ReedWeave: Faster Reed-Solomon Polynomial Commitments from Interleaving and Folding

Yuhao Jia[1], Zhe Li[2], Chaoping Xing[1], Yizhou Yao[1], Chen Yuan[1], Jielong Zhang[1]

[1] School of Computer Science,
Shanghai Jiao Tong University, Shanghai, China
{jyh1529400,xingcp,yaoyizhou0620,chen_yuan,jielongzhang}@sjtu.edu.cn
[2] Institute of Mathematics and Interdisciplinary Sciences,
Xidian University, Xi'an, China
lizh0048@e.ntu.edu.sg

Abstract. Polynomial commitment schemes (PCSs) allow a prover to commit to a polynomial and later prove its evaluations succinctly. Among hash-based constructions, FRI (Ben-Sasson et al., ICALP 2018) and its follow-up works achieve polylogarithmic proofs by recursively "folding" Reed-Solomon (RS) codes. However, their concrete evaluation costs remain relatively high, as the first few folding rounds operate on the largest codewords and dominate the prover work. Another line of works, initiated by Ligero (Ames et al., CCS 2017) construct PCSs from interleaved codes. Although through "interleaving", RS encoding seems no longer to be the major bottleneck, their approach destroys the foldable structure, consequently only yielding a square-root proof size.

We present *ReedWeave*, a Reed-Solomon PCS that successfully incorporates interleaving with folding, achieving the best of the two worlds. ReedWeave decomposes a degree-d polynomial into m smaller components and commits to their RS encodings over a common domain. Under our novel decomposition, a random linear combination of these codewords is exactly an m-ary folding. At a high level, ReedWeave starts with an m-ary folding, followed by standard binary folding as in FRI. The prover costs are substantially reduced in the sense that at the very beginning it suffices to do interleaved-RS encoding rather than standard RS encoding. That is, we reduce prover costs from $\mathcal{O}(d \log d)$ to $\mathcal{O}(d \log(d/m))$ for constant code rate while maintaining efficient polylogarithmic verification. Moreover, our approach essentially works for arbitrary m, relaxing the smoothness requirements of the underlying fields by a multiplicative factor of m.

We benchmark our Rust implementation over Goldilocks using 32 threads and targeting 100-bit security. At $d = 2^{24}$, $\rho = 1/4$, and $m = 64$, the DEEP variant commits in 475.0 ms, opens in 111.6 ms, verifies in 2.07 ms, and produces a 594.7 KiB proof. Compared with FRI (ICALP 2018), commitment, opening, and verification are 6.9×, 51.5×, and 4.2× faster, respectively. Against STIR (CRYPTO 2024), the corresponding speedups are 2.9×, 68.5×, and 2.1×. Its proof is only 0.69× the size of FRI's and 2.31× that of STIR's.

1 Introduction

A polynomial commitment scheme (PCS) [KZG10] allows a prover to commit succinctly to a polynomial and later prove its evaluations without sending the full polynomial. By emulating access to polynomial oracles through commitments and evaluation proofs, PCSs are central building blocks for succinct arguments [BFS20,GLS+23,ZCF24]. Among PCS constructions based on different cryptographic assumptions, code-based approaches are attractive for their transparent setup and fast provers based on lightweight hashing and field arithmetic [AHIV17,BCR+19,ZXZS20,BFH+20,XZS22]. Code-based polynomial commitments are typically evaluated by four main costs: commitment time, evaluation time, verifier time, and proof size. These costs and the underlying fields are largely determined by the underlying code [GLH+25,BCF+25,BFRW26].

Foldable-code PCSs. One line of work builds polynomial commitments from codes that support recursive folding. FRI [BBHR18] uses Reed-Solomon (RS) codes and recursively folds a large RS codeword into progressively smaller ones, achieving polylogarithmic proof size and verifier time. Subsequent constructions such as STIR [ACFY24] and WHIR [ACFY25] improve the query complexity and evaluation protocol, while BaseFold [ZCF24] generalizes the folding paradigm beyond RS codes to field-agnostic code families. These constructions provide succinct verification, but the resulting prover cost remains a major practical expense, particularly in the first rounds where the codewords are largest.

Interleaved-code PCSs. A different approach, used by Ligero [AHIV17] and Brakedown [GLS+23], organizes a large input as several shorter codewords and commits to them in an interleaved form. Working with shorter codewords can substantially reduce prover cost, and Brakedown further combines this construction with linear-time encodable codes. The tradeoff is less succinct verification. In these frameworks, proof size and verifier time typically grow roughly with the square root of the input size. Code switching [RZR24] provides a way to recover succinct verification. Constructions such as Blaze [BCF+25] and BrakingBase [NST25] first use a fast encodable code and then prove a transition to a foldable code with polylogarithmic verification. However, this transition introduces an additional proof layer and increases evaluation and verification costs. Consequently, the benefit becomes pronounced only for sufficiently large polynomials.

For interleaved RS codes, however, the starting codewords already have the algebraic structure required by FRI folding. This raises the following natural question.

> *Can we acheive the prover effiency of interleaving and the succinct verifier of FRI simutaniously?*

Table 1. Code-based PCS comparison. Here d is the polynomial size and λ the security parameter. Rates and proximity parameters are constant. For ReedWeave, m is the interleaving parameter and ρ is the RS rate.

Scheme	Code	Subgroup required	Commit time	Open time	Verifier time / Proof size
FRI [BBHR18]/ DEEP-FRI [BGKS20]	RS	d/ρ	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(\lambda \log^2 d)$
Shockwave [AHIV17]/ Ligero [GLS+23]	RS	$\Theta(\sqrt{\lambda d}/\rho)$	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(\sqrt{\lambda d})$
BaseFold [ZCF24]	RFC	None	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(\lambda \log^2 d)$
Blaze [BCF+25]	Packed RAA	None†	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(\lambda \log^2 d)$
Brakedown [GLS+23]	Spielman	None	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(\sqrt{\lambda d})$
BrakingBase [NST25]	Spielman	None	$\mathcal{O}(d)$	$\mathcal{O}(d)$	$\mathcal{O}(\lambda \log^2 d)$
QAPCS [LLX+26]	QA	None	$\mathcal{O}(d \log d)$	$\mathcal{O}(d)$	$\mathcal{O}(\sqrt{\lambda d})$
ReedWeave (Ours)	RS	$d/(m\rho)$	$\mathcal{O}(d \log \frac{d}{m})$	$\mathcal{O}(d)$	$\mathcal{O}(\lambda(m + \log^2 \frac{d}{m}))$

Codes. RS = Reed-Solomon codes. RFC = Random foldable codes. RAA = Repeat-Accumulate-Accumulate. QA = Quasi-Abelian. Blaze and BrakingBase use auxiliary BaseFold-style proofs.

Fields. Subgroup orders refer to power-of-two multiplicative subgroups. "None" does not waive soundness-related field-size requirements. † Blaze uses binary extension fields and auxiliary additive RS domains.

Costs. Time counts field/hash operations, while size counts field elements/hash values. Public preprocessing is excluded. Square-root bounds use optimized row length $\Theta(\sqrt{\lambda d})$.

1.1 Our Results

We answer this question affirmatively with *ReedWeave*, a PCS for univariate polynomials. Our approach incorporates three main techniques. First, we allow smaller encoding domains by an interleaved RS encoding. We then interleave evaluation-claim reductions with FRI proximity testing, using the same verifier challenges to fold the oracle and reduce the evaluation claim in each round. Finally, we apply the DEEP mechanism [BGKS20] to support soundness analyses in the Johnson regime, which allows fewer query repetitions. Building on prior work [AHIV17,GLS+23,BCF+25,ZZW+26], these ingredients yield two main results.

For $d = mk$ and rate ρ, ReedWeave decomposes the polynomial $f(X) = \sum_{i=0}^{m-1} X^i f_i(X^m)$ into m polynomials $f_0(X), \ldots, f_{m-1}(X)$ of degree less than k and encodes them over a common subgroup of order $N = k/\rho$, rather than encoding the full polynomial over a domain of size d/ρ. To connect the evalua-

tions of $f(X)$ with the evaluations of $f_0(X), \ldots, f_{m-1}(X)$, we show that if there exists a group $\mathcal{L} \subseteq \mathbb{F}_q^*$ has order 2^n, then there always exists a group $\mathcal{L}' \subseteq \mathbb{F}_{q^t}^*$ of order $2^n \cdot m$ containing $\mathcal{L}$. Thus, to fold $f(X)$ at a point $\alpha \in \mathcal{L}'$, the FRI protocol needs to query $f(\alpha\omega^i), i = 0, \ldots, m-1$ where ω is an element of order m. This is equivalent to query $f_0(\alpha^m), \ldots, f_{m-1}(\alpha^m)$ with $\alpha^m \in \mathcal{L}$. Therefore, it suffices to compute and commit the evaluations of $f_0(X), \ldots, f_{m-1}(X)$ on $\mathcal{L}$ for the initial folding.

One may find out that the initial folding in ReedWeave is very similar to Brakedown protocol. It is indeed the case and not surprisingly ReedWeave has very fast prover time. Moreover, after the initial folding, we obtain a polynomial $g(Y) = \sum_{i=0}^{m-1} \gamma^i f_i(Y)$ of degree less than k over the challenge γ. Because we can treat the initial folding as part of FRI protocol, we can also follow the protocol to fold $g(Y)$ multiple times until obtaining a constant-degree polynomial. This guarantees ReedWeave a short proof size comparable to the FRI protocol. Another merit of ReedWeave is that we require a relatively small subgroup for folding which might be interesting for some specific finite field $\mathbb{F}_q$. For example, at $d = 2^{24}$ and $\rho = 1/4$, choosing $m = 64$ reduces the required subgroup order from 2^{26} to 2^{20}. Table 1 compares representative code-based commitments by their codes, subgroup requirements, and asymptotic costs.

Since we do not compute the evaluation of $f(X)$ directly, it is impossible to prove $f(z) = y$ by showing that $\frac{f(X)-y}{X-z}$ has low degree. ReedWeave proves the claim $f(z) = y$ in a folding manner, i.e., our protocol folds both the FRI oracle and the evaluation claim. Instead of sending $f(z) = y$, the prover sends $f_i(z^m) = v_i$ for $i = 1, \ldots, m$ and the verifier checks $\sum_{i=0}^{m-1} z^i v_i = y$. Then, we fold $f(X)$ to $g(Y) = \sum_{i=0}^{m-1} \gamma^i f_i(Y)$ and reduce the claim $f(z) = y$ to $g(z^m) = \sum_{i=0}^{m-1} \gamma^i v_i$. One can find the details in Section 1.2.

By exploiting DEEP we also enjoy proximity parameters that are arbitrarily close to the Johnson radius, which reduce the total number of queries. Our single-thread and 32-thread measurements show lower total prover time than the tested FRI, STIR, WHIR, and BaseFold implementations, with the verification and communication tradeoffs quantified below.

Our main result is the following.

Theorem 1 (Informal). *Let $\lambda, m \geq 1$ and $d = mk$, with $k \geq 4$ a power of two. Fix constants $\rho \in (0, 1)$ and $0 < \eta < 1 - \sqrt{\rho}$. If $\mathbb{F}$ is sufficiently large and contains a multiplicative subgroup of power-of-two order $N = d/(m\rho)$, ReedWeave's DEEP variant has a binding commitment and a perfectly complete evaluation IOP with round-by-round knowledge soundness at $\delta = 1 - \sqrt{\rho} - \eta$, total binding and knowledge error at most $2^{-\lambda}$, and $\mathcal{O}(\log k)$ rounds. With Merkle instantiation, it has the following properties.*

- Commitment time. $\mathcal{O}(d \log N)$ *field operations and* $\mathcal{O}(mN)$ *hash operations.*
- Open time. $\mathcal{O}(d)$ *field operations and* $\mathcal{O}(N)$ *hash operations.*
- Verification time. $\mathcal{O}(\lambda(m + \log^2 N))$ *field and hash operations.*
- Proof size. $\mathcal{O}(\lambda(m + \log^2 N))$ *field elements and hash values.*

For fixed security and rate parameters, verification time and proof size remain polylogarithmic in d when m is polylogarithmic in d. ReedWeave thus relaxes, rather than eliminates, the multiplicative-subgroup requirement.

Implementation and evaluation. We implement both variants in Rust and benchmark them over Goldilocks, i.e., $\mathbb{F}_{2^{64}-2^{32}+1}$, targeting 100-bit security. At $d = 2^{24}$ and rate 1/4 with 32 threads, ReedWeave$_{\mathsf{JB}}$, our DEEP variant, takes 586.6 ms in total prover time, verifies in 2.07 ms, and produces a 594.7 KiB proof. It achieves total-prover speedups of 15.4× over the tested FRI implementation and 13.8× over BaseFold, with lower verification time and smaller proofs than both. By reducing the required subgroup size by up to 64×, ReedWeave also scales to $d = 2^{34}$ over Goldilocks, whereas FRI, STIR, and WHIR cannot reach this scale due to the field's 2^{32} two-adicity limit. Figure 1 summarizes representative costs, and Section 5 includes other metrics such as commit time, open time and verifier time.

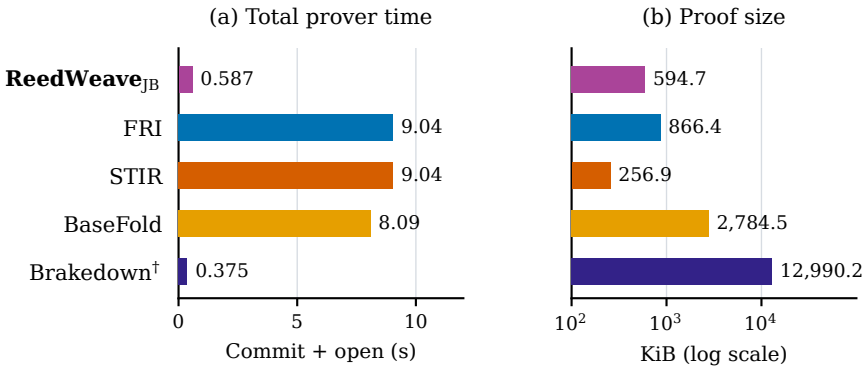


Fig. 1. PCS costs at $d = 2^{24}$ over Goldilocks, targeting 100-bit security with 32 threads. Values are averages of five measurements. The rate is 1/4, except for Brakedown (†), which uses approximately 0.657. ReedWeave$_{\mathsf{JB}}$ uses $m = 64$, requiring subgroup order 2^{20} rather than 2^{26} for direct encoding. Total prover time is commitment plus opening. Proof sizes include commitments and use a logarithmic axis. Lower is better.

1.2 Technical Overview

We first recall the folding operation of FRI. We then explain how interleaved RS encoding avoids the full-size domain, how a shared challenge couples evaluation and proximity checks, and how DEEP improves query efficiency without changing the encoding.

FRI overview. FRI [BBHR18] is an interactive oracle proof of proximity (IOPP) for Reed-Solomon codes. Let $\mathbb{F}$ have odd characteristic, and let $\mathcal{L}$ be a multiplicative subgroup of power-of-two order N. For a polynomial g of degree less than a power-of-two bound $k < N$, binary folding writes

$$g(X) = g_{\mathsf{e}}(X^2) + Xg_{\mathsf{o}}(X^2), \qquad g_\gamma = g_{\mathsf{e}} + \gamma g_{\mathsf{o}},$$

where $\gamma \in \mathbb{F}$ is a random verifier challenge. The folded polynomial has degree less than $k/2$ and is evaluated over the squared domain $\mathcal{L}^2$, preserving the rate $\rho = k/N$. Each folded oracle value can be computed locally from the values at x and $-x$. Repeating this operation reduces the instance to a constant-degree polynomial, which the prover sends directly. After that, the verifier queries the oracles to check consistency between successive folds.

ReedWeave retains this recursion but changes its starting point. Rather than constructing the full-size RS oracle, it starts from a smaller virtual oracle obtained from an interleaved commitment.

Starting from an interleaved commitment. ReedWeave begins by representing the polynomial so that the first fold can be computed without materializing the full-size RS codeword. Let $d = mk$ and write

$$f(X) = \sum_{i=0}^{m-1} X^i f_i(X^m),$$

where each f_i has degree less than k. ReedWeave encodes the m component polynomials independently over the same RS domain $\mathcal{L}$ of size $N = k/\rho$. Their evaluations form an $m \times N$ matrix $\widehat{U}$, whose column at $x \in \mathcal{L}$ is

$$\widehat{U}(x) = \big(f_0(x), \ldots, f_{m-1}(x)\big)^{\mathsf{T}}.$$

ReedWeave commits to this matrix column-wise using a Merkle tree, so the commitment fixes all component evaluations over $\mathcal{L}$. During the evaluation protocol, the verifier samples a random challenge α. This challenge defines

$$g^{(0)} = \sum_{i=0}^{m-1} \alpha^i f_i,$$

which has degree less than k. Note that ReedWeave does not commit separately to the RS codeword of $g^{(0)}$. Instead, whenever the verifier queries a point $x \in \mathcal{L}$, the prover opens the authenticated column $\widehat{U}(x)$, and then the verifier computes

$$g^{(0)}(x) = \sum_{i=0}^{m-1} \alpha^i \widehat{U}_i(x).$$

The RS codeword of $g^{(0)}$ therefore serves as a *virtual oracle*, and the subsequent FRI folding starts directly from this instance with degree bound k over the domain $\mathcal{L}$.

Commitment requires m FFTs of length N, for a total of $\mathcal{O}(mN \log N)$ field operations. The total encoding length remains $mN = d/\rho$, but the prover performs shorter FFTs and avoids constructing the full-size RS oracle.

Viewing the first step as an m-ary FRI fold. The preceding construction admits a useful interpretation in terms of FRI. Conceptually, consider a compatible multiplicative subgroup $\mathcal{L}'$ of size mN such that the map $x \mapsto x^m$ maps $\mathcal{L}'$ onto the smaller domain $\mathcal{L}$. Such a domain always exists over a suitable finite extension whenever m is coprime to the field characteristic. More precisely, if $\mathcal{L}$ has order N, then some finite extension contains an element ω of order mN satisfying $\mathcal{L} = \langle \omega^m \rangle$. We give the detailed proof in Section A.3.

For every $y \in \mathcal{L}$, its m preimages in $\mathcal{L}'$ form one folding block of an m-ary FRI step. The component values $(f_0(y), \ldots, f_{m-1}(y))$ provide an alternative representation of exactly the same block information, and the two representations are related by an invertible linear change of basis.

Ordinary FRI stores the block evaluations and computes their folded value after receiving α. ReedWeave instead commits directly to the component representation. Once α is sampled, taking the linear combination $\sum_i \alpha^i f_i(y)$ produces exactly the value of the corresponding m-ary FRI fold. Thus, ReedWeave does not remove the first fold itself. It removes the need to materialize the large oracle *before* that fold.

This viewpoint explains both the prover savings and the smaller subgroup requirement. ReedWeave stores only the information needed to compute every possible first folded oracle, while encoding each component over the smaller domain of size N. The compatible domain of size mN is used only for this interpretation and is never constructed by the protocol. Consequently, ReedWeave requires only the smaller RS domain while retaining the same folded polynomial that ordinary arity-m FRI would produce.

Folding evaluation claims. The virtual oracle $g^{(0)}$ over $\mathcal{L}$ lets us test proximity on a smaller domain, but proximity alone does not establish the claimed evaluation $f(z) = y$. To connect the claim to this oracle, the prover sends values v_i claimed to equal $f_i(z^m)$. The verifier checks $y = \sum_i z^i v_i$ before sampling the initial folding challenge α. The same challenge that defines $g^{(0)} = \sum_i \alpha^i f_i$ then reduces the component claims to $g^{(0)}(z^m) = \sum_i \alpha^i v_i$.

Each binary round follows the same principle. Given a current claim $g(u) = w$, the prover sends claimed evaluations $a = g_{\mathsf{e}}(u^2)$ and $b = g_{\mathsf{o}}(u^2)$. The verifier checks $w = a + ub$ before sampling the folding challenge γ. The next claim is $g_\gamma(u^2) = a + \gamma b$. Thus, the same challenge folds the oracle and reduces the evaluation claim, with both claimed component values fixed before the challenge is sampled. Once the degree becomes constant, the verifier checks the remaining claim directly.

This follows BaseFold's principle of sharing randomness between evaluation and proximity reductions [ZCF24], applied here to univariate claims with-

out a multilinear sumcheck. Direct evaluation folding also appears in DeepFold [GLH+25]. The full protocol is given in Protocol 3.1.

The initial evaluations and combination require $\mathcal{O}(mk+mN)$ field operations. Subsequent coefficient vectors and oracle tables shrink geometrically, contributing only $\mathcal{O}(k+N)$ additional field operations. All later oracles are obtained by folding rather than new FFTs, so opening remains linear in d at constant rate.

DEEP variant. Each initial column opening contains m field elements, making query efficiency particularly important. We apply DEEP [BGKS20] to support proximity parameters beyond the unique-decoding radius, where several polynomial tuples may agree with the committed oracle on sufficiently many columns. After the interleaved oracle is fixed, the verifier samples an out-of-domain point ζ during commitment, and the prover sends the claimed component evaluations $f_i(\zeta)$. Except with small probability over ζ, these answers are consistent with at most one nearby tuple.

During evaluation, the initial folding challenge combines both the target evaluations and the out-of-domain evaluations into two claims about the virtual oracle. Both claims are then reduced through the same binary folds, without sampling fresh out-of-domain points. Over sufficiently large fields, the resulting protocol admits knowledge-soundness analysis at proximity parameters arbitrarily close to the Johnson radius, allowing fewer query repetitions. The variant retains the encoding, subgroup requirement, and query structure, while carrying only two evaluation claims per round. The construction and security analysis appear in Section 4.

1.3 Related Work

RS-based polynomial commitments. FRI [BBHR18] introduced recursive folding for testing RS proximity, and DEEP-FRI [BGKS20] strengthened its soundness using out-of-domain evaluations. STIR [ACFY24] decreases the code rate across rounds to reduce query complexity, while WHIR [ACFY25] extends this approach to constrained RS codes and polynomial evaluation claims. These protocols benefit from strong RS agreement guarantees, including results up to the Johnson radius [Hab24,Hab25,BSCH+25,BCGM26]. ReedWeave retains rate-preserving folding and obtains its smaller initial domain through interleaved RS encoding before commitment.

DeepFold [GLH+25] already combines direct evaluation folding with DEEP for multilinear polynomial commitments, without a separate sumcheck. Its protocol samples a fresh out-of-domain point in each round and propagates the accumulated evaluation claims. ReedWeave instead samples its DEEP point during commitment and, after the initial batching step, maintains only the target claim and that out-of-domain claim through the binary folds. The resulting protocol carries a constant number of evaluation claims per round.

Field-agnostic polynomial commitments. Several works avoid large multiplicative subgroups by changing the underlying codes. Brakedown [GLS+23] uses

linear-time encodable codes, while BaseFold [ZCF24] generalizes FRI-style folding to code families that can be instantiated over arbitrary sufficiently large fields. BrakingBase [NST25] combines linear-time encoding with BaseFold-based evaluation, and Lightning [QGZ26] develops systematic codes for the Brakedown framework with lower encoding cost. Other approaches change the evaluation domain while retaining RS structure. ECFFT [BCKL22] constructs suitable evaluation sets over arbitrary sufficiently large fields. Circle STARKs [HLP24] instead use the circle group. ReedWeave retains ordinary RS encoding over multiplicative subgroups but reduces the required subgroup size.

Interleaved polynomial commitments. Ligero and Shockwave [AHIV17,GLS+23] use interleaved RS encodings, with the square-root proof-size bounds shown in Table 1. Blaze [BCF+25] uses lifting through interleaving to apply a more expensive evaluation protocol to a smaller instance. ReedWeave follows this approach using ordinary RS codes. Its recursive evaluation protocol starts from a virtual RS oracle defined by a random linear combination of the committed rows. It retains the tradeoff between subgroup size and column width while obtaining proof size $\mathcal{O}(\lambda(m + \log^2 N))$ for $N = d/(m\rho)$.

Concurrent work. Superficially similar but fundamentally different from our work is LiftWHIR [ZZW+26]. It combines interleaved commitments with WHIR. Its RS lifting reduction maps the target multilinear evaluation and DEEP check to a smaller constrained-RS instance. Subsequent WHIR rounds re-encode folded polynomials and check consistency across domains. Writing $k = d/m$, its evaluation-time bound is $\mathcal{O}(d + k \log k)$, and its analysis takes $m = \Theta(\log d)$ to obtain linear time.

ReedWeave instead makes the initial row combination exactly an arity-m FRI fold, bypassing the full-size RS encoding, and reduces the target and DEEP claims through rate-preserving binary folds. This avoids domain shifting, re-encoding, and associated sumchecks, yielding $\mathcal{O}(d)$ opening time at constant rate for every admissible m, with no further FFTs. Under the conditions of Theorem 1, FRI-style recursion gives verification time and proof size $\mathcal{O}(\lambda(m + \log^2 N))$, where $N = d/(m\rho)$.

Organization Section 2 gives the main preliminaries. Sections 3 and 4 present the base and DEEP constructions, their security bounds, and the key DEEP arguments. Section 5 reports our implementation and 32-thread evaluation. Additional preliminaries and full security proofs appear in Appendices A-C, and Appendix D gives single-thread measurements.

2 Preliminaries

2.1 Notation

Let $\lambda \in \mathbb{N}$ be the security parameter, and let $\mathsf{negl}(\lambda)$ denote a negligible function. PPT abbreviates probabilistic polynomial time. Let $\mathbb{F}$ be a finite field of size q,

and write $\mathbb{F}^\star = \mathbb{F} \setminus \{0\}$. We denote by $\mathbb{F}^{<d}[X]$ the set of univariate polynomials over $\mathbb{F}$ of degree less than d. We use $x \xleftarrow{\$} S$ for uniform sampling from a finite set S and $[a] = \{1, \ldots, a\}$. Matrix rows and columns are indexed from zero. All logarithms are base two. For a relation $\mathcal{R}$, write $\mathcal{L}(\mathcal{R}) = \{x : \exists w,\ (x, w) \in \mathcal{R}\}$.

Let Σ be a finite alphabet and $\mathcal{L}$ a nonempty ordered set of size N. We identify a function $u : \mathcal{L} \to \Sigma$ with its value vector in Σ^N. For $S \subseteq \mathcal{L}$, write $u|_S$ for the restriction of u to S. For functions $u, v : \mathcal{L} \to \Sigma$, their relative Hamming distance is $\Delta_{\mathcal{L}}(u, v) = |\{x \in \mathcal{L} : u(x) \neq v(x)\}|/N$. A code of length N over Σ is a nonempty subset $\mathcal{C} \subseteq \Sigma^N$. We write $\Delta_{\mathcal{L}}(u, \mathcal{C}) = \min_{c \in \mathcal{C}} \Delta_{\mathcal{L}}(u, c)$. For a code with at least two codewords, its relative minimum distance is $\Delta_{\mathcal{C}} = \min_{c, c' \in \mathcal{C},\, c \neq c'} \Delta_{\mathcal{L}}(c, c')$.

2.2 Reed-Solomon Codes

Definition 1 (Reed-Solomon codes). *Let $\mathcal{L} \subseteq \mathbb{F}$ be an ordered evaluation domain of N distinct elements, and let $k \in [N]$. The Reed-Solomon code of degree bound k over $\mathcal{L}$ is* $\mathrm{RS}[\mathbb{F}, \mathcal{L}, k] = \{P|_{\mathcal{L}} : P \in \mathbb{F}^{<k}[X]\} \subseteq \mathbb{F}^N$.

Encoding. For $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, k]$ and a coefficient vector $u = (u_0, \ldots, u_{k-1}) \in \mathbb{F}^k$, let $P_u(X) = \sum_{a=0}^{k-1} u_a X^a$ and define $\mathsf{Enc} : \mathbb{F}^k \to \mathcal{C}$ by $\mathsf{Enc}(u) = P_u|_{\mathcal{L}}$. If $\mathcal{L}$ is a multiplicative subgroup of power-of-two order N, the encoding can be computed by padding the coefficient vector with zeros to length N and applying an FFT. This takes $\mathcal{O}(N \log N)$ field operations.

The code has rate $\rho = k/N$ and relative minimum distance $\Delta_{\mathcal{C}} = 1 - \rho + 1/N$. Standard RS properties and unique decoding are recalled in Section A.1.

Definition 2 (Interleaved codes). *For a code $\mathcal{C} \subseteq \Sigma^N$ and an integer $m \geq 1$, the m-interleaved code is $\mathcal{C}^m = \{\widehat{U} \in \Sigma^{m \times N} : \widehat{U}_i \in \mathcal{C}$ for $0 \leq i < m\}$. Each column is regarded as one symbol, so $\mathcal{C}^m$ is a code of length N over the alphabet Σ^m.*

For $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, k]$, we extend Enc row-wise to $U \in \mathbb{F}^{m \times k}$ by setting $\mathsf{Enc}(U)_i = \mathsf{Enc}(U_i)$. An oracle $\widehat{U} \in \mathbb{F}^{m \times N}$ is indexed by $x \in \mathcal{L}$, and a query returns the full column $\widehat{U}(x) \in \mathbb{F}^m$. Distances between such oracles are measured over the alphabet $\mathbb{F}^m$. A coordinate contributes one disagreement if any entry in that column differs.

Lemma 1 (Distance of interleaved codes). *Let $\mathcal{C} \subseteq \Sigma^N$ have relative minimum distance $\Delta_{\mathcal{C}}$. For every $m \geq 1$, the relative minimum distance of $\mathcal{C}^m$ over Σ^m is $\Delta_{\mathcal{C}^m} = \Delta_{\mathcal{C}}$.*

Proof (Proof of Lemma 1). Two distinct matrices differ in some row, giving distance at least $\Delta_{\mathcal{C}}$ over Σ^m. Equality is attained by matrices differing only in one row, where the codewords have distance $\Delta_{\mathcal{C}}$. □

2.3 Mutual Correlated Agreement

We use the proximity-generator formulation of WHIR. For a set S of coordinates, write $\mathcal{C}|_S = \{c|_S : c \in \mathcal{C}\}$. When applying the definition to a Reed-Solomon code, we identify $[N]$ with its ordered evaluation domain $\mathcal{L}$.

Definition 3 (Proximity Generator with Mutual Correlated Agreement [ACFY25]). *Let $\mathcal{C} \subseteq \mathbb{F}^n$ be a linear code. We say that* Gen *is a* proximity generator *for $\mathcal{C}$ with* mutual correlated agreement *with proximity bound* B *and error* err *if for every $f_1, \ldots, f_\ell : [n] \to \mathbb{F}$ and $\delta \in (0, 1 - \mathsf{B}(\mathcal{C}, \ell))$ the following holds:*

$$\Pr_{(r_1,\ldots,r_\ell)\leftarrow\mathsf{Gen}(\ell)} \left[\begin{array}{c} \exists S \subseteq [n] \text{ s.t. } \quad |S| \geq (1-\delta)\cdot n \\ \wedge\, \exists u \in \mathcal{C},\ u|_S = \sum_{j\in[\ell]} r_j \cdot f_j|_S \\ \wedge\, \exists j \in [\ell],\ \forall u' \in \mathcal{C},\ u'|_S \neq f_j|_S \end{array} \right] \leq \mathsf{err}(\mathcal{C}, \ell, \delta).$$

Haböck [Hab25] established mutual correlated agreement for Reed-Solomon codes up to the Johnson radius. Ben-Sasson et al. [BSCH+25] give an improved bound on the number of exceptional challenges. The following statement combines that bound with the unique decoding guarantee.

Theorem 2 ([ACFY25]). *Let $\mathcal{C} := \mathrm{RS}[\mathbb{F}, \mathcal{L}, k]$ have length $N := |\mathcal{L}|$, dimension $2 \leq k < N$, and rate $\rho := k/N$. For every integer $\ell \geq 1$, the generator $\mathsf{Gen}(\ell; \alpha) = (1, \alpha, \ldots, \alpha^{\ell-1})$, with $\alpha \xleftarrow{\$} \mathbb{F}$, has* mutual correlated agreement *for $\mathcal{C}$ with proximity bound $\mathsf{B}(\mathcal{C}, \ell) := \sqrt{\rho}$. Its error $\mathsf{err}(\mathcal{C}, \ell, \delta)$ can be taken as follows:*

- *If $0 < \delta \leq (1-\rho)/2$, take $\mathsf{err}(\mathcal{C}, \ell, \delta) := (\ell - 1)N/|\mathbb{F}|$.*
- *If $(1-\rho)/2 < \delta < 1 - \sqrt{\rho}$, set $\bar{\rho} := (k-1)/N$, $\eta := 1 - \sqrt{\bar{\rho}} - \delta$, and $s := \max\{\lceil \sqrt{\bar{\rho}}/\eta \rceil, 3\}$. Then*

$$\mathsf{err}(\mathcal{C}, \ell, \delta) := \frac{\ell - 1}{|\mathbb{F}|} \left(\frac{2(s+\frac{1}{2})^5 + 3(s+\frac{1}{2})\delta\bar{\rho}}{3\bar{\rho}^{3/2}} N + \frac{s+\frac{1}{2}}{\sqrt{\bar{\rho}}} \right).$$

Moreover, in the first case, if $\Delta_{\mathcal{C}} = 1 - \rho + 1/N \geq 3\sqrt{2/N}$ and $\Delta_{\mathcal{C}}/3 \leq \delta \leq \Delta_{\mathcal{C}}/2 - 3/(\Delta_{\mathcal{C}} N)$, the improved unique-decoding result of [BSCH+25] yields $\mathsf{err}(\mathcal{C}, \ell, \delta) := (\ell - 1)(\delta N + 1)/|\mathbb{F}|$.

2.4 Interactive Oracle Proofs and Polynomial Commitments

Interactive oracle proofs (IOPs). We briefly recall the definition of interactive oracle proofs (IOPs) [BCS16]. A k-round public-coin IOP, $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$, for a relation $\mathcal{R}$ proceeds as follows. Initially, $\mathcal{P}$ sends an oracle string π_0. In each round $i \in [1, k]$, the verifier samples and sends a random challenge α_i, and the prover replies with an oracle string π_i. After k rounds of interaction, the verifier $\mathcal{V}$ queries some entries of the oracle strings $\pi_0, \pi_1, \ldots, \pi_k$ and outputs a bit b.

Definition 4 (IOPs). *Let* $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$ *be a* k*-round public-coin IOP for a relation* $\mathcal{R}$*. We say that* IOP *is perfectly complete if for every* $(x, w) \in \mathcal{R}$*,*

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

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

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

Round-by-round knowledge soundness. We use the state-function framework of STIR [ACFY24] and WHIR [ACFY25] to analyze each verifier challenge. The formal definitions, including the per-round extraction condition, appear in Section A.4.

Polynomial commitment schemes.

Definition 5 (Polynomial commitment scheme). *We adapt the definition in [GLS+23] to univariate polynomials. A univariate polynomial commitment scheme over* $\mathbb{F}$ *consists of* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$*.*

- $\mathsf{Setup}(1^\lambda, d) \to \mathsf{pp}$ *takes the security parameter and a degree bound* d*, and outputs public parameters.*
- $\mathsf{Commit}(\mathsf{pp}, f) \to C$ *takes* $f \in \mathbb{F}^{<d}[X]$ *and outputs a commitment* C*.*
- $\mathsf{Open}(\mathsf{pp}, C, f) \to b$ *checks whether* f *opens* C *and outputs* $b \in \{0, 1\}$*.*
- $\mathsf{Eval}(\mathsf{pp}, C, z, y; f)$ *is an interactive protocol between* $\mathcal{P}$ *and* $\mathcal{V}$ *with public input* C*,* $z \in \mathbb{F}$*, and* $y \in \mathbb{F}$*. The prover additionally knows* f *and convinces the verifier that* $\mathsf{Open}(\mathsf{pp}, C, f) = 1$ *and* $f(z) = y$*.*

The scheme is perfectly complete if, for every $d \in \mathbb{N}$*,* $f \in \mathbb{F}^{<d}[X]$*, and* $z \in \mathbb{F}$*, honestly generated parameters and* $C \leftarrow \mathsf{Commit}(\mathsf{pp}, f)$ *satisfy* $\mathsf{Open}(\mathsf{pp}, C, f) = 1$ *and the honest execution of* $\mathsf{Eval}(\mathsf{pp}, C, z, f(z); f)$ *accepts with probability one.*

It is binding if for every $d \in \mathbb{N}$ *and PPT adversary* $\mathcal{A}$*,*

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

The scheme is knowledge sound if Eval *is an argument of knowledge for the relation whose instances are* (C, z, y) *and whose witnesses are polynomials* $f \in \mathbb{F}^{<d}[X]$ *satisfying* $f(z) = y$ *and* $\mathsf{Open}(\mathsf{pp}, C, f) = 1$*. Denote this relation by*

$\mathcal{R}_{\mathsf{Eval},\mathsf{pp}}$. *Concretely, for every* $d \in \mathbb{N}$ *and PPT algorithms* $\mathcal{A}$ *and* $\mathcal{P}^*$*, there is an expected polynomial-time extractor* Ext *such that, for every randomness* r*,*

$$\Pr\begin{bmatrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d), \\ (C, z, y) \leftarrow \mathcal{A}(\mathsf{pp}, r), \\ \langle \mathcal{P}^*(r), \mathcal{V} \rangle_{\mathsf{Eval}(\mathsf{pp}, C, z, y)} = 1 \end{bmatrix} \approx \Pr\begin{bmatrix} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, d), \\ (C, z, y) \leftarrow \mathcal{A}(\mathsf{pp}, r), \\ f \leftarrow \mathsf{Ext}(\mathsf{pp}, r), \\ ((C, z, y); f) \in \mathcal{R}_{\mathsf{Eval},\mathsf{pp}} \end{bmatrix}.$$

The symbol $\approx$ means that the two probabilities differ by a negligible function of λ.

Remark 1. Our DEEP construction replaces Commit with a public-coin interaction whose transcript is C, with oracle access in the IOP setting. In the binding experiment, after Setup, the adversary interacts with the honest commitment verifier and outputs two openings. Its probability of an accepted commitment and valid openings to distinct polynomials, over all parties' randomness, is bounded by the stated binding error for the specified adversary class.

3 A Base Construction in the Unique-Decoding Regime

We present the base construction of ReedWeave in the IOP model. It reduces the required subgroup size and relies on unique decoding. We denote its opening predicate by $\mathsf{Open}_{\mathsf{base}}$ and its evaluation protocol by $\mathsf{Eval}_{\mathsf{base}}$, reserving Eval for the DEEP variant in Section 4. Both variants share the polynomial representation and folding operations but have separate opening relations and security guarantees. Following the BCS transformation [BCS16], we replace each oracle message with a Merkle commitment and answer each verifier query with the requested value and a Merkle authentication path. Fiat-Shamir then derives verifier challenges by hashing the public input and preceding transcript, yielding a non-interactive argument in the random oracle model.

3.1 Construction

Parameters. Let $\mathbb{F}$ be a finite field of size q and odd characteristic. Let $d = mk$ be the polynomial degree bound, where $m \geq 1$ and $k = 2^\ell$ for $\ell \geq 2$. Fix $N = 2^n > k$ and an element $\omega \in \mathbb{F}^\star$ of order N, and set $\rho = k/N$. Choose $1 \leq t < \ell$ so that $k_t = k/2^t > 1$ is a fixed constant, and let $Q \geq 1$ be the number of repetitions in the query phase. For $0 \leq j \leq t$, set $\mathcal{L}_j = \langle \omega^{2^j} \rangle$, $N_j = N/2^j$, $k_j = k/2^j$, and $\mathcal{C}_j = \mathrm{RS}[\mathbb{F}, \mathcal{L}_j, k_j]$. Every code $\mathcal{C}_j$ has rate ρ. The public parameters pp specify $\mathbb{F}, m, k, N, \omega, t$, and Q.

Notation. For $f(X) = \sum_{u=0}^{d-1} c_u X^u \in \mathbb{F}^{<d}[X]$, let $U_f \in \mathbb{F}^{m \times k}$ be the matrix with $(U_f)_{i,a} = c_{ma+i}$ for $0 \leq i < m$ and $0 \leq a < k$. Its i-th row is the coefficient vector of f_i in the decomposition $f(X) = \sum_{i=0}^{m-1} X^i f_i(X^m)$ of Definition 6 in Appendix A.3, where $\deg f_i < k$. We instantiate Enc with the code $\mathcal{C}_0$ and apply it to each row of U_f. Thus $\mathsf{Enc}(U_f)(x) = (f_0(x), \ldots, f_{m-1}(x))^\mathsf{T}$ for every $x \in \mathcal{L}_0$.

Commitment Phase Given a univariate polynomial $f \in \mathbb{F}^{<d}[X]$ with coefficient matrix U_f, let $\widehat{U} := \mathsf{Enc}(U_f) \in \mathcal{C}_0^m$ be its encoding. In the IOP setting, the commitment to f is the oracle $\widehat{U}$, where a query at $x \in \mathcal{L}_0$ returns the column $\widehat{U}(x) \in \mathbb{F}^m$. The corresponding Merkle commitment C is the root of a Merkle tree whose leaves are the columns of $\widehat{U}$. The prover sends C to the verifier and retains $\widehat{U}$ for subsequent openings.

Opening Phase The prover opens a commitment C by sending a univariate polynomial $f \in \mathbb{F}^{<d}[X]$ and a word $\widehat{U} \in \mathbb{F}^{m \times N}$ to the verifier. The verifier checks that (i) the Merkle commitment of $\widehat{U}$ equals C, and (ii) $\Delta_{\mathcal{L}_0}(\widehat{U}, \mathsf{Enc}(U_f)) < \Delta_{\mathcal{C}_0}/2$, where $\Delta_{\mathcal{C}_0}$ is the relative minimum distance of $\mathcal{C}_0$. The distance in (ii) is measured over the alphabet $\mathbb{F}^m$. The predicate $\mathsf{Open}_{\mathsf{base}}(\mathsf{pp}, C, f, \widehat{U})$ returns one if and only if both checks pass.

Evaluation Phase Given a commitment C, a point $z \in \mathbb{F}$, and a claimed evaluation $y \in \mathbb{F}$, the prover wishes to convince the verifier that it knows an opening $(f, \widehat{U})$ with $f \in \mathbb{F}^{<d}[X]$ such that $\mathsf{Open}_{\mathsf{base}}(\mathsf{pp}, C, f, \widehat{U}) = 1$ and $f(z) = y$. The protocol $\mathsf{Eval}_{\mathsf{base}}$ interleaves the reduction of this evaluation claim with the FRI IOPP.

The decomposition of f gives $f(z) = \sum_{i=0}^{m-1} z^i f_i(z^m)$. For the initial reduction, the prover sends $v = (v_0, \ldots, v_{m-1})$, where v_i is claimed to equal $f_i(z^m)$. The verifier checks $y = \sum_{i=0}^{m-1} z^i v_i$ and then samples $\alpha \xleftarrow{\$} \mathbb{F}$. This challenge reduces these evaluation claims to the single claim $g^{(0)}(z^m) = \sum_{i=0}^{m-1} \alpha^i v_i$, where $g^{(0)} = \mathsf{Fold}_m(f, \alpha) = \sum_i \alpha^i f_i \in \mathbb{F}^{<k}[X]$. The same linear combination of the rows of $\widehat{U}$ defines the virtual oracle π_0. For $\widehat{U} = \mathsf{Enc}(U_f)$, we have $\pi_0 = g^{(0)}|_{\mathcal{L}_0}$. Each entry of π_0 is computed from the corresponding column of $\widehat{U}$, so the prover does not send a separate oracle for π_0.

The protocol then proceeds in t binary folding rounds using Definition 6. In each round, the prover sends the claimed evaluations of the even and odd parts of the current polynomial. The verifier checks their consistency with the current evaluation claim and sends a random challenge, which is used both to fold the polynomial and to reduce the claim. Since k_t is a fixed constant, the prover sends the final polynomial directly. The verifier checks its degree and the final evaluation claim, and performs the FRI consistency checks. The complete base evaluation protocol is given in Protocol 3.1, with oracle access to $\widehat{U}$ from the commitment phase.

Protocol 3.1: Base evaluation protocol $\mathsf{Eval}_{\mathsf{base}}$

Public input: parameters pp, oracle $\widehat{U} \in \mathbb{F}^{m \times N}$, point $z \in \mathbb{F}$, and claimed evaluation $y \in \mathbb{F}$.
Prover witness: the polynomial $f \in \mathbb{F}^{<d}[X]$.
The verifier rejects if any check below fails.

1. **Initial folding.**

(a) $\mathcal{P}$ sends $v = (v_0, \ldots, v_{m-1})$, where $v_i = f_i(z^m)$ for $0 \le i < m$.
(b) $\mathcal{V}$ checks $y = \sum_{i=0}^{m-1} z^i v_i$, then samples and sends $\alpha \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
(c) $\mathcal{P}$ computes $g^{(0)} = \mathsf{Fold}_m(f, \alpha)$. Both parties set $z_0 = z^m$ and $y_0 = \sum_{i=0}^{m-1} \alpha^i v_i$. Define the oracle by $\pi_0(x) := \sum_{i=0}^{m-1} \alpha^i \widehat{U}_i(x)$ for $x \in \mathcal{L}_0$. $\mathcal{V}$ computes each queried entry of π_0 from a column of $\widehat{U}$.

2. **Binary folding.** For $j = 0, \ldots, t-1$:
(a) $\mathcal{P}$ writes $g^{(j)}(X) = g_{\mathsf{e}}^{(j)}(X^2) + X g_{\mathsf{o}}^{(j)}(X^2)$ and sends $a_j = g_{\mathsf{e}}^{(j)}(z_j^2)$ and $b_j = g_{\mathsf{o}}^{(j)}(z_j^2)$.
(b) $\mathcal{V}$ checks $y_j = a_j + z_j b_j$, then samples and sends a fresh challenge $\gamma_j \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
(c) $\mathcal{P}$ computes $g^{(j+1)} = \mathsf{Fold}_2(g^{(j)}, \gamma_j)$. If $j < t-1$, $\mathcal{P}$ outputs the oracle $\pi_{j+1} := g^{(j+1)}|_{\mathcal{L}_{j+1}}$.
(d) Both parties set $z_{j+1} = z_j^2$ and $y_{j+1} = a_j + \gamma_j b_j$.

3. **Final polynomial.** $\mathcal{P}$ sends $g^{(t)}$ directly. $\mathcal{V}$ checks that $\deg g^{(t)} < k_t$ and $g^{(t)}(z_t) = y_t$. For the query phase, define $\pi_t := g^{(t)}|_{\mathcal{L}_t}$.

4. **Query phase.** After all oracle messages and the final polynomial have been fixed, $\mathcal{V}$ performs the following test Q times independently:
(a) Sample $x_0 \xleftarrow{\$} \mathcal{L}_0$ and set $x_j = x_0^{2^j} \in \mathcal{L}_j$ for $0 \le j \le t$.
(b) Query the columns $\widehat{U}(x_0)$ and $\widehat{U}(-x_0)$ to compute $\pi_0(x_0)$ and $\pi_0(-x_0)$.
(c) For each $1 \le j < t$, query $\pi_j(x_j)$ and $\pi_j(-x_j)$. Compute $\pi_t(x_t) = g^{(t)}(x_t)$ directly.
(d) For each $0 \le j < t$, check

$$\pi_{j+1}(x_{j+1}) = \frac{\pi_j(x_j) + \pi_j(-x_j)}{2} + \gamma_j \frac{\pi_j(x_j) - \pi_j(-x_j)}{2x_j}.$$

The verifier accepts if all checks pass.

We analyze the complexity of the commitment and evaluation phases separately. For the simplified bounds, fix m, ρ, k_t, and a proximity parameter $0 < \delta \le (1-\rho)/2$. Choose the constant in $Q = \Theta(\lambda)$ and the field size q so that the soundness error in Corollary 1 is at most $2^{-\lambda}$. Then $k = d/m = \Theta(d)$, $N = d/(m\rho) = \Theta(d)$, and $t = \log_2(k/k_t) = \mathcal{O}(\log d)$. We also give the general bounds to retain the dependence on m and Q.

We count field operations and hash operations separately, and measure proof size in field elements and hash values. Note that hash operations take a constant number of words as input.

Commitment time. Encoding the m rows requires $\mathcal{O}(mN \log N) = \mathcal{O}(d \log d)$ field operations. Hashing the N columns requires $\mathcal{O}(mN)$ hash operations, and computing the internal nodes requires $\mathcal{O}(N)$. Thus commitment uses $\mathcal{O}(d \log d)$ field operations and $\mathcal{O}(mN) = \mathcal{O}(d)$ hash operations. The public commitment consists of one hash value.

Prover time. Evaluating all m polynomials f_i at z^m and the coefficients of $g^{(0)}$ requires $\mathcal{O}(mk)$ field operations. Forming the virtual oracle π_0 from the encoded rows requires $\mathcal{O}(mN)$. The coefficient vectors and oracle tables in the remaining folding rounds have total lengths $\mathcal{O}(k)$ and $\mathcal{O}(N)$, respectively. Computing the evaluation messages and folds requires $\mathcal{O}(k + N)$ field operations. Constructing the additional Merkle trees requires $\mathcal{O}(N)$ hash operations. In total, the prover performs $\mathcal{O}(mk + mN) = \mathcal{O}(d)$ field operations and $\mathcal{O}(N) = \mathcal{O}(d)$ hash operations after commitment.

Verifier time. The evaluation and folding checks, including computing the query points and evaluating the final polynomial, require $\mathcal{O}(Q(m + t + k_t + \log N)) = \mathcal{O}(\lambda \log d)$ field operations. Verifying the queried columns and oracle entries using individual Merkle authentication paths requires $\mathcal{O}(Q(m+t \log N)) = \mathcal{O}(\lambda \log^2 d)$ hash operations. The term Qm accounts for hashing the initial column openings.

Proof size. The evaluation messages, final polynomial, and query responses contain $\mathcal{O}(m + t + k_t + Q(m + t)) = \mathcal{O}(\lambda \log d)$ field elements. The intermediate oracle commitments and their Merkle authentication paths contribute $\mathcal{O}(t + Qt \log N) = \mathcal{O}(\lambda \log^2 d)$ hash values. The initial query responses account for $\mathcal{O}(Qm)$ field elements in the general bound.

3.2 Security Analysis

Completeness and binding For an honestly generated commitment, the opening and evaluation checks accept for every choice of verifier randomness, establishing perfect completeness. For binding, consider valid openings $(f_0, \widehat{U}_0)$ and $(f_1, \widehat{U}_1)$ of the same commitment, with $f_0 \neq f_1$. If the matrices differ, their Merkle trees yield a hash collision. Otherwise, the triangle inequality gives $\Delta_{\mathcal{L}_0}(\mathsf{Enc}(U_{f_0}), \mathsf{Enc}(U_{f_1})) < \Delta_{\mathcal{C}_0}$, contradicting the minimum distance $\Delta_{\mathcal{C}_0^m} = \Delta_{\mathcal{C}_0}$ from Lemma 1. Thus any PPT adversary breaks binding only with negligible probability, assuming collision resistance.

Round-by-round knowledge soundness We analyze $\mathsf{Eval}_{\mathsf{base}}$ in the language of IOPs. For an instance $(\widehat{U}, z, y)$ with input oracle $\widehat{U}$, the extraction relation requires $f \in \mathbb{F}^{<d}[X]$ with $\Delta_{\mathcal{L}_0}(\widehat{U}, \mathsf{Enc}(U_f)) < \Delta_{\mathcal{C}_0}/2$ and $f(z) = y$. The verifier makes column queries, whereas the extractor may read the entire oracle.

We first combine mutual correlated agreement with the evaluation checks. The following lemma applies both to the rows of $\widehat{U}$ and to the words $(\pi_j)_{\mathsf{e}}, (\pi_j)_{\mathsf{o}}$.

Lemma 2 (Folding and evaluation in the unique-decoding regime). *Let $\mathcal{D} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, K]$ have length $M = |\mathcal{L}|$ and dimension $2 \leq K < M$. Fix $0 < \delta \leq (1 - K/M)/2$, an integer $r \geq 1$, words $w_0, \ldots, w_{r-1} : \mathcal{L} \to \mathbb{F}$, a point $\zeta \in \mathbb{F}$, and values $v_0, \ldots, v_{r-1} \in \mathbb{F}$.*

Except with probability at most $(r-1)(M+1)/q$ *over* $\alpha \xleftarrow{\$} \mathbb{F}$*, the following implication holds simultaneously for every* $h \in \mathbb{F}^{<K}[X]$ *and* $S \subseteq \mathcal{L}$*. If*

$$|S| > (1-\delta)M, \qquad h|_S = \sum_{i=0}^{r-1} \alpha^i w_i|_S, \qquad h(\zeta) = \sum_{i=0}^{r-1} \alpha^i v_i,$$

then there exist $h_0, \ldots, h_{r-1} \in \mathbb{F}^{<K}[X]$ *such that*

$$h = \sum_{i=0}^{r-1} \alpha^i h_i, \qquad h_i|_S = w_i|_S, \quad h_i(\zeta) = v_i \quad (0 \le i < r).$$

The proof appears in Section B.1.

Theorem 3 (Round-by-round knowledge soundness of $\mathsf{Eval}_{\mathsf{base}}$). *Fix the parameters of Protocol 3.1 and* $0 < \delta \le (1-\rho)/2$*. For the extraction relation specified in Section 3.2,* $\mathsf{Eval}_{\mathsf{base}}$ *has round-by-round knowledge soundness with the following errors:*

$$\varepsilon^{\mathsf{init}} = \frac{(m-1)(N+1)}{q}, \quad \varepsilon_j^{\mathsf{fold}} = \frac{N_{j+1}+1}{q} (0 \le j < t), \quad \varepsilon^{\mathsf{qry}} = (1-\delta)^Q.$$

The Q *points sampled from* $\mathcal{L}_0$ *in the query phase are treated as a single final verifier challenge.*

The deterministic extractor E *queries all* N *columns of* $\widehat{U}$ *and uses* $\mathcal{O}(mN^3)$ *field operations. Its running time is polynomial in* $m, N,$ *and* $\log q$*.*

The extractor uniquely decodes each row, reconstructs f, and checks the column distance and $f(z) = y$, returning $\perp$ if decoding or a check fails. Appendix B.2 gives the algorithm and proof.

Corollary 1 (Knowledge soundness of $\mathsf{Eval}_{\mathsf{base}}$). *Under the conditions of Theorem 3, let* $\kappa = \varepsilon^{\mathsf{init}} + \sum_{j=0}^{t-1} \varepsilon_j^{\mathsf{fold}} + \varepsilon^{\mathsf{qry}}$*. For every evaluation instance and every malicious prover* $\mathcal{P}^*$*, the probability that* $\mathsf{Eval}_{\mathsf{base}}$ *accepts but* E *returns* $\perp$ *is at most* κ*, over the randomness of the verifier and* $\mathcal{P}^*$*. Equivalently,* $\kappa = A/q + (1-\delta)^Q$*, where* $A = (m-1)(N+1) + N - N_t + t$*.*

For every PPT prover $\mathcal{P}^*$ *whose acceptance probability is* p*, there exists a polynomial-time extractor* $\mathsf{Ext}^{\mathcal{P}^*}$ *that outputs a valid witness with probability at least* $p - \kappa$*.*

The proof appears in Section B.3.

From Unique Decoding to List Decoding. The base construction needs only a subgroup of order $N = d/(m\rho)$, but its binding and evaluation analyses rely on unique decoding. Beyond that radius, several nearby polynomials may correspond to the same oracle, and the folding challenge may select among them. The next section adds out-of-domain evaluations to distinguish these candidates and proves knowledge soundness by a union bound over the joint decoding list, without enlarging the evaluation domain.

4 ReedWeave up to the Johnson Bound

We extend the base construction to proximity parameters $\delta = 1-\sqrt{\rho}-\eta$ by using the DEEP technique [BGKS20]. The commitment phase adds an out-of-domain evaluation of each component polynomial to distinguish nearby candidates. The evaluation protocol, denoted by Eval, reduces these evaluations together with the target claim through the same binary folding rounds. The encoding and subgroup requirement are unchanged.

4.1 List Decoding of Interleaved Codes

For $\mathcal{D} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, K]$ and words $\mathbf{w} = (w_0, \ldots, w_{r-1})$ on $\mathcal{L}$, define the joint decoding list

$$\mathcal{J}_{\mathcal{D}}(\mathbf{w}, \delta) = \left\{ \mathbf{h} \in (\mathbb{F}^{<K}[X])^r : \Delta_{\mathcal{L}}\big(\mathbf{w}, (h_i|_{\mathcal{L}})_{i=0}^{r-1}\big) < \delta \right\}.$$

The distance is over $\mathbb{F}^r$, so all rows must agree on a common set of more than $(1-\delta)|\mathcal{L}|$ coordinates.

Lemma 3 (Johnson bound for interleaved codes). *Let $\mathcal{D} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, K]$ have length M and rate $\rho = K/M$. For every $r \geq 1$, every $\mathbf{w} : \mathcal{L} \to \mathbb{F}^r$, and every $0 < \delta < 1-\sqrt{\rho}$, the joint decoding list $\mathcal{J}_{\mathcal{D}}(\mathbf{w}, \delta)$ has size at most $\lfloor 1/((1-\delta)^2-\rho) \rfloor$.*

Proof. Two distinct polynomial tuples differ in at least one component and hence agree as column symbols at at most $K-1$ coordinates. The counting argument in the proof of Lemma 7 (Appendix A.2) therefore applies over the alphabet $\mathbb{F}^r$ and gives the stated bound, independently of r. □

We use the deterministic RS list decoder of Chatterjee, Harsha, and Kumar [CHK25], which attains the Guruswami-Sudan radius [GS99]. It recovers all polynomials of degree less than K agreeing with a received word at more than $\sqrt{(K-1)M}$ coordinates, in time polynomial in M and $\log q$. Retaining only candidates whose evaluations are at relative distance less than δ from the received word recovers every component needed here, since $(1-\delta)M > \sqrt{KM}$. List decoding is used only for extraction.

Lemma 4 (Enumerating the joint decoding list). *Under the conditions of Lemma 3, write $B = \lfloor 1/((1-\delta)^2 - K/M) \rfloor$. There is a deterministic algorithm that enumerates $\mathcal{J}_{\mathcal{D}}(\mathbf{w}, \delta)$ using r invocations of the RS list decoder and $\mathcal{O}(rBMK + rB^2M + rBK)$ additional field operations and comparisons. Its running time is polynomial in $r, M, B,$ and $\log q$.*

Proof. Row decoding. List decode each row, retaining only candidates at relative distance less than δ from that row. For every retained candidate, compute its agreement set by evaluation.

Incremental enumeration. Process the rows one at a time. Extend each retained tuple by every candidate for the next row, retaining an extension exactly

when the intersection of its agreement sets has size greater than $(1-\delta)M$. Store intermediate tuples by references to row polynomials.

Correctness. Every tuple in the full joint decoding list satisfies the agreement condition on each prefix of rows, so it is never discarded. Conversely, every final retained tuple satisfies the defining joint agreement condition. The algorithm therefore enumerates exactly $\mathcal{J}_{\mathcal{D}}(\mathbf{w}, \delta)$.

Running time. By Lemma 3, each row has at most B retained candidates, and at most B tuples survive at every intermediate stage. Computing the row agreement sets costs $\mathcal{O}(rBMK)$ field operations. Each extension stage uses at most B^2 set intersections, each costing $\mathcal{O}(M)$ comparisons, for a total of $\mathcal{O}(rB^2M)$ comparisons. Writing the final coefficients costs $\mathcal{O}(rBK)$ operations. Together with the r decoder calls, these costs give the stated bound. □

4.2 Construction

Parameters. Use the parameters, codes $\mathcal{C}_j$, coefficient matrix U_f, and component polynomials f_i from Section 3.1. The public parameters additionally specify $\delta = 1 - \sqrt{\rho} - \eta$ for $0 < \eta < 1 - \sqrt{\rho}$. Set $B := \lfloor 1/(2\eta\sqrt{\rho} + \eta^2) \rfloor$. Since every $\mathcal{C}_j$ has rate ρ, Lemma 3 gives the same list-size bound at every layer.

We describe the protocols in the language of IOPs. Unlike the non-interactive Commit algorithm in Definition 5, the commitment phase below is interactive, with ζ sampled by an honest verifier.

Commitment Phase The prover augments the interleaved oracle with component evaluations at a random out-of-domain point, sampled after the oracle is fixed. These evaluations are retained for subsequent evaluation proofs.

Protocol 4.1: DEEP commitment phase

Public input: parameters pp.
Prover input: $f \in \mathbb{F}^{<d}[X]$.

1. $\mathcal{P}$ computes $\widehat{U} := \mathsf{Enc}(U_f)$ and sends it as an oracle indexed by $\mathcal{L}_0$. A query at $x \in \mathcal{L}_0$ returns the column $\widehat{U}(x) \in \mathbb{F}^m$.
2. $\mathcal{V}$ samples and sends $\zeta \xleftarrow{\$} \mathbb{F} \setminus \mathcal{L}_0$.
3. $\mathcal{P}$ sends $c = (c_0, \ldots, c_{m-1}) \in \mathbb{F}^m$, where $c_i = f_i(\zeta)$. The verifier rejects a malformed response.

The commitment is $C_{\mathsf{IOP}} = (\widehat{U}, \zeta, c)$, with oracle access to $\widehat{U}$. The prover retains $\widehat{U}$ together with f.

Opening Phase To open $C_{\mathsf{IOP}} = (\widehat{U}, \zeta, c)$, the prover sends $f \in \mathbb{F}^{<d}[X]$. The predicate $\mathsf{Open}^{\mathsf{IOP}}(\mathsf{pp}, C_{\mathsf{IOP}}, f)$ accepts if and only if $\Delta_{\mathcal{L}_0}(\widehat{U}, \mathsf{Enc}(U_f)) < \delta$ and $f_i(\zeta) = c_i$ for every $0 \le i < m$. The distance is measured over $\mathbb{F}^m$ against the oracle fixed during commitment.

Evaluation Phase In $\mathcal{R}^{\mathsf{IOP}}_{\mathsf{Eval},\mathsf{pp}}$, the instance is (C_{IOP}, z, y) and the witness is $f \in \mathbb{F}^{<d}[X]$ satisfying $\mathsf{Open}^{\mathsf{IOP}}(\mathsf{pp}, C_{\mathsf{IOP}}, f) = 1$ and $f(z) = y$.

As in the base protocol, the prover sends the component claims $v_i = f_i(z^m)$ before the verifier samples α. The same challenge now combines both v and c, giving two claims about $g^{(0)} = \mathsf{Fold}_m(f, \alpha)$ at both z^m and ζ. Each binary round reduces both claims using the same folding challenge. The oracle messages and query phase are unchanged.

Protocol 4.2: DEEP evaluation protocol Eval

Public input: pp, commitment $C_{\mathsf{IOP}} = (\widehat{U}, \zeta, c)$, point $z \in \mathbb{F}$, and claimed evaluation $y \in \mathbb{F}$.
Prover witness: $f \in \mathbb{F}^{<d}[X]$.
The verifier rejects if any message is malformed or any check fails.

1. **Initial folding.**
   (a) $\mathcal{P}$ sends $v = (v_0, \ldots, v_{m-1})$, where $v_i = f_i(z^m)$.
   (b) $\mathcal{V}$ checks $y = \sum_{i=0}^{m-1} z^i v_i$, then samples and sends $\alpha \xleftarrow{\$} \mathbb{F}$.
   (c) $\mathcal{P}$ computes $g^{(0)} = \mathsf{Fold}_m(f, \alpha)$. Define the virtual oracle by $\pi_0(x) := \sum_{i=0}^{m-1} \alpha^i \widehat{U}_i(x)$ for $x \in \mathcal{L}_0$. Both parties set $z_0 = z^m$ and $y_0 = \sum_i \alpha^i v_i$ for the target evaluation, and $\zeta_0 = \zeta$ and $u_0 = \sum_i \alpha^i c_i$ for the out-of-domain evaluation. Each queried entry of π_0 is computed from a column of $\widehat{U}$.
2. **Binary folding.** For $j = 0, \ldots, t-1$:
   (a) $\mathcal{P}$ writes $g^{(j)}(X) = g_{\mathsf{e}}^{(j)}(X^2) + X g_{\mathsf{o}}^{(j)}(X^2)$. It sends $a_j = g_{\mathsf{e}}^{(j)}(z_j^2)$ and $b_j = g_{\mathsf{o}}^{(j)}(z_j^2)$ for the target evaluation, and $\widetilde{a}_j = g_{\mathsf{e}}^{(j)}(\zeta_j^2)$ and $\widetilde{b}_j = g_{\mathsf{o}}^{(j)}(\zeta_j^2)$ for the out-of-domain evaluation.
   (b) $\mathcal{V}$ checks $y_j = a_j + z_j b_j$ and $u_j = \widetilde{a}_j + \zeta_j \widetilde{b}_j$, then samples and sends a fresh challenge $\gamma_j \xleftarrow{\$} \mathbb{F}$.
   (c) $\mathcal{P}$ computes $g^{(j+1)} = \mathsf{Fold}_2(g^{(j)}, \gamma_j)$. If $j < t-1$, it sends the oracle $\pi_{j+1} = g^{(j+1)}|_{\mathcal{L}_{j+1}}$.
   (d) Both parties set $z_{j+1} = z_j^2$ and $y_{j+1} = a_j + \gamma_j b_j$, and update the out-of-domain claim by setting $\zeta_{j+1} = \zeta_j^2$ and $u_{j+1} = \widetilde{a}_j + \gamma_j \widetilde{b}_j$.
3. **Final polynomial.** $\mathcal{P}$ sends $g^{(t)}$ directly. $\mathcal{V}$ checks $\deg g^{(t)} < k_t$, $g^{(t)}(z_t) = y_t$, and $g^{(t)}(\zeta_t) = u_t$. Define $\pi_t = g^{(t)}|_{\mathcal{L}_t}$ for the query phase.
4. **Query phase.** After all oracle messages and the final polynomial have been sent, $\mathcal{V}$ repeats the following test Q times independently:
   (a) Sample $x_0 \xleftarrow{\$} \mathcal{L}_0$ and set $x_j = x_0^{2^j}$ for $0 \le j \le t$.
   (b) Query $\widehat{U}(x_0)$ and $\widehat{U}(-x_0)$ to compute $\pi_0(x_0)$ and $\pi_0(-x_0)$. For $1 \le j < t$, query $\pi_j(x_j)$ and $\pi_j(-x_j)$. Compute $\pi_t(x_t)$ from $g^{(t)}$.
   (c) For every $0 \le j < t$, check

$$\pi_{j+1}(x_{j+1}) = \frac{\pi_j(x_j) + \pi_j(-x_j)}{2} + \gamma_j \frac{\pi_j(x_j) - \pi_j(-x_j)}{2x_j}.$$

The verifier accepts if all checks pass.

Remark 2 (Extension-field challenges). For both protocols, if the base field $\mathbb{F}$ is too small for the target soundness error, the verifier random challenges may be sampled over a sufficiently large finite extension $\mathbb{E}/\mathbb{F}$. The initial RS encoding and its Merkle commitment remain over $\mathbb{F}$, with unchanged evaluation domains and subgroup requirements. Subsequent algebraic computations are performed over $\mathbb{E}$.

For the simplified bounds, fix m, ρ, η, k_t, take $Q = \Theta(\lambda)$, and choose a field satisfying the subgroup requirement so that the sum of the binding and knowledge errors in Theorem 4 and Corollary 2 is at most $2^{-\lambda}$. Then $N = \Theta(d)$ and $t = \mathcal{O}(\log d)$.

Commitment time. Encoding the rows and computing c require $\mathcal{O}(mN \log N + mk) = \mathcal{O}(d \log d)$ field operations. The Merkle tree costs $\mathcal{O}(mN) = \mathcal{O}(d)$ hash operations. The commitment transcript (R, ζ, c) contains one Merkle root and $m + 1$ field elements.

Prover time. Computing $v, g^{(0)}$, and π_0 costs $\mathcal{O}(mk + mN)$ field operations. The remaining coefficient vectors and oracle tables have total lengths $\mathcal{O}(k)$ and $\mathcal{O}(N)$, so folding and computing both sets of evaluation messages cost $\mathcal{O}(k+N)$, without further FFTs. The total evaluation cost is $\mathcal{O}(mk + mN) = \mathcal{O}(d)$ field operations and $\mathcal{O}(N) = \mathcal{O}(d)$ hash operations. Relative to the base protocol, the additional evaluations cost $\mathcal{O}(k)$ field operations.

Verifier time. The evaluation and folding checks require $\mathcal{O}(m + t + k_t + Q(m + t + k_t + \log N)) = \mathcal{O}(\lambda \log d)$ field operations. Verifying the column openings and Merkle authentication paths requires $\mathcal{O}(Q(m + t \log N)) = \mathcal{O}(\lambda \log^2 d)$ hash operations. The out-of-domain evaluations introduce no additional oracle queries or authentication paths.

Proof size. The evaluation messages, final polynomial, and query responses contain $\mathcal{O}(m + t + k_t + Q(m + t)) = \mathcal{O}(\lambda \log d)$ field elements. The intermediate oracle commitments and their Merkle authentication paths contribute $\mathcal{O}(t + Qt \log N) = \mathcal{O}(\lambda \log^2 d)$ hash values. Relative to the base protocol at the same Q, the evaluation proof contains exactly $2t$ additional field elements. The m out-of-domain answers are counted in the commitment phase.

Remark 3 (Shifting to improve rate). The virtual RS oracle obtained after the initial interleaving reduction is a natural starting point for a rate-improving recursion as in STIR [ACFY24] and WHIR [ACFY25]. Rather than shrinking the degree bound and the evaluation domain by the same factor, these protocols reencode the folded polynomial on a larger domain than its natural folding domain, thereby decreasing the code rate and the number of queries required in subsequent iterations. Consistency is enforced through shift queries and out-of-domain samples. STIR incorporates the resulting evaluation constraints by quotienting, whereas WHIR batches them into a sumcheck constraint on a constrained RS code.

We retain rate-preserving FRI folding to prioritize prover efficiency in the evaluation phase. After the initial encoding, all subsequent oracles are obtained by folding, without additional FFTs. Shifting instead requires re-encoding the folded polynomial on a new evaluation domain, introducing prover work that our construction avoids. We therefore leave this prover-verifier tradeoff unexplored here. Incorporating shifting after the initial interleaving reduction remains a promising direction for reducing verifier time and proof size.

4.3 Security Analysis

Completeness and binding For honestly generated commitments, linearity and the even-odd decomposition ensure that the honest prover's messages satisfy both evaluation checks in every round. The folding consistency checks also hold, establishing perfect completeness. Binding follows by distinguishing the candidates in the joint decoding list with the out-of-domain evaluations.

Theorem 4 (Binding). *Fix the public parameters. Let a malicious prover send an arbitrary oracle $\widehat{U}$ in Protocol 4.1, receive the honest verifier's challenge ζ, and send c. The probability that the resulting C_{IOP} admits two distinct polynomials accepted by $\mathsf{Open}^{\mathsf{IOP}}$ is at most $\varepsilon^{\mathsf{bind}} := \binom{B}{2}(k-1)/(q-N)$. The prover may be computationally unbounded and may choose c and both polynomials after receiving ζ.*

Proof. Condition on $\widehat{U}$, whose joint decoding list has size at most B by Lemma 3. Any two distinct tuples differ in a component polynomial of degree less than k, so their evaluation vectors coincide at a random $\zeta \in \mathbb{F} \setminus \mathcal{L}_0$ with probability at most $(k-1)/(q-N)$. A union bound over pairs gives $\varepsilon^{\mathsf{bind}}$. Outside this event, every evaluation vector c is consistent with at most one tuple. Since the component decomposition of f is unique, at most one polynomial opens the commitment. □

Folding with evaluation constraints The following extension of Lemma 2 uses MCA and the polynomial identity lemma, taking a union bound over the fixed joint decoding list. As in WHIR [ACFY25], the component polynomials must agree with their respective words on the same set.

Lemma 5 (Folding with evaluation constraints). *Let $\mathcal{D} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, K]$ have length $M = |\mathcal{L}|$ and dimension $2 \le K < M$, and fix $0 < \delta < 1 - \sqrt{K/M}$. Fix integers $r, s \ge 1$, words $\mathbf{w} = (w_0, \ldots, w_{r-1})$ on $\mathcal{L}$, points $\xi_1, \ldots, \xi_s \in \mathbb{F}$, and values $v_{i,b} \in \mathbb{F}$ for $0 \le i < r$ and $1 \le b \le s$. Let $B_{\mathcal{D}}$ bound $|\mathcal{J}_{\mathcal{D}}(\mathbf{w}, \delta)|$, and let err denote the MCA bound in Theorem 2.*

Except with probability at most

$$\mathsf{err}(\mathcal{D}, r, \delta) + \frac{(r-1)B_{\mathcal{D}}}{q}$$

over $\alpha \xleftarrow{\$} \mathbb{F}$*, the following implication holds simultaneously for every* $h \in \mathbb{F}^{<K}[X]$ *and* $S \subseteq \mathcal{L}$*. If*

$$|S| > (1-\delta)M, \quad h|_S = \sum_{i=0}^{r-1} \alpha^i w_i|_S, \quad h(\xi_b) = \sum_{i=0}^{r-1} \alpha^i v_{i,b}$$

for $1 \le b \le s$*, then there exist* $h_0, \ldots, h_{r-1} \in \mathbb{F}^{<K}[X]$ *such that*

$$h = \sum_{i=0}^{r-1} \alpha^i h_i, \quad h_i|_S = w_i|_S, \quad h_i(\xi_b) = v_{i,b}$$

for $0 \le i < r,\ 1 \le b \le s$*.*

Proof. Except with probability at most $\mathsf{err}(\mathcal{D}, r, \delta)$, MCA gives polynomials h_i agreeing with the respective words on S. They form a tuple in $\mathcal{J} := \mathcal{J}_{\mathcal{D}}(\mathbf{w}, \delta)$, and $|S| > (1-\delta)M > K-1$ implies $h = \sum_i \alpha^i h_i$.

The list $\mathcal{J}$ is fixed before α. For each tuple $\mathbf{h} \in \mathcal{J}$ violating an evaluation constraint, choose the least index b with some $h_i(\xi_b) \neq v_{i,b}$. Then $p_{\mathbf{h}}(T) := \sum_{i=0}^{r-1} T^i (h_i(\xi_b) - v_{i,b})$ is a fixed nonzero polynomial of degree at most $r-1$. A union bound over at most $B_{\mathcal{D}}$ tuples bounds the probability that any $p_{\mathbf{h}}(\alpha)$ vanishes by $(r-1)B_{\mathcal{D}}/q$. Selecting one violated constraint per tuple avoids a factor of s.

Outside both bad events, a tuple obtained from h, S cannot violate an evaluation constraint. Such a violation, together with the identity $h = \sum_i \alpha^i h_i$ and the claimed evaluation at its selected ξ_b, would imply $p_{\mathbf{h}}(\alpha) = 0$. The two bounds apply simultaneously to all h, S, proving the claim. □

Round-by-round knowledge soundness We adapt the base analysis in Appendix B.2, maintaining both evaluation claims in the state function and using list decoding for extraction.

Theorem 5 (Round-by-round knowledge soundness of Eval). *Fix the public parameters of Protocol 4.2. For every fixed commitment transcript* C_{IOP}*,* Eval *has round-by-round knowledge soundness for* $\mathcal{R}^{\mathsf{IOP}}_{\mathsf{Eval},\mathsf{pp}}$*, with the following errors:*

$$\varepsilon^{\mathsf{init}} = \mathsf{err}(\mathcal{C}_0, m, \delta) + \frac{(m-1)B}{q}, \quad \varepsilon^{\mathsf{fold}}_j = \mathsf{err}(\mathcal{C}_{j+1}, 2, \delta) + \frac{B}{q} \quad (0 \le j < t), \quad \varepsilon^{\mathsf{qry}} = (1-\delta)^Q.$$

The Q *query points are treated as a single final verifier challenge.*

The deterministic extractor E *makes* N *column queries to* $\widehat{U}$ *and runs in time polynomial in* m, N, B*, and* $\log q$*. It makes* m *calls to the RS list decoder and uses*

$$\mathcal{O}(mBNk + mB^2N + Bmk)$$

additional field operations and comparisons.

The extractor enumerates the joint decoding list and reconstructs a polynomial satisfying the opening predicate and $f(z) = y$, or returns $\perp$ if none exists. Appendix C.1 gives the algorithm and proof.

Corollary 2 (Knowledge soundness of Eval). *Let*

$$\kappa_{\mathsf{Eval}} := \mathsf{err}(\mathcal{C}_0, m, \delta) + \sum_{j=0}^{t-1} \mathsf{err}(\mathcal{C}_{j+1}, 2, \delta) + \frac{B(m-1+t)}{q} + (\sqrt{\rho} + \eta)^Q.$$

For every evaluation instance and malicious prover, the probability that Eval *accepts but* E *returns* $\perp$ *is at most* κ_{Eval}.

For every PPT prover $\mathcal{P}^*$ *whose acceptance probability is* p*, there exists a polynomial-time extractor* $\mathsf{Ext}^{\mathcal{P}^*}$ *that outputs a valid witness with probability at least* $p - \kappa_{\mathsf{Eval}}$.

The proof appears in Section C.2.

5 Implementation and Evaluation

We compare both ReedWeave variants with existing PCSs in commitment time, opening time, verification time, and proof size.

5.1 Implementation and Experimental Setup

Implementation and baselines. Our implementation is available anonymously at `https://anonymous.4open.science/r/ReedWeave/`. Our Rust implementation uses the Goldilocks base field $\mathbb{F}_p$, where $p = 2^{64} - 2^{32} + 1$, and BLAKE3 for hashing. We denote the unique-decoding construction of Section 3 by $\text{ReedWeave}_{\mathsf{UB}}$ and the DEEP-enhanced construction of Section 4 by $\text{ReedWeave}_{\mathsf{JB}}$. We compare with PCSs based on FRI [BBHR18] and STIR [ACFY24], WHIR [ACFY25], BaseFold [ZCF24], and the Brakedown and Shockwave PCSs [GLS+23]. We use the Plonky3 implementations of FRI, STIR, and WHIR,[3] Scroll's BaseFold implementation,[4] and the `lcpc` implementations of Brakedown and Shockwave.[5]

Experimental setup. All experiments run on the same server, equipped with an AMD EPYC 9754 128-Core Processor and 512 GB of RAM, running Ubuntu 22.04.5 LTS. We use 32 threads and report the average of five runs, with single-thread results in Appendix D. All schemes target 100-bit security. Initial encoding is performed over the Goldilocks base field, while verifier challenges are sampled from extension fields to achieve the target soundness error. We benchmark a single univariate evaluation proof for each scheme except WHIR, which proves a multilinear evaluation. The polynomial size d ranges from 2^{20} to 2^{34},

[3] `https://github.com/Plonky3/Plonky3`
[4] `https://github.com/scroll-tech/gkr-backend`
[5] `https://github.com/conroi/lcpc`

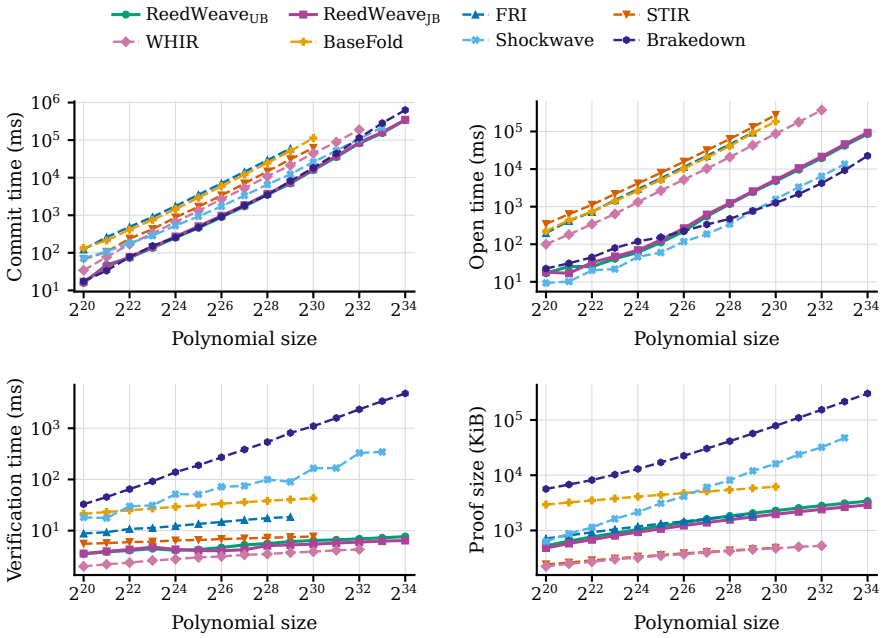


Fig. 2. PCS performance over Goldilocks at initial code rate 1/2, targeting 100-bit security with 32 threads. Points are averages of five runs. Brakedown uses its native code rate of approximately 0.657. A curve ends where a scheme runs out of memory or its FFT domain exceeds 2^{32}. Vertical axes are logarithmic. Lower is better.

denoting the number of coefficients for the univariate schemes and the number of evaluations over the Boolean hypercube for WHIR.

We benchmark initial code rates $\rho \in \{1/2, 1/4\}$, except for Brakedown, which uses its native code rate of approximately 0.657. ReedWeave uses interleaving parameter $m \in \{32, 64\}$, reducing the required multiplicative subgroup order by a factor of 32 or 64 relative to direct RS encoding.

We measure commitment and opening times separately and report their sum as total prover time. Times are reported in milliseconds or seconds, and proof sizes in KiB. All benchmarked implementations use Merkle multiproofs with path pruning to eliminate duplicate authentication nodes.

5.2 Comparison with Existing Polynomial Commitments

Figures 2 and 3 plot performance across both code rates, and Tables 2 and 3 report representative concrete costs at rates 1/2 and 1/4. Unless stated otherwise, numerical comparisons below refer to ReedWeave$_{\mathsf{JB}}$ at $d = 2^{24}$ and rate 1/4.

Comparison with RS-based PCSs. Both ReedWeave variants achieve lower commitment and opening times than the benchmarked FRI, STIR, and WHIR im-

Table 2. Concrete PCS costs over Goldilocks at initial code rate 1/2 with 32 threads and 100-bit security. Entries are averages of five runs. WHIR proves a multilinear evaluation, whereas the other schemes prove a univariate evaluation. Shaded rows correspond to our constructions. A dash indicates an unavailable configuration. At $d = 2^{32}$, FRI and STIR exceed the 2^{32} two-adicity limit of Goldilocks and BaseFold requires ∼1.7 TiB of RAM. At $d = 2^{34}$, we report only our constructions, which use ∼464 GiB of RAM, whereas all rate-1/2 baselines exceed the 2^{32} two-adicity limit or our 512 GB memory. † Brakedown uses its native code rate of approximately 0.657 rather than 1/2.

	PCS	Commit time (s)	Open time (s)	Verify time (ms)	Proof size (KiB)
$d = 2^{24}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	0.254	0.061	4.13	1041.4
	ReedWeave$_{\mathbf{JB}}$	0.271	0.069	4.30	932.6
	FRI	1.756	2.933	12.33	1179.8
	STIR	0.873	4.140	6.43	336.1
	WHIR	0.657	1.321	2.78	321.2
	Shockwave	0.529	0.046	51.56	2142.4
	BaseFold	1.457	2.653	29.16	4092.7
	Brakedown†	0.256	0.119	138.91	12990.2
$d = 2^{28}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	3.523	1.178	5.60	1828.5
	ReedWeave$_{\mathbf{JB}}$	3.707	1.243	5.08	1565.7
	FRI	29.233	45.602	17.55	1777.6
	STIR	14.639	63.018	7.26	435.5
	WHIR	10.885	20.764	3.49	422.5
	Shockwave	6.559	0.344	99.39	8125.3
	BaseFold	24.770	40.790	38.21	5434.8
	Brakedown†	3.477	0.474	536.74	41256.3
$d = 2^{32}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	82.336	19.625	6.91	2808.0
	ReedWeave$_{\mathbf{JB}}$	86.068	21.367	5.98	2408.4
	FRI	–	–	–	–
	STIR	–	–	–	–
	WHIR	189.102	371.635	4.29	528.5
	Shockwave	108.003	6.447	329.05	32000.1
	BaseFold	–	–	–	–
	Brakedown†	114.399	4.201	2340.45	153032.7
$d = 2^{34}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	340.564	85.427	7.65	3396.6
	ReedWeave$_{\mathbf{JB}}$	347.091	92.898	6.41	2896.2

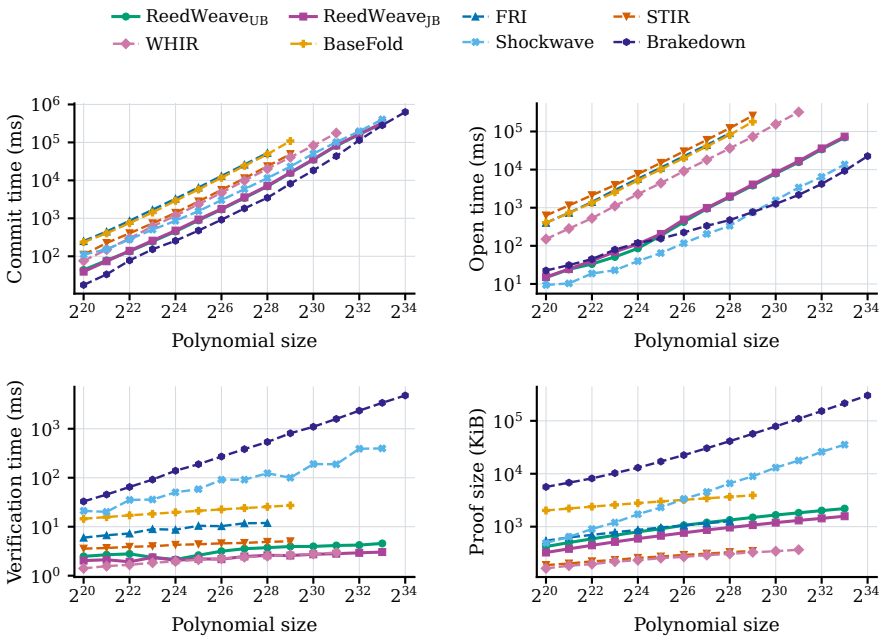


Fig. 3. PCS performance over Goldilocks at initial code rate 1/4, under the same experimental setup as Figure 2. Brakedown retains its native code rate of approximately 0.657, so its curve is unchanged. A curve ends where a scheme runs out of memory or its FFT domain exceeds 2^{32}. Vertical axes are logarithmic. Lower is better.

plementations across all measured polynomial sizes and code rates. At $d = 2^{24}$, ReedWeave$_{\mathsf{JB}}$ takes 340.7 ms in total prover time at rate 1/2 and 586.6 ms at rate 1/4, compared with 3.44 s for WHIR and 9.04 s for FRI and STIR at rate 1/4. It also improves over FRI in both proof size and verification time, whereas STIR and WHIR produce smaller proofs of 256.9 and 232.2 KiB at rate 1/4. Verification is faster than in STIR and within 5.7% of WHIR. Compared with WHIR, the prover speedup comes primarily from opening, which drops from 2270.2 to 111.6 ms at rate 1/4. Shockwave opens in 39.8 ms at the same point, but its slower commitment brings total prover time to 900.6 ms, while its proof is 2.89× larger and verification is 24.4× slower than in ReedWeave$_{\mathsf{JB}}$.

At larger polynomial sizes, $m = 64$ interleaving reduces the required two-adic subgroup order by 64×, allowing ReedWeave to scale beyond the 2^{32} two-adicity limit of Goldilocks. At $d = 2^{32}$ for rate 1/2 and $d = 2^{31}$ for rate 1/4, FRI and STIR can no longer be instantiated, while ReedWeave$_{\mathsf{JB}}$ remains over 5× faster than WHIR in total prover time and over 17× faster in opening. At the same sizes, its proofs are over 13× smaller and verify more than 55× faster than in Shockwave. At the largest measured sizes, ReedWeave uses ∼400 GiB of RAM at $d = 2^{33}$ and rate 1/4, and ∼464 GiB at $d = 2^{34}$ and rate 1/2, whereas WHIR

Table 3. Concrete PCS costs over Goldilocks at initial code rate 1/4 with 32 threads and 100-bit security. Entries are averages of five runs. WHIR proves a multilinear evaluation, whereas the other schemes prove a univariate evaluation. Shaded rows correspond to our constructions. A dash indicates an unavailable configuration. At $d = 2^{31}$, FRI and STIR exceed the 2^{32} two-adicity limit of Goldilocks and BaseFold requires ∼1.6 TiB of RAM. At $d = 2^{33}$, the largest size at rate 1/4, we report only our constructions, which use ∼400 GiB of RAM, whereas FRI, STIR, and WHIR exceed the 2^{32} two-adicity limit and BaseFold requires >6 TiB of RAM. † Brakedown uses its native code rate of approximately 0.657 rather than 1/4.

	PCS	Commit time (s)	Open time (s)	Verify time (ms)	Proof size (KiB)
$d = 2^{24}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	0.451	0.086	2.14	798.0
	ReedWeave$_{\mathbf{JB}}$	0.475	0.112	2.07	594.7
	FRI	3.288	5.748	8.68	866.4
	STIR	1.395	7.648	4.27	256.9
	WHIR	1.171	2.270	1.95	232.2
	Shockwave	0.861	0.040	50.42	1716.6
	BaseFold	2.908	5.185	19.68	2784.5
	Brakedown†	0.256	0.119	138.91	12990.2
$d = 2^{28}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	6.943	1.898	3.73	1335.2
	ReedWeave$_{\mathbf{JB}}$	7.174	1.979	2.61	964.1
	FRI	52.872	90.123	11.94	1262.8
	STIR	23.492	122.343	4.91	329.8
	WHIR	20.080	36.068	2.51	305.9
	Shockwave	11.464	0.336	123.71	6576.2
	BaseFold	48.677	81.200	25.51	3666.0
	Brakedown†	3.477	0.474	536.74	41256.3
$d = 2^{31}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	81.370	15.671	4.14	1824.9
	ReedWeave$_{\mathbf{JB}}$	83.183	16.683	2.82	1311.1
	FRI	–	–	–	–
	STIR	–	–	–	–
	WHIR	176.794	323.371	2.96	366.6
	Shockwave	101.776	3.399	188.35	17782.0
	BaseFold	–	–	–	–
	Brakedown†	43.529	2.172	1587.30	109544.9
$d = 2^{33}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	315.570	69.712	4.56	2195.9
	ReedWeave$_{\mathbf{JB}}$	321.552	73.434	3.04	1573.2

also exceeds the 2^{32} two-adicity limit and Shockwave requires over 512 GiB at $d = 2^{34}$, exceeding our server's 512 GB memory.

Comparison with field-agnostic PCSs. ReedWeave$_{\mathsf{JB}}$ outperforms BaseFold in total prover time, verification time, and proof size across all measured polynomial sizes and code rates, with total-prover speedups of 12.1× at rate 1/2 and 13.8× at rate 1/4 for $d = 2^{24}$, while BaseFold runs out of memory beyond $d = 2^{30}$ and requires over 6 TiB by $d = 2^{33}$. Brakedown achieves lower prover time at rate 1/4, but at $d = 2^{24}$ its proof is 21.8× larger and verification is 67.3× slower. At rate 1/2, ReedWeave$_{\mathsf{JB}}$ also achieves lower total prover time than Brakedown up to $d = 2^{24}$ and again from $d = 2^{32}$ onward, with a 104× smaller proof and 746× faster verification at $d = 2^{34}$. Unlike both baselines, ReedWeave requires a subgroup of order $d/(m\rho)$.

5.3 Comparison of Two ReedWeave Variants

ReedWeave$_{\mathsf{JB}}$ reduces proof size relative to ReedWeave$_{\mathsf{UB}}$ by 9.1-14.7% at rate 1/2 and by 22.8-28.7% at rate 1/4 across all measured polynomial sizes. Because both variants share the same commitment phase, the relative prover-time overhead of ReedWeave$_{\mathsf{JB}}$ decreases as commitment dominates at larger scales. At rate 1/4 with $m = 64$, lowering the query count Q from 149 to 101 at $d = 2^{24}$ reduces proof size by 25.5% with a 9.2% prover-time overhead, whereas by $d = 2^{33}$ the proof-size reduction reaches 28.4% with only a 2.5% overhead. Similarly, at rate 1/2, the proof-size reduction grows from 10.4% at $d = 2^{24}$ to 14.7% at $d = 2^{34}$, while the prover-time overhead drops from 8.2% to 3.3%.

AI Disclosure

During the preparation of this work, the author(s) used GPT 6-Astra to assist in drafting parts of the proofs and for language editing and stylistic polishing and code implementations. After using this tool, the author(s) thoroughly reviewed, independently verified, and revised all mathematical derivations and text. The author(s) take full responsibility for the integrity, correctness, and originality of the content of this publication.

References

ACFY24. G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. STIR: reed-solomon proximity testing with fewer queries. In *CRYPTO (10)*, *Lecture Notes in Computer Science* 14929, pages 380–413. Springer, 2024. https://doi.org/10.1007/978-3-031-68403-6_12.

ACFY25. G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev. WHIR: reed-solomon proximity testing with super-fast verification. In *EUROCRYPT (4)*, *Lecture Notes in Computer Science* 15604, pages 214–243. Springer, 2025. https://doi.org/10.1007/978-3-031-91134-7_8.

AHIV17. S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS*, pages 2087–2104. ACM, 2017. https://doi.org/10.1145/3133956.3134104.

BBHR18. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *ICALP*, *LIPIcs* 107, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. https://doi.org/10.4230/LIPIcs.ICALP.2018.14.

BCF+25. M. Brehm, B. Chen, B. Fisch, N. Resch, R. D. Rothblum, and H. Zeilberger. Blaze: Fast SNARKs from interleaved RAA codes. In *EUROCRYPT (4)*, *Lecture Notes in Computer Science* 15604, pages 123–152. Springer, 2025. https://doi.org/10.1007/978-3-031-91134-7_5.

BCGM26. S. Bordage, A. Chiesa, Z. Guan, and I. Manzur. All polynomial generators preserve distance with mutual correlated agreement. In *CCC*, *LIPIcs* 383, pages 24:1–24:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026. https://doi.org/10.4230/LIPIcs.CCC.2026.24.

BCKL22. E. Ben-Sasson, D. Carmon, S. Kopparty, and D. Levit. Scalable and transparent proofs over all large fields, via elliptic curves – (ECFFT part II). In *TCC (1)*, *Lecture Notes in Computer Science* 13747, pages 467–496. Springer, 2022. https://doi.org/10.1007/978-3-031-22318-1_17.

BCR+19. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT (1)*, *Lecture Notes in Computer Science* 11476, pages 103–128. Springer, 2019. https://doi.org/10.1007/978-3-030-17653-2_4.

BCS16. E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *TCC (B2)*, *Lecture Notes in Computer Science* 9986, pages 31–60, 2016. https://doi.org/10.1007/978-3-662-53644-5_2.

BFH+20. R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang. Ligero++: A new optimized sublinear IOP. In *CCS*, pages 2025–2038. ACM, 2020. https://doi.org/10.1145/3372297.3417893.

BFRW26. B. Bünz, G. Fenzi, R. D. Rothblum, and W. Wang. TensorSwitch: Nearly optimal polynomial commitments from tensor codes. In *CRYPTO (9)*, *Lecture Notes in Computer Science* 16808, pages 297–329. Springer, 2026. https://doi.org/10.1007/978-3-032-35424-2_11.

BFS20. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT (1)*, *Lecture Notes in Computer Science* 12105, pages 677–706. Springer, 2020. https://doi.org/10.1007/978-3-030-45721-1_24.

BGKS20. E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf. DEEP-FRI: Sampling outside the box improves soundness. In *ITCS*, *LIPIcs* 151, pages 5:1–5:32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. https://doi.org/10.4230/LIPIcs.ITCS.2020.5.

BSCH+25. E. Ben-Sasson, D. Carmon, U. Haböck, S. Kopparty, and S. Saraf. On proximity gaps for reed–solomon codes. Cryptology ePrint Archive, Paper 2025/2055, 2025. https://eprint.iacr.org/2025/2055.

CHK25. S. Chatterjee, P. Harsha, and M. Kumar. Deterministic list decoding of reed-solomon codes. arXiv preprint arXiv:2511.05176, 2025. https://doi.org/10.48550/arXiv.2511.05176.

GLH+25. Y. Guo, X. Liu, K. Huang, W. Qu, T. Tao, and J. Zhang. DeepFold: Efficient multilinear polynomial commitment from reed-solomon code and its application to zero-knowledge proofs. In *USENIX Security Symposium*, pages 3497–3516. USENIX Association, 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/guo-yanpei.

GLS+23. A. Golovnev, J. Lee, S. T. V. Setty, J. Thaler, and R. S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *CRYPTO (2)*, *Lecture Notes in Computer Science* 14082, pages 193–226. Springer, 2023. https://doi.org/10.1007/978-3-031-38545-2_7.

GS99. V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. https://doi.org/10.1109/18.782097.

Hab24. U. Haböck. Basefold in the list decoding regime. *IACR Cryptology ePrint Archive*, page 1571, 2024. https://eprint.iacr.org/2024/1571.

Hab25. U. Haböck. A note on mutual correlated agreement for reed–solomon codes. Cryptology ePrint Archive, Paper 2025/2110, 2025. https://eprint.iacr.org/2025/2110.

HLP24. U. Haböck, D. Levit, and S. Papini. Circle STARKs. *IACR Cryptology ePrint Archive*, page 278, 2024. https://eprint.iacr.org/2024/278.

KZG10. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, *Lecture Notes in Computer Science* 6477, pages 177–194. Springer, 2010. https://doi.org/10.1007/978-3-642-17373-8_11.

LLX+26. Z. Li, H. Liu, C. Xing, Y. Yao, and C. Yuan. More efficient SNARKs via quasi-abelian codes: Faster, smaller, and field-agnostic. *IACR Cryptol. ePrint Arch.*, 2026:939, 2026. https://eprint.iacr.org/2026/939.

NST25. V. Nair, A. Sharma, and B. Thankey. BrakingBase – a linear prover, poly-logarithmic verifier, field agnostic polynomial commitment scheme. In *ASIACRYPT (5)*, *Lecture Notes in Computer Science* 16249, pages 427–455. Springer, 2025. https://doi.org/10.1007/978-981-95-5116-3_14.

QGZ26. W. Qu, Y. Guo, and J. Zhang. Lightning, field-agnostic super-efficient polynomial commitment scheme. *IACR Cryptology ePrint Archive*, page 258, 2026. https://eprint.iacr.org/2026/258.

RZR24. N. Ron-Zewi and R. D. Rothblum. Local proofs approaching the witness length. *Journal of the ACM*, 71(3), 2024. https://doi.org/10.1145/3661483.

WB86. L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes. US Patent 4633470, 1986. https://patents.google.com/patent/US4633470A.

XZS22. T. Xie, Y. Zhang, and D. Song. Orion: Zero knowledge proof with linear prover time. In *CRYPTO (4)*, *Lecture Notes in Computer Science* 13510, pages 299–328. Springer, 2022. https://doi.org/10.1007/978-3-031-15985-5_11.

ZCF24. H. Zeilberger, B. Chen, and B. Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In *CRYPTO (10)*, *Lecture Notes in Computer Science* 14929, pages 138–169. Springer, 2024. https://doi.org/10.1007/978-3-031-68403-6_5.

ZXZS20. J. Zhang, T. Xie, Y. Zhang, and D. Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 859–876. IEEE, 2020. https://doi.org/10.1109/SP40000.2020.00052.

ZZW+26. Z. Zhang, X. Zhang, Y. Wei, L. Qin, and Y. Deng. LiftWHIR: A prover-efficient polynomial commitment with short proofs. Cryptology ePrint Archive, Paper 2026/1561, 2026. https://eprint.iacr.org/2026/1561.

Supplementary Material

A Additional Preliminaries

A.1 Standard Reed-Solomon Facts

Lemma 6 (Basic properties). *Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, k]$ be as in Definition 1.*

1. *The map Enc is linear and injective. The code $\mathcal{C}$ has dimension k, rate $\rho = k/N$, and relative minimum distance $\Delta_{\mathcal{C}} = (N-k+1)/N = 1-\rho+1/N$.*
2. *A polynomial in $\mathbb{F}^{<k}[X]$ is uniquely determined by its evaluations at any k distinct points of $\mathcal{L}$.*
3. *Every word $w : \mathcal{L} \to \mathbb{F}$ has at most one $P \in \mathbb{F}^{<k}[X]$ with $\Delta_{\mathcal{L}}(w, P|_{\mathcal{L}}) < \Delta_{\mathcal{C}}/2$.*

Proof (Proof of Lemma 6). Linearity is immediate. The root bound gives injectivity and the claimed distance and uniqueness. The distance bound is attained by zero and $\prod_{x \in S}(X - x)$ for $S \subseteq \mathcal{L}$ with $|S| = k - 1$. The unique-decoding claim follows from the triangle inequality. □

Unique decoding. The Berlekamp-Welch algorithm [WB86] deterministically recovers the unique polynomial of degree less than k differing from a received word in at most $e = \lfloor (N-k)/2 \rfloor$ positions, or reports failure. Solving its linear system by Gaussian elimination and verifying the result takes $\mathcal{O}(N^3)$ field operations, for any evaluation domain of distinct points.

A.2 List Decoding and Out-of-Domain Separation

For $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, k]$, a received word $w : \mathcal{L} \to \mathbb{F}$, and $\delta \geq 0$, define the decoding list $\mathsf{List}_{\mathcal{C}}(w, \delta) = \{P \in \mathbb{F}^{<k}[X] : \Delta_{\mathcal{L}}(w, P|_{\mathcal{L}}) \leq \delta\}$. The code is (δ, B)-list decodable if every such list has size at most B.

Lemma 7 (Johnson bound). *Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, k]$ have length N and rate $\rho = k/N \in (0,1)$. For every $0 \leq \delta < 1 - \sqrt{\rho}$ and every received word w, the list $\mathsf{List}_{\mathcal{C}}(w, \delta)$ has size at most $1/((1-\delta)^2 - \rho)$. In particular, at $\delta = 1 - \sqrt{\rho} - \eta$ with $0 < \eta < 1 - \sqrt{\rho}$, its size is at most $1/(2\eta\sqrt{\rho} + \eta^2)$.*

Proof (Proof of Lemma 7). Let L be the list size and assume $L > 0$. Put $a = 1 - \delta$ and $\bar{\rho} = (k-1)/N$. For each $x \in \mathcal{L}$, let r_x count the candidates agreeing with w at x. Every candidate agrees at at least aN positions, so $\sum_x r_x \geq LaN$. Distinct candidates agree with each other at at most $k-1$ positions. Consequently, Cauchy-Schwarz gives $L^2 a^2 N \leq \sum_{x \in \mathcal{L}} r_x^2$. Double counting gives $\sum_x r_x^2 = \sum_x r_x + 2\sum_x \binom{r_x}{2} \leq LN + L(L-1)(k-1)$. Thus $L(a^2 - \bar{\rho}) \leq 1 - \bar{\rho}$. Since $a^2 > \rho > \bar{\rho}$, this implies $L \leq 1/(a^2 - \rho)$. Substituting $a = \sqrt{\rho} + \eta$ gives the stated special case. □

The DEEP technique [BGKS20] uses random out-of-domain evaluations to distinguish candidates. The following elementary fact applies to a candidate list fixed before the challenge is sampled.

Lemma 8 (Separation by an out-of-domain evaluation). *Let $\mathcal{L} \subsetneq \mathbb{F}$ have size N, with $|\mathbb{F}| = q$, let $B \geq 1$ be an integer, and let $\mathcal{S} \subseteq \mathbb{F}^{<k}[X]$ be a fixed set of at most B distinct polynomials. For $\zeta \xleftarrow{\$} \mathbb{F} \setminus \mathcal{L}$, the probability that two distinct $P, P' \in \mathcal{S}$ satisfy $P(\zeta) = P'(\zeta)$ is at most $\binom{B}{2}(k-1)/(q-N)$. Outside this event, for every $v \in \mathbb{F}$ at most one $P \in \mathcal{S}$ satisfies $P(\zeta) = v$.*

Proof (Proof of Lemma 8). For each distinct pair P, P', the nonzero polynomial $P - P'$ has at most $k-1$ roots in $\mathbb{F}$. Sampling from $\mathbb{F} \setminus \mathcal{L}$ therefore yields a collision for this pair with probability at most $(k-1)/(q-N)$. A union bound over the at most $\binom{B}{2}$ pairs proves the claim. □

A.3 Folding

Definition 6 (Fold). *Let $f \in \mathbb{F}^{<d}[X]$ and let $r \geq 1$ be an integer. Write $f(X) = \sum_{i=0}^{r-1} X^i f_i(X^r)$. For a challenge $\gamma \in \mathbb{F}$, define the r-*fold *of f by $\mathsf{Fold}_r(f, \gamma)(Y) = \sum_{i=0}^{r-1} \gamma^i f_i(Y)$. This polynomial has degree less than $\lceil d/r \rceil$.*

For binary folding of oracle words, suppose that $\mathbb{F}$ has odd characteristic and $\mathcal{L} \subseteq \mathbb{F}^\star$ satisfies $-\mathcal{L} = \mathcal{L}$. Put $\mathcal{L}' = \{x^2 : x \in \mathcal{L}\}$. For $w : \mathcal{L} \to \mathbb{F}$, define $w_\mathsf{e}, w_\mathsf{o} : \mathcal{L}' \to \mathbb{F}$ by $w_\mathsf{e}(x^2) = (w(x) + w(-x))/2$ and $w_\mathsf{o}(x^2) = (w(x) - w(-x))/(2x)$. The denominators are nonzero, and both expressions are invariant under $x \mapsto -x$, so these words are well-defined. Set $\mathsf{Fold}_2(w, \gamma) = w_\mathsf{e} + \gamma w_\mathsf{o}$.

The polynomial decomposition in Definition 6 is unique. For $r = 2$, we write $g(X) = g_\mathsf{e}(X^2) + X g_\mathsf{o}(X^2)$, so $\mathsf{Fold}_2(g, \gamma) = g_\mathsf{e} + \gamma g_\mathsf{o}$. If $\deg g < d$ and d is even, then $\deg g_\mathsf{e}, \deg g_\mathsf{o} < d/2$. On the domains $\mathcal{L}, \mathcal{L}'$ above, we have $(g|_\mathcal{L})_\mathsf{e} = g_\mathsf{e}|_{\mathcal{L}'}$ and $(g|_\mathcal{L})_\mathsf{o} = g_\mathsf{o}|_{\mathcal{L}'}$. In particular, $\mathsf{Fold}_2(g|_\mathcal{L}, \gamma) = \mathsf{Fold}_2(g, \gamma)|_{\mathcal{L}'}$.

Remark 4. Let $\mathcal{L} \subseteq \mathbb{F}^\star$ be a subgroup of order $N = 2^n$. For every integer $a \geq 1$, the decomposition $f(X) = \sum_{i=0}^{a-1} X^i f_i(X^a)$ gives $\mathsf{Fold}_a(f, \alpha)(y) = \sum_i \alpha^i f_i(y)$ for $\alpha \in \mathbb{F}$ and $y \in \mathcal{L}$. These values require only the evaluations of the polynomials f_i on $\mathcal{L}$, so encoding requires $N \mid q-1$, not $aN \mid q-1$.

Lemma 9. *Let p be a prime and $a \geq 1$ an integer with $\gcd(a, p) = 1$. Then there exists a positive integer t such that $a \mid p^t - 1$.*

Proof. For $a = 1$, take $t = 1$. Otherwise, since $\gcd(a, p) = 1$, Euler's theorem gives $p^{\varphi(a)} \equiv 1 \pmod{a}$, so taking $t = \varphi(a)$ suffices. □

Corollary 3. *Let q be a prime power and let $\mathcal{L} \subseteq \mathbb{F}_q^\star$ be a subgroup of order N, where $N \mid q-1$. For every integer $a \geq 1$ with $\gcd(a, q) = 1$, there exist a positive integer t and an element $\omega \in \mathbb{F}_{q^t}^\star$ of order aN such that $\mathcal{L} = \langle \omega^a \rangle$. In particular, $\mathcal{L}' = \langle \omega \rangle$ contains $\mathcal{L}$, and the map $x \mapsto x^a$ from $\mathcal{L}'$ onto $\mathcal{L}$ is a-to-1.*

Proof. Write $q = p^s$, where p is prime. Since $N \mid q-1$ and $\gcd(a,q) = 1$, we have $\gcd(aN, p) = 1$. By Lemma 9, there exists a positive integer t such that $aN \mid p^t - 1$. Since $p^t - 1 \mid p^{st} - 1 = q^t - 1$, we also have $aN \mid q^t - 1$.

Let γ generate $\mathbb{F}_{q^t}^\star$ and set $\omega = \gamma^{(q^t-1)/(aN)}$. Then ω has order aN, and ω^a has order N. Both $\mathcal{L}$ and $\langle \omega^a \rangle$ are subgroups of order N in the cyclic group $\mathbb{F}_{q^t}^\star$, so they coincide. The power map on $\langle \omega \rangle$ has image $\langle \omega^a \rangle = \mathcal{L}$ and a kernel of size a, proving the final claim. □

A.4 Round-by-Round Knowledge Soundness

We adapt the following definitions from STIR [ACFY24] and WHIR [ACFY25] to IOPs.

Definition 7 (State function). *Let $(\mathcal{P}, \mathcal{V})$ be an IOP for a relation $\mathcal{R} = \{(x, w)\}$. A* state function *for $(\mathcal{P}, \mathcal{V})$ is a possibly inefficient function* State *that receives as inputs x and a transcript* tr*, outputs a bit, and has the following properties:*

- Empty transcript. *If* $\mathsf{tr} = \emptyset$*, then* $\mathsf{State}(x, \mathsf{tr}) = 1$ *if and only if* $x \in \mathcal{L}(\mathcal{R})$.
- Prover moves. *If* tr *is a transcript where the prover is about to move and* $\mathsf{State}(x, \mathsf{tr}) = 0$*, then, for every prover message* π*,* $\mathsf{State}(x, \mathsf{tr} \parallel \pi) = 0$.[6]
- Full transcript. *If* tr *is a full transcript and* $\mathsf{State}(x, \mathsf{tr}) = 0$*, then* $\mathcal{V}$ *rejects given this interaction transcript.*

Definition 8 (Round-by-round knowledge soundness). *A k-round IOP $(\mathcal{P}, \mathcal{V})$ for a relation $\mathcal{R} = \{(x, w)\}$ has* round-by-round knowledge soundness *with errors $(\varepsilon_1, \ldots, \varepsilon_k)$ and extraction time* et *if the IOP has a state function* State *and there exists a deterministic extractor* E *that runs in time at most* et *with the following property. For every x, round $i \in [k]$, and transcript* $\mathsf{tr} = (\pi_0, \alpha_1, \pi_1, \ldots, \alpha_{i-1}, \pi_{i-1})$*, if*

$$\mathsf{State}(x, \mathsf{tr}) = 0 \text{ and } \Pr_{\alpha_i}\big[\mathsf{State}(x, \mathsf{tr} \parallel \alpha_i) = 1\big] > \varepsilon_i(x),$$

then $(x, \mathsf{E}(x, \mathsf{tr})) \in \mathcal{R}$.

If et *is unbounded, we omit the word "knowledge" and say that the IOP has* round-by-round soundness.

B Proofs for the Base Construction

B.1 Folding and evaluation in the unique-decoding regime

Proof (Proof of Lemma 2). The parameter δ satisfies $\delta \le (1 - K/M)/2 < 1 - \sqrt{K/M}$. Let $\mathsf{Bad}_{\mathsf{MCA}}$ be the event that mutual correlated agreement fails. This happens with probability at most $(r-1)M/q$ according to Theorem 2. Since $2\delta \le 1 - K/M < \Delta_{\mathcal{D}}$, Lemma 6 guarantees that each w_i has at most one

[6] $\parallel$ denotes transcript concatenation.

polynomial of degree less than K at relative distance less than δ. If some w_i has no such polynomial, no pair h, S can satisfy the hypotheses outside $\mathsf{Bad}_{\mathsf{MCA}}$, and the claim follows.

Otherwise, let h_i be the unique polynomial satisfying $\Delta_{\mathcal{L}}(w_i, h_i|_{\mathcal{L}}) < \delta$. These polynomials depend only on the words w_i, not on α. Outside $\mathsf{Bad}_{\mathsf{MCA}}$, every pair h, S satisfying the hypotheses has $h_i|_S = w_i|_S$ for all i. Since $|S| > (1-\delta)M > K-1$, Lemma 6 gives the polynomial identity $h = \sum_i \alpha^i h_i$. If some $h_i(\zeta) \neq v_i$, the polynomial $p(T) = \sum_{i=0}^{r-1} T^i(h_i(\zeta) - v_i)$ is nonzero, has degree at most $r-1$, and is independent of α. The claimed evaluation of h implies $p(\alpha) = 0$, which occurs with probability at most $(r-1)/q$. Taking a union bound over the $\mathsf{Bad}_{\mathsf{MCA}}$ failure event and the event $p(\alpha) = 0$ gives the stated error bound, simultaneously for all h and S. □

B.2 Round-by-round knowledge soundness of the base protocol

Proof (Proof of Theorem 3). We define the state function and extractor, and bound the round-by-round knowledge soundness errors. We apply Definition 8 with instance $(\widehat{U}, z, y)$, where $\widehat{U}$ is accessed as an input oracle. Consecutive prover messages without an intervening verifier challenge are treated as one move. Specifically, after γ_{j-1} the prover sends (π_j, a_j, b_j) for $1 \leq j < t$, and after α it sends (a_0, b_0). We include the vector of Q points sampled from $\mathcal{L}_0$ in the transcript as the final verifier message, followed by an empty prover response. The verifier then performs the prescribed oracle queries and checks. This regrouping of messages does not change the verifier's acceptance probability.

Notation. For $x \in \mathcal{L}_0$, write $x_j = x^{2^j}$. Whenever the relevant messages have been sent, define $w_{j+1} = \mathsf{Fold}_2(\pi_j, \gamma_j)$ and let $A_j \subseteq \mathcal{L}_0$ be the set of points x for which the first j folding checks pass. Thus $A_0 = \mathcal{L}_0$, and $A_{j+1} = \{x \in A_j : w_{j+1}(x_{j+1}) = \pi_{j+1}(x_{j+1})\}$. Set $\pi_t = g^{(t)}|_{\mathcal{L}_t}$ as in the base protocol. For $j < t$ and $h \in \mathbb{F}^{<k_j}[X]$, set $S_j(h) = \{x \in A_j : \pi_j(x_j) = h(x_j) \text{ and } \pi_j(-x_j) = h(-x_j)\}$. These sets are used only in the state function. The verifier does not compute them.

State function. We suppress the instance $(\widehat{U}, z, y)$ in the notation for State. After any rejection, $\mathsf{State} = 0$ at that prefix and all its extensions. Otherwise, the state is one exactly when the corresponding condition below holds.

1. *Empty transcript.* There exists $f \in \mathbb{F}^{<d}[X]$ with $\Delta_{\mathcal{L}_0}(\widehat{U}, \mathsf{Enc}(U_f)) < \Delta_{\mathcal{C}_0}/2$ and $f(z) = y$.
2. *After v.* There exist $h_0, \ldots, h_{m-1} \in \mathbb{F}^{<k}[X]$ with $h_i(z^m) = v_i$ for every i and
$$|\{x \in \mathcal{L}_0 : \widehat{U}_i(x) = h_i(x) \text{ for every } i\}| > (1-\delta)N.$$
3. *After α.* There exists $h \in \mathbb{F}^{<k}[X]$ with $h(z_0) = y_0$ and $|\{x \in \mathcal{L}_0 : \pi_0(x) = h(x)\}| > (1-\delta)N$.
4. *Before γ_j ($0 \leq j < t$).* There exists $h \in \mathbb{F}^{<k_j}[X]$ with $|S_j(h)| > (1-\delta)N$, $h_{\mathsf{e}}(z_j^2) = a_j$, and $h_{\mathsf{o}}(z_j^2) = b_j$, where $h(X) = h_{\mathsf{e}}(X^2) + X h_{\mathsf{o}}(X^2)$.

5. *After γ_j ($0 \le j < t$).* There exists $h \in \mathbb{F}^{<k_{j+1}}[X]$ with $h(z_{j+1}) = y_{j+1}$ and $|\{x \in A_j : w_{j+1}(x_{j+1}) = h(x_{j+1})\}| > (1-\delta)N$.
6. *After the final polynomial.* $|A_t| > (1-\delta)N$.
7. *After the final verifier message.* All Q sampled points belong to A_t.

Prover moves preserve the zero state. We show that a state of one after a prover move implies a state of one before it. After v, the polynomials in Item 2 give $f(X) = \sum_i X^i h_i(X^m)$, with $\Delta_{\mathcal{L}_0}(\widehat{U}, \mathsf{Enc}(U_f)) < \delta < \Delta_{\mathcal{C}_0}/2$. The verifier's check gives $f(z) = \sum_i z^i v_i = y$. Thus Item 2 implies Item 1.

For the move (a_0, b_0), a polynomial satisfying Item 4 also satisfies Item 3, since $S_0(h)$ is contained in its agreement set with π_0 and $h(z_0) = a_0 + z_0 b_0 = y_0$. For $1 \le j < t$, suppose that Item 4 holds after (π_j, a_j, b_j). For every $x \in S_j(h)$, the definition of A_j gives $w_j(x_j) = \pi_j(x_j) = h(x_j)$ and $x \in A_{j-1}$. The verifier's check also gives $h(z_j) = a_j + z_j b_j = y_j$. Hence the same polynomial satisfies Item 5 for the preceding challenge γ_{j-1}.

After the final polynomial, Item 6 implies Item 5 for γ_{t-1}, because A_t is exactly the agreement set of the checked polynomial $g^{(t)}$ with w_t within A_{t-1}. The empty final prover response preserves the state. Thus prover moves cannot change the state from zero to one. Item 1 gives the condition at the empty transcript, and Item 7 ensures rejection at a full transcript with $\mathsf{State} = 0$, establishing the remaining conditions of Definition 7.

Initial folding. Fix a prefix ending with v. Apply Lemma 2 to the m rows of $\widehat{U}$, the code $\mathcal{C}_0$, the point z^m, and the values v_i. Except with probability at most $(m-1)(N+1)/q$ over α, any polynomial satisfying Item 3 yields component polynomials satisfying Item 2 on the same agreement set. Therefore, for every prefix with $\mathsf{State} = 0$ before α, the probability that $\mathsf{State} = 1$ after α is at most $\varepsilon^{\mathsf{init}}$.

Binary folding. Fix $0 \le j < t$ and a prefix immediately before γ_j. Apply Lemma 2 to the words $(\pi_j)_{\mathsf{e}}, (\pi_j)_{\mathsf{o}}$ on $\mathcal{L}_{j+1}$, the code $\mathcal{C}_{j+1}$, the point z_j^2, and the values a_j, b_j. The hypotheses hold because $k_{j+1} \ge k_t > 1$ and $k_{j+1}/N_{j+1} = \rho$. The lemma gives an exceptional set of challenges of probability at most $(N_{j+1}+1)/q$.

Fix a challenge outside this set and suppose that $\mathsf{State} = 1$ after γ_j, with a polynomial h satisfying Item 5. Let $T = \{x \in A_j : w_{j+1}(x_{j+1}) = h(x_{j+1})\}$ and $S = \{x_{j+1} : x \in T\} \subseteq \mathcal{L}_{j+1}$. Each $y \in S$ has at most 2^{j+1} preimages in T under the map $x \mapsto x^{2^{j+1}}$. Hence $|S| \ge |T|/2^{j+1} > (1-\delta)N_{j+1}$. On S, the polynomial h agrees with w_{j+1}, and its evaluation is $h(z_j^2) = a_j + \gamma_j b_j$. The lemma therefore gives $h_{\mathsf{e}}, h_{\mathsf{o}} \in \mathbb{F}^{<k_{j+1}}[X]$ agreeing with these two words on S, with $h_{\mathsf{e}}(z_j^2) = a_j$ and $h_{\mathsf{o}}(z_j^2) = b_j$.

Set $\widetilde{h}(X) = h_{\mathsf{e}}(X^2) + X h_{\mathsf{o}}(X^2)$. For every $x \in T$, agreement of both words at $x_{j+1} = x_j^2$ implies $\pi_j(x_j) = \widetilde{h}(x_j)$ and $\pi_j(-x_j) = \widetilde{h}(-x_j)$. Since $\deg \widetilde{h} < 2k_{j+1} = k_j$ and $T \subseteq S_j(\widetilde{h})$, the polynomial $\widetilde{h}$ satisfies Item 4. Therefore $\mathsf{State} = 1$ after a challenge outside the exceptional set implies $\mathsf{State} = 1$ before it. For every prefix

with $\mathsf{State} = 0$, the probability of a transition to 1 is consequently at most $\varepsilon_j^{\mathsf{fold}}$. The bound holds for every prefix, including prefixes generated by an adaptive prover.

Query phase. After the final polynomial, all oracle tables and A_t are determined. If $\mathsf{State} = 0$ and the verifier has not rejected, Item 6 gives $|A_t| \le (1-\delta)N$. Since the Q points are sampled independently and uniformly from $\mathcal{L}_0$, the probability that all belong to A_t is $(|A_t|/N)^Q \le (1-\delta)^Q$. This proves the bound $\varepsilon^{\mathsf{qry}}$. No independence between queries to different oracles within the same repetition is used.

Extractor. The extractor $\mathsf{E}(\widehat{U}, z, y, \mathsf{tr})$ ignores tr and proceeds as follows.

1. Read all N columns of $\widehat{U}$ and uniquely decode each row in $\mathcal{C}_0$ up to $e = \lfloor (N-k)/2 \rfloor$ errors. Return $\perp$ if any row has no codeword within this radius.
2. Let $h_0, \ldots, h_{m-1}$ be the decoded polynomials and reconstruct
$$f(X) = \sum_i X^i h_i(X^m).$$
3. Check $\Delta_{\mathcal{L}_0}(\widehat{U}, \mathsf{Enc}(U_f)) < \Delta_{\mathcal{C}_0}/2$ and $f(z) = y$. Return f if both checks pass, and $\perp$ otherwise.

The distance check is over $\mathbb{F}^m$, since separate row-distance bounds would not suffice.

Correctness. Whenever a witness exists, each row differs from its encoding in at most $(N-k)/2$ positions. Unique decoding therefore recovers all its component polynomials, and the final checks succeed. Conversely, every returned polynomial satisfies the required distance and evaluation conditions.

Running time. The decoder recalled in Section A.1 uses $\mathcal{O}(N^3)$ field operations per row. Reconstruction and the final checks use $\mathcal{O}(mNk)$ field operations. The total cost is therefore $\mathcal{O}(mN^3)$, with N column queries.

If a witness exists, E recovers it on every prefix. Otherwise, the transition bounds above rule out a 0-to-1 transition probability exceeding the corresponding error. Thus the state function and extractor satisfy Definition 8. □

B.3 Knowledge soundness of the base protocol

Proof (Proof of Corollary 1). Fix an evaluation instance $I = (\widehat{U}, z, y)$ and a malicious prover $\mathcal{P}^*$. Let Accept denote the acceptance event in an execution with the honest verifier, and let tr be the resulting transcript. If $\mathsf{E}(I, \mathsf{tr}) = \perp$, no witness satisfying the stated conditions exists, and the empty-transcript state is zero. An accepting execution ends in state one. Since prover moves cannot change zero to one, some verifier challenge must cause such a transition. For each verifier move, condition on the preceding transcript and apply the corresponding transition bound from Theorem 3. Averaging over prefixes and taking a union

bound gives κ, also for adaptive prover strategies. The expression for A follows from $\sum_{j=0}^{t-1} N_{j+1} = N - N_t$.

Define $\mathsf{Ext}^{\mathcal{P}^*}(I)$ to run one execution with $\mathcal{P}^*$ and the honest verifier. If the verifier rejects, it returns $\perp$. Otherwise, it returns $\mathsf{E}(I, \mathsf{tr})$. The extractor succeeds exactly when the verifier accepts and E returns a valid witness. Since E returns only valid witnesses or $\perp$, writing $p = \Pr[\mathsf{Accept}]$ gives

$$\begin{aligned}&\Pr[\mathsf{Ext}^{\mathcal{P}^*}(I) \text{ outputs a valid witness}]\\ &\quad = p - \Pr[\mathsf{Accept} \wedge \mathsf{E}(I, \mathsf{tr}) = \perp] \geq p - \kappa.\end{aligned}$$

The extractor runs the protocol once and invokes E at most once, using $\mathcal{O}(mN^3)$ additional field operations. Its running time is therefore polynomial for every PPT prover. □

C Additional Proofs for the DEEP Construction

C.1 Round-by-round knowledge soundness of the DEEP protocol

Proof (Proof of Theorem 5). Fix $I = (C_{\mathsf{IOP}}, z, y)$. As in Theorem 3, group consecutive prover messages into one move and treat the Q query points as a single final verifier challenge, followed by an empty prover response.

Notation. For $x \in \mathcal{L}_0$, write $x_j = x^{2^j}$. Whenever the relevant messages have been sent, define $w_{j+1} = \mathsf{Fold}_2(\pi_j, \gamma_j)$ and let $A_j \subseteq \mathcal{L}_0$ be the set of points x for which the first j folding checks pass. Thus $A_0 = \mathcal{L}_0$, and $A_{j+1} = \{x \in A_j : w_{j+1}(x_{j+1}) = \pi_{j+1}(x_{j+1})\}$. Set $\pi_t = g^{(t)}|_{\mathcal{L}_t}$ as in the DEEP protocol. For $j < t$ and $h \in \mathbb{F}^{<k_j}[X]$, set $S_j(h) = \{x \in A_j : \pi_j(x_j) = h(x_j) \text{ and } \pi_j(-x_j) = h(-x_j)\}$. These sets are used only in the state function. The verifier does not compute them.

State function. We suppress I in the notation. After any rejection, $\mathsf{State} = 0$ at that prefix and all its extensions. Otherwise, the state is one exactly when the corresponding condition below holds.

1. *Empty transcript.* $I \in \mathcal{L}(\mathcal{R}^{\mathsf{IOP}}_{\mathsf{Eval},\mathsf{pp}})$.
2. *After v.* There exists $\mathbf{h} \in \mathcal{J}_{\mathcal{C}_0}(\widehat{U}, \delta)$ with $h_i(z^m) = v_i$ and $h_i(\zeta) = c_i$ for every i.
3. *After α.* There exists $h \in \mathbb{F}^{<k}[X]$ with $h(z_0) = y_0$, $h(\zeta_0) = u_0$, and $|\{x \in \mathcal{L}_0 : \pi_0(x) = h(x)\}| > (1-\delta)N$.
4. *Before γ_j ($0 \leq j < t$).* There exists $h \in \mathbb{F}^{<k_j}[X]$ with $|S_j(h)| > (1-\delta)N$ and
$$h_{\mathsf{e}}(z_j^2) = a_j, \quad h_{\mathsf{o}}(z_j^2) = b_j,$$
$$h_{\mathsf{e}}(\zeta_j^2) = \widetilde{a}_j, \quad h_{\mathsf{o}}(\zeta_j^2) = \widetilde{b}_j.$$
5. *After γ_j ($0 \leq j < t$).* There exists $h \in \mathbb{F}^{<k_{j+1}}[X]$ with $h(z_{j+1}) = y_{j+1}$ and $h(\zeta_{j+1}) = u_{j+1}$ such that more than $(1-\delta)N$ points $x \in A_j$ satisfy $w_{j+1}(x_{j+1}) = h(x_{j+1})$.
6. *After the final polynomial.* $|A_t| > (1-\delta)N$.
7. *After the final verifier challenge.* All Q sampled points belong to A_t.

Prover moves preserve the zero state. We show that a state of one after a prover move implies a state of one before it. After v, a tuple in Item 2 gives $f(X) = \sum_i X^i h_i(X^m)$ satisfying the opening predicate. The initial evaluation check gives $f(z) = y$. Thus Item 2 implies Item 1.

For the move after α, Item 4 with $j = 0$ implies Item 3 by the two evaluation checks and the containment of $S_0(h)$ in the agreement set with π_0. For $1 \le j < t$, every $x \in S_j(h)$ satisfies $w_j(x_j) = \pi_j(x_j) = h(x_j)$ and $x \in A_{j-1}$. The evaluation checks also give $h(z_j) = y_j$ and $h(\zeta_j) = u_j$. Hence Item 4 implies Item 5 for the preceding challenge.

After the final polynomial, Item 6 implies Item 5 for γ_{t-1}, using the checked polynomial $g^{(t)}$. The empty final response preserves the state. Thus prover moves cannot change the state from zero to one. Items 1 and 7 give the remaining conditions of Definition 7.

Initial folding. Apply Lemma 5 to the rows of $\widehat{U}$, code $\mathcal{C}_0$, points (z^m, ζ), and value vectors (v, c), all fixed before α. Since the joint decoding list has size at most B, Item 3 implies Item 2 except with probability $\varepsilon^{\mathsf{init}}$.

Binary folding. Fix a prefix before γ_j. Apply Lemma 5 to $(\pi_j)_{\mathsf{e}}, (\pi_j)_{\mathsf{o}}$ on $\mathcal{L}_{j+1}$, code $\mathcal{C}_{j+1}$, points (z_j^2, ζ_j^2), and value pairs (a_j, b_j) and $(\widetilde{a}_j, \widetilde{b}_j)$. Here $k_{j+1} \ge k_t > 1$ and $k_{j+1}/N_{j+1} = \rho$, giving exceptional probability at most $\varepsilon_j^{\mathsf{fold}}$.

Outside this event, let h satisfy Item 5. Set $T = \{x \in A_j : w_{j+1}(x_{j+1}) = h(x_{j+1})\}$ and $S = \{x_{j+1} : x \in T\}$. Each $y \in S$ has at most 2^{j+1} preimages in T under the map $x \mapsto x^{2^{j+1}}$. Hence $|S| \ge |T|/2^{j+1} > (1-\delta)N_{j+1}$. The lemma gives polynomials $h_{\mathsf{e}}, h_{\mathsf{o}}$ agreeing with the two words on S and satisfying all four evaluation constraints. Set $\widetilde{h}(X) = h_{\mathsf{e}}(X^2) + X h_{\mathsf{o}}(X^2)$. It has degree less than k_j and agrees with π_j at $\pm x_j$ for every $x \in T$. Hence $T \subseteq S_j(\widetilde{h})$ and Item 4 holds. The probability that the state changes from zero to one is therefore at most $\varepsilon_j^{\mathsf{fold}}$, also for adaptively generated prefixes.

Query phase. If the verifier has not rejected and the state is zero before the query challenge, then $|A_t| \le (1-\delta)N$. The probability that all Q independent uniform samples lie in A_t is $(|A_t|/N)^Q \le (1-\delta)^Q$, giving $\varepsilon^{\mathsf{qry}}$.

Extractor. The extractor ignores the transcript and proceeds as follows.

1. Read all N input columns and enumerate $\mathcal{J}_{\mathcal{C}_0}(\widehat{U}, \delta)$ in a fixed order.
2. Find the first tuple $\mathbf{h}$ satisfying $h_i(\zeta) = c_i$ for every i and $\sum_i z^i h_i(z^m) = y$. Return $\perp$ if no tuple passes these checks.
3. Form $f(X) = \sum_i X^i h_i(X^m)$ and verify the opening predicate and $f(z) = y$. Return f if both checks pass, and $\perp$ otherwise.

Correctness. Whenever a witness exists, its component tuple belongs to the joint decoding list and passes the evaluation checks. Every tuple selected in Step 2 therefore yields a valid witness in Step 3. Conversely, every output is valid.

Running time. By Lemma 4, enumeration uses m calls to the RS list decoder and $\mathcal{O}(mBNk+mB^2N+mBk)$ additional operations. The evaluation tests cost $\mathcal{O}(Bmk)$ operations, and the final distance check costs $\mathcal{O}(mNk)$. Together these give the stated bound.

If a witness exists, the extractor recovers a witness on every prefix. Otherwise, the transition bounds above imply the extraction condition of Definition 8. □

C.2 Knowledge soundness of the DEEP protocol

Proof (Proof of Corollary 2). Fix an evaluation instance $I = (C_{\mathsf{IOP}}, z, y)$ and a malicious prover $\mathcal{P}^*$. Let Accept denote the acceptance event in an execution with the honest verifier, and let tr be the resulting transcript. If $\mathsf{E}(I, \mathsf{tr}) = \perp$, then I has no valid witness and the initial state is zero. An accepting execution ends in state one. Since prover moves preserve the zero state, some verifier challenge must cause a transition from zero to one. Conditioning on the preceding transcript and summing the bounds of Theorem 5 gives

$$\Pr[\mathsf{Accept} \wedge \mathsf{E}(I, \mathsf{tr}) = \perp] \leq \kappa_{\mathsf{Eval}}.$$

Define $\mathsf{Ext}^{\mathcal{P}^*}(I)$ to run one execution with $\mathcal{P}^*$ and the honest verifier. If the verifier rejects, it returns $\perp$. Otherwise, it returns $\mathsf{E}(I, \mathsf{tr})$. The extractor succeeds exactly when the verifier accepts and E returns a valid witness. Since E returns only valid witnesses or $\perp$, writing $p = \Pr[\mathsf{Accept}]$ gives

$$\begin{aligned}&\Pr[\mathsf{Ext}^{\mathcal{P}^*}(I) \text{ outputs a valid witness}] \\ &\quad = p - \Pr[\mathsf{Accept} \wedge \mathsf{E}(I, \mathsf{tr}) = \perp] \geq p - \kappa_{\mathsf{Eval}}.\end{aligned}$$

The extractor runs one protocol execution and invokes E at most once, so its running time is polynomial for every PPT prover. □

D Single-Thread Evaluation

D.1 Rate 1/2

We evaluate the same implementations using a single thread at initial code rate 1/2, while Brakedown uses its native code rate of approximately 0.657. Each ReedWeave variant uses the same protocol parameters as in the 32-thread evaluation. Figure 4 and Table 4 report the results. We use $\text{ReedWeave}_{\mathsf{JB}}$ for representative comparisons below and report commitment plus opening time as total prover time.

Comparison with RS-based PCSs. Both ReedWeave variants achieve lower commitment and opening times than the benchmarked FRI, STIR, and WHIR implementations across all measured polynomial sizes. At $d = 2^{24}$, $\text{ReedWeave}_{\mathsf{JB}}$ takes 2.095 s in total prover time, compared with 18.78 s for WHIR, 31.83 s for STIR, and 38.76 s for FRI, and also improves over FRI in both proof size and

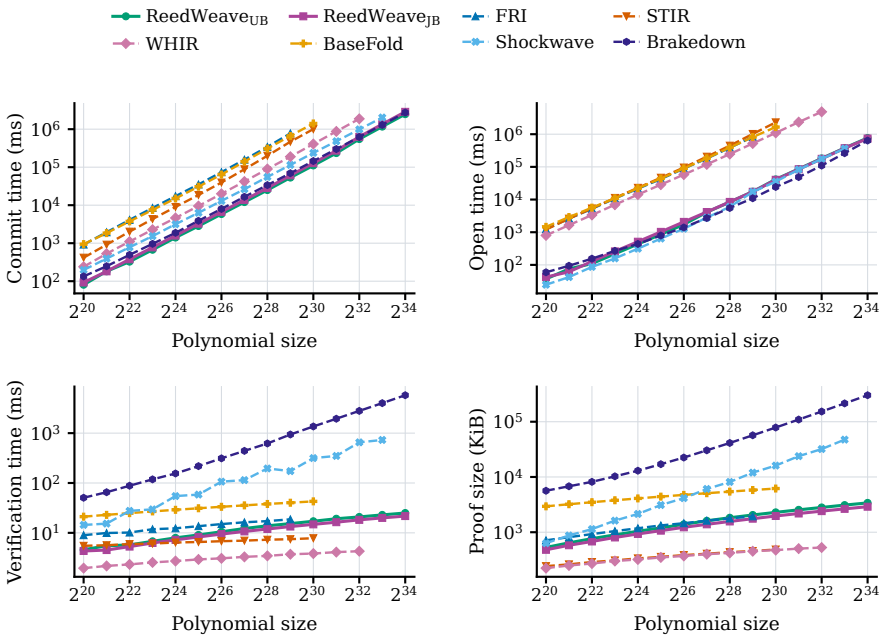


Fig. 4. Single-thread PCS performance over Goldilocks at initial code rate 1/2 and 100-bit security. Brakedown uses its native code rate of approximately 0.657. Points are averages of five runs. A curve ends where a scheme runs out of memory or its FFT domain exceeds 2^{32}. Vertical axes are logarithmic. Lower is better.

verification time. STIR and WHIR produce smaller proofs of 336.1 and 321.2 KiB and verify in 6.40 and 2.72 ms, while ReedWeave$_{\mathsf{JB}}$ is 2.9× faster than WHIR in commitment and 27.7× faster in opening. At $d = 2^{32}$, where FRI and STIR exceed the 2^{32} two-adicity limit, this gap grows to an 8.3× total-prover speedup over WHIR.

Shockwave trades a slower commitment for a faster opening. At $d = 2^{24}$, its higher commitment cost brings total prover time to 3.468 s, while its proof is 2.30× larger and verification is 7.56× slower than in ReedWeave$_{\mathsf{JB}}$. By $d = 2^{32}$, ReedWeave$_{\mathsf{JB}}$ produces a 13.3× smaller proof and verifies 35.9× faster than Shockwave, while also taking less total prover time.

Comparison with field-agnostic PCSs. ReedWeave$_{\mathsf{JB}}$ outperforms BaseFold in total prover time, verification time, and proof size across all measured polynomial sizes, achieving a 17.8× total-prover speedup at $d = 2^{24}$. ReedWeave$_{\mathsf{UB}}$ also achieves lower total prover time than Brakedown across all measured polynomial sizes in this single-thread setting, and ReedWeave$_{\mathsf{JB}}$ does so up to $d = 2^{30}$, while Brakedown's proofs and verification times are orders of magnitude larger.

Comparison of two ReedWeave variants. The DEEP variant reduces proof size by 9.3-14.7% across all measured polynomial sizes, from 10.6% at $d = 2^{24}$ to 14.7% at $d = 2^{34}$, with a 10.1-13.7% prover-time overhead. At $d = 2^{34}$, our constructions require ∼464 GiB of RAM, whereas no rate-1/2 baseline completes due to the 2^{32} two-adicity limit or our 512 GB memory limit.

Table 4. Concrete single-thread costs under the setup of Figure 4. WHIR proves a multilinear evaluation, whereas the other schemes prove a univariate evaluation. Shaded rows correspond to our constructions. A dash indicates an unavailable configuration. At $d = 2^{32}$, FRI and STIR exceed the 2^{32} two-adicity limit of Goldilocks and BaseFold requires ∼1.7 TiB of RAM. At $d = 2^{34}$, we report only our constructions, which use ∼464 GiB of RAM, whereas all rate-1/2 baselines exceed the 2^{32} two-adicity limit or our 512 GB memory. † Brakedown uses its native code rate of approximately 0.657 rather than 1/2.

	PCS	Commit time (s)	Open time (s)	Verify time (ms)	Proof size (KiB)
$d = 2^{24}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	1.409	0.434	7.93	1041.1
	ReedWeave$_{\mathbf{JB}}$	1.583	0.512	7.23	930.8
	FRI	17.283	21.481	12.33	1179.8
	STIR	8.984	22.847	6.40	336.1
	WHIR	4.616	14.163	2.72	321.2
	Shockwave	3.154	0.314	54.66	2142.4
	BaseFold	15.039	22.244	29.00	4092.7
	Brakedown†	1.874	0.438	155.15	12990.2
$d = 2^{28}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	25.032	8.360	13.85	1828.5
	ReedWeave$_{\mathbf{JB}}$	28.305	8.471	11.94	1565.7
	FRI	342.851	385.619	17.20	1777.6
	STIR	199.673	445.622	7.28	435.5
	WHIR	89.676	246.352	3.45	422.5
	Shockwave	55.320	5.980	195.68	8125.3
	BaseFold	309.116	390.517	37.99	5434.8
	Brakedown†	33.807	5.519	620.63	41256.3
$d = 2^{32}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	550.185	181.469	20.86	2808.0
	ReedWeave$_{\mathbf{JB}}$	629.293	176.466	18.22	2408.4
	FRI	–	–	–	–
	STIR	–	–	–	–
	WHIR	1872.716	4840.817	4.28	528.5
	Shockwave	986.747	174.210	653.93	32000.1
	BaseFold	–	–	–	–
	Brakedown†	622.775	109.437	2793.62	153032.7
$d = 2^{34}$ rate = 1/2	**ReedWeave$_{\mathbf{UB}}$**	2485.420	766.534	25.01	3396.6
	ReedWeave$_{\mathbf{JB}}$	2862.646	738.000	21.70	2896.2

D.2 Rate 1/4

We next evaluate initial code rate 1/4 under the same single-thread setup as in Section D.1. Brakedown retains its native code rate of approximately 0.657, so its measurements are unchanged. Each ReedWeave variant uses the same protocol parameters as in the 32-thread evaluation. Figure 5 and Table 5 report the results.

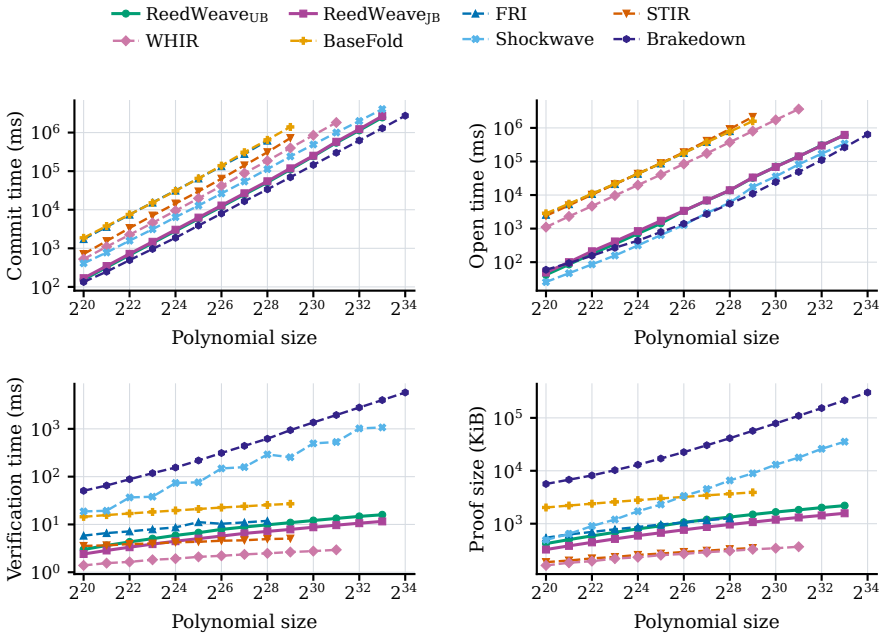


Fig. 5. Single-thread PCS performance over Goldilocks at initial code rate 1/4 and 100-bit security. Brakedown retains its native code rate of approximately 0.657. Points are averages of five runs. A curve ends where a scheme runs out of memory or its FFT domain exceeds 2^{32}. Vertical axes are logarithmic. Lower is better.

Comparison with RS-based PCSs. Both ReedWeave variants achieve lower commitment and opening times than the benchmarked FRI, STIR, and WHIR implementations across all measured polynomial sizes. At $d = 2^{24}$, ReedWeave$_{\mathsf{JB}}$ takes 3.873 s in total prover time, compared with 29.19 s for WHIR, 57.48 s for STIR, and 74.29 s for FRI, and also improves over FRI in both proof size and verification time. STIR and WHIR produce smaller proofs of 256.9 and 232.2 KiB and verify in 4.23 and 1.92 ms, while ReedWeave$_{\mathsf{JB}}$ is 3.1× faster than WHIR in commitment and 23.5× faster in opening. At $d = 2^{31}$, where FRI and STIR exceed the 2^{32} two-adicity limit, ReedWeave$_{\mathsf{JB}}$ achieves a 7.6× total-prover speedup over WHIR.

Shockwave opens faster at $d = 2^{24}$, but its commitment cost brings total prover time to 6.730 s, while its proof is 2.89× larger and verification is 16.5× slower than in ReedWeave$_{\mathsf{JB}}$. By $d = 2^{31}$, ReedWeave$_{\mathsf{JB}}$ produces a 13.6× smaller proof and verifies 54.9× faster than Shockwave, while also taking less total prover time.

Comparison with field-agnostic PCSs. ReedWeave$_{\mathsf{JB}}$ outperforms BaseFold in total prover time, verification time, and proof size across all measured polynomial sizes, achieving a 19.1× total-prover speedup at $d = 2^{24}$. Unlike at rate 1/2, Brakedown achieves lower total prover time than both ReedWeave variants across all measured polynomial sizes, at the cost of a 21.8× larger proof and 34.8× slower verification at $d = 2^{24}$.

Comparison of two ReedWeave variants. Across all measured polynomial sizes, ReedWeave$_{\mathsf{JB}}$ reduces proof size by 22.8-28.7% relative to ReedWeave$_{\mathsf{UB}}$, from 25.5% at $d = 2^{24}$ to 28.4% at $d = 2^{33}$, with a 5.4-9.4% prover-time overhead. At $d = 2^{33}$, our constructions require ∼400 GiB of RAM, whereas FRI, STIR, and WHIR exceed the 2^{32} two-adicity limit and BaseFold requires over 6 TiB.

Table 5. Concrete single-thread costs under the setup of Figure 5. WHIR proves a multilinear evaluation, whereas the other schemes prove a univariate evaluation. Shaded rows correspond to our constructions. A dash indicates an unavailable configuration. At $d = 2^{31}$, FRI and STIR exceed the 2^{32} two-adicity limit of Goldilocks and BaseFold requires ∼1.6 TiB of RAM. At $d = 2^{33}$, the largest size at rate 1/4, we report only our constructions, which use ∼400 GiB of RAM, whereas FRI, STIR, and WHIR exceed the 2^{32} two-adicity limit and BaseFold requires >6 TiB of RAM. † Brakedown uses its native code rate of approximately 0.657 rather than 1/4.

	PCS	Commit time (s)	Open time (s)	Verify time (ms)	Proof size (KiB)
$d = 2^{24}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	2.830	0.710	5.90	798.0
	ReedWeave$_{\mathbf{JB}}$	3.033	0.840	4.46	594.7
	FRI	31.346	42.947	8.67	866.4
	STIR	14.476	43.008	4.23	256.9
	WHIR	9.425	19.761	1.92	232.2
	Shockwave	6.414	0.316	73.75	1716.6
	BaseFold	30.770	43.136	19.61	2784.5
	Brakedown†	1.874	0.438	155.15	12990.2
$d = 2^{28}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	51.246	13.754	9.82	1335.2
	ReedWeave$_{\mathbf{JB}}$	54.670	13.872	7.16	964.1
	FRI	618.227	801.947	11.87	1262.8
	STIR	309.436	921.592	4.95	329.8
	WHIR	185.371	374.098	2.49	305.9
	Shockwave	112.952	5.893	291.14	6576.2
	BaseFold	646.716	771.151	25.56	3666.0
	Brakedown†	33.807	5.519	620.63	41256.3
$d = 2^{31}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	545.159	141.287	13.35	1824.9
	ReedWeave$_{\mathbf{JB}}$	577.640	141.256	9.66	1311.1
	FRI	–	–	–	–
	STIR	–	–	–	–
	WHIR	1822.264	3634.616	2.94	366.6
	Shockwave	999.944	81.919	530.24	17782.0
	BaseFold	–	–	–	–
	Brakedown†	299.973	48.761	1944.89	109544.9
$d = 2^{33}$ rate = 1/4	**ReedWeave$_{\mathbf{UB}}$**	2464.395	613.700	15.93	2195.9
	ReedWeave$_{\mathbf{JB}}$	2675.456	619.663	11.58	1573.2