f}_{i-1}(X) - \hat{f}_{i-1}(\beta^2)}{X - \beta^2}\Big|_{L_{i-1}} \in \mathrm{RS}[L_{i-1}, (d_{i-1}-1)/|L_{i-1}|]\ (i \neq 1).$$

Here, $d_i = d_{i-1}/2$, $L_i = \{x^2 | x \in L_{i-1}\}$, $d_0 = n = 2^\mu$. To prove the above three equations, for each $i \in [\mu]$ $\mathcal{P}$ and $\mathcal{V}$ invoke a batch FRI to show that $\hat{p}_{i-1}|_{L_{i-1}} \in \mathrm{RS}[L_{i-1}, d_{i-1}/|L_{i-1}|]$, where $\hat{p}_{i-1}$ is a random linear combination of

$$\hat{f}_{i-1}, \quad X \cdot \frac{\hat{f}_{i-1}(X) - \hat{f}_{i-1}(\pm\beta)}{X \mp \beta}, \quad X \cdot \frac{\hat{f}_{i-1}(X) - \hat{f}_{i-1}(\beta^2)}{X - \beta^2}.$$

HyperPlonk achieves a succinct FRI-based multilinear PCS without proof composition (*cf.* [37]), but it has limitations:

(1) The soundness error is directly related to the field size due to the Schwartz–Zippel lemma. Attaining the desired security level needs a large field, yet it impacts efficiency.

(2) As shown in Theorem 2.2, there is a tension for HyperPlonk to achieve $O(n)$ prover complexity, $O(\log^2 n)$ verifier complexity, and $O(\log^2 n)$ communication complexity.

(3) The verifier may get additional knowledge from the multiple evaluations it opens on $\{\hat{f}_i\}_{i\in[0,\mu-1]}$, which is inapplicable to AVSS for secrecy violation, *i.e.*, parties less than the threshold may obtain extra information about the secret.

Our proposed PCS overcomes the above limitations.

(1) The verifier picks β from L_0 instead of $\mathbb{F}\setminus\{0\}$ and for $i \in [\mu]$, its queries change to $\hat{f}_{i-1}(\pm\beta^{2^{i-1}})$ instead of $\hat{f}_{i-1}(\pm\beta)$. Now, the verifier can obtain queries by directly opening the Merkle trees instead of invoking the FRI-PC. Consequently, the prover and verifier only need to invoke the FRI to prove Equation (1) other than the two equations below it. This avoids their computation of obtaining additional inverses of field elements and enlarging the polynomial degrees to the same. Besides, the soundness error originating from the Schwartz–Zippel lemma is no longer related to the field size.

(2) Based on the idea (1), the prover now needs to show $\hat{f}_{i-1}|_{L_{i-1}} \in \mathrm{RS}[L_{i-1}, d_{i-1}/|L_{i-1}|]$ for $i \in [\mu]$ instead of $\hat{p}_{i-1}$. This statement is exactly the same as that of rolling batch FRI in Section 3.1, and the prover and verifier can invoke the rolling batch FRI to complete these low-degree tests.

(3) To mask the evaluations on $\{\hat{f}_i\}_{i\in[\mu]}$, the prover chooses a random μ-variate multilinear polynomial $\tilde{r}(X_1,\ldots,X_\mu)$, and sets $\hat{f}_0$ to be the univariate polynomial determined by $\tilde{f} + \alpha\tilde{r}$ instead of $\tilde{f}$ itself for $\alpha \leftarrow_\$ \mathbb{F}$ from the verifier. Now, the evaluations $\{\hat{f}_i\}_{i\in[\mu]}$ are uniformly distributed because of $\tilde{r}$. Although the verifier may still obtain evaluations on $\hat{f}_0|_{L_0}$, the number of leaked evaluations is limited by an upper bound, and it suffices for AVSS (in Section 4).

Below, we present PC_m, our improved multilinear PCS.

- $\mathsf{pp} \leftarrow \mathsf{Gen}_m(1^\lambda, \mu)$: takes security parameter λ and the number of variable μ; outputs the public parameter $\mathsf{pp} = \{\mathbb{F}, L_0\}$.
- $C \leftarrow \mathsf{Commit}_m(\mathsf{pp}, \tilde{f}, \tilde{r})$: takes as inputs pp, $\tilde{f}$, and a random multilinear polynomial $\tilde{r}$; outputs $C = \{C_{\hat{f}}, C_{\hat{r}}\}$ for $C_{\hat{f}} \leftarrow \mathsf{MT.Commit}(\hat{f}|_{L_0})$ and $C_{\hat{r}} \leftarrow \mathsf{MT.Commit}(\hat{r}|_{L_0})$.
- $b \leftarrow \mathsf{VerPoly}_m(\mathsf{pp}, C, \hat{f}|_{L_0}, \hat{r}|_{L_0}, \tilde{f}, \tilde{r})$: receives $\hat{f}|_{L_0}$ and $\hat{r}|_{L_0}$ as openings of C, checks the validity using $\mathsf{MT.Verify}$. Then, decodes $\hat{f}|_L$ and $\hat{r}|_{L_0}$, and checks the consistency between the decoding of $\hat{f}|_L$, $\hat{r}|_{L_0}$ and $\tilde{f}$, $\tilde{r}$. Outputs 1 iff all checks pass.
- $b \leftarrow \mathsf{Eval}_m(\mathsf{pp}, C, \boldsymbol{x}, y, \{\tilde{f}, \tilde{r}\})$: Given evaluation point $\boldsymbol{x} = (x_1,\ldots,x_\mu)$, $y = \tilde{f}(\boldsymbol{x})$, $\tilde{f}$, and $\tilde{r}$, $\mathcal{P}$ and $\mathcal{V}$ run the following:
  1: $\mathcal{P}$ sends $y = \tilde{f}(\boldsymbol{x})$ and $y_r = \tilde{r}(\boldsymbol{x})$ to $\mathcal{V}$.
  2: $\mathcal{V}$ sends $\alpha \leftarrow_\$ \mathbb{F}$ to $\mathcal{P}$.
  3: $\mathcal{P}$ sets $\hat{f}_0(X) \leftarrow \hat{f}(X) + \alpha\cdot\hat{r}(X)$, and for $i \in [\mu]$, computes

$$\hat{f}_i(X) \leftarrow \hat{g}_{i-1}(X) + x_i \cdot \hat{h}_{i-1}(X) \tag{2}$$

by uniquely decomposing $\hat{f}_{i-1}(X)$ into $\hat{g}_{i-1}(X^2)+X\cdot\hat{h}_{i-1}(X^2)$. For every $i\in[\mu-1]$, $\mathcal{P}$ sets $L_i=\{x^2|x\in L_{i-1}\}$ and sends $C_i\leftarrow\mathsf{MT.Commit}(\hat{f}_i|_{L_i})$ to $\mathcal{V}$. When $i=\mu$, $\mathcal{V}$ can compute $\hat{f}_\mu=\tilde{f}(\boldsymbol{x})+\alpha\tilde{r}(\boldsymbol{x})$ locally.

4: Set $\hat{\varphi}_0(X)=\hat{f}_0(X)$. For each $i\in[\mu]$:
   a. $\mathcal{V}$ sends $\alpha_i\leftarrow_\$\mathbb{F}$ to $\mathcal{P}$.
   b. $\mathcal{P}$ derives $\hat{p}_i(X)\leftarrow\hat{g}_{i-1}(X)+\alpha_i\cdot\hat{h}_{i-1}(X)$ by uniquely decomposing $\hat{f}_{i-1}(X)$ into $\hat{g}_{i-1}(X^2)+X\cdot\hat{h}_{i-1}(X^2)$. When $i=\mu$, $\mathcal{P}$ sends the coefficients of $\hat{p}_\mu$ to $\mathcal{V}$; else, $\mathcal{P}$ computes $\hat{\varphi}_i(X)\leftarrow\hat{p}_i(X)+\alpha_i^2\cdot\hat{f}_i(X)$, and sends $\bar{C}_i\leftarrow\mathsf{MT.Commit}(\hat{p}_i|_{L_i})$ to $\mathcal{V}$. $\hat{\varphi}_i(X)$ represents the i-th round polynomial to be invoked in FRI.
5: Repeat the following $q=O(\lambda)$ times.
   a. $\mathcal{V}$ sends $\beta\leftarrow_\$ L_0$ to $\mathcal{P}$.
   b. $\mathcal{P}$ invokes $\mathsf{MT.Open}$ to open $\{\hat{f}_i(\pm\beta^{2^i})\}_{i\in[0,\mu-1]}$ and $\{\hat{p}_i(\pm\beta^{2^i})\}_{i\in[\mu-1]}$.
   c. $\mathcal{V}$ runs $\mathsf{MT.Verify}$ to check the Merkle trees and if, $\forall i\in[\mu]$, triples $(\pm\beta^{2^{i-1}},\pm\hat{f}_{i-1}(\beta^{2^{i-1}})),(x_i,\hat{f}_i(\beta^{2^i}))$ and $(\pm\beta^{2^{i-1}},\pm\hat{\varphi}_{i-1}(\beta^{2^{i-1}})),(\alpha_i,\hat{p}_i(\beta^{2^i}))$ (note the $+$ and $-$ points) are on two common lines, respectively.
6: Iff all above verifications pass, $\mathcal{V}$ outputs 1; 0 otherwise.

Theorem 3.2. *PC_m is a polynomial commitment scheme as described in Definition 2.3. PC_m also has $2q$-bound knowledge leaking property. (The proof is in Appendix D.)*

Complexity. The computation of Commit is $O(2^\mu\cdot\mu)$ due to the FFT and Merkle tree construction for $\hat{f}|_{L_0}$ and $\hat{r}|_{L_0}$. As the main procedure of Eval can be seen as a tensor product argument (as in [10] and HyperPlonk) and a rolling batch FRI, the complexity overhead comes from these two protocols. The prover complexity in each round is linear to the current polynomial degree and is $\sum_{i\in[0,\log(2^\mu)-1]}O(2^\mu/2^i)=O(2^\mu)$ in total. The verifier complexity and communication complexity in each round are both logarithmic to the current polynomial degree and are $\sum_{i\in[0,\log(2^\mu)-1]}O(\log(2^\mu/2^i))=O(\mu^2)$.

3.3 Construction of PolyFRIM

We present PolyFRIM, our multivariate PCS. Suppose $\tilde{f}(X_1,\ldots,X_\mu)$ is a μ-variate d-degree polynomial where $d=2^t-1$. Any evaluation $\tilde{f}(x_1,\ldots,x_\mu)$ is equivalent to the evaluation $\tilde{f}'(x_{1,1},\ldots,x_{1,t},\ldots,x_{\mu,1},\ldots,x_{\mu,t})$, where $\tilde{f}'$ is a $\mu\cdot t$-variate multilinear polynomial, and for every $i\in[\mu]$ and $j\in[t]$, $x_{i,j}=x_i^{2^{j-1}}$. So, the evaluation proof for $\tilde{f}$ can be transformed equivalently into that for $\tilde{f}'$, which can be proved by PC_m.

The algorithms of PolyFRIM are described below.

- $\mathsf{pp}\leftarrow\mathsf{Gen}_\mathsf{F}(1^\lambda,d,\mu)$: Takes as inputs security parameter λ, degree d, and variable number μ; outputs $\mathsf{pp}\leftarrow\mathsf{Gen}_m(1^\lambda,\mu\cdot t)$.
- $C\leftarrow\mathsf{Commit}_\mathsf{F}(\mathsf{pp},\tilde{f},\tilde{r})$: Takes as inputs pp, polynomials $\tilde{f}$ and $\tilde{r}$, and sets $\tilde{f}'$ and $\tilde{r}'$ be multilinear polynomials with $\mu\cdot\log(d+1)$ variables constructed by $\tilde{f}$ and $\tilde{r}$, respectively. The algorithm runs $C\leftarrow\mathsf{Commit}_m(\mathsf{pp},\tilde{f}',\tilde{r}')$.
- $b\leftarrow\mathsf{VerPoly}_\mathsf{F}(\mathsf{pp},C,\tilde{f},\tilde{r})$: Takes as inputs pp, C, $\tilde{f}$, and $\tilde{r}$, and sets $\tilde{f}'$ and $\tilde{r}'$ be multilinear polynomials as in $\mathsf{Commit}_\mathsf{F}$. The algorithm runs $\mathsf{VerPoly}_m(\mathsf{pp},C,\hat{f}'|_{L_0},\hat{r}'|_{L_0},\tilde{f}',\tilde{r}')$
- $b\leftarrow\mathsf{Eval}_\mathsf{F}(\mathsf{pp},C,\boldsymbol{x},y,\{\tilde{f},\tilde{r}\})$: Parse $\boldsymbol{x}=(x_1,\ldots,x_\mu)$. The prover and verifier invoke the interactive argument $\mathsf{Eval}_m(\mathsf{pp},C,\boldsymbol{x}',y,\{\tilde{f}',\tilde{r}'\})$ to prove $\tilde{f}'(\boldsymbol{x}')=\tilde{f}(\boldsymbol{x})$, where $\boldsymbol{x}'=(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_\mu)$ and for every $j\in[\mu]$, $\boldsymbol{x}_j=(x_j^{2^0},x_j^{2^1},\ldots,x_j^{2^{t-1}})$.

Complexity. The complexity of PolyFRIM is similar to PC_m, but substituting the term 2^μ as d^μ. The time complexity of Commit is $O(\mu d^\mu\log d)$, the prover complexity is $O(d^\mu)$, and the verifier and communication complexities are $O(\mu^2\log^2 d)$.

Theorem 3.3. PolyFRIM *is a polynomial commitment scheme with $2q$-bound knowledge leaking property.*

Due to the equivalence of transformation from $\tilde{f}$ to $\tilde{f}'$, Theorem 3.3 follows Theorem 3.2 directly.

3.4 Extending to One-to-many Proofs

We extend PolyFRIM for (n,n)-degree bivariate polynomials to one-to-many proofs with $O(n^2\log n)$ prover complexity.

We first explain why reducing the prover complexity is challenging. When invoking PolyFRIM to prove $f(x_i,y_j)$ for $i,j\in[n]$, the evaluation proof will be transformed into a proof for a multilinear polynomial on $(x_i^{2^0},\ldots,x_i^{2^{t-1}},y_j^{2^0},\ldots,y_j^{2^{t-1}})$, where $n=2^t-1$. Therefore, Equation (2) in each round is different for every verifier. Besides, the random challenges (α and α_i) in Steps 3 and 4 of Protocol 1 can also be different for every verifier. Hence, the prover has to repeat PolyFRIM for $O(n^2)$ times, leading to an $O(n^3\log n)$ prover complexity.

We reduce the prover complexity by making evaluation vectors "partially the same" and random challenges "sharable" even for different verifiers. This enables the prover to save the computation overhead by reusing procedures as much as possible. The details are described below.

Improvement for evaluation vectors. First, we transform the evaluation vector $(x^{2^0},\ldots,x^{2^{t-1}},y^{2^0},\ldots,y^{2^{t-1}})$ into its variable-inverse $(x^{2^{t-1}},\ldots,x^{2^0},y^{2^{t-1}},\ldots,y^{2^0})$. The outcome is equivalent after reversing the order of coefficient vector $\boldsymbol{f}$.

Second, we make evaluation vectors "partially the same" by picking special values for every pair (x_i,y_j). This is feasible as AVSS only requires no repetition between any two distinct pairs. Specifically, for every $i,j\in[n]$, we set $(x_i,y_j)=(w_N^i,w_N^j)$, where $N=n=2^t$ and w_N^z denotes the z-th root of unity. This results in only 2^k choices for $\{x_i^{2^k}\}_{i\in[n]}$ in the k-th round, *i.e.*, the prover only needs 2^k operations instead of $n=2^t$. The case with Y is alike. The prover computation overhead related to the evaluation vector becomes $O(2\cdot\sum_{k\in[t]}2^k)=O(n^2\log n)$.

Improvement for random challenges. We construct "shared random challenges" [34, 36] for different verifiers to make

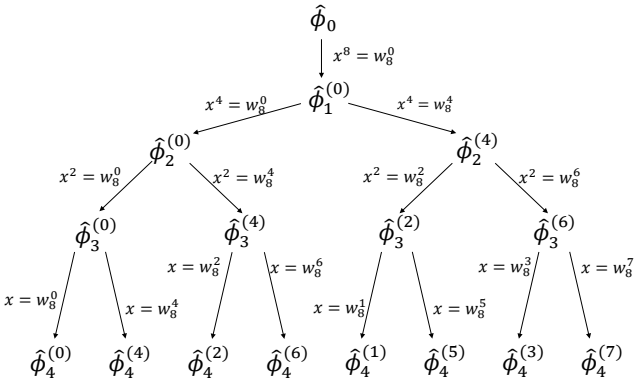


Figure 1: One-to-many proof for a 15-degree polynomial $\hat{\phi}(X)$. $\hat{\phi}(X)$ is firstly transformed into a multilinear polynomial $\tilde{\phi}_0(X_1,X_2,X_3,X_4)$ and then transformed into $\tilde{\phi}_1^{(0)}(w_8^0,X_2,X_3,X_4)$. For $k\in\{1,2,3\}$ and $\ell\in[0,2^{k-1}-1]$, $\tilde{\phi}_k^{(2^{3-k}\cdot\ell)}$ is converted into 2^k polynomials $\{\tilde{\phi}_{k+1}^{(2^{2-k}\cdot\ell)}\}_{\ell\in[0,2^k-1]}$.

PolyFRIM non-interactive. Let the set of random challenges in the ℓ-th round be $A_{r_\ell}=\{\alpha_{1,\ell},\ldots,\alpha_{n,\ell}\}$, where $\ell\in[2t]$ and $\alpha_{i,\ell}$ represents the random challenge from the i-th verifier. The prover computes $\alpha_\ell\leftarrow\mathsf{MT.Commit}(\{\alpha_{i,\ell}\}_{\ell\in[n]})$ and uses α_ℓ as the (final) challenge of the ℓ-th round. The verification path from the leaf $\alpha_{i,\ell}$ to α_ℓ is also sent to the i-th verifier. Figure 1 considers a univariate example, but the process naturally extends to any multivariate polynomial. Our non-interactive $\mathsf{Eval}_{\mathsf{otm}}$ algorithm consists of $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Open}$ and $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Verify}$, calling $\mathsf{Commit}_{\mathsf{F}}$ and $\mathsf{Eval}_{\mathsf{F}}$ of PolyFRIM.

• $\{(w_N^i,w_N^j),f(w_N^i,w_N^j),\pi_{i,j}\}\leftarrow\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Open}(\mathsf{pp},f,r,V_{n,n})$: $\mathcal{P}$ computes proofs $\{\pi_{i,j}\}_{i,j\in[n]}$ for bivariate polynomials f,r and the evaluation-point set $V_{n,n}=\{(w_N^i,w_N^j)\}_{i,j\in[n]}$ as below.

1. For every $i,j\in[n]$, $\mathcal{P}$ computes

$$\alpha_{i,j}\leftarrow\mathsf{H}\big(C_f||C_r||(w_N^i,w_N^j)||f(w_N^i,w_N^j)||r(w_N^i,w_N^j)\big),$$

where $(C_f,C_r)\leftarrow\mathsf{Commit}_{\mathsf{F}}(\mathsf{pp},f,r)$. Then $\mathcal{P}$ computes $\alpha\leftarrow\mathsf{MT.Commit}(\{\alpha_{i,j}\}_{i,j\in[n]})$. Let $\mathsf{path}_{\alpha_{i,j}}$ be the verification path of $\alpha_{i,j}$. Set $\tilde{f}_0$ as the multilinear polynomial determined by $f+\alpha_0\cdot r$. Denote by $\hat{f}_0$ the univariate polynomial determined by vector $\boldsymbol{f}_0$. Set $n=N=2^t$.

2. Generate proofs about variable X. Let $\hat{f}_0^{(0)}=\hat{f}_0$. To avoid long superscripts, for every $k\in[t]$ and $\ell\in[0,2^{k-1}-1]$, let $\xi_{\ell,k}=\ell\cdot2^{t-k}$ and $-\xi_{\ell,k}=\ell\cdot2^{t-k}+2^{t-1}$. Polynomials with $-\xi_{\ell,k}$ go from the same upper-layer polynomials with $\xi_{\ell,k}$. Intuitively, $\xi_{\ell,k}$ covers $0,2,1,3$ in Figure 1, and $-\xi_{\ell,k}$ covers $4,5,6,7$. $\mathcal{P}$ then computes

$$\begin{aligned}\hat{f}_k^{(\xi_{\ell,k})}(X)&\leftarrow\hat{g}_{k-1}^{(2\cdot\xi_{\ell,k})}(X)+w_N^{\ell\cdot2^{t-k}}\cdot\hat{h}_{k-1}^{(2\cdot\xi_{\ell,k})}(X),\\ \hat{f}_k^{(-\xi_{\ell,k})}(X)&\leftarrow\hat{g}_{k-1}^{(2\cdot\xi_{\ell,k})}(X)-w_N^{\ell\cdot2^{t-k}}\cdot\hat{h}_{k-1}^{(2\cdot\xi_{\ell,k})}(X),\\ C_k^{(\xi_{\ell,k})}&\leftarrow\mathsf{MT.Commit}(\hat{f}_k^{(\xi_{\ell,k})}|_{L_k}),\\ C_k^{(-\xi_{\ell,k})}&\leftarrow\mathsf{MT.Commit}(\hat{f}_k^{(-\xi_{\ell,k})}|_{L_k}),\end{aligned}$$

by uniquely decomposing $\hat{f}_{k-1}^{(2\cdot\xi_{\ell,k})}$ as $\hat{g}_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)+X\cdot\hat{h}_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)$ to get $\hat{g}_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)$ and $\hat{h}_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)$ After the first t rounds, $\mathcal{P}$ will get polynomials $\hat{f}_t^{(1)},\ldots,\hat{f}_t^{(n)}$, corresponding to $w_N^1,w_N^2,\ldots,w_N^n$. Here, polynomials with $2\xi_{\ell,k}$ denote the decomposition of those with $\xi_{\ell,k}$.

3. The subsequent computation corresponding to variable Y is similar to Step 2. For every $i\in[n]$, let $\hat{f}_t^{(i,0)}=\hat{f}_t^{(i)}$. For every $k\in[t]$ and $\ell\in[0,2^{k-1}-1]$, $\mathcal{P}$ computes

$$\begin{aligned}\hat{f}_{k+t}^{(i,\xi_{\ell,k})}(X)&\leftarrow\hat{g}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X)+w_N^{\ell\cdot2^{t-k}}\cdot\hat{h}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X),\\ \hat{f}_{k+t}^{(i,-\xi_{\ell,k})}(X)&\leftarrow\hat{g}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X)-w_N^{\ell\cdot2^{t-k}}\cdot\hat{h}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X),\\ C_{k+t}^{(i,\xi_{\ell,k})}&\leftarrow\mathsf{MT.Commit}(\hat{f}_{k+t}^{(i,\xi_{\ell,k})}|_{L_{k+t}}),\\ C_{k+t}^{(i,-\xi_{\ell,k})}&\leftarrow\mathsf{MT.Commit}(\hat{f}_{k+t}^{(i,-\xi_{\ell,k})}|_{L_{k+t}}),\end{aligned}$$

by uniquely decomposing $\hat{f}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}$ as $\hat{g}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)+X\cdot\hat{h}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)$ to get $\hat{g}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)$ and $\hat{h}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)$. $\mathcal{P}$ will eventually get all paths reaching all evaluation points in $\{(w_N^i,w_N^j)\}_{i,j\in[n]}$, all forking from a single root of $\hat{f}_0$.

4. For every $i,j\in[n]$, we have commitments $\{C_k^{(i\cdot2^{t-k})}\}_{k\in[t]}$ and $\{C_{k+t}^{(i,j\cdot2^{t-k})}\}_{k\in[t]}$ in the path from the root for $\hat{f}_0$ to the evaluation point (w_N^i,w_N^j), where $C_k^{(x)}\leftarrow C_k^{(x\bmod 2^t)}$ and $C_{k+t}^{(x,y)}\leftarrow C_{k+t}^{(x\bmod 2^t,\,y\bmod 2^t)}$. $\mathcal{P}$ computes the challenge

$$\alpha_1^{(i,j)}\leftarrow\mathsf{H}\big(\alpha||\{C_k^{(i\cdot2^{t-k})}\}_{k\in[t]}||\{C_{k+t}^{(i,j\cdot2^{t-k})}\}_{k\in[t]}\big),$$

and $\alpha_1\leftarrow\mathsf{MT.Commit}(\{\alpha_1^{(i,j)}\}_{i,j\in[n]})$ in this step. Set $\hat{\varphi}_0^{(0)}=\hat{f}_0$. $\mathcal{P}$ computes polynomials for rolling batch FRI

$$\begin{aligned}q_1^{(\xi_{\ell,1})}(X)&\leftarrow g_0^{(2\cdot\xi_{\ell,1})}(X)+\alpha_1\cdot h_0^{(2\cdot\xi_{\ell,1})}(X),\\ \hat{\varphi}_1^{(\xi_{\ell,1})}(X)&\leftarrow q_1^{(\xi_{\ell,1})}(X)+\alpha_1^2\cdot\hat{f}_1^{(\xi_{\ell,1})}(X),\\ \bar{C}_1^{(\xi_{\ell,1})}&\leftarrow\mathsf{MT.Commit}(q_1^{(\xi_{\ell,1})}|_{L_1}),\end{aligned}$$

by uniquely decomposing $\hat{\varphi}_0^{(2\cdot\xi_{\ell,1})}(X)$ as $g_0^{(2\cdot\xi_{\ell,1})}(X^2)+Xh_0^{(2\cdot\xi_{\ell,1})}(X^2)$ to get $g_0^{(2\cdot\xi_{\ell,1})}(X^2)$ and $h_0^{(2\cdot\xi_{\ell,1})}(X^2)$. Let $(x)\leftarrow(x\bmod 2^t)$ for $x\in\mathbb{Z}$. Next, for $k\in[2,t]$ and $\ell\in[0,2^{k+1}-1]$, $\mathcal{P}$ computes the Fiat–Shamir challenges

$$\begin{aligned}\alpha_k^{(\xi_{\ell,k})}&\leftarrow\mathsf{H}\big(\alpha_{k-1}||\bar{C}_{k-1}^{(2\cdot\xi_{\ell,k-1})}\big),\\ \alpha_k&\leftarrow\mathsf{MT.Commit}\big(\{\alpha_k^{(\xi_{\ell,k})}\}_{\ell\in[0,2^{k-1}-1]}\big).\end{aligned}$$

Then for $\ell\in[0,2^k-1]$, $\mathcal{P}$ computes for rolling batch FRI

$$\begin{aligned}q_k^{(\xi_{\ell,k})}(X)&\leftarrow g_{k-1}^{(2\cdot\xi_{\ell,k})}(X)+\alpha_k\cdot h_{k-1}^{(2\cdot\xi_{\ell,k})}(X),\\ \hat{\varphi}_k^{(\xi_{\ell,k})}(X)&\leftarrow q_k^{(\xi_{\ell,k})}(X)+\alpha_k^2\cdot\hat{f}_k^{(\xi_{\ell,k})}(X),\\ \bar{C}_k^{(\xi_{\ell,k})}&\leftarrow\mathsf{MT.Commit}(q_k^{(\xi_{\ell,k})}|_{L_k}),\end{aligned}$$

by uniquely decomposing $\hat{\phi}_{k-1}^{(2\cdot\xi_{\ell,k})}(X)$ as $g_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)+X\cdot h_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)$ to get $g_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)$ and $h_{k-1}^{(2\cdot\xi_{\ell,k})}(X^2)$.

5. For $i\in[n]$, set $\hat{\phi}_t^{(i,0)}=\hat{\phi}_t^{(i)}$ and $\bar{C}_t^{(i,0)}=\bar{C}_t^{(i)}$. For $k\in[t],\ell\in[0,2^{k-1}-1]$, $\mathcal{P}$ computes the Fiat–Shamir challenges

$$\alpha_{k+t}^{(i,\xi_{\ell,k})}\leftarrow \mathsf{H}\big(\alpha_{k+t-1}||\bar{C}_{k+t-1}^{(i,2\cdot\xi_{\ell,k-1})}\big),$$
$$\alpha_{k+t}\leftarrow \mathsf{MT.Commit}\big(\{\alpha_{k+t}^{(i,\xi_{\ell,k})}\}_{\ell\in[0,2^{k+1}-1]}^{i\in[n]}\big).$$

Then for $\ell\in[0,2^k-1]$, $\mathcal{P}$ computes for rolling batch FRI

$$q_{k+t}^{(i,\xi_{\ell,k})}(X)\leftarrow g_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X)+\alpha_{k+t}\cdot h_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X),$$
$$\hat{\phi}_{k+t}^{(i,\xi_{\ell,k})}(X)\leftarrow q_{k+t}^{(i,\xi_{\ell,k})}(X)+\alpha_{k+t}^2\cdot \hat{f}_{k+t}^{(i,\xi_{\ell,k})}(X),$$
$$\bar{C}_{k+t}^{(i,\xi_{\ell,k})}\leftarrow \mathsf{MT.Commit}(q_{k+t}^{(i,\xi_{\ell,k})}|_{L_{k+t}}),$$

by uniquely decomposing $\hat{\phi}_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X)$ as $g_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)+X\cdot h_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)$ to get $g_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)$ and $\cdot h_{k+t-1}^{(i,2\cdot\xi_{\ell,k})}(X^2)$.

6. For every $i,j\in[n]$, $\mathcal{P}$ gets $q_{i,j}\leftarrow \mathsf{H}(\alpha_{2t}||\bar{C}_{2t}^{(i,j)})$ and computes $q\leftarrow \mathsf{MT.Commit}\big(\{q_{i,j}\}_{i,j\in[n]}\big)$ as the random evaluation set, which $\mathcal{P}$ will open for all polynomials in Eval_m. Every proof $\pi_{i,j}$ includes all the Merkle tree commitments, opened evaluations and their verification paths, and all verification paths for aggregated random challenges.

• $b\leftarrow \mathsf{Eval}_{\mathsf{otm}}.\mathsf{Verify}(\mathsf{pp},\{(w_N^i,w_N^j),f(w_N^i,w_N^j),\pi_{i,j}\})$ for $i,j\in[n]$: $\mathcal{V}$ uses MT.Verify to check the consistency of the Merkle tree commitments against their verification paths. $\mathcal{V}$ outputs 0 if anyone fails. $\mathcal{V}$ then outputs $\mathsf{Eval}_{\mathsf{F}}.\mathsf{Verify}$.

Theorem 3.4. *The algorithm* $\mathsf{Eval}_{\mathsf{otm}}$ *is a non-interactive argument for the relation* $((C,\boldsymbol{x},\boldsymbol{y});\tilde{f})$.

From the standpoint of an individual verifier, the major distinction between $\mathsf{Eval}_{\mathsf{F}}$ and $\mathsf{Eval}_{\mathsf{otm}}$ is the random challenge generation. Instead of deriving it from the hash of preceding messages in $\mathsf{Eval}_{\mathsf{F}}$, $\mathsf{Eval}_{\mathsf{otm}}$ utilizes a Merkle tree root. This adaptation aligns with the equivalence of this mechanism to $\log N$ rounds of dummy messages [36]. The proof detailing this Fiat–Shamir transformation variant has been comprehensively addressed [36, Theorem 4].

Complexity. $\mathsf{Eval}_{\mathsf{otm}}$ computes and commits $O(2^k)$ polynomials of degree $O(2^{2t-k})$, $\forall k\in[2t]$, where the evaluations of a polynomial on L_k can be efficiently computed by the evaluations of polynomials on L_{k-1} within $O(2^{2t-k})$ computations. So, the prover complexity is $\sum_{k\in[2t]}O(2^k\cdot 2^{2t-k})=O(t\cdot 2^{2t})=O(n^2\log n)$. The communication and the verifier complexities per party are $O(\log^2 n)$, the same as PolyFRIM.

4 Construction of FRISS

Overview. For threshold t and n parties $\{P_1,P_2,\ldots,P_n\}$, FRISS uses a (t,t)-degree bivariate polynomial $f(X,Y)$ to share a secret $s=f(z,0)$ for a prescribed z. We set $V_{n,n}=\{(w_N^i,w_N^j)\}_{i,j\in[n]}$ as evaluation points, where $n=3t+1$ and $N=2^{\lceil\log n\rceil}$. On a high level, FRISS is similar with eAVSS [3] using PolyFRIM instead of KZG, where the dealer firstly sends $\hat{f}(x_i,Y)$ and its commitment as the share to party P_i, who then sends $f(x_i,y_j)$ and $\pi_{i,j}$ to party P_j. After receiving enough evaluations and proofs, P_j reconstructs $\hat{f}(X,y_j)$ and sends $\{f(x_i,y_j),\pi_{i,j}\}$ to P_i to help P_i obtain its share $\hat{f}(x_i,Y)$. However, trivially following the above idea poses challenges.

Firstly, eAVSS needs KZG to be zero-knowledge. FRISS achieves secrecy from PolyFRIM's $2q$-bound knowledge leaking by setting threshold $t=p+2q$ for p corrupted parties.

Secondly, without the homomorphism of KZG, generating $\pi_{k,j}$ for P_j to aid some honest party P_k to recover its share using proofs $\{\pi_{i,j}\}_{i\neq j}$ is not straightforward. As described in Section 1.2, this can be resolved by the dealer additionally sending $\{f(x_j,y_i),\pi_{j,i}\}_{j\in[n]}$ to P_i instead of only $\hat{f}(x_i,Y)$.

Thirdly, again facing the lack of homomorphism, it is unclear how to efficiently ensure the consistency among commitments to $\{\hat{f}(x_i,Y)\}_{i\in[n]}$ and $f(X,Y)$, which is necessary for the correctness of AVSS. In a KZG-based AVSS, the dealer just broadcasts $t+1$ commitments to $\hat{f}(x_1,Y),\ldots,\hat{f}(x_{t+1},Y)$, and every party P_i ($i\notin[t]$) can construct the commitment to $\hat{f}(x_i,Y)$ by homomorphism. Nevertheless, our unique construction of PolyFRIM allows an almost cost-free solution. Specifically, all commitments to $\hat{f}(x_i,Y)$ are generated from one commitment to $\hat{f}(X,Y)$, and this naturally guarantees commitment consistency. Each party needs only to use $\hat{f}(x_i,Y)$ for an extra validity check by invoking VerPoly of FRI-PC. It is unknown how to handle this problem using other FRI-based PCSs like Virgo without such special construction.

Below, we present FRISS, consisting of Algorithms 2-4.

Algorithm 2 $\mathsf{Deal}(s)$ for the dealer P_d

1: Sample (t,t)-degree polynomials $f(X,Y)$ and $r(X,Y)$ randomly, such that $f(z,0)=s$.
2: $C\leftarrow \mathsf{Commit}_{\mathsf{F}}(\mathsf{pp},f,r)$; $\mathsf{frv}\leftarrow(\mathsf{pp},f,r,V_{n,n})$.
3: $\{(w_N^i,w_N^j),f(w_N^i,w_N^j),\pi_{i,j}\}_{i,j\in[n]}\leftarrow \mathsf{Eval}_{\mathsf{otm}}.\mathsf{Open}(\mathsf{frv})$.
4: **for** $i=1$ to n **do** $\hat{f}_i\leftarrow\hat{f}(w_N^i,Y)$, $\hat{r}_i\leftarrow\hat{r}(w_N^i,Y)$.
5: Broadcast $\langle\mathsf{Commit},C,\{C_i^*\}_{i\in[n]}\rangle$.
6: **for** $i=1$ to n **do**
7: **for** $j=1$ to n **do** $f_{j,i}\leftarrow f(w_N^j,w_N^i)$, $r_{j,i}\leftarrow r(w_N^j,w_N^i)$.
8: Send $\langle\mathsf{Share},\hat{f}_i,\hat{r}_i,\{f_{j,i},r_{j,i},\pi_{j,i}\}_{j\in[n]}\rangle$ to P_i.

Dealing phase (Algorithm 2). Given the secret s, the dealer P_d randomly picks (t,t)-degree polynomials $f(X,Y)$ and $r(X,Y)$ where $f(z,0)=s$. P_d generates proofs of f,r on $V_{n,n}$ by $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Open}$. In this algorithm, P_d gets $\{C_i^*\}_{i\in[n]}$ where C_i^* is the commitment of $\hat{f}(w_N^i,Y)+\alpha\cdot\hat{r}(w_N^i,Y)$. Then P_d broadcasts all the commitments and sends the Share message to P_i for $i\in[n]$, which includes $\hat{f}(w_N^i,Y)$, $\hat{r}(w_N^i,Y)$, and evaluations/proofs of f and r on $\{(w_N^j,w_N^i)\}_{j\in[n]}$.

Algorithm 3 $\mathsf{Share}_i()$ for party P_i

1: $\mathsf{honest\_dealer} \leftarrow 0$, $\mathsf{share}_i \leftarrow \emptyset$, $\mathsf{echo\_set} \leftarrow \emptyset$.
2: **upon** receiving from P_d $\langle \mathsf{Commit}, C, \{C_i^*\}_{i\in[n]} \rangle$ and $\langle \mathsf{Share}, \hat{f}_i, \hat{r}_i, \{f_{j,i}, r_{j,i}, \pi_{j,i}\}_{j\in[n]} \rangle$ **do**
3: **if** $\mathsf{Commit}_{\mathsf{F}}(\mathsf{pp}, \hat{f}_i + \alpha\hat{r}_i) = C_i^*$ holds and $\forall j \in [n]$: $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Verify}(\mathsf{pp}, (w_N^j, w_N^i), \{f_{j,i}, r_{j,i}\}, \pi_{j,i}) = 1$ **then**
4: $\mathsf{honest\_dealer} \leftarrow 1$; $\mathsf{share}_i \leftarrow \hat{f}_i$.
5: Send $\mathsf{ec}_{j,i} = \langle \mathsf{Echo}, f_{j,i}, r_{j,i}, \pi_{j,i} \rangle$ to $P_j, \forall j \in [n]$.
6: **upon** receiving $\mathsf{ec}_{i,j}$ from P_j for the first time **do**
7: **if** $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Verify}(\mathsf{pp}, (w_N^i, w_N^j), \{f_{i,j}, r_{i,j}\}, \pi_{i,j})$ **then**
8: Mark Echo as valid.
9: $\mathsf{echo\_set} \leftarrow \mathsf{echo\_set} \cup \{(w_N^i, w_N^j), \{f_{i,j}, r_{i,j}\}\}$.
10: **upon** getting $2t+1$ valid Echo & $\mathsf{honest\_dealer} = 1$ **do**
11: Send $\langle \mathsf{Ready} \rangle$ to all parties.
12: **upon** receiving $t+1$ Ready and not yet sent Ready **do**
13: Send $\langle \mathsf{Ready} \rangle$ to all parties.
14: **upon** receiving $2t+1$ Ready **do**
15: Wait until $t+1$ valid Echo.
16: Interpolate $\hat{f}_i$ and $\hat{r}_i$ using the set echo_set.
17: **output** $\mathsf{share}_i \leftarrow \hat{f}_i$ and **terminate**

Sharing phase (Algorithm 3). Honest party P_i first checks the correctness of C_i^*, then checks if the received evaluations are valid by $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Verify}$. If all the verifications pass, P_i will set $\mathsf{honest\_dealer} = 1$, and send an Echo message with $f(w_N^j, w_N^i)$, its proof, and $r(w_N^j, w_N^i)$ to other parties. Through a typical "Echo + Ready" approach [3] (Steps 6-16, Algorithm 3), a party who has not received the Share message will get its share by $t+1$ valid Echo messages. When Share_i terminates, P_i gets $\hat{f}(w_N^i, Y)$ as its share.

At the end of the Share phase, every honest party receives at least $2t+1$ valid Echo messages. At least $t+1$ of them are sent by honest parties with $\mathsf{honest\_dealer} = 1$. Further, every such honest P_i has a valid commitment C_i^* and valid polynomial $\hat{f}(x_i, Y)$, so is able to generate a valid proof for $\hat{f}_i(0)$.

Reconstruction phase (Algorithm 4). Every (at least $t+1$) honest P_i with $\mathsf{honest\_dealer} = 1$ sending $\hat{f}_i(0)$ and corresponding proof to other parties would suffice to recover the secret. However, at most t honest parties with $\mathsf{honest\_dealer} = 0$ could not participate in the reconstruction as they cannot generate valid proofs. We propose an alternative to enable these "weaker" parties to contribute to the reconstruction.

Specifically, if P_i is with $\mathsf{honest\_dealer} = 1$, it would send Rec_strong, evaluations, and corresponding proof to other parties. Otherwise, it just sends Rec_weak and evaluations. For another honest party P_j, if it receives at least $t+1$ proofs, it can verify the validity of evaluations and get its share directly by interpolation. If it receives sufficient evaluations (some with proofs but less than t and some without), it can still compute $f(X,0)$ by an efficient RS decoding algorithm such as Berlekamp–Welch, hence computing $f(z,0)$.

Algorithm 4 $\mathsf{Reconstruction}_i()$ for party P_i

1: $\mathsf{valid\_rec} \leftarrow \emptyset$, $\mathsf{decode\_set} \leftarrow \emptyset$.
2: **if** $\mathsf{honest\_dealer} = 1$ **then**
3: $\pi_i \leftarrow \mathsf{Eval.Open}(\mathsf{pp}, C_i^*, \hat{f}_i + \alpha\hat{r}_i, 0)$.
4: Send $\mathsf{rs}_i = \langle \mathsf{Rec\_strong}, \hat{f}_i(0), \hat{r}_i(0), \pi_i \rangle$ to all parties.
5: **if** $\mathsf{honest\_dealer} = 0$ **then**
6: Send $\mathsf{rw}_i = \langle \mathsf{Rec\_weak}, \hat{f}_i(0), \hat{r}_i(0) \rangle$ to all parties.
7: **upon** receiving rs_j for the first time **do**
8: **if** $\mathsf{Eval.Verify}(\mathsf{pp}, C_j^*, 0, \hat{f}_j(0) + \alpha\hat{r}_j(0), \pi_j) = 1$ **then**
9: $\mathsf{valid\_rec} \leftarrow \mathsf{valid\_rec} \cup \{\hat{f}_j(0)\}$.
10: $\mathsf{decode\_set} \leftarrow \mathsf{decode\_set} \cup \{\hat{f}_j(0)\}$.
11: **if** $|\mathsf{valid\_rec}| \geq t+1$ **then**
12: Get $\hat{f}(X,0)$ by interpolation and obtain $\hat{f}(z,0)$.
13: **terminate**
14: **upon** receiving rw_j from P_j for the first time **do**
15: $\mathsf{decode\_set} \leftarrow \mathsf{decode\_set} \cup \{\hat{f}_j(0)\}$.
16: **if** $|\mathsf{decode\_set}| + 2|\mathsf{valid\_rec}| \geq 3t+1$ **then**
17: Get $\hat{f}(X,0)$ by decoding and obtain $\hat{f}(z,0)$.
18: **terminate**

Theorem 4.1. FRISS *is an* (n,t)-AVSS *scheme as in Definition 2.4. Appendix E provides a formal proof.*

Complexity. The dealer complexity mainly comes from generating proofs in Step 3, Algorithm 2 using the one-to-many PolyFRIM, which is $O(n^2 \log n)$ as $t = O(n)$, $N = O(n)$.

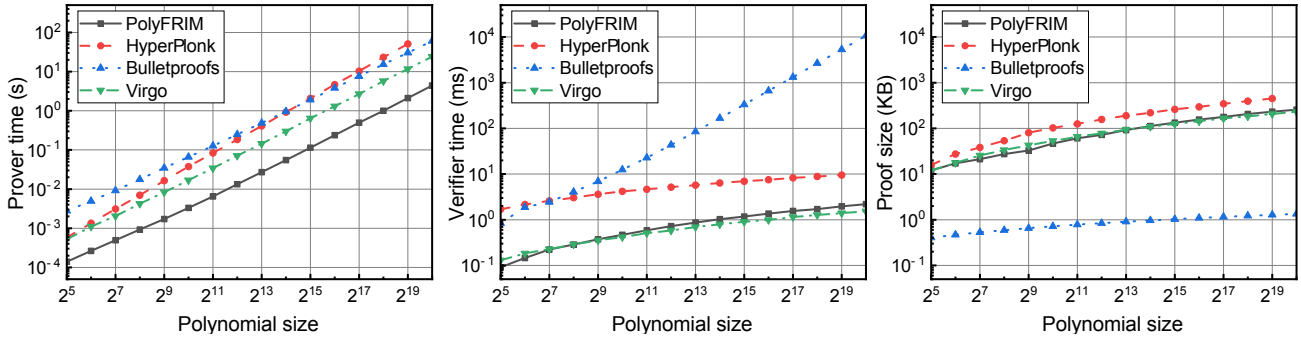
The per-party computation overhead comes from:
(1) verifying proofs for $O(n)$ times using $\mathsf{Eval}_{\mathsf{otm}}.\mathsf{Verify}$ in Steps 3 and 7, Algorithm 3: $O(n) \cdot O(\log^2 n) = O(n \log^2 n)$;
(2) generating proofs using the FRI-PC Eval algorithm in Step 3, Algorithm 4: $O(n)$;
(3) verifying proofs for $O(n)$ times by the FRI-PC Verify algorithm in Step 8, Algorithm 4: $O(n) \cdot O(\log^2 n) = O(n \log^2 n)$.
So, the per-party computation overhead is $O(n \log^2 n)$.

The communication overhead comes from:
(1) the dealer sending shares and proofs in Step 8, Algorithm 2: $O(n) + O(n^2) + O(n^2) \cdot O(\log^2 n) = O(n^2 \log^2 n)$;
(2) the dealer broadcasting commitments in Step 5, Algorithm 2: $O(n^2)$ using reliable broadcast (*e.g.*, [14]);
(3) party sending shares and proofs in Step 5, Algorithm 3 and Step 4, Algorithm 4: $O(n^2) \cdot O(\log^2 n) = O(n^2 \log^2 n)$.
So, the total communication complexity is $O(n^2 \log^2 n)$.

5 Empirical Conclusion

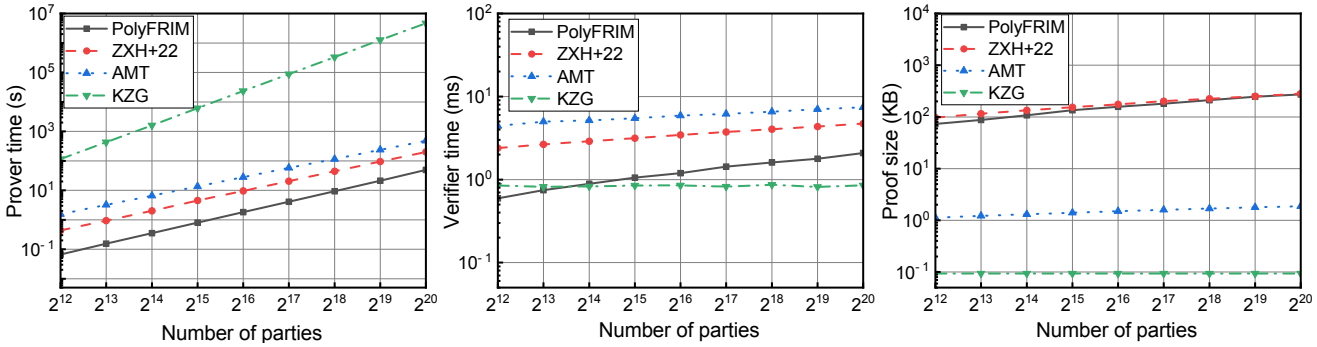
We wrote $\sim$1,600 lines for PolyFRIM and $\sim$1,800 lines for FRISS in Rust. We ran all experiments on an AMD Ryzen 3900X processor with 80GB RAM and Ubuntu 22.04 LTS operating system. Reported figures are averages over 10 executions. We assume no parallelization.

We use $\mathbb{F}_{p^2}$ as our field, $p = 2^{61} - 1$ [37], with a multiplicative coset of size up to 2^{60}. For all FRI-based schemes (Virgo [37],



(a) prover time (b) verifier time (c) proof size

Figure 2: Comparison of transparent multivariate polynomial commitments



(a) prover time (b) per-verifier time (c) per-proof size

Figure 3: Comparison of one-to-many proofs of univariate polynomial commitments with the number of parties as the degree

ZXH$^+$22 [36], HyperPlonk [16], and PolyFRIM), we set the code rate as $1/8$, and the query repetition number as 34, providing 100 bits of security [6, Conjecture 8.4]. Besides, we set the FRI reduction parameter as 2, reducing the polynomial degree by half in each round. Our hash function is blake3.[5]

5.1 Performance of PolyFRIM

We compare PolyFRIM with other transparent PCSs, including Bulletproofs [11][6], Virgo [37][7], and HyperPlonk [16]. As HyperPlonk is not open-sourced, we implement it ourselves.

Figure 2 shows their prover time, verifier time, and proof size. The size of the multivariate polynomial (number of monomials) varies from 2^5 to 2^{20} (as in [36]). We run out of memory for HyperPlonk when the polynomial size is 2^{19}. As shown, for size being 2^{20}, the prover time of PolyFRIM takes only 4.4s, which is 6×, 10×, and 25× faster than Virgo, HyperPlonk, and Bulletproofs, respectively.

PolyFRIM has a 4-10× faster verifier and a 2-3× smaller proof size than HyperPlonk. This speed stems from avoiding FRI-PC by changing the queries from the whole field to the committed domain. Its smaller proof size is due to the rolling batch FRI, which reduces Merkle tree depth and verification path size. The verifier time and proof size of PolyFRIM are competitive with Virgo (less than 50% and 10% worse, respectively). As for Bulletproofs, PolyFRIM has a 100-200× larger proof size but can be 1000× faster for verification.

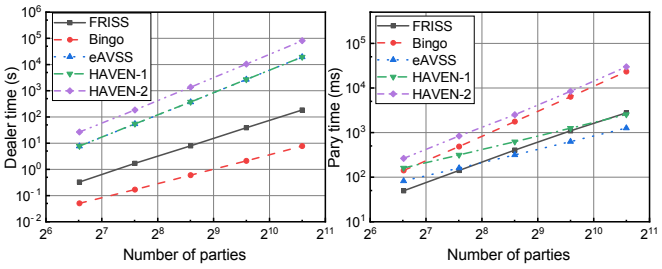
5.2 Performance of One-to-Many PolyFRIM

PolyFRIM is the first multivariate PCS with one-to-many proof. Figure 3a shows the prover time. One-to-many PolyFRIM is 800-10^6× faster than its trivial repetition. It only takes 0.07s (resp., 49.4s) to generate 2^{12} (resp., 2^{20}) proofs for a 2^{12}-degree (resp., 2^{20}-degree) polynomial. Using only lightweight crypto operations (RS codes and hash), our prover time is 25-100× and 10^3-10^6× faster than AMT[8] and KZG, respectively. For one-to-many ZXH$^+$22[9], our prover time is 4-7× faster for eliminating the need for the GKR protocol [21, 36] (and PolyFRIM is faster than Virgo).

As in Figure 2b, our per-verifier time is 3-9× (resp., 2-4×) faster than AMT (resp., ZXH$^+$22). Even compared with constant-verifier KZG, ours is concretely faster for $n < 2^{14}$.

Figure 2c shows the proof size. Ours is larger than trusted-setup-based AMT and KZG. PolyFRIM features up to 20%

[5] github.com/BLAKE3-team/BLAKE3
[6] github.com/dalek-cryptography/bulletproofs
[7] github.com/sunblaze-ucb/Virgo (in C++)
[8] github.com/alinush/libpolycrypto
[9] github.com/sunblaze-ucb/eVSS



(a) dealer time (b) per-party time

Figure 4: Performance of PCSs in different AVSS schemes

smaller proof size than transparent ZXH+22, thanks to its avoidance of additional GKR protocol for delegation.

5.3 Performance of PolyFRIM in AVSS

Figure 4 compares the performance of PCSs in different AVSS schemes (eAVSS [3], HAVEN [2], and Bingo [1]) for varying numbers of parties $n = 3t+1$ and degree-t polynomials.

For FRISS, the dealer generates n^2 proofs for a (t,t)-degree polynomial using PolyFRIM, and each party verifies n proofs. In eAVSS and HAVEN, everything is the same except using different PCSs (KZG for eAVSS and HAVEN-1, and Bulletproofs for HAVEN-2). Following these works, we achieve a bivariate KZG by committing n t-degree univariate polynomials and generating n proofs for every polynomial. For Bingo, the dealer just commits $t+1$ $2t$-degree univariate polynomials, and every party generates n proofs for a $2t$-degree univariate polynomial using KZG.

Figure 4 shows the generation time of commitments and evaluation proofs for the dealer (Steps 2-3 in Algorithm 2) and verification time (Steps 3 and 7 in Algorithm 3) for per party. We omit PCS-irrelevant parts (*e.g.*, broadcast/encryption) to focus on how PCS improves AVSS. The dealer time of FRISS caused by the PCS is 4-800× faster than AVSS using a similar method (eAVSS and HAVEN-1/2). Our per-party time is 0.5-1.2× of eAVSS and HAVEN-1. In Bingo, proof generation is partially "shifted" from the dealer to the parties, so FRISS is 10× slower for the dealer but 10× faster for parties.

References

[1] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, and Gilad Stern. Bingo: Adaptively secure packed asynchronous verifiable secret sharing and asynchronous distributed key generation. In *CRYPTO*, 2023.

[2] Nicolas Alhaddad, Mayank Varia, and Haibin Zhang. High-threshold AVSS with optimal communication complexity. In *FC*, 2021.

[3] Michael Backes, Amit Datta, and Aniket Kate. Asynchronous computational VSS with reduced communication complexity. In *CT-RSA*, 2013.

[4] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *STOC*, 1993.

[5] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, 2018.

[6] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for Reed-Solomon codes. In *FOCS*, 2020.

[7] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT*, 2019.

[8] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC*, 2016.

[9] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In *CCS*, 2020.

[10] Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for diverse environments. In *EUROCRYPT*, 2022.

[11] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *S&P*, 2018.

[12] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT*, 2020.

[13] Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *CCS*, 2002.

[14] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *DISC*, 2005.

[15] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: a toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In *ASIACRYPT*, 2021.

[16] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In *EUROCRYPT*, 2023.

[17] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination & its applications. In *CCS*, 2021.

[18] Sourav Das, Zhuolun Xiang, Alin Tomescu, Alexander Spiegelman, Benny Pinkas, and Ling Ren. Verifiable secret sharing simplified. IACR ePrint 2023/1196.

[19] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *S&P*, 2022.

[20] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR ePrint 2019/953, 2019.

[21] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015.

[22] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *CRYPTO*, 2023.

[23] Bin Hu, Zongyang Zhang, Han Chen, You Zhou, Huazu Jiang, and Jianwei Liu. DyCAPS: Asynchronous dynamic-committee proactive secret sharing. IACR ePrint 2022/1169.

[24] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, 2010.

[25] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. Redshift: Transparent SNARKs from list polynomial commitments. In *CCS*, 2019.

[26] Tohru Kohrita and Patrick Towa. Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. IACR ePrint 23/917.

[27] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *TCC*, 2021.

[28] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *TCC*, 2013.

[29] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *CRYPTO*, 2020.

[30] Victor Shoup and Nigel P. Smart. Lightweight asynchronous verifiable secret sharing with optimal resilience. *J. Cryptol.*, 2024. To appear.

[31] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *S&P*, 2020.

[32] Alexander Vlasov and Konstantin Panarin. Transparent polynomial commitment scheme with polylogarithmic communication complexity. IACR ePrint 2019/1020.

[33] Harry W. H. Wong, Jack P. K. Ma, and Sherman S. M. Chow. Secure multiparty computation of threshold signatures made more efficient. In *NDSS*, 2024.

[34] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. hbACSS: How to robustly share many secrets. In *NDSS*, 2022.

[35] Thomas Yurek, Zhuolun Xiang, Yu Xia, and Andrew Miller. Long live the honey badger: robust asynchronous DPSS and its applications. In *Usenix Sec. (USS)*, 2023.

[36] Jiaheng Zhang, Tiancheng Xie, Thang Hoang, Elaine Shi, and Yupeng Zhang. Polynomial commitment with a one-to-many prover and applications. In *USS*, 2022.

[37] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *S&P*, 2020.

A The FRI Protocol and FRI-PC

Protocol 5 The FRI protocol to test if $\hat{f}|_L \in \mathrm{RS}[L,d/|L|]$

Inputs: $\big((\mathbb{F},L,d),(\hat{f})\big)$. L is a multiplicative coset. Without loss of generality, we assume $d = 2^{\mathsf{r}}$ for a positive integer r.

1: P sets $L_0 = L, \hat{f}_0 = \hat{f}$, computes $\hat{f}_0|_{L_0}$, executes $\mathsf{rt}_0 \leftarrow \mathsf{MT.Commit}(\hat{f}_0|_{L_0})$, and sends rt_0 to $\mathcal{V}$.
2: **for** $i = 1$ to $\mathsf{r}+1$ **do**
3: $\quad\mathcal{P}$ decomposes $\hat{f}_{i-1}$ as $\hat{g}_i(X^2) + X \cdot \hat{h}_i(X^2)$.
4: $\quad\mathcal{V}$ sends $\alpha_i \leftarrow_{\$} \mathbb{F}$ to $\mathcal{P}$.
5: $\quad\mathcal{P}$ computes $\hat{f}_i(X) \leftarrow \hat{g}_i(X) + \alpha_i \cdot \hat{h}_i(X)$.
6: $\quad$If $i \neq \mathsf{r}+1$, $\mathcal{P}$ computes $\hat{f}_i|_{L_i}$ where $L_i = \{x^2 | x \in L_{i-1}\}$, runs $\mathsf{rt}_i \leftarrow \mathsf{MT.Commit}(\hat{f}_i|_{L_i})$, and sends rt_i to $\mathcal{V}$.
7: $\quad$If $i = \mathsf{r}+1$, $\mathcal{P}$ directly sends $\hat{f}_i$ to $\mathcal{V}$.
8: **for** $j = 1$ to $q = O(\lambda)$ **do**
9: $\quad\mathcal{V}$ sends $\beta \leftarrow_{\$} L_0$ to $\mathcal{P}$.
10: $\quad$**for** $i = 1$ to $\mathsf{r}+1$ **do**
11: $\quad\quad\mathcal{P}$ uses $\mathsf{MT.Open}$ to open $\hat{f}_{i-1}(\pm\beta^{2^{i-1}})$, $\hat{f}_i(\beta^{2^i})$.
12: $\quad\quad\mathcal{V}$ checks the correctness by $\mathsf{MT.Verify}$.
13: $\quad\quad\mathcal{V}$ checks if the pairs $(\pm\beta^{2^{i-1}}, \hat{f}_{i-1}(\pm\beta^{2^{i-1}}))$ and $(\alpha_i, \hat{f}_i(\beta^{2^i}))$ are on a common line.
14: If all the above checks pass, $\mathcal{V}$ outputs 1; 0 otherwise.

We recall FRI in Protocol 5, and FRI-PC below.

• $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, d)$: Takes as inputs the security parameter λ and the degree bound d; outputs public parameter pp, which includes the field $\mathbb{F}$ and a multiplicative coset L.

• $C \leftarrow \mathsf{Commit}(\mathsf{pp}, \hat{f})$: Executes $C \leftarrow \mathsf{MT.Commit}(\hat{f}|_L)$.

• $b \leftarrow \mathsf{VerPoly}(\mathsf{pp}, C, \hat{f}|_L, \hat{f})$: Receives $\hat{f}|_L$ as the opening of C; outputs 1 iff the decoding of $\hat{f}|_L$ and $\hat{f}$ are consistent.

• $b \leftarrow \mathsf{Eval}(\mathsf{pp}, C, y, x, \hat{f})$: $\mathcal{P}$ and $\mathcal{V}$ invoke the batch FRI on proving $\hat{f}|_L \in \mathrm{RS}[L, d/|L|] \wedge \hat{f}'|_L \in \mathrm{RS}[L, (d-1)/|L|]$, where $\hat{f}'(X) = (\hat{f}(X) - y)/(X - x)$. $\mathcal{V}$ outputs 1 iff FRI outputs 1.

The batch version of FRI-PC for multiple evaluation points on multiple polynomials can be obtained from the batch FRI.

B HyperPlonk

We recall HyperPlonk, which uses the FRI-PC algorithms $(\mathsf{Gen}_u, \mathsf{Commit}_u, \mathsf{VerPoly}_u, \mathsf{Eval}_u)$.

• $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, \mu)$: Takes as inputs λ and a variable number μ, outputs public parameter $\mathsf{pp} \leftarrow \mathsf{Gen}_u(1^\lambda, n)$ where $n = 2^\mu$.

• $C \leftarrow \mathsf{Commit}(\mathsf{pp}, \tilde{f})$: Takes as inputs pp and $\tilde{f}$, outputs the commitment $C \leftarrow \mathsf{Commit}_u(\mathsf{pp}, \hat{f})$.

• $b \leftarrow \mathsf{VerPoly}(\mathsf{pp}, C, \hat{f}|_L, \tilde{f})$: Runs $\mathsf{VerPoly}_u(\mathsf{pp}, C, \hat{f}|_L, \hat{f})$.

• $b \leftarrow \mathsf{Eval}(\mathsf{pp}, C, \boldsymbol{x}, y, \tilde{f})$: Given pp, C, the evaluation point $\boldsymbol{x} = (x_1, \ldots, x_\mu)$, $y = \tilde{f}(\boldsymbol{x})$, and the multilinear polynomial $\tilde{f}$, $\mathcal{P}$ and $\mathcal{V}$ invoke the following argument:

1: Let $\hat{f}_0(X) = \hat{f}(X)$. For every $i \in [\mu]$, $\mathcal{P}$ first decomposes $\hat{f}_{i-1}(X)$ into $\hat{g}_i(X^2) + X \cdot \hat{h}_i(X^2)$, and then computes $\hat{f}_i(X) \leftarrow \hat{g}_i(X) + x_i \cdot \hat{h}_i(X)$. If $i < \mu$, $\mathcal{P}$ runs $C_i \leftarrow \mathsf{MT.Commit}(\hat{f}_i|_{L_i})$, and sends $\{C_i\}_{i \in [\mu]}$ to $\mathcal{V}$, where $L_i = \{x^2 | x \in L_{i-1}\}$. For $i = \mu$, $\mathcal{P}$ sends $\hat{f}_\mu$ directly to $\mathcal{V}$.

2: $\mathcal{V}$ picks $\beta \leftarrow_\$ \mathbb{F} \setminus \{0\}$ and sends it to $\mathcal{P}$.

3: For every $i \in [\mu]$, $\mathcal{P}$ computes $\hat{f}_{i-1}(\beta)$, $\hat{f}_{i-1}(-\beta)$, and $\hat{f}_i(\beta^2)$. Note that $\hat{f}_\mu$ is a constant polynomial and equals $\tilde{f}(\boldsymbol{x})$. $\mathcal{P}$ and $\mathcal{V}$ then invoke Eval_u to prove the validity of $\{\hat{f}_{i-1}(\beta), \hat{f}_{i-1}(-\beta), \hat{f}_i(\beta^2)\}_{i \in [\mu]}$ except $\hat{f}_\mu(\beta^2)$. If Eval_u outputs 0, $\mathcal{V}$ outputs 0 and aborts.

4: For every $i \in [\mu]$, $\mathcal{V}$ checks if $\hat{f}_i(\beta^2)$ equals $\big(\hat{f}_{i-1}(\beta) + \hat{f}_{i-1}(-\beta)\big)/2 + x_i \cdot \big(\hat{f}_{i-1}(\beta) - \hat{f}_{i-1}(-\beta)\big)/2\beta$. If so, $\mathcal{V}$ outputs 1; otherwise, $\mathcal{V}$ outputs 0 and aborts.

We give a detailed proof of Theorem 2.2 (inspired by [10]).

Proof. **Completeness.** Supposing $\tilde{f}(x_1, \ldots, x_\mu) = y$, we prove that the verifier $\mathcal{V}$ will output 1 in the interactive argument Eval. Setting $\boldsymbol{f} = (a_0, \ldots, a_{2^\mu - 1})$ as the coefficient vector of $\tilde{f}$, we first prove by induction that for all $i \in [0, \mu]$,

$$\hat{f}_i(X) = \sum_{\substack{j \in [0, 2^\mu - 1], \\ (j_1, \ldots, j_\mu)_2 \leftarrow j}} (a_j x_1^{j_1} \cdots x_i^{j_i})(X^{j_{i+1} + 2 \cdot j_{i+2} + \cdots + 2^{\mu - i - 1} \cdot j_\mu}). \tag{3}$$

In Equation (3) and below, $(j_1, \ldots, j_\mu)_2$ denotes the binary representation of j. When $i = 0$,

$$\hat{f}_i(X) = \sum_{j \in [0, 2^\mu - 1]} a_j X^j = \sum_{j \in [0, 2^\mu - 1]} a_j X^{j_1 + 2 \cdot j_2 + \cdots + 2^{\mu - 1} \cdot j_\mu}.$$

Suppose Equation (3) holds when $i = k$, and we next prove Equation (3) also holds when $i = k + 1$. Since $\hat{f}_k(X) = \hat{f}_E^{(k-1)}(X^2) + X \cdot \hat{f}_O^{(k-1)}(X^2)$, we have

$$\hat{f}_E^{(k-1)}(X^2) = \sum_{j \in [0, 2^\mu - 1], j_{k+1} = 0} a_j x_1^{j_1} \cdots x_k^{j_k} X^{2 \cdot j_{k+2} + \cdots + 2^{\mu - k - 1} \cdot j_\mu},$$

$$\hat{f}_O^{(k-1)}(X^2) = \sum_{j \in [0, 2^\mu - 1], j_{k+1} = 1} a_j x_1^{j_1} \cdots x_k^{j_k} X^{2 \cdot j_{k+2} + \cdots + 2^{\mu - k - 1} \cdot j_\mu}.$$

We thus have $\hat{f}_{k+1}(X) \leftarrow \hat{f}_E^{(k-1)}(X) + x_{k+1} \cdot \hat{f}_O^{(k-1)}(X)$ by the procedure of HyperPlonk, then we have $\hat{f}_{k+1}(X)$ equals to $\sum_{j \in [0, 2^\mu - 1]} a_j x_1^{j_1} \cdots x_k^{j_k} x_{k+1}^{j_{k+1}} X^{j_{(k+1)+1} + \cdots + 2^{\mu - (k+1) - 1} \cdot j_\mu}$. So, Equation (3) holds for all $i \in [0, \mu]$.

To see $\hat{f}_\mu(\pm\beta) = y$, we show by induction that $\hat{f}_\mu(X)$ equals to $\sum_{j \in [0, 2^\mu - 1]} a_j x_1^{j_1} \cdots x_\mu^{j_\mu} = \langle \boldsymbol{f}, (1, x_1) \otimes \cdots \otimes (1, x_\mu) \rangle = y$.

Besides, for every $i \in [\mu]$, $\hat{f}_i(X) \leftarrow \hat{f}_E^{(i-1)}(X) + x_i \cdot \hat{f}_O^{(i-1)}(X)$, where $\hat{f}_E^{(i-1)}(X^2) = \big(\hat{f}_{i-1}(X) + \hat{f}_{i-1}(-X)\big)/2$ and $\hat{f}_O^{(i-1)}(X^2) = \big(\hat{f}_{i-1}(X) - \hat{f}_{i-1}(-X)\big)/2X$, hence $\hat{f}_{i+1}(\beta^2) = \big(\hat{f}_i(\beta) + \hat{f}_i(-\beta)\big)/2 + x_i \cdot \big(\hat{f}_i(\beta) - \hat{f}_i(-\beta)\big)/2\beta$.

Polynomial binding. For proximity parameter $\delta \le (1 - \rho)/2$, by the soundness of batch FRI, if the verifier accepts in VerPoly, with probability $1 - \mathsf{negl}(\lambda)$ each opened vector of commitment binds to the encoding of a unique univariate polynomial. The polynomial binding property hence follows.

Soundness. Suppose that $\tilde{f}(x_1, \ldots, x_\mu) \neq y$, and $\hat{f}'_1, \ldots, \hat{f}'_\mu$ are actually committed polynomials. We argue by two cases:

• Case 1. There exists some $\hat{f}'_i$ for $i \in [0, \mu - 1]$ of degree at least $2^{\mu - i}$ or the evaluation proof for $\hat{f}'_i$ is invalid. According to Theorem 2.1, this probability is at most $\varepsilon_{\mathsf{FRI}}$.

• Case 2. Degrees of $\{\hat{f}'_i\}$ are less than $2^{\mu - i}$ for $i \in [0, \mu]$ and the evaluation proofs are all valid. Since $\tilde{f}(\boldsymbol{x}) \neq y$, there must be some i such that $\hat{f}'_i(X) \neq \hat{f}'^{(i-1)}_E(X) + x_i \hat{f}'^{(i-1)}_O(X)$ but it holds for all queries. Assume j is the smallest of such is. By definition, $\hat{f}'^{(j-1)}_E(X^2) = (\hat{f}_{j-1}(X) + \hat{f}_{j-1}(-X))/2$, and $\hat{f}'^{(j-1)}_O(X^2) = (\hat{f}_{j-1}(X) - \hat{f}_{j-1}(-X))/2X$. Set $\hat{g}(X)$ to be:

$$\hat{f}'_j(X^2) - \frac{\hat{f}_{j-1}(X) + \hat{f}_{j-1}(-X)}{2} - x_j \cdot \frac{\hat{f}_{j-1}(X) - \hat{f}_{j-1}(-X)}{2\beta}.$$

$\hat{g}(X)$ is a non-zero polynomial with a degree at most $2^{\mu - j + 1}$, so the soundness error is bounded by $2^{\mu - j + 1}/|\mathbb{F} \setminus \{0\}| \le 2^\mu/|\mathbb{F} \setminus \{0\}|$ according to the Schwartz–Zippel lemma.

Using the union bound argument, the total soundness error of HyperPlonk is $\varepsilon_{\mathsf{FRI}} + 2^\mu/|\mathbb{F} \setminus \{0\}|$. By appropriate parameter choices for $\mathbb{F}, L$, and q, the soundness error can be negligible.

Knowledge soundness. HyperPlonk is an argument of knowledge with extractability of Merkle trees proven in the random oracle model [8, 37]. Roughly, given the root and sufficiently many authentication paths, there exists an efficient extractor that reconstructs the leaves with high probability. Additionally, the leaves are RS codewords and can be efficiently decoded by decoding algorithms such as Berlekamp–Welch.

Specifically, for any PPT adversary $\mathcal{P}^*$ with Commit^* and Eval^*, there exists a PPT extractor $\mathcal{E}$ such that given the random access tape of $\mathcal{P}^*$, the following probability is $\mathsf{negl}(\lambda)$:

$$\Pr\begin{bmatrix} C^* \leftarrow \mathsf{Commit}^*(\mathsf{pp}, \tilde{f}^*) & \tilde{f}^* \leftarrow \mathcal{E}(\mathsf{pp}, C^*, \boldsymbol{x}, y^*) \\ 1 \leftarrow \mathsf{Eval}^*(\mathsf{pp}, C^*, \boldsymbol{x}, y^*, \tilde{f}^*) : & y^* \neq \tilde{f}^*(\boldsymbol{x}) \end{bmatrix}.$$

Suppose the Merkle tree is based on a random oracle $\mathcal{H}: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$. We then construct a PPT extractor $\mathcal{E}$ with the same random type of $\mathcal{P}^*$ working as follows:

1. $\mathcal{E}$ mirrors the process of $\mathcal{P}^*$ querying $\mathcal{H}$. Let the queries made by $\mathcal{P}^*$ to $\mathcal{H}$ be $(q_1, q_2, \ldots, q_{\max})$ in the order they are made with duplicates excluded, which implies that queries are not generated from parent nodes to child ones. Define $q_i \in \mathcal{H}(q_j)$ if the first λ bits or the last λ bits of q_i is $\mathcal{H}(q_j)$, *i.e.*, q_i could be the parent node of q_j. If there exists some $i \neq j$, $\mathcal{H}(q_i) = \mathcal{H}(q_j)$ or some $i \leq j$, $q_i \in \mathcal{H}(q_j)$, $\mathcal{E}$ outputs random polynomial $\tilde{f}$ and aborts.
2. $\mathcal{E}$ constructs a directed graph G according to the query set $Q = \{q_1, \ldots, q_{\max}\}$. There is an edge from q_i to q_j in G if and only if $q_i \in \mathcal{H}(q_j)$. The outdegree of each node is at most 2. When $\mathcal{P}^*$ generates C^* in the Commit^* algorithm, if C^* does not equal $\mathcal{H}(q)$ for some $q \in Q$ with depth $\lceil \mu \rceil$ of the binary tree, $\mathcal{E}$ outputs a random polynomial $\tilde{f}$ and aborts; otherwise we suppose that $\mathcal{H}(q_r) = C^*$ for some r. If any verification path is invalid, $\mathcal{E}$ outputs a random polynomial $\tilde{f}$ and aborts.
3. $\mathcal{E}$ reads all leaves from the root q_r. If there exists any missing leaf, $\mathcal{E}$ outputs a random $\tilde{f}$ and aborts; otherwise, it concatenates these leaf strings and decodes the string using an efficient RS decoding algorithm. Therefore, $\mathcal{E}$ could efficiently output the coefficients of $\tilde{f}$.

Let E_1 be the event $\mathcal{V}$ accepts, and E_2 be $y^* \neq \tilde{f}^*(\boldsymbol{x})$. Consider $\Pr[E_1 \wedge E_2]$. Suppose that $\mathcal{E}$ aborts before constructing the graph G. If for some $i \neq j$, $\mathcal{H}(q_i) = \mathcal{H}(q_j)$, $\mathcal{E}$ finds a collision. If for some $i \leq j$, $q_i \in \mathcal{H}(q_j)$, $\mathcal{E}$ could generate a Merkle tree violating the logical order. Either probability is $\mathsf{negl}(\lambda)$ since $\mathcal{H}$ is collision-resistant and non-invertible.

Otherwise, $\mathcal{E}$ has successfully constructed a graph G. If some node on a verification path does not lie in G, $\mathcal{P}^*$ has to guess the value to construct a valid verification path, and this probability is $\mathsf{negl}(\lambda)$ since $\mathcal{H}$ is non-invertible. Additionally, if any leaf is missing, $\mathcal{V}$ will be convinced with probability $\mathsf{negl}(\lambda)$ once it queries this leaf. The probability that this leaf is not queried by $\mathcal{V}$ is at most $(1-1/|L_0|)^q = \mathsf{negl}(\lambda)$ as the query repetition number of FRI-PC q satisfies $q = O(\lambda)$. If $\mathcal{E}$ does not abort, it could always efficiently extract the coefficient vector of $\tilde{f}$. In this case, $\mathcal{V}$ accepts the statement with probability $\mathsf{negl}(\lambda)$ with an appropriate choice of parameters according to the soundness. So, we can bound $\Pr[E_1 \wedge E_2]$:

$$\begin{aligned} &\Pr[E_1 \wedge E_2 \,|\, \mathcal{E} \text{ aborts}] + \Pr[E_1 \wedge E_2 \,|\, \mathcal{E} \text{ does not abort}] \\ \leq\ &\Pr[\mathcal{E} \text{ aborts}] + \Pr[E_1 \wedge E_2 \,|\, \mathcal{E} \text{ does not abort}] \leq \mathsf{negl}(\lambda). \end{aligned}$$

Complexity. Commit takes $O(n \log n)$, mainly caused by the FFT operation and the construction of the Merkle tree for $\hat{f}|_L$. The other complexities depend on the concrete algorithm.

• Case 1. The prover shows that $\{\hat{f}_i(\pm\beta),\ \hat{f}_i(\beta^2)\}_{i\in[0,\mu]}$ is valid round by round independently. The prover complexity is $\sum_{i\in[0,\log n]} O(n/2^i) = O(n)$. The verifier and communication complexities are both $\sum_{i\in[0,\log n]} O(\log^2(n/2^i)) = O(\log^3 n)$.

• Case 2. The prover invokes a batch FRI to prove the validity of all the evaluations, where all $\{\hat{f}_i\}_{i\in[\mu]}$ are computed and evaluated on the same multiplicative coset. The prover complexity is hence $O(n \log n)$ caused by constructing Merkle trees for $\log n$ n-length vectors. The verifier and communication complexities are both $O(\log^2 n)$ due to the batch FRI. □

C Proof of Theorem 3.1

Proof. The completeness comes directly from that of the FRI protocol. Similar to the proof in Appendix B, the argument of knowledge property holds according to the extractability of Merkle trees. We mainly prove the soundness property below from the following two cases.

• Case 1. If any of the Merkle tree commitments $\{\mathsf{rt}_i\}_{i\in[0,\mu]}$ is not valid, or the leaves are inconsistent with the roots, the verification passes with probability no more than $\mathsf{negl}(\lambda)$.

• Case 2. All the Merkle tree commitments are valid, and the leaves are consistent with corresponding roots. For this case, we first define functions $\{\mathrm{back}_i\}_{i\in[0,\mu]}$ and sets $\{\mathrm{err}_i\}_{i\in[0,\mu]}$.

Define $\mathrm{back}_0(A_0) = A_0$ for $A_0 \subseteq L_0$, and $\mathrm{back}_i(A_i) = \mathrm{back}_{i-1}(A_{i-1})$ for $A_i = \{x^2 | x \in A_{i-1}\}$ and $i \geq 1$.

Define $\mathrm{err}_0 = \emptyset$. For $i \in [\mu]$, let the set E be $\{x \in L_{i-1} : \hat{g}_i(x^2) + \alpha\hat{h}_i(x^2) + \alpha_i^2\hat{f}_i(x^2) \neq \hat{f}^{(i)}(x^2)\}$, where $\hat{f}^{(i-1)}(X) = \hat{g}_i(X^2) + X \cdot \hat{h}_i(X^2)$. Define $\mathrm{err}_i = \mathrm{err}_{i-1} \cup \mathrm{back}_{i-1}(E)$.

For $i \in [\mu]$, suppose that $\hat{h}^{(i)}|_{L_i} \in \mathrm{RS}[L_i, \rho]$ and set $A_i = \{x \in L_i | \hat{f}^{(i)}(x) = \hat{h}^{(i)}(x)\}$. Define $\varepsilon_i(\cdot) : L_i \to \{0,1\}$ such that $\varepsilon_i(A_i) = |\mathrm{back}_i(A_i) \cap \mathrm{err}_i|/|L_0| + (1 - |A_i|/|L_i|)$.

By definition, $|\mathrm{back}_i(A_i)| = 2^i|A_i|$, $A_i \subseteq L_i$, and $\mathrm{err}_i \subseteq L_i$. Further, $\varepsilon_\mu(A_\mu)$ means the probability that the verifier finds inconsistency and rejects with a single query in Step 13 of Protocol 1. Suppose:

$$\Pr[\varepsilon_\mu(A_\mu) \geq (1-\rho)/2] \geq 1 - |L_0|/|\mathbb{F}|. \tag{4}$$

That is, with probability at least $1 - |L_0|/|\mathbb{F}|$, the verifier can catch about $\delta = (1-\rho)/2$ inconsistency if it picks only one query. For q queries, the probability that a malicious prover cheats successfully will be $(1-\delta)^q = ((1+\rho)/2)^q$.

By the union bound argument, the total soundness error is $|L_0|/|\mathbb{F}| + ((1+\rho)/2)^q + \mathsf{negl}(\lambda)$. □

To prove Inequality (4), we first describe three lemmas.

Lemma C.1 ([6]). *For $\delta < (1-\rho)/2$, $\{\hat{f}\}_{i\in[n-1]}$ with degree bounds d and multiplicative coset L where $\rho = d/|L|$, if*

$$\Pr_{\alpha \leftarrow_\$ \mathbb{F}}[\Delta(\sum_{i=0}^{n-1} \alpha^i \cdot \hat{f}_i|_L, \mathrm{RS}[L,\rho]) \leq \delta] \geq |L|/|\mathbb{F}|,$$

where $\Delta(\boldsymbol{a},\boldsymbol{b})$ is the relative Hamming distance between $\boldsymbol{a}$ and $\boldsymbol{b}$, there exists $L' \subset L$ and $\hat{p}_0,\ldots,\hat{p}_{n-1}$ with degree bound d such that $|L'|/|L| \geq 1-\delta$ and $\hat{f}_i|_{L'} = \hat{p}_i|_{L'}$ for $i \in [0,n-1]$.

Lemma C.2. *For any $S_1 \subseteq S_2 \subseteq L_i$, $\varepsilon_i(S_1) \geq \varepsilon_i(S_2)$.*

Proof. By definition, $\varepsilon_i(S_1) - \varepsilon_i(S_2)$

$$= \frac{(|S_2| - |S_1|)}{|L_i|} - \frac{(|\text{back}_i(S_2) \cap \text{err}_i| - |\text{back}_i(S_1) \cap \text{err}_i|)}{|L_0|}$$
$$= |S_2 \backslash S_1|/|D_i| - |(\text{back}_i(S_2) \backslash \text{back}_i(S_1)) \cap \text{err}_i|/|\text{back}_i(L_i)|$$
$$= \frac{|\text{back}_i(S_2 \backslash S_1)|}{|\text{back}(L_i)|} - \frac{|\text{back}_i(S_2 \backslash S_1) \cap \text{err}_i|}{|\text{back}_i(L_i)|} \geq 0. \qquad \square$$

Lemma C.3. *For every $i \in [\mu]$ and $\hat{h}_{L_i}^{(i)} \in \text{RS}[L_i,\rho]$, if $\Delta(\hat{g}_i|_{L_i} + \alpha_i \hat{h}_i|_{L_i} + \alpha_i^2 \hat{f}_i|_{L_i}, \hat{h}^{(i)}|_{L_i}) \geq \theta$, then $\varepsilon_i(A_i) > \theta$.*

Proof. Let $\hat{f} \leftarrow \hat{f}_E^{(i)} + \alpha_i \hat{f}_O^{(i)} + \alpha_i^2 \hat{f}_i$. Define set $B = A_i \cup \{x_i \in L_i | \hat{f}(x) \neq \hat{f}^{(i)}(x)\}$. Define polynomial $\hat{f}'$ such that $\hat{f}'(x) = h^{(i)}(x)$ when $x \in B$ and $\hat{f}'(x) = f(x)$ when $x \notin B$.

For all $x \in B$, if $\hat{f}(x) \neq \hat{f}'(x)$, $\hat{f}(x) \neq \hat{h}^{(i)}(x)$. Besides, for all $x \notin B$, $\hat{f}(x) = \hat{h}^{(i)}(x)$. Thus, the number of inconsistency points in B is not less than the Hamming distance between $\hat{f}|_{L_i}$ and $\hat{f}'|_{L_i}$, *i.e.*, $|\text{back}_i(B) \cap \text{err}_i|/|L_0| \geq \Delta(\hat{f}|_{L_i}, \hat{f}'|_{L_i})$.

Also, as $\hat{f}'(x) = \hat{h}^{(i)}(x)$ holds for all $x \in B$, $1 - |B|/|L_i| \geq \Delta(\hat{f}'|_{L_i}, \hat{h}^{(i)}|_{L_i})$. So, $\varepsilon_i(B) \geq \Delta(\hat{f}|_{L_i}, \hat{f}'|_{L_i}) + \Delta(\hat{f}'|_{L_i}, \hat{h}^{(i)}|_{L_i}) \geq \Delta(\hat{f}|_{L_i}, \hat{h}^{(i)}|_{L_i}) \geq \delta$. As $A_i \subseteq B$, $\varepsilon_i(A_i) \geq \varepsilon_i(B) \geq \delta$. $\square$

Proof of Inequality (4). We prove by induction. When $\mu = 1$, Protocol 1 is a standard FRI, and soundness holds naturally. If Inequality (4) holds $\forall \mu \leq t$, we show it holds for $\mu = t+1$.

When $\mu = t$, the protocol can be seen as a partial protocol starting from $\hat{f}^{(1)}$ for $\mu = t+1$. Set $\delta = (1-\rho)/2$. By induction, when $\mu = t+1$, the following holds:

- If there exists $i \geq 1$, such that $\Delta(\hat{f}^{(i)}|_{L_i}, \text{RS}[L_i,\rho]) \geq \delta$, then $\Pr[\varepsilon_\mu(A_\mu) \geq \delta] \geq 1 - |L_i|/2|\mathbb{F}| > 1 - |L_0|/|\mathbb{F}|$.
- If there exists $i \geq 2$, such that $\Delta(\hat{f}_i|_{L_i}, \text{RS}[L_i,\rho]) \geq \delta$, then $\Pr[\varepsilon_\mu(A_\mu) \geq \delta] \geq 1 - |L_i|/2|\mathbb{F}| > 1 - |L_0|/|\mathbb{F}|$.

Therefore, we only need to prove Inequality (4) in the following case: (a) For all $i \geq 1$, $\Delta(\hat{f}^{(i)}|_{L_i}, \text{RS}[L_i,\rho]) \leq \delta$; (b) For all $i \geq 2$, $\Delta(\hat{f}_i|_{L_i}, \text{RS}[L_i,\rho]) \leq \delta$; (c) At least one of $\Delta(\hat{f}^{(0)}|_{L_0}, \text{RS}[L_0,\rho]) > \delta$ and $\Delta(\hat{f}_1|_{L_1}, \text{RS}[L_1,\rho]) > \delta$ holds.

As (3) holds, there does not exist $A_1' \subseteq L_1$ and $|A_1'|/|L_1| \geq 1-\delta$ such that $\hat{g}_1|_{L_1}, \hat{h}_1|_{L_1}, \hat{f}_1|_{L_1}$ are all δ-close to $\text{RS}[L_1,\rho]$, respectively. Otherwise, $\hat{f}^0|_{L_0}$ and $\hat{f}_1|_{L_0}$ are both close to some $\text{RS}[L_0,\rho]$ by the linearity of RS code, respectively. Therefore, by Lemma C.1, for random α_1 and codeword $\hat{h}^{(1)}|_{L_1} \in \text{RS}[L_1,\rho]$, with probability $1 - |L_0|/|\mathbb{F}|$, $\Delta\big((\hat{g}_1 + \alpha_1 \hat{h}_1 + \alpha_1^2 \hat{f}_1)|_{L_1}, \hat{h}^{(1)}|_{L_1}\big) \geq \delta$. Thus, by Lemma C.3,

$$\Pr[\varepsilon_1(A_1) \geq \delta] \geq 1 - |L_0|/|\mathbb{F}|. \tag{5}$$

Next, we prove that in the case mentioned above, it holds

$$\Pr[\varepsilon_{i+1}(A_{i+1}) \geq \min\{\varepsilon_i(A_i), \delta\}] \geq 1 - |L_i|/|\mathbb{F}|. \tag{6}$$

Note that if Inequalities (5) and (6) hold, Inequality (4) will hold. Below are two cases underlying Inequality (6).

On one hand, if there does not exist $B' \subseteq L_{i+1}$ and $|B'|/|L_{i+1}| \geq 1-\delta$, such that $\hat{f}_E^{(i+1)}|_{B'}, \hat{f}_O^{(i+1)}|_{B'}, \hat{f}_{i+1}|_{B'}$ are some RS codeword, respectively. By Lemma C.1, with probability $1 - |L_{i+1}|/|\mathbb{F}|$, the relative Hamming distance between $(\hat{g}_{i+1} + \alpha_{i+1}\hat{h}_{(i+1)} + \alpha_{i+1}^2 \hat{f}_{i+1})|_{L_{i+1}}$ and $\text{RS}[L_{i+1},\rho]$ is greater than δ. Then by Lemma C.3, $\varepsilon_{i+1}(A_{i+1}) \geq \delta$.

On the other hand, there exists such $B' \subseteq L_{i+1}$ and $|B'|/|L_{i+1}| \geq 1-\delta$. Set $\hat{h}_E^{(i+1)}|_{L_{i+1}}, \hat{h}_O^{(i+1)}|_{L_{i+1}}, \hat{h}_{i+1}|_{L_{i+1}} \in \text{RS}[L_{i+1},\rho]$, and suppose the following relation hold iff $x \in B'$:

$$\hat{g}_{(i+1)}(x) = \hat{h}_E^{(i+1)}(x), \hat{h}_{(i+1)}(x) = \hat{h}_O^{(i+1)}(x), \hat{f}_{i+1}(x) = \hat{h}_{i+1}(x).$$

By properties of random linear combination, with probability at least $1 - |L_{i+1}|/|\mathbb{F}|$, the relative Hamming distance between $(\hat{f}_E^{(i+1)} + \alpha_{i+1}\hat{f}_O^{(i+1)} + \alpha_{i+1}^2 \hat{f}_{i+1})|_{L_{i+1}}$ and $(\hat{h}_E^{(i+1)} + \alpha_{i+1}\hat{h}_O^{(i+1)} + \alpha_{i+1}^2 \hat{h}_{i+1})|_{L_{i+1}}$ equals $1 - |B'|/|L_{i+1}|$. If $\hat{h}^{(i+1)}|_{L_{i+1}} \neq (\hat{h}_E^{(i+1)} + \alpha_{i+1}\hat{h}_O^{(i+1)} + \alpha_{i+1}^2 \hat{h}_{i+1})|_{L_{i+1}}$; by the code distance of the RS code,

$$\Delta(\hat{f}_E^{(i+1)} + \alpha_{i+1}\hat{f}_O^{(i+1)} + \alpha_{i+1}^2 \hat{f}_{i+1})|_{L_{i+1}}, \hat{h}^{(i+1)}|_{L_{i+1}}) \geq \delta.$$

Hence, by Lemma C.3, $\varepsilon_{i+1}(A_{i+1}) \geq \delta$. Therefore, below, we only need to consider the following case:

$$\hat{h}^{(i+1)}|_{L_{i+1}} = (\hat{h}_E^{(i+1)} + \alpha_{i+1}\hat{h}_O^{(i+1)} + \alpha_{i+1}^2 \hat{h}_{i+1})|_{L_{i+1}}.$$

In this case, for $x \in D_{i+1} \backslash B$, with probability at least $1 - |L_{i+1}|/|\mathbb{F}|$, $\hat{h}^{(i+1)}(x) \neq \hat{f}_E^{(i+1)}(x) + \alpha \cdot \hat{f}_O^{(i+1)}(x) + \alpha^2 \hat{f}_{i+1}(x)$. Let $B = \{x \in L_i : x^2 \in B'\}$. By definition, $B \subseteq A_i$. Define set $C = A_{i+1} \cup B'$. To prove that $\varepsilon_{i+1}(A_{i+1}) \geq \varepsilon_i(A_i)$, it suffices to prove $\varepsilon_{i+1}(C) \geq \varepsilon_i(B)$ according to Lemma C.2 as $B \subseteq A_i$ and $A_{i+1} \subseteq C$. By the definition of B', for every $x \in A_{i+1} \backslash B'$,

$$\hat{f}^{(i+1)}(x) = \hat{h}^{(i+1)}(x) \neq \hat{g}_{(i+1)}(x) + \alpha \cdot \hat{g}_{(i+1)}(x) + \alpha^2 \hat{f}_{i+1}(x).$$

So, $\text{back}_{i+1}(A_{i+1} \backslash B') = \text{back}_{i+1}(C \backslash B') \subseteq \text{err}_{i+1}$. Define sets $M = \text{back}_i(B) \cap \text{err}_i$ and $N = \text{back}_{i+1}(C \backslash B')$. As $\text{err}_i \subseteq \text{err}_{i+1}$,

$$M \cup N \subseteq \text{err}_i \cup \text{back}_{i+1}(C \backslash B') \subseteq \text{err}_{i+1}. \tag{7}$$

Besides, as $B' \subseteq C$ and $\text{back}_i(B) = \text{back}_{i+1}(B')$, we have

$$M \cup N = (\text{back}_{i+1}(B') \cap \text{err}_i) \cup N \subseteq \text{back}_{i+1}(C). \tag{8}$$

By Relations (7) and (8),

$$M \cup N \subseteq \text{back}_{i+1}(C) \cap \text{err}_{i+1}. \tag{9}$$

Since $\text{back}_i(B) = \text{back}_{i+1}(B')$,

$$M \cap N \subseteq \text{back}_{i+1}(B') \cap \text{back}_{i+1}(C \backslash B') = \emptyset. \tag{10}$$

According to Relation (10), Relation (9) can be rewritten as

$$|\text{back}_{i+1}(C) \cap \text{err}_{i+1}| \geq |\text{back}_{i+1}(C \backslash B')| + |\text{back}_i(B) \cap \text{err}_i|.$$

Since $\frac{|\text{back}_{i+1}(C\backslash B')|}{|D|} = \frac{|C\backslash B'|}{|D_{i+1}|} = \frac{|C|}{|D_{i+1}|} - \frac{|B|}{|D_i|}$, we have

$$|\text{back}_{i+1}(C)\cap \text{err}_{i+1}| - \frac{|C|}{|D_{i+1}|} \geq |\text{back}_i(B)\cap \text{err}_i| - \frac{|B|}{|D_i|}.$$

So $\varepsilon_{i+1}(C) \geq \varepsilon_i(B)$, Inequalities (6) and (4) hold. □

D Proof of Theorem 3.2

Proof. **Completeness and polynomial binding.** Polynomial binding directly comes from HyperPlonk. The main differences between PC_m and HyperPlonk lie in the usage of rolling batch FRI and random polynomial $\tilde{r}$. The completeness of PC_m follows from that of HyperPlonk and rolling batch FRI.

Soundness. We bound the soundness error via two cases.

• Case 1. There exists some polynomial $\hat{f}_i$ for $i \in [0,\mu-1]$, which has a degree at least $2^{\mu-i}$. According to Theorem 3.1, the verification passes with a probability bounded by $\varepsilon_{\mathsf{FRI}}$.

• Case 2. For every $i \in [0,\mu-1]$, the degree of $\hat{f}_i$ is less than $2^{\mu-i}$. Suppose that $\tilde{f}(\boldsymbol{x}) \neq y$. We have $\hat{f}(\boldsymbol{x}) + \alpha\cdot\hat{r}(\boldsymbol{x}) = y + \alpha\cdot y_r$ holds with a probability bounded by $1/|\mathbb{F}|$ due to the randomness of α. Otherwise, there must exist some i such that $\hat{f}_i(X) \neq \hat{g}_{i-1}(X) + x_i\cdot\hat{h}_{i-1}(X)$ but for every query, the equation holds. Assume that j is the smallest of these i's. By definition, $\hat{g}_{j-1}(X^2) = \big(\hat{f}_{j-1}(X) + \hat{f}_{j-1}(-X)\big)/2$, and $\hat{h}_{j-1}(X^2) = \big(\hat{f}_{j-1}(X) - \hat{f}_{j-1}(-X)\big)/2X$. Set $\hat{\phi}(X)$ to be

$$\hat{f}_j(X^2) - \frac{\hat{f}_{j-1}(X)+\hat{f}_{j-1}(-X)}{2} - x_j\cdot\frac{\hat{f}_{j-1}(X)-\hat{f}_{j-1}(-X)}{2\beta}.$$

As $\hat{f}_j(X) \neq \hat{g}_{j-1}(X) + x_j\cdot\hat{h}_{j-1}(X)$, $\hat{\phi}$ is a non-zero polynomial of degree less than $2^{\mu-j}$. By the Schwartz–Zippel lemma, the verification passes with a probability of no more than $(2^{\mu-j}/|L_j|)^q \leq (2^\mu/|L_0|)^q$.

By the union bound argument, the total soundness error is less than $\varepsilon_{\mathsf{FRI}} + (2^\mu/|L_0|)^q + 1/|\mathbb{F}|$ for $\varepsilon_{\mathsf{FRI}}$ in Theorem 3.1.

Knowledge soundness. Similar to HyperPlonk, knowledge soundness comes from the extractability of Merkle trees and efficient decoding algorithms for RS code. We omit the proof.

$2q$-bound knowledge leaking. Figure 5 gives the simulator. In Step 3 of Eval_m, a random $\hat{f}'$ can always be found when $2q+m \leq 2^\mu - 1$. The consistency check in the first round always passes as only evaluations corresponding to $\{\pm\beta_i\}_{i\in Q}$ would be opened and $\hat{f}(\pm\beta_i) = \hat{f}'(\pm\beta_i)$. In addition, $\{\hat{f}'_i\}_{i\in[\mu]}$ are valid, so consistency checks in other rounds will pass. Due to random $\tilde{r}$, $\hat{f}'_i$ is uniformly distributed. So, no verifier can notice the simulation with non-negligible probability. □

E Proof of Theorem 4.1

Proof. **Liveness.** If the dealer P_d is honest, then every party will receive their share in the Share phase, so there will be at least $2t+1$ valid Echo messages sent by honest parties. Also, every honest party will send a Ready message, which means every honest party will complete the Share phase.

Agreement. Suppose an honest party P_i completes the Share phase; we consider the case for every other honest party P_j. We first prove that they will receive $2t+1$ Ready's. Completion means P_i must have received $2t+1$ Ready's according to Step 14 in Algorithm 3 with at least $t+1$ sent by honest parties. From Step 12, if an honest party who has not sent Ready receives $t+1$ Ready's, it will send Ready. Thus, every honest party will eventually receive at least $2t+1$ Ready's.

Next, we prove every honest party will receive $t+1$ Echo messages. An honest party sends Ready only via Step 10 or Step 12. If all honest parties send Ready by Step 12, there must be $t+1$ Ready's sent by dishonest parties, which is impossible. Thus, at least one honest party has sent Ready by Step 10. Consequently, at least $t+1$ honest parties have sent the Echo's. So, P_j will eventually receive $t+1$ valid Echo's. It can then obtain its share f_j and complete the Share phase.

Correctness. If the dealer P_d is honest, then $f(X,Y)$ only corresponds to a set of the same secret that $s = f(z,0)$, so honest parties will reconstruct the same secret. If an honest party P_i reconstructs z_i, every honest party receives at least $t+1$ valid Echo messages corresponding to a common commitment of a bivariate polynomial, so they reconstruct the same secret.

Secrecy. If the dealer P_d is honest and the corrupted set of $\mathcal{A}$ is $\{P_{c_i}\}_{i\in[t-\ell]}$, then $\mathcal{A}$ will get univariate polynomials $\hat{f}(w_N^{c_i}, Y)$ and $\hat{f}(X, w_N^{c_i})$ for all $i \in [t-\ell]$ along with the proof of these polynomials' evaluations. According to Theorem 3.2, the evaluation proof only leaks ℓ queried evaluations of $\hat{f}$. As the variable degree of X and Y are both t, $\mathcal{A}$ has no extra information about polynomial $\hat{f}(X,0)$ and the secret $s = f(z,0)$. □

• $C \leftarrow \mathcal{S}_1\left(1^\lambda, \mathsf{pp}, r_{\mathcal{A}}\right)$

1: Get $\{\pm\beta_i\}_{i\in Q}$ from $r_{\mathcal{A}}$ and $\{\hat{f}(\pm\beta_i)\}_{i\in[Q]}$ by the oracle access to $\hat{f}$, where Q is the set of all queries.
2: Construct a map $\hat{F}: L_0 \to \mathbb{F}$ such that $\hat{F}(\pm\beta_i) = \hat{f}(\pm\beta_i)$ for every $i \in Q$, and take random values on other points.
3: Randomly pick a multilinear polynomial $\tilde{r}'(X_1,\ldots,X_\mu)$.
4: Output $C'_f \leftarrow \mathsf{MT.Commit}(\hat{F}|_{L_0})$ and $C'_r \leftarrow \mathsf{MT.Commit}(\hat{r}'|_{L_0})$.

• $\mathcal{S}_2\big(\mathsf{pp}, \{C'_f, C'_r\}, \{\boldsymbol{x}_j = (x_1^{(j)},\ldots,x_\mu^{(j)})\}_{j\in[m]}\big)$

1: Send $\{y_j, \tilde{r}(\boldsymbol{x}_j)\}_{j\in[m]}$ to $\mathcal{V}$ given oracle access to $y_j = \tilde{f}(\boldsymbol{x}_j)$ for all $j \in [m]$.
2: Receive α from $\mathcal{V}$, and generate a random polynomial $\hat{f}'$ with coefficient vector $\boldsymbol{f}'$ such that for every $i \in Q$, $\hat{f}'(\pm\beta_i) = \hat{f}(\pm\beta_i)$ and for every $j \in [m]$, $\langle \boldsymbol{f}', (1, x_1^{(j)}) \otimes \cdots \otimes (1, x_\mu^{(j)})\rangle = \tilde{f}(\boldsymbol{x}_j)$. Let $\hat{f}'_0(X) = \hat{f}'(X) + \alpha\cdot\hat{r}'(X)$. $\mathcal{S}_2$ acts like an honest prover with input $\hat{f}'_0$.

Figure 5: Simulator $\mathcal{S}$ of the PC_m