LiftWHIR: A Prover-Efficient Polynomial Commitment with Short Proofs

Zhongliang Zhang[1,2] Xinxuan Zhang[1,2] Yuanju Wei[1,2]
Lang Qin[1,2] Yi Deng[3]

[1]*State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences*
[2]*School of Cyber Security, University of Chinese Academy of Sciences*
[3]*School of Cryptology, Xidian University, Xi'an, China*

{zhangzhongliang,zhangxinxuan,weiyuanju,qinlang}@iie.ac.cn
ydeng.cas@gmail.com

Abstract

Polynomial commitment schemes allow a prover to commit to a large polynomial and later prove a claimed evaluation at a chosen point. They are a core component of many efficient SNARKs, and the cost of their evaluation phase directly affects SNARK prover time. Reed–Solomon-based schemes already offer small proofs and fast verification, but generating an evaluation proof for a large polynomial remains expensive.

We present LiftWHIR, a Reed–Solomon-based polynomial commitment scheme that reduces prover time in the evaluation phase. LiftWHIR combines interleaved coding with the DEEP(ITCS'20) technique to reduce proving an evaluation of a large polynomial to two smaller tasks: a proximity test on a shorter codeword and evaluation of a smaller polynomial. The use of DEEP simultaneously reduces the number of queries required by the proximity test. We then use WHIR(EUROCRYPT'25) to prove both resulting tasks, keeping verification and communication costs low. LiftWHIR trades a modest increase in proof size and verifier time for a substantial reduction in prover time.

At $n = 2^{20}$ over a 255-bit prime field and code rate $1/2$ (resp., $1/4$), LiftWHIR reduces the evaluation phase to 167 ms (resp., 169 ms), yielding a $4.4\times$ (resp., $6.7\times$) speedup over WHIR. Including commitment, LiftWHIR achieves total prover times of 808 ms (resp., 1,474 ms), corresponding to overall prover speedups of $1.61\times$ (resp., $1.55\times$). Verification time increases from 0.55 ms to 0.76 ms (resp., 0.40 ms to 0.51 ms), while proof size is $1.46\times$ (resp., $1.33\times$) that of WHIR. We further instantiate Spartan(CRYPTO'20) with LiftWHIR and compare it with a Spartan variant instantiated with WHIR. LiftWHIR speeds up proving by $1.9\times$, while verification time increases only from 3.86ms to 4.62ms, at the cost of a 42% increase in proof size.

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) [11, 25, 30, 34] allow a prover to produce a short proof that a computation is correct and a verifier to check the proof at much lower cost than re-executing the computation. In recent years, SNARKs have seen widespread use in blockchain applications [8, 13], verifiable computation [38], and privacy-preserving computation [32]. As these applications process larger computations, the practical efficiency of SNARKs becomes increasingly important. In particular, prover time, verifier time, and proof size are three key metrics that determine the practical utility of a SNARK. Many modern SNARKs [16, 18, 19, 24, 40] achieve succinctness by combining polynomial interactive oracle proofs (PIOPs) [9] with polynomial commitment schemes (PCSs) [29, 37]. This makes the PCS a key component of modern SNARK constructions, and the efficiency and communication cost of the underlying PCS largely determine the overall performance of the resulting SNARK.

A PCS allows a prover to commit to a polynomial f with a short commitment. Given a point x, the prover can later prove an evaluation claim $f(x) = y$ without sending the polynomial itself. The prover's main work in a PCS typically falls into two phases. During the *commitment phase*, the prover commits to the polynomial; during the *evaluation phase*, the prover generates a proof for a given evaluation claim. We call the combined cost of these phases the PCS *prover time*. An efficient PCS needs to provide small proofs and efficient verification while minimizing prover costs *in both phases*.

Polynomial commitment schemes can be constructed using different techniques. Pairing-based PCSs such as KZG [29] rely on bilinear groups and require a trusted setup to generate a structured reference string (SRS). By contrast, code-based PCSs use error-correcting codes and hash functions [1, 42] to provide a transparent alternative. In these PCSs, the choice of encoding technique directly affects prover time, verifier time, and proof size [14, 26, 42]. Brakedown [26] uses a linear-time encodable code and achieves a fast prover, but it typically requires larger proofs and higher verification costs. By contrast, PCSs built from Reed–Solomon proximity tests can achieve lower verification complexity and more compact proofs, making this line of work an important approach to

building transparent PCSs. FRI [6] and its DEEP-FRI extension [10] form the foundation of this line of work. Subsequent work follows two related directions: BaseFold [42] and DeepFold [27] construct multilinear PCSs, while STIR [2] reduces the query complexity of Reed–Solomon proximity testing. WHIR [3] extends STIR's low-query approach to constrained Reed–Solomon codes that support multilinear polynomial evaluations. Compared with BaseFold and DeepFold, WHIR requires fewer queries, thereby reducing proof size and verification cost. These improvements, however, require additional prover computation.

This additional prover computation occurs primarily during the evaluation phase. The distinction between one-time preprocessing and per-instance evaluation is important in practice. Many practical SNARKs separate preprocessing from online proving [19, 38]. When the same model or circuit is used for many inputs, the prover can complete one-time work, such as generating the proving key, before the inputs arrive and reuse the resulting data across proofs. Each new instance, however, still requires a fresh proof once its concrete input is known. A PCS evaluation proof depends on the evaluation claim for that instance [29, 37]; it must therefore be generated anew and cannot be amortized by preprocessing. Its cost recurs on the critical path of every online proof, directly affecting proving latency and system throughput.

This distinction is especially relevant to computation-intensive applications. In zkML [17], for example, the prover can commit to a fixed model during preprocessing and reuse the commitment across many inferences, but must still generate a fresh proof online for each new input. Similar repeated-instance workloads arise in verifiable cloud computing [38] and verifiable FHE [31], where provers must produce proofs for many computations. As computation sizes and the number of instances grow, reducing WHIR's evaluation cost removes work that would otherwise recur in every online proof. This motivates the central question of this work:

> *Can we substantially reduce WHIR's prover time, especially in the evaluation phase, while largely preserving its low verification time and small proof size?*

1.1 Our Results

We answer this question affirmatively. Our contributions can be summarized as follows:

LiftWHIR. We present LiftWHIR, a Reed–Solomon-based polynomial commitment scheme that reduces WHIR's evaluation time while retaining its favorable verification time and proof size. Furthermore, it supports flexible parameter choices to trade off evaluation time, verification time, and proof size.

LiftWHIR is built around a new lifting reduction that we propose. Combining the interleaved code with the DEEP technique, the lifting reduction transforms the evaluation proof into two lighter tasks: a proximity proof for an implicit Reed–Solomon code—expressed as a linear combination of the interleaved code—and two point-evaluation proofs for the polynomial encoded by that Reed–Solomon code. Crucially, the encoded polynomial is much smaller than the original one, and the DEEP technique ensures that we only need to prove Reed–Solomon proximity with the distance-preserving bound. By invoking WHIR to prove this Reed–Solomon proximity and the small-scale evaluations, we obtain LiftWHIR. Since the reduction shrinks the encoded polynomial, we significantly reduce the running time of WHIR's evaluation protocol, while only modestly increasing the proof size and verification time.

Directly sending the linear combination of the interleaved code also yields a simple standalone PCS. This lifting-only variant further reduces prover work at the cost of larger proofs and higher verification overhead, though its proof size and verification overhead remain substantially smaller than those of Brakedown and Shockwave.

Performance and applications. We mainly compare the efficiency of our scheme with WHIR. Since WHIR supports a flexible choice of the initial folding parameter f, we use WHIR's default setting $f = 4$ as the proof-size baseline: under proof-size caps of $1.25\times$ and $1.5\times$ this baseline, we select the optimal f for WHIR and the optimal parameters for LiftWHIR, respectively, and compare their evaluation times. The results are reported in Table 1. Under the $1.25\times$ cap, LiftWHIR achieves $2.26\times$ and $4.11\times$ evaluation-time speedups at code rates $1/2$ and $1/4$, respectively. Under the $1.5\times$ cap, the corresponding speedups increase to $4.16\times$ and $9.29\times$. Detailed comparisons with WHIR and other PCSs are deferred to Section 5.

To demonstrate the practical impact of LiftWHIR, we integrate it into Spartan, a widely used zkSNARK system built on PCS. Under the same instance and machine settings, replacing WHIR with LiftWHIR reduces Spartan's online prover time from 50.7 seconds to 26.8 seconds, a $1.89\times$ speedup. In this Spartan instantiation, the reduction in PCS evaluation time therefore translates directly into lower online proving time, confirming that LiftWHIR's gains carry over to end-to-end proof generation in real systems.

1.2 Technique Overview

Reed–Solomon lifting reduces WHIR's initial recursion cost. WHIR uses rate-improving folding to reduce the query complexity of Reed–Solomon proximity testing [3]. After each folding round, WHIR encodes the reduced polynomial using a Reed–Solomon code of lower rate. The lower rate increases the code's relative distance, so the verifier needs fewer queries and openings and the proof becomes smaller. However, these benefits add prover work. After each fold, the

Code rate	Proof-size cap	Best WHIR evaluation	Best LiftWHIR evaluation	LiftWHIR speedup
1/2	1.25× (205.4 KiB)	725.6 ms	321.0 ms	2.26×
1/2	1.50× (246.4 KiB)	693.0 ms	166.7 ms	4.16×
1/4	1.25× (139.3 KiB)	1134.3 ms	275.9 ms	4.11×
1/4	1.50× (167.2 KiB)	1134.3 ms	122.1 ms	9.29×

Table 1: Evaluation time under proof-size caps of 1.25× and 1.5× the default WHIR proof size. For each code rate and cap, we report the fastest measured configuration of each scheme; speedups are relative to WHIR.

prover performs a new Reed–Solomon encoding and creates a new oracle for the next recursive round. And an additional sumcheck is introduced to ensure the consistence of these encodings.

Our key insight is to replace WHIR's first reduction on the largest evaluation instance with a different reduction. This initial reduction produces a smaller instance before running WHIR. WHIR therefore begins its recursion on the smaller instance and avoids folding the largest object.

Blaze's lifting technique motivates this approach [14]. Blaze uses interleaving to reduce a large multilinear polynomial evaluation to a smaller evaluation problem and an IOPP for the code. Its lifting uses an RAA code, primarily to support efficient encoding. The resulting object does not directly match the Reed–Solomon structure that WHIR expects, so composing the two requires a conversion between code structures together with additional commitments and openings.

LiftWHIR instead uses Reed–Solomon lifting so that the reduced instance composes directly with WHIR. We arrange the vector representation of f as a matrix $U \in \mathbb{F}^{m\times k}$ and write U_i for row i. For an evaluation at z, let $q_1 \in \mathbb{F}^m$ and $q_2 \in \mathbb{F}^k$ be such that

$$f(z) = q_1^\top U q_2.$$

The prover computes

$$v = U q_2, \qquad v_i = \langle U_i, q_2 \rangle.$$

Each v_i is an evaluation of the smaller polynomial represented by U_i, and the original evaluation is $f(z) = q_1^\top v$.

After v is fixed, the verifier samples $r = (r_1, \ldots, r_m)$, which determines

$$U_r = \sum_i r_i U_i, \qquad \langle U_r, q_2 \rangle = \langle r, v \rangle.$$

The reduction therefore leaves two tasks: proving this relation and testing the Reed–Solomon encoding of U_r for proximity to the code.

The linearity of Reed–Solomon encoding supplies the folded codeword directly from the row encodings. Writing $\widehat{U}_i = \mathsf{Enc}_{\mathsf{row}}(U_i)$, we have

$$\widehat{U}_r = \sum_i r_i \widehat{U}_i = \mathsf{Enc}_{\mathsf{row}}(U_r).$$

The row encodings thus determine the Reed–Solomon encoding of U_r. LiftWHIR reuses this codeword without creating another commitment or converting between code structures.

Because the outer commitment and r determine $\widehat{U}_r$, LiftWHIR uses $\widehat{U}_r$ as a virtual commitment to U_r when running WHIR. When WHIR queries a coordinate j, the prover opens the corresponding column of the outer oracle, and the verifier computes

$$\widehat{U}_{r,j} = \sum_i r_i \widehat{U}_{i,j}.$$

WHIR then verifies both the Reed–Solomon proximity of $\widehat{U}_r$ and the relation $\langle U_r, q_2 \rangle = \langle r, v \rangle$, without constructing another commitment.

Using DEEP to Reduce First-Round Queries in LiftWHIR. Although the lifting reduction described above reduces the size of the (implicit) Reed–Solomon code that WHIR needs to prove—thereby lowering evaluation time—it substantially increases communication and verification time. The reason is that, as currently constructed, the reduction requires WHIR to prove that the word is very close to the Reed–Solomon code, that is, within a very small proximity parameter, and WHIR's communication and verification overhead grows as this parameter shrinks.

The small proximity parameter stems from how the lifting reduction converts the well-formedness check of the commitment—namely that of the Reed–Solomon codes before folding—into a proximity claim on the folded code. Soundness of this fold relies on correlated agreement for Reed–Solomon codes [7]: roughly, for a random folding, if the folded word is close to a valid Reed–Solomon code, then so are the original codes before folding.

Although correlated agreement supports a considerably larger distance-preserving bound,our lifting reduction can only rely on the unique-decoding radius. This is because we must ensure that every word of the interleaved code has at most one valid Reed–Solomon codeword within that distance, which is necessary to guarantee the binding property of the commitment. In this case, we must substantially increase the number of queries in WHIR's first round to reach the target soundness error.

Beyond the unique-decoding radius, a received word can be close to several low-degree polynomials, so the analysis can

no longer rely on a unique nearby polynomial. LiftWHIR addresses this issue by applying the DEEP technique [10, 27] to the correlated-agreement analysis. In addition to the ordinary evaluation claim, the protocol introduces a polynomial evaluation at a random out-of-domain point α. This value adds a constraint on the candidate polynomials and distinguishes these candidates with high probability in the list-decoding range. The resulting analysis therefore extends beyond the unique-decoding radius.

This mechanism fits directly into Reed–Solomon lifting. For the target evaluation, the prover computes $v = Uq_2$. For the DEEP check, let P_i be the degree-$< k$ univariate polynomial whose coefficient vector is U_i, and let

$$c_i = P_i(\alpha), \qquad c = (c_1, \ldots, c_m).$$

Using the same random coefficients $r = (r_i)_i$ gives

$$\langle U_r, q_2 \rangle = \langle r, v \rangle, \qquad P_r(\alpha) = \langle r, c \rangle,$$

where $U_r = \sum_i r_i U_i$ and $P_r = \sum_i r_i P_i$. Both equations are linear in U_r. The protocol combines them into one constraint and proves the resulting constrained Reed–Solomon instance with WHIR.

The DEEP check therefore uses the same random linear combination as the target evaluation and requires no additional folding step. Its out-of-domain constraint yields a stronger correlated-agreement bound, so LiftWHIR needs fewer queries in WHIR's first round for the same soundness target. This retains the prover savings from Reed–Solomon lifting while reducing the proof-size overhead from first-round oracle openings.

1.3 Related Work

FRI and code folding. FRI introduced recursive folding for transparent Reed–Solomon proximity testing [6]. DEEP-FRI improves its soundness using out-of-domain evaluations, allowing proximity testing with fewer queries [10]. More recent work further reduces the query complexity of Reed–Solomon proximity testing. STIR progressively lowers the rate of the Reed–Solomon codes across rounds to reduce the number of queries required for proximity testing [2]. WHIR extends this idea to constrained Reed–Solomon codes, supporting both proximity testing and polynomial evaluation claims [3]. FRI-style folding has also been used to construct multilinear polynomial commitments. BaseFold combines code folding with Sumcheck to obtain a multilinear PCS for foldable linear codes [42]. DeepFold develops the Reed–Solomon setting further by applying the DEEP technique in the list-decoding regime, reducing the number of queries while retaining efficient proving [27]. The soundness of such folding protocols has also been studied through proximity-gap and correlated-agreement techniques [7, 12].

Interleaved codes. Several code-based proof systems use interleaving to organize committed data as a matrix and apply a base code along one dimension. Ligero introduced this approach in succinct proof systems [1]. Brakedown combines interleaving with linear-time encodable codes, obtaining linear prover time at the cost of larger proofs [26]. Orion instead uses tensor codes together with proof composition to obtain polylogarithmic proof size, albeit with a high query count [41]. More recent work integrates interleaved-code proximity testing more directly with polynomial evaluation proofs. When the evaluation point is sampled uniformly after the commitment, the proximity and evaluation checks can share the same random linear combination of the rows [22]. Ligerito supports arbitrary evaluation points through a partial Sumcheck reduction [36].

Code switching. Code switching separates a fast-encodable source code from a code that supports succinct verification [39]. Blaze and BrakingBase instantiate this paradigm using different combinations of fast-encodable codes and BaseFold-style multilinear evaluation protocols, achieving linear prover time with polylogarithmic verifier complexity and proof size [14, 35]. Subsequent work further reduces the query complexity of this approach, first to $O(\lambda \log\log n + \log n)$ in FICS and later to the optimal $O(\lambda)$ for linear-time encodable codes [4, 5]. TensorSwitch commits using tensor codes and recursively reduces evaluation claims to instances over smaller tensor codes [15]. SwitchFold instead performs recursive switching within a single code family, avoiding an auxiliary foldable code; its BrakeFold instantiation uses the Brakedown code sequence and achieves linear prover time with polylogarithmic proof size and verifier time [20].

2 Preliminaries

2.1 Notation

Let $\lambda \in \mathbb{N}$ be the security parameter, and let $\mathsf{negl}(\lambda)$ denote a negligible function. Throughout, $\mathbb{F}$ denotes a finite field, and PPT abbreviates probabilistic polynomial time. We write $[t] = \{1, \ldots, t\}$ and use $x \xleftarrow{\$} S$ for a uniformly random element of a finite set S. All vectors are column vectors unless stated otherwise. For $a, b \in \mathbb{F}^t$, their inner product is $\langle a, b \rangle = a^\top b$.

Let $D = (\gamma_1, \ldots, \gamma_N)$ be an ordered set and let $f, g \colon D \to \mathbb{F}$. Their relative Hamming distance is

$$\Delta_D(f, g) = \frac{1}{N} \left| \{ i \in [N] : f(\gamma_i) \neq g(\gamma_i) \} \right|.$$

We identify a function on D with its value vector in $\mathbb{F}^N$. For a code $\mathcal{C} \subseteq \mathbb{F}^N$, define $\Delta_D(f, \mathcal{C}) = \min_{c \in \mathcal{C}} \Delta_D(f, c)$.

2.2 Multilinear Extensions

A polynomial is multilinear if each variable has degree at most one. We write $\mathbb{F}_{\le 1}[X_1,\dots,X_\mu]$ for the set of μ-variate multilinear polynomials over $\mathbb{F}$.

Lemma 2.1 (Multilinear extension)**.** *For every function $f\colon \{0,1\}^\mu \to \mathbb{F}$, there is a unique polynomial $\widetilde{f} \in \mathbb{F}_{\le 1}[X_1,\dots,X_\mu]$ satisfying $\widetilde{f}(b) = f(b)$ for every $b \in \{0,1\}^\mu$. This polynomial is the multilinear extension of f, and*

$$\widetilde{f}(X) = \sum_{b\in\{0,1\}^\mu} f(b)\,\widetilde{\mathsf{eq}}(b,X),$$

$$\text{where}\quad \widetilde{\mathsf{eq}}(b,X) = \prod_{j=1}^{\mu}\bigl(b_jX_j + (1-b_j)(1-X_j)\bigr).$$

2.3 Reed–Solomon Codes

Let $L = (\alpha_1,\dots,\alpha_N)$ be an ordered tuple of distinct elements of a finite field $\mathbb{F}$ of size q, and let $d \le N$. For $P \in \mathbb{F}[X]$, write $P|_L = (P(\alpha_1),\dots,P(\alpha_N))$. The Reed–Solomon code of degree bound d over L is

$$\mathsf{RS}_{L,d} = \{P|_L : P \in \mathbb{F}[X],\ \deg P < d\} \subseteq \mathbb{F}^N.$$

Its rate is $\rho = d/N$, and its relative minimum distance is $(N-d+1)/N = 1-\rho+1/N$.

Distance and list decoding. For $f \in \mathbb{F}^N$, define

$$\Delta_L(f,\mathsf{RS}_{L,d}) = \min_{P\in\mathbb{F}[X],\,\deg P<d} \Delta_L(f,P|_L).$$

For a radius $\delta \in [0,1]$, the corresponding list of nearby polynomials is

$$\mathcal{L}_\delta(f) = \{P \in \mathbb{F}[X] : \deg P < d \text{ and } \Delta_L(f,P|_L) \le \delta\}.$$

$\mathsf{RS}_{L,d}$ is (δ,B)-list decodable if $|\mathcal{L}_\delta(f)| \le B$ for every $f \in \mathbb{F}^N$. For every $\eta \in (0, 1-\sqrt{\rho})$, the code is $(1-\sqrt{\rho}-\eta, 1/(2\eta\sqrt{\rho}))$-list decodable by the Johnson bound [28].

Correlated and mutual correlated agreement. We recall correlated agreement [7] and mutual correlated agreement [3, 12].

Definition 2.2 (Correlated and mutual correlated agreement)**.** Let $\mathcal{C} \subseteq \mathbb{F}^N$ be a linear code, let $G\colon \mathcal{X} \to \mathbb{F}^s$, and let a seed x be sampled uniformly from $\mathcal{X}$. For $f_1,\dots,f_s \in \mathbb{F}^N$, write $f_x = \sum_{j=1}^s G(x)_j f_j$. For $S \subseteq [N]$, write $\mathcal{C}|_S = \{c|_S : c \in \mathcal{C}\}$.

Correlated agreement. The generator G is a proximity generator for $\mathcal{C}$ with proximity bound B and error $\varepsilon_{\mathsf{CA}}$ if, for every $f_1,\dots,f_s \in \mathbb{F}^N$ and $\delta \in (0, 1-B)$, the condition

$$\Pr_x[\Delta_L(f_x,\mathcal{C}) \le \delta] > \varepsilon_{\mathsf{CA}}(\delta)$$

implies that there exist $c_1,\dots,c_s \in \mathcal{C}$ and $T \subseteq [N]$ such that $|T| \ge (1-\delta)N$ and $f_j|_T = c_j|_T$ for every $j \in [s]$.

Mutual correlated agreement. The generator G is a proximity generator for $\mathcal{C}$ with mutual correlated agreement, proximity bound $B^\star$, and error $\varepsilon_{\mathsf{MCA}}$ if, for every $f_1,\dots,f_s \in \mathbb{F}^N$ and $\delta \in (0, 1-B^\star)$, with probability at least $1-\varepsilon_{\mathsf{MCA}}(\delta)$ over x, every $S \subseteq [N]$ with $|S| \ge (1-\delta)N$ satisfies

$$f_x|_S \in \mathcal{C}|_S \quad\Longrightarrow\quad f_j|_S \in \mathcal{C}|_S \quad \text{for every } j \in [s].$$

Thus, mutual correlated agreement requires the original words to agree with the code on the same set as their random linear combination.

Theorem 2.3 (Mutual correlated agreement for Reed—Solomon codes [12, Theorem 5])**.** *Let $G\colon \mathbb{F}^s \to \mathbb{F}^s$ be given by $G(r) = r$. For every integer $t \ge 3$ and*

$$0 < \delta \le 1 - \left(1 + \frac{1}{2t}\right)\sqrt{\rho},$$

G has mutual correlated agreement for $\mathsf{RS}_{L,d}$ with error

$$\varepsilon_{\mathsf{MCA}}(\delta) = O\left(\frac{t^7N^2s}{\rho^{3/2}q}\right).$$

In particular, this gives mutual correlated agreement at every positive radius $\delta = 1-\sqrt{\rho}-\eta$ with fixed $\eta \in (0, 1-\sqrt{\rho})$.

For prime fields, the capacity conjecture of Crites and Stewart [21, Conjecture 3] specializes to the affine-line generator. The affine-line-to-affine-space reduction and sub-generator closure of Bordage et al. [12, Lemma 7.1 and Corollary 4.2] give the identity-generator form below. The reduction is independent of the output dimension and increases the error by at most a factor $(1-1/p)^{-1}$. For $0 \le x \le 1-1/p$, define the p-ary entropy H_p and write $C_p = 1-H_p$:

$$H_p(x) = x\log_p(p-1) - x\log_p x - (1-x)\log_p(1-x),$$
$$C_p(x) = 1 - H_p(x),$$

with $0\log_p 0 = 0$.

Conjecture 2.4 (Mutual correlated agreement over prime fields)**.** There exist absolute constants $c_1,c_2 > 0$ with the following property. For every $\eta > 0$, prime p, and $0 < \delta < 1-1/p$ satisfying

$$\rho + \frac{1}{N} + \eta < C_p(\delta),$$

let $\mathbb{F} = \mathbb{F}_p$ and let $\mathsf{RS}_{L,d}$ have a smooth evaluation domain in the sense of [21]. For every $s \ge 1$, the identity generator

$$G_{\mathsf{id}}\colon \mathbb{F}^s \longrightarrow \mathbb{F}^s, \qquad G_{\mathsf{id}}(r) = r,$$

has mutual correlated agreement for $\mathsf{RS}_{L,d}$ at radius δ with error at most

$$\varepsilon_{\mathsf{cap}} := \frac{d^{c_2}}{(1-1/p)\,p\,\eta^{c_1}\rho^{c_1+c_2}}.$$

Out-of-domain evaluations. The DEEP technique evaluates at a random point outside the code domain to distinguish candidate low-degree polynomials [10]. For a list of size at most B, a root-counting argument gives the following bound.

Lemma 2.5. *Let $f \in \mathbb{F}^N$ and suppose $|\mathcal{L}_\delta(f)| \le B$. For a uniformly random α in $\mathbb{F} \setminus L$,*

$$\Pr[\exists P \neq Q \in \mathcal{L}_\delta(f) : P(\alpha) = Q(\alpha)] \le \binom{B}{2} \frac{d-1}{q-N}.$$

Proof. For each distinct pair $P, Q \in \mathcal{L}_\delta(f)$, the nonzero polynomial $P - Q$ has degree at most $d-1$ and therefore at most $d-1$ roots in $\mathbb{F} \setminus L$. A union bound over the at most $\binom{B}{2}$ pairs proves the claim. □

2.4 Interactive Oracle Proofs and Polynomial Commitments

For a relation $\mathcal{R}$, let $\mathcal{L}(\mathcal{R}) = \{x : \exists w \ (x,w) \in \mathcal{R}\}$.

Interactive oracle proofs. A t-round public-coin IOP [9] $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ proceeds as follows. The prover first sends an oracle string π_0. In each round $i \in [t]$, the verifier samples a random challenge α_i, and the prover replies with an oracle string π_i. After the last round, the verifier queries entries of $\pi_0, \ldots, \pi_t$ and outputs a bit.

Definition 2.6 (IOPs)**.** Let $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$ be a t-round public-coin IOP for $\mathcal{R}$. It is complete if, for every $(x,w) \in \mathcal{R}$, the following probability equals 1:

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

It has soundness error ν if, for every $x \notin \mathcal{L}(\mathcal{R})$ and every unbounded adversary $\mathcal{A}$, the following probability is at most ν:

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

IOPs of proximity. An IOP of proximity (IOPP) additionally gives the verifier oracle access to the purported witness w. The verifier queries only a sublinear number of its entries. The soundness states that if w is far from any valid witness, then the verifier rejects with high probability.

Definition 2.7 (IOPPs)**.** A public-coin IOP $\mathsf{IOP} = (\mathcal{P}, \mathcal{V})$ for $\mathcal{R}$ is an IOPP if it is complete and has $\nu(\cdot)$-IOPP soundness for some function ν. For every (x,w) such that w is δ-far in relative Hamming distance from every w' satisfying $(x,w') \in \mathcal{R}$, and for every unbounded adversary $\mathcal{A}$, the following probability is at most $\nu(\delta)$:

$$\Pr_{\alpha_1,\ldots,\alpha_t}\left[\mathcal{V}^{w,\pi_0,\ldots,\pi_t}(x;\alpha_1,\ldots,\alpha_t) = 1 \;\middle|\; \begin{array}{l} \pi_0 \leftarrow \mathcal{A}(x,w), \\ \pi_1 \leftarrow \mathcal{A}(x,w,\alpha_1), \\ \cdots \\ \pi_t \leftarrow \mathcal{A}(x,w,\alpha_1,\ldots,\alpha_t) \end{array}\right]$$

Polynomial commitment schemes.

Definition 2.8 (Polynomial commitment scheme)**.** A multilinear polynomial commitment scheme over $\mathbb{F}$ consists of $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$:

- $\mathsf{Setup}(1^\lambda, \mu) \to \mathsf{pp}$ takes the security parameter and the number of variables and outputs public parameters.
- $\mathsf{Commit}(\mathsf{pp}, f) \to C$ takes $f \in \mathbb{F}_{\le 1}[X_1, \ldots, X_\mu]$ and outputs a commitment.
- $\mathsf{Open}(\mathsf{pp}, C, f) \to b$ 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, an evaluation point $z \in \mathbb{F}^\mu$, and a value $y \in \mathbb{F}$. The prover additionally knows f and convinces the verifier that f opens C and $f(z) = y$. The verifier outputs a bit at the end of the protocol.

The scheme is complete if, for every $\mu \in \mathbb{N}$, $f \in \mathbb{F}_{\le 1}[X_1, \ldots, X_\mu]$, and $z \in \mathbb{F}^\mu$,

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

It is binding if, for every $\mu \in \mathbb{N}$ and every PPT adversary $\mathcal{A}$, the following probability is negligible in λ:

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

For fixed pp, define

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

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

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

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

3 LiftWHIR

LiftWHIR first reduces the original multilinear evaluation claim to a smaller instance and then invokes WHIR on the reduced instance. The prover encodes the rows of the vector representation of the polynomial using a common Reed–Solomon code and commits to the resulting matrix. After the verifier samples a random linear-combination challenge r, the outer commitment together with r determines a virtual Reed–Solomon oracle for the folded vector U_r, without requiring a separate encoding or commitment. The construction also records an out-of-domain evaluation of the polynomial associated with each row, which will later be used in the correlated-agreement analysis.

Review of WHIR. We recall the form of the constraints supported by WHIR [3]. Let $u = (u_b)_{b\in\{0,1\}^c}$ be a vector indexed by the Boolean hypercube. WHIR represents a constraint on u using a weight polynomial $w(Z,X)$ and a claimed value σ, with the constraint

$$\sum_{b\in\{0,1\}^c} w(u_b, b) = \sigma.$$

Here, the first argument of w is instantiated with the entry u_b, while the remaining arguments identify the hypercube point b. LiftWHIR only uses constraints that are linear in u. In this case, for a polynomial $\omega(X)$, the weight polynomial takes the form $w(Z,X) = Z\cdot\omega(X)$, and the constraint becomes

$$\sum_{b\in\{0,1\}^c} u_b\cdot\omega(b) = \big\langle u, (\omega(b))_{b\in\{0,1\}^c}\big\rangle = \sigma.$$

We therefore write the WHIR constraints used below directly as inner-product claims. WHIR can also batch several such constraints on the same polynomial using a random linear combination. Alongside the linear claim, WHIR tests whether its initial oracle is close to the corresponding Reed–Solomon code.

Matrix representation and row encoding. Let $n = 2^\mu = mk$, where $m = 2^a$, $k = 2^c$, and $a + c = \mu$. Let $u \in \mathbb{F}^n$ be the chosen tensor-product representation of the committed multilinear polynomial f. We arrange u row-wise as a matrix $U \in \mathbb{F}^{m\times k}$, write $U_i \in \mathbb{F}^k$ for its i-th row, and have $u = \mathrm{vec}_{\mathrm{row}}(U)$.

For a target point $\mathbf{z} = (z_1,\dots,z_\mu) \in \mathbb{F}^\mu$, let $\mathbf{z}_{\mathsf{col}} = (z_1,\dots,z_c) \in \mathbb{F}^c$ and $\mathbf{z}_{\mathsf{row}} = (z_{c+1},\dots,z_\mu) \in \mathbb{F}^a$. We use row-major ordering: the high-variable block indexes the rows, the low-variable block indexes the columns, and all entries of each row are consecutive in u. Let $q \in \mathbb{F}^n$ be the evaluation vector induced by the chosen representation of f, so that $f(\mathbf{z}) = \langle q, u\rangle$. The tensor-product structure gives

$$q = q_1 \otimes q_2, \qquad q_1 \in \mathbb{F}^m, \quad q_2 \in \mathbb{F}^k.$$

Here q_1 and q_2 are induced by $\mathbf{z}_{\mathsf{row}}$ and $\mathbf{z}_{\mathsf{col}}$, respectively. Hence

$$f(\mathbf{z}) = \langle q_1 \otimes q_2, u\rangle = q_1^\top U q_2.$$

Let $L = (\ell_1,\dots,\ell_N)$ be the evaluation domain of the row Reed–Solomon code, and let $\mathsf{Enc}_{\mathsf{row}} : \mathbb{F}^k \to \mathbb{F}^N$ denote its linear encoding map. LiftWHIR applies this encoder to each row, giving $\widehat{U}_i := \mathsf{Enc}_{\mathsf{row}}(U_i)$. The encoded rows form a vector-valued outer oracle $\widehat{U} \in \mathbb{F}^{m\times N}$ indexed by column position. The prover commits to this oracle, and an opening at position $j \in [N]$ reveals the column $\widehat{U}_{1,j},\dots,\widehat{U}_{m,j}$.

DEEP evaluations. We interpret each row U_i as the coefficient vector of a degree-$< k$ polynomial $P_i \in \mathbb{F}[X]$. After the prover commits to the outer oracle, the verifier samples $\alpha \in \mathbb{F}\setminus L$, and the prover provides $c_i := P_i(\alpha)$ and $c = (c_1,\dots,c_m)$. Let $d_\alpha := (1,\alpha,\dots,\alpha^{k-1})^\top \in \mathbb{F}^k$. Then $c_i = \langle U_i, d_\alpha\rangle$. These values provide the out-of-domain information used in the correlated-agreement analysis.

Reducing the evaluation claim. For an evaluation claim $f(\mathbf{z}) = y$, the prover computes the row evaluations $v := Uq_2$, where $v_i = \langle U_i, q_2\rangle$. The original evaluation can therefore be written as $\langle q_1, v\rangle = y$. This equality alone does not ensure that the entries of v are the claimed row evaluations. It remains to verify $v_i = \langle U_i, q_2\rangle$ for all $i \in [m]$. After v is fixed, the verifier samples $r = (r_1,\dots,r_m) \in \mathbb{F}^m$, and we define

$$U_r := \sum_{i=1}^m r_i U_i.$$

The row-evaluation claims are then combined into

$$\langle U_r, q_2\rangle = \sum_{i=1}^m r_i v_i = \langle r, v\rangle.$$

Thus, after checking $\langle q_1, v\rangle = y$, the verifier reduces the remaining row-evaluation consistency check to a linear constraint on the smaller vector U_r. The same challenge also combines the DEEP evaluations. Since $c_i = \langle U_i, d_\alpha\rangle$, we obtain

$$\langle U_r, d_\alpha\rangle = \sum_{i=1}^m r_i c_i = \langle r, c\rangle.$$

The WHIR instance. By linearity of the Reed–Solomon encoder,

$$\widehat{U}_r := \sum_{i=1}^m r_i \widehat{U}_i = \mathsf{Enc}_{\mathsf{row}}(U_r).$$

Thus, the outer commitment together with the challenge r determines a virtual WHIR initial oracle $\widehat{U}_r$, and no separate encoding or commitment is required.

The folded evaluation constraint and the DEEP constraint are both linear constraints on U_r:

$$\langle U_r, q_2\rangle = \langle r, v\rangle, \qquad \langle U_r, d_\alpha\rangle = \langle r, c\rangle.$$

Following WHIR's batching procedure [3, Section 5.2] for linear constraints, a random challenge $\gamma \in \mathbb{F}$ combines them into

$$\langle U_r, q_2 + \gamma d_\alpha \rangle = \langle r, v \rangle + \gamma \langle r, c \rangle.$$

LiftWHIR invokes WHIR on the virtual oracle $\widehat{U}_r$ with the above batched linear constraint.

3.1 The LiftWHIR Protocol

We present LiftWHIR as a public-coin IOP and write Π_{WHIR} for the WHIR protocol instantiated with the row Reed–Solomon code defined above. We use the notation established above for the outer oracle and the two linear constraints on the folded row. The linear functionals represented by q_2 and d_α are supplied to Π_{WHIR} through their succinct descriptions; the verifier need not materialize either vector explicitly.

In the concrete PCS instantiation, a single Merkle tree commits to the outer oracle in column-major order. An authenticated opening at position $j \in [N]$ reveals the entire column $\widehat{U}_{1,j}, \ldots, \widehat{U}_{m,j}$. Subsequent WHIR oracle messages are committed as prescribed by Π_{WHIR}, and the public-coin protocol is made non-interactive via the Fiat–Shamir transformation.

Commitment phase.

1. The prover $\mathcal{P}$ computes $\widehat{U}_i := \mathsf{Enc}_{\mathsf{row}}(U_i)$ for each $i \in [m]$, obtaining the outer oracle $\widehat{U} \in \mathbb{F}^{m \times N}$, and sends $\widehat{U}$ to the verifier $\mathcal{V}$.
2. The verifier sends $\alpha \xleftarrow{\$} \mathbb{F} \setminus L$ to the prover.
3. Upon receiving α, the prover computes $c_i := P_i(\alpha)$ for each $i \in [m]$ and sends $c = (c_1, \ldots, c_m) \in \mathbb{F}^m$ to the verifier.

Evaluation phase. Given an evaluation claim $f(\mathbf{z}) = y$, where $\mathbf{z} \in \mathbb{F}^\mu$ and $y \in \mathbb{F}$, let $q = q_1 \otimes q_2$ be the corresponding evaluation vector under the tensor decomposition above.

1. The prover computes $v := U q_2 \in \mathbb{F}^m$ and sends v to the verifier.

   The verifier checks $\langle q_1, v \rangle \stackrel{?}{=} y$. If the equality does not hold, the verifier rejects.

2. The verifier sends $r = (r_1, \ldots, r_m) \xleftarrow{\$} \mathbb{F}^m$ to the prover.

   Upon receiving r, the prover computes

   $$U_r := \sum_{i=1}^{m} r_i U_i \in \mathbb{F}^k.$$

   The outer oracle and the challenge r define the virtual folded oracle

   $$\widehat{U}_r := \sum_{i=1}^{m} r_i \widehat{U}_i.$$

   The verifier computes $y_{\mathsf{eval}} := \langle r, v \rangle$ and $y_{\mathsf{deep}} := \langle r, c \rangle$.

3. The verifier sends $\gamma \xleftarrow{\$} \mathbb{F}$ to the prover and computes $y^\star := y_{\mathsf{eval}} + \gamma y_{\mathsf{deep}}$.

4. The prover and verifier run Π_{WHIR} with $\widehat{U}_r$ as the virtual initial oracle and with the batched linear claim

   $$\langle U_r, q_2 + \gamma d_\alpha \rangle = y^\star.$$

   Whenever the WHIR verifier queries a position $j \in [N]$ of the virtual initial oracle $\widehat{U}_r$, the prover opens the j-th column of the outer oracle $\widehat{U}$. After authenticating the opening, the verifier computes

   $$\widehat{U}_{r,j} := \sum_{i=1}^{m} r_i \widehat{U}_{i,j}$$

   and uses $\widehat{U}_{r,j}$ as the response to the corresponding initial-oracle query of Π_{WHIR}.

   All remaining WHIR messages, oracle queries, and verification steps are performed according to Π_{WHIR}.

5. The verifier accepts if all queried outer-oracle columns are authenticated successfully and the WHIR verifier accepts. Otherwise, it rejects.

3.2 Complexity Analysis

We analyze one commitment followed by one evaluation. We assume that the row Reed–Solomon code has constant rate, so that $N = \Theta(k)$, and choose

$$m = \Theta(\log n), \qquad k = \frac{n}{m} = \Theta\left(\frac{n}{\log n}\right).$$

Let ℓ_0 denote the number of queries that WHIR makes to its initial oracle $\widehat{U}_r$. We count field operations and hash evaluations separately, and treat field elements and hash outputs as constant-size words.

The choice $m = \Theta(\log n)$ gives the asymptotic bounds below. In concrete instantiations, m is tunable: increasing m reduces the WHIR instance size $k = n/m$, but increases the lifting-layer communication and verification work because c, v, and each opened column contain m field elements. Section 5.3 evaluates this tradeoff.

Commitment. The prover encodes the m rows of U using the Reed–Solomon code, which requires $O(mN \log N) = O(n \log n)$ field operations. Computing the m out-of-domain evaluations $c_i = P_i(\alpha)$ requires an additional $O(mk) = O(n)$ field operations. Constructing the outer Merkle commitment requires $O(mN) = O(n)$ hash evaluations. Hence, the commitment phase takes $O(n \log n)$ field operations and $O(n)$ hashes.

In the Merkle-tree instantiation, the commitment-phase communication consists of the outer Merkle root and $c \in \mathbb{F}^m$, and therefore has size $O(\log n)$ words.

Evaluation. The prover computes $v = Uq_2$ and $U_r = \sum_{i=1}^{m} r_i U_i$, each using $O(mk)$ field operations. The folded Reed–Solomon word $\widehat{U}_r = \sum_{i=1}^{m} r_i \widehat{U}_i$ is not materialized, re-encoded, or separately committed; instead, it is used as the virtual initial oracle of WHIR.

The remaining prover work is therefore that of WHIR starting from the virtual initial oracle determined by the outer commitment and the challenge r. LiftWHIR thus avoids WHIR's initial encoding and commitment costs. The subsequent WHIR recursion is quasi-linear in k, and since $k = \Theta(n/\log n)$, the overall evaluation cost is

$$O(mk + k\log k) = O(n)$$

field operations. No additional Merkle commitment is constructed for $\widehat{U}_r$.

Verification. The verifier checks $\langle q_1, v\rangle = y$ and computes $\langle r, v\rangle$ and $\langle r, c\rangle$, which together require $O(m)$ field operations.

For each of the ℓ_0 queries to the virtual initial oracle, the verifier computes $\widehat{U}_{r,j} = \sum_{i=1}^{m} r_i \widehat{U}_{i,j}$ from the authenticated j-th column of $\widehat{U}$. This contributes $O(m\ell_0)$ field operations. Thus, in addition to the underlying WHIR verification, LiftWHIR introduces

$$O(m\ell_0 + m) = O(\ell_0 \log n)$$

field operations.

Under the outer Merkle commitment, authenticating the ℓ_0 opened columns requires

$$O(m\ell_0 + \ell_0 \log(N/\ell_0))$$

hash operations, in addition to the hashes required by the remaining WHIR verification.

Proof size. Besides the WHIR transcript, the evaluation proof contains $v \in \mathbb{F}^m$ and the ℓ_0 authenticated columns of the outer oracle. The latter contribute $m\ell_0$ field elements and a shared Merkle multiproof of size $O(\ell_0 \log(N/\ell_0))$ hash outputs. Therefore, the lifting layer adds

$$O(m + m\ell_0 + \ell_0 \log(N/\ell_0)) = O(\ell_0 \log n)$$

words to the underlying WHIR proof.

3.3 Security

We prove that LiftWHIR is perfectly complete and knowledge sound in the random-oracle model, using the mutual correlated agreement theorem of Bordage et al. [12] within the Johnson bound, and conditionally, under Conjecture 2.4 together with efficient list decoding as formalized in Assumption B.6, in the conjectured capacity regime. The formal statements and proofs appear in Appendix A.

4 A Standalone Variant

The reduction underlying LiftWHIR can also be used without the final WHIR invocation. In this standalone variant, the prover sends the folded vector U_r explicitly, and the verifier checks its Reed–Solomon consistency at randomly sampled positions together with the two linear constraints of the reduction. Removing the WHIR proof avoids the additional WHIR proving overhead, at the cost of higher verifier time and proof size.

4.1 Protocol

We use the matrix layout, row encoder, and query notation from Section 3. In the standalone variant, we use the square layout $m = k = \sqrt{n}$, and write $U \in \mathbb{F}^{m\times k}$ and $\widehat{U} \in \mathbb{F}^{m\times N}$ for the unencoded and encoded matrices, respectively. The protocol also reuses the column-wise Merkle commitment from Section 3.1. We describe the interactive public-coin protocol below; the Merkle-tree commitment and Fiat–Shamir transformation follow the same procedure as in LiftWHIR.

Commitment phase.

1. The prover sends the outer oracle $\widehat{U} \in \mathbb{F}^{m\times N}$. For an honest prover,
$$\widehat{U}_i = \mathsf{Enc}_{\mathsf{row}}(U_i) = P_i|_L \qquad \text{for every } i \in [m].$$
2. The verifier samples $\alpha \xleftarrow{\$} \mathbb{F} \setminus L$ and sends α to the prover.
3. The prover sends $c = (c_1, \ldots, c_m) \in \mathbb{F}^m$, where $c_i = P_i(\alpha)$ for every $i \in [m]$.

Evaluation phase. Given an evaluation claim $f(\mathbf{z}) = y$, where $\mathbf{z} \in \mathbb{F}^\mu$ and $y \in \mathbb{F}$, let $q = q_1 \otimes q_2$ be the corresponding evaluation vector as in Section 3.

1. The prover sends $v \in \mathbb{F}^m$, where $v = Uq_2$. The verifier checks $\langle q_1, v\rangle \stackrel{?}{=} y$. If the equality does not hold, the verifier rejects.
2. The verifier samples $r = (r_1, \ldots, r_m) \xleftarrow{\$} \mathbb{F}^m$ and sends r to the prover.
3. The prover sends $U_r \in \mathbb{F}^k$ claimed to satisfy
$$U_r = \sum_{i=1}^{m} r_i U_i.$$
Let P_r be the degree-$< k$ polynomial whose coefficient vector is U_r.

4. After U_r is fixed, the verifier samples a uniformly random set $Q \subseteq [N]$ of size ℓ and sends Q to the prover. For every $j \in Q$, the prover opens the j-th column of $\widehat{U}$. The verifier authenticates the column and checks

$$\mathsf{Enc}_{\mathsf{row}}(U_r)_j \stackrel{?}{=} \sum_{i=1}^{m} r_i \widehat{U}_{i,j}.$$

5. The verifier checks

$$\langle U_r, q_2 \rangle \stackrel{?}{=} \langle r, v \rangle \qquad \text{and} \qquad P_r(\alpha) \stackrel{?}{=} \langle r, c \rangle.$$

6. The verifier accepts if all authenticated openings and all checks above accept. Otherwise, it rejects.

The sampled-column test ensures that the explicitly provided vector U_r is consistent with the committed oracle by the linearity of the Reed–Solomon encoder. The two remaining checks correspond to the two linear constraints inherited from the LiftWHIR reduction: $\langle U_r, q_2 \rangle = \langle r, v \rangle$ checks the folded row-evaluation claim, while $P_r(\alpha) = \langle r, c \rangle$ checks consistency with the out-of-domain evaluations provided during the commitment phase.

4.2 Complexity Analysis

We analyze one commitment followed by one evaluation. Since the row code has constant rate, $N = \Theta(k)$. The standalone variant uses the square layout $m = k = \sqrt{n}$.

Commitment time. The prover encodes all rows and computes the DEEP values c. Therefore,

$$T_{\mathcal{P}}^{\mathsf{com}} = O(mN \log N + mk) = O(n \log n),$$

with $O(n)$ hash computations for the outer Merkle tree. The commitment-phase communication consists of the Merkle root and $c \in \mathbb{F}^m$, and therefore has size $O(\sqrt{n})$ words.

Evaluation prover time. The prover computes $v = Uq_2$ and $U_r = \sum_{i=1}^{m} r_i U_i$. Therefore,

$$T_{\mathcal{P}}^{\mathsf{eval}} = O(mk) = O(n).$$

No additional Reed–Solomon encoding or Merkle commitment is constructed for U_r.

Verifier time. The verifier can encode U_r in $O(N \log N)$ field operations. Combining the m authenticated values in each of the ℓ sampled columns costs $O(m\ell)$ operations, while the two linear checks cost $O(k+m)$. Hence,

$$\begin{aligned} T_{\mathcal{V}} &= O(N \log N + m\ell + k + m) \\ &= O(\sqrt{n} \log n + \ell\sqrt{n}). \end{aligned}$$

Alternatively, the verifier can compute only the queried Reed–Solomon coordinates of U_r, reducing the arithmetic cost to $O(\ell\sqrt{n})$.

Communication. The evaluation communication consists of v, the explicit folded vector U_r, and the sampled column openings. Thus, $O(m + k + m\ell)$ field elements are communicated during evaluation. A shared Merkle multiproof adds $O(\ell \log(N/\ell))$ hash outputs.

4.3 Security

We prove that the standalone variant is perfectly complete and knowledge sound in the random-oracle model. The proof uses the mutual correlated agreement theorem of Bordage et al. [12] within the Johnson bound, and conditionally, under Conjecture 2.4 together with efficient list decoding as formalized in Assumption B.6, in the conjectured capacity regime. The formal statements and proofs appear in Appendix B.

5 Performance Evaluation

We examine how Reed–Solomon lifting affects prover time, verification time, and proof size in practice. We compare LiftWHIR with prior PCSs across input sizes and characterize the tradeoffs exposed by its parameter choices. We then replace WHIR with LiftWHIR in Spartan to test whether the reduction in prover time persists end to end.

5.1 Experimental Setup

We evaluate LiftWHIR against WHIR [3], DeepFold [27], BaseFold [42], Shockwave, and Brakedown [26]. We also evaluate the standalone variant from Section 4 to isolate the performance of the lifting reduction without the subsequent WHIR invocation. All configurations target 128-bit security. LiftWHIR, WHIR, the standalone variant, and DeepFold use the list-decoding-capacity setting of Conjecture 2.4. We use the same setting for Shockwave after augmenting it with the DEEP check. BaseFold and Brakedown use their respective unique-decoding parameters. Brakedown uses its native code of rate approximately $2/3$; all other curves in the rate-$1/2$ scaling comparison use Reed–Solomon codes of rate $1/2$.

For Shockwave and Brakedown, we use the fresh-random-point optimization of Diamond and Posen [22], which lets proximity testing and evaluation reuse the same random row fold. Their measurements in Figure 1 therefore correspond to fresh random evaluation points sampled after commitment.

The implementation uses the 255-bit prime field $\mathbb{F}_p$, where

$$\begin{aligned} p = {} & 46242760681095663677370860714659204618 \cdot 10^{39} \\ & + 85964256042920260721392983675019408 1793. \end{aligned}$$

We run the benchmarks in Ubuntu 20.04 under WSL2 on an Intel Core i7-13700 host with 32 GB of memory. Every PCS and Spartan measurement uses one thread. The PCS results are medians over 20 measured executions, while the Spartan results are means over three executions. We report

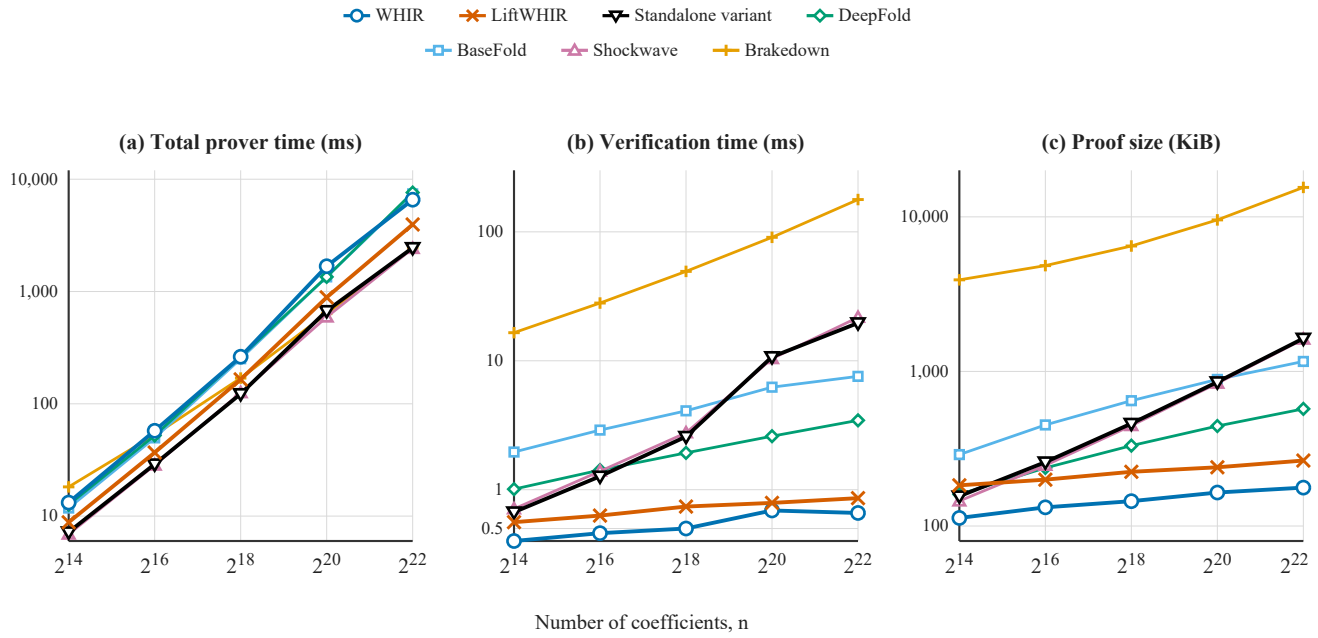

Figure 1: Scaling comparison over the 255-bit prime field. The Reed–Solomon-based schemes use code rate $1/2$, whereas Brakedown uses Brakedown's native rate of approximately $2/3$. WHIR uses $f = 4$, and LiftWHIR uses $m = 8, f = 2$. Each point is the median of 20 single-threaded executions. All vertical axes are logarithmic.

commitment time, evaluation time, their sum as total prover time, verification time, and proof size. Here n denotes the coefficient-vector length.

We use two independent PCS benchmark batches. The scaling batch fixes WHIR at $f = 4$ and LiftWHIR at $m = 8, f = 2$, and varies n from 2^{14} to 2^{22}. The configuration batch fixes $n = 2^{20}$ and sweeps m and f to expose the evaluation-time–proof-size frontier. Ratios below compare measurements only within the same batch.

5.2 Scaling Across Schemes

Figure 1 reports the scaling results. Its panels show total prover time, verification time, and proof size. As the number of coefficients grows, LiftWHIR remains faster than WHIR, DeepFold, and BaseFold in total prover time at every measured size. At $n = 2^{22}$, it is $1.66\times$, $1.92\times$, and $1.87\times$ faster than those three schemes in total prover time, respectively. Relative to WHIR, its proof is $1.50\times$ as large, while verification rises by only 0.20 ms and remains below 1 ms.

The standalone variant, Shockwave, and Brakedown occupy a different part of the tradeoff. At $n = 2^{22}$, they are $1.61\times$, $1.64\times$, and $1.62\times$ faster than LiftWHIR in total prover time. In return, their verification times are $22.8\times$, $25.2\times$, and $207.0\times$ higher, and their proofs are $6.2\times$, $6.1\times$, and $58.5\times$ larger. Shockwave has lower total prover time at every measured size, with its advantage over LiftWHIR increasing from $1.28\times$ to $1.64\times$. Brakedown is slower than LiftWHIR through $n = 2^{18}$ and becomes faster between 2^{18} and 2^{20}, whereas its verification and communication costs remain substantially higher throughout the sweep.

The standalone variant also clarifies the cost profile of the lifting reduction itself. It tracks Shockwave closely across the sweep: their total prover times differ by at most 14%, their proof sizes by at most 8%, and their verification times by at most 11%. At $n = 2^{22}$, the standalone variant and Brakedown also have nearly identical total prover time, while the standalone variant verifies $9.07\times$ faster and produces a $9.48\times$ smaller proof.

To match the standalone variant's support for openings at arbitrary evaluation points, Shockwave and Brakedown require a preceding partial Sumcheck [33] that reduces the target evaluation to a fresh random point, as in Ligerito [36]. Figure 1 does not include this additional cost. Against these optimized fresh-random-point baselines, the standalone variant closely matches Shockwave in all three metrics and remains near Brakedown in prover time while sharply reducing Brakedown's verification and communication costs. Thus, LiftWHIR's core lifting reduction supports arbitrary-point openings while retaining the low-prover-time profile of Shockwave and Brakedown.

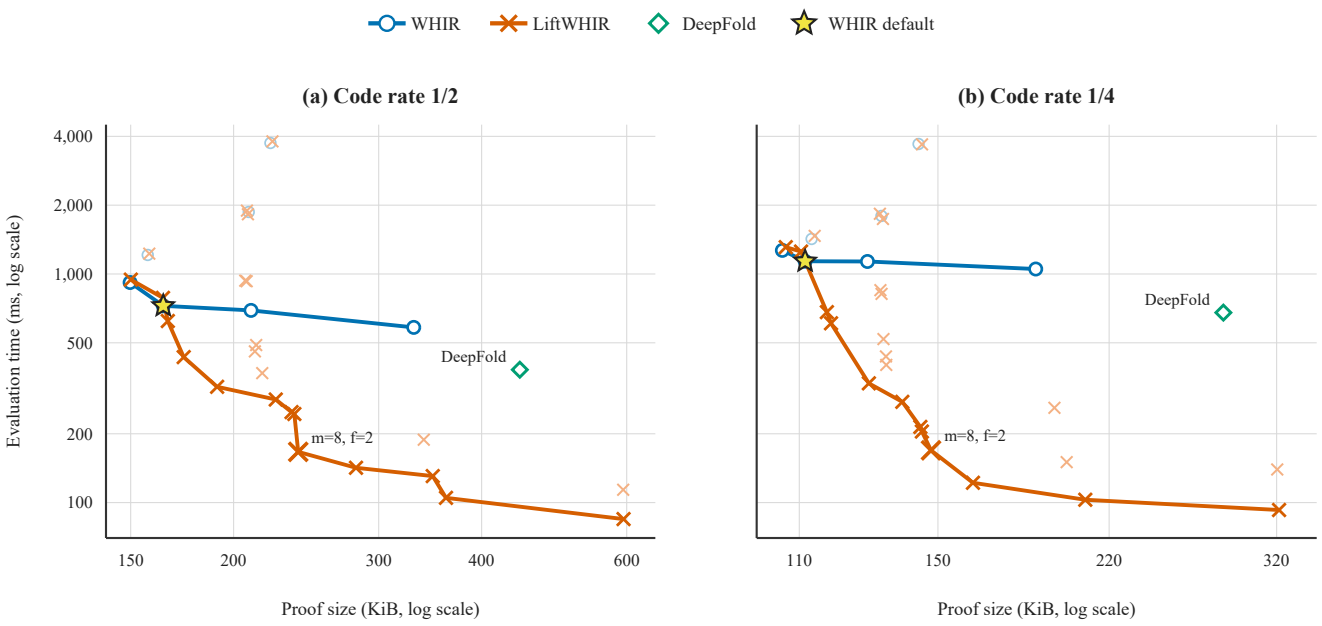

Figure 2: Evaluation-time–proof-size tradeoffs at $n = 2^{20}$ for code rates (a) $1/2$ and (b) $1/4$. We sweep m and f for LiftWHIR, and f for WHIR. The star identifies the default WHIR configuration, and the DeepFold point provides a standalone reference. Both axes are logarithmic, and every point is the median of 20 single-threaded executions.

5.3 Evaluation-Time–Proof-Size Tradeoffs

LiftWHIR introduces the row count m in addition to WHIR's first-fold parameter f. Increasing m reduces the instance handled recursively by WHIR, but enlarges the lifting layer. Figure 2 plots the resulting measured tradeoffs at $n = 2^{20}$. Solid lines show the Pareto frontiers. Faint markers are measured configurations dominated in both evaluation time and proof size. The star marks WHIR's default $f = 4$ configuration.

The $m = 8, f = 2$ configuration illustrates the main operating point. At code rate $1/2$, it reduces evaluation time from 725.6 ms for default WHIR to 166.7 ms, a $4.35\times$ speedup. Verification time increases from 0.549 ms to 0.758 ms, and the proof is $1.46\times$ as large. At rate $1/4$, evaluation time falls from 1136.9 ms to 169.3 ms, a $6.72\times$ speedup, while verification time and proof size are $1.28\times$ and $1.33\times$ those of WHIR, respectively. The same LiftWHIR configuration also improves both plotted metrics over DeepFold: its proof is $1.85\times$ smaller and its evaluation is $2.29\times$ faster at rate $1/2$, with corresponding factors of $1.92\times$ and $4.00\times$ at rate $1/4$.

The frontier provides further operating points when a larger proof is acceptable. Among measured configurations whose proofs are at most $1.25\times$ the default WHIR size, the fastest LiftWHIR configuration gives $2.26\times$ and $4.11\times$ evaluation-time speedups at rates $1/2$ and $1/4$. At $1.5\times$ the default size, the corresponding speedups are $4.16\times$ and $9.29\times$. LiftWHIR does not dominate WHIR at every proof size. Rather, the lifting parameters extend the measured frontier toward lower evaluation latency once a moderate increase in communication is allowed.

5.4 Spartan Integration

We finally replace WHIR with LiftWHIR as the PCS in Spartan [40]. Both versions use $n = 2^{20}$, code rate $1/2$, and the same 255-bit prime field. Spartan+WHIR uses $f = 4$, while Spartan+LiftWHIR uses $m = 8, f = 2$.

Metric	WHIR	LiftWHIR
Offline time (s)	23.0	21.9
PCS opening (s)	28.2	5.27
Online prover (s)	50.7	26.8
Verification (ms)	3.86	4.62
Proof size (bytes)	812,613	1,150,387

Table 2: Spartan integration at $n = 2^{20}$ and code rate $1/2$. Each entry is the mean of three single-threaded executions.

Replacing WHIR with LiftWHIR accelerates the PCS-opening phase by $5.35\times$ and reduces end-to-end online prover time by $1.89\times$. The offline phase changes little. The online improvement therefore comes primarily from faster PCS evaluation. Verification time and proof size increase by $1.20\times$ and $1.42\times$, respectively.

Ethical Considerations

This work develops and evaluates a general-purpose cryptographic primitive. The research uses mathematical analysis, synthetic benchmark instances, and experiments on author-controlled hardware. It does not involve human participants, personal or sensitive data, interactions with deployed services, or the discovery or exploitation of vulnerabilities.

The stakeholders include researchers and implementers of polynomial commitments and proof systems, developers who integrate these primitives, and users of applications built from them. More efficient polynomial commitments can reduce the cost of verifiable computation. As with other general-purpose cryptographic tools, the eventual effects depend on the applications in which the primitive is deployed. A separate risk is that implementers could treat experimental parameter choices as unconditional security guarantees. We mitigate this risk by distinguishing the proven security statements from the parameter settings that rely on the stated correlated-agreement conjecture, and by reporting the benchmark configuration needed to interpret the measurements.

Publishing the construction, analysis, implementation, and evaluation materials allows the community to examine both the security arguments and the performance claims. Given that the research does not act on real users or production systems, we find that these benefits outweigh the limited indirect risks of publication.

Open Science

Artifact access. An artifact for anonymous review is available at https://anonymous.4open.science/r/liftwhir-F689. If the paper is accepted, we will replace this anonymous snapshot with a stable public release.

Implementation and benchmarks. The artifact contains the Rust implementation of LiftWHIR, the standalone variant, the benchmarked PCS baselines, and the Spartan integration evaluated in Section 5. It also contains the benchmark driver, correctness tests, a pinned Rust toolchain and dependency lockfile, and a Dockerfile. The README gives build, test, smoke-test, and benchmark commands and documents the principal parameter controls and output columns.

Evaluation data. The `results/` directory contains the five CSV tables underlying the PCS scaling, headline comparison, rate-specific configuration frontiers, and Spartan experiment. The tables record the input size, code rate, configuration, repetition count, aggregation rule, thread setting, timing components, and proof size used for the reported results. These files contain the aggregate measurements used in the paper rather than the individual per-run timing logs.

Reproduction scope. The archived CSV files permit independent checking of the values reported in the paper, while the implementation and benchmark driver support fresh measurements on compatible hardware. The README explains how to force the single-threaded setting used in our experiments. Absolute timings may vary across machines.

References

[1] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2087–2104. ACM, 2017. doi:10.1145/3133956.3134104.

[2] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-Solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Proceedings, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 380–413. Springer, 2024. doi:10.1007/978-3-031-68403-6_12.

[3] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed-Solomon proximity testing with super-fast verification. In *Advances in Cryptology – EUROCRYPT 2025, Proceedings, Part IV*, volume 15604 of *Lecture Notes in Computer Science*, pages 214–243. Springer, 2025. doi:10.1007/978-3-031-91134-7_8.

[4] Anubhav Baweja, Pratyush Mishra, Tushar Mopuri, and Matan Shtepel. FICS and FACS: Fast IOPPs and accumulation via code-switching. Cryptology ePrint Archive, Paper 2025/737, 2025. URL: https://eprint.iacr.org/2025/737.

[5] Anubhav Baweja, Pratyush Mishra, Tushar Mopuri, and Matan Shtepel. Query-optimal IOPPs for linear-time encodable codes. In *Advances in Cryptology – EUROCRYPT 2026, Proceedings, Part VII*, volume 16547 of *Lecture Notes in Computer Science*. Springer, 2026. doi:10.1007/978-3-032-25336-1_4.

[6] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics*, pages 14:1–14:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.14.

[7] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for Reed-Solomon codes. *Journal of the ACM*, 70(5):31:1–31:57, 2023. doi:10.1145/3614423.

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

[9] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography 2016-B, Proceedings, Part II*, Lecture Notes in Computer Science, pages 31–60. Springer, 2016. doi:10.1007/978-3-662-53644-5_2.

[10] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151 of *Leibniz International Proceedings in Informatics*, pages 5:1–5:32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.5.

[11] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012*, pages 326–349. ACM, 2012. doi:10.1145/2090236.2090263.

[12] Sarah Bordage, Alessandro Chiesa, Ziyi Guan, and Ignacio Manzur. All polynomial generators preserve distance with mutual correlated agreement. Cryptology ePrint Archive, Paper 2025/2051, 2025. Full version of the CCC 2026 paper. URL: https://eprint.iacr.org/2025/2051.

[13] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964. IEEE, 2020. doi:10.1109/SP40000.2020.00050.

[14] Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. Blaze: Fast SNARKs from interleaved RAA codes. In *Advances in Cryptology – EUROCRYPT 2025, Proceedings, Part IV*, volume 15604 of *Lecture Notes in Computer Science*, pages 123–152. Springer, 2025. doi:10.1007/978-3-031-91134-7_5.

[15] Benedikt Bünz, Giacomo Fenzi, Ron D. Rothblum, and William Wang. TensorSwitch: Nearly optimal polynomial commitments from tensor codes. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology – CRYPTO 2026, Proceedings, Part IX*, volume 16808 of *Lecture Notes in Computer Science*, pages 297–329. Springer, 2026. doi:10.1007/978-3-032-35424-2_11.

[16] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In *Advances in Cryptology – EUROCRYPT 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 677–706. Springer, 2020. doi:10.1007/978-3-030-45721-1_24.

[17] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. ZKML: An optimizing system for ML inference in zero-knowledge proofs. In *Proceedings of the Nineteenth European Conference on Computer Systems*, EuroSys '24, pages 560–574. ACM, April 2024. doi:10.1145/3627703.3650088.

[18] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In *Advances in Cryptology – EUROCRYPT 2023, Proceedings, Part II*, volume 14005 of *Lecture Notes in Computer Science*, pages 499–530. Springer, 2023. doi:10.1007/978-3-031-30617-4_17.

[19] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology – EUROCRYPT 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768. Springer, 2020. doi:10.1007/978-3-030-45721-1_26.

[20] Mingshu Cong, Tsz Hon Yuen, and Siu-Ming Yiu. SwitchFold: Code-agnostic succinct polynomial commitments via recursive code switching. Cryptology ePrint Archive, Paper 2026/1489, 2026. URL: https://eprint.iacr.org/2026/1489.

[21] Elizabeth Crites and Alistair Stewart. On Reed-Solomon proximity gaps conjectures. Cryptology ePrint Archive, Paper 2025/2046, 2025. URL: https://eprint.iacr.org/2025/2046.

[22] Benjamin E. Diamond and Jim Posen. Proximity testing with logarithmic randomness. *IACR Communications in Cryptology*, 1(1):2, April 2024. doi:10.62056/aksdkp10.

[23] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987. doi:10.1007/3-540-47721-7_12.

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

[25] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 99–108. ACM, 2011. doi:10.1145/1993636.1993651.

[26] Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In *Advances in Cryptology – CRYPTO 2023, Proceedings, Part II*, Lecture Notes in Computer Science, pages 193–226. Springer, 2023. doi:10.1007/978-3-031-38545-2_7.

[27] Yanpei Guo, Xuanming Liu, Kexi Huang, Wenjie Qu, Tianyang Tao, and Jiaheng Zhang. DeepFold: Efficient multilinear polynomial commitment from Reed-Solomon code and its application to zero-knowledge proofs. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3497–3516, Seattle, WA, August 2025. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/guo-yanpei.

[28] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. doi:10.1109/18.782097.

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

[30] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732. ACM, 1992. doi:10.1145/129712.129782.

[31] Christian Knabenhans, Alexander Viand, Antonio Merino-Gallardo, and Anwar Hithnawi. vFHE: Verifiable fully homomorphic encryption. In *Proceedings of the 12th Workshop on Encrypted Computing and Applied Homomorphic Cryptography*, CCS '24, pages 11–22. ACM, October 2024. doi:10.1145/3689945.3694806.

[32] Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985. ACM, 2021. doi:10.1145/3460120.3485379.

[33] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.

[34] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science*, pages 436–453. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365746.

[35] Vineet Nair, Ashish Sharma, and Bhargav Thankey. BrakingBase—a linear prover, poly-logarithmic verifier, field agnostic polynomial commitment scheme. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025, Proceedings, Part V*, volume 16249 of *Lecture Notes in Computer Science*, pages 427–455. Springer, 2025. doi:10.1007/978-981-95-5116-3_14.

[36] Andrija Novakovic and Guillermo Angeris. Ligerito: A small and concretely fast polynomial commitment scheme. Cryptology ePrint Archive, Paper 2025/1187, 2025. URL: https://eprint.iacr.org/2025/1187.

[37] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography – 10th Theory of Cryptography Conference, TCC 2013*, volume 7785 of *Lecture Notes in Computer Science*, pages 222–242. Springer, 2013. doi:10.1007/978-3-642-36594-2_13.

[38] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013. doi:10.1109/SP.2013.47.

[39] Noga Ron-Zewi and Ron D. Rothblum. Local proofs approaching the witness length. *Journal of the ACM*, 71(3):18, 2024. doi:10.1145/3661483.

[40] Srinath T. V. Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Advances in Cryptology – CRYPTO 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 704–737. Springer, 2020. doi:10.1007/978-3-030-56877-1_25.

[41] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 299–328. Springer, 2022. doi:10.1007/978-3-031-15985-5_11.

[42] Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In *Advances in Cryptology – CRYPTO 2024, Proceedings, Part X*, volume 14929 of *Lecture Notes in Computer Science*, pages 138–169. Springer, 2024. doi:10.1007/978-3-031-68403-6_5.

A Security of LiftWHIR

This appendix proves perfect completeness and knowledge soundness for the LiftWHIR IOP. LiftWHIR reduces a multilinear evaluation claim for the outer interleaved oracle to a constrained Reed–Solomon instance. The extractor first invokes the WHIR extractor on the folded instance. Mutual correlated agreement maps the resulting witness to the joint decoding list of the outer oracle, and the DEEP values select a unique matrix $U^\star$ from that list.

The proof uses WHIR's knowledge-soundness result for one linear constraint. LiftWHIR batches its two constraints with the challenge γ from Section 3.1. Lemma A.5 derives extraction for the two original constraints and bounds the error introduced by batching.

A.1 Relations and extraction

Recall that $n = mk$, that the row code is $\mathsf{RS}_{L,k} \subseteq \mathbb{F}^N$, and that $\widehat{U} \in \mathbb{F}^{m\times N}$ denotes the outer oracle. For $U \in \mathbb{F}^{m\times k}$, write U_i for its i-th row and define

$$\mathsf{Enc}_{\mathsf{row}}(U) := \begin{pmatrix} \mathsf{Enc}_{\mathsf{row}}(U_1) \\ \vdots \\ \mathsf{Enc}_{\mathsf{row}}(U_m) \end{pmatrix}.$$

Definition A.1 (Common agreement)**.** For $\delta \in (0,1)$, we write $\mathsf{Agree}_\delta(\widehat{U}, U) = 1$ if there exists a set $S \subseteq [N]$ with $|S| \geq (1-\delta)N$ such that

$$\widehat{U}_{i,j} = \mathsf{Enc}_{\mathsf{row}}(U_i)_j \qquad \text{for every } i \in [m] \text{ and } j \in S.$$

Common agreement is stronger than requiring each row to be individually close to the row code. It is the witness property supplied by correlated agreement and ensures that all extracted rows describe one matrix.

For $\alpha \in \mathbb{F} \setminus L$, recall the vector

$$d_\alpha = (1, \alpha, \ldots, \alpha^{k-1})^\top \in \mathbb{F}^k.$$

If P_i is the degree-$< k$ polynomial with coefficient vector U_i, then $P_i(\alpha) = \langle U_i, d_\alpha \rangle$.

Definition A.2 (Lifted evaluation relation)**.** The relation $\mathcal{R}_{\mathsf{Lift}}$ contains

$$\Big((\widehat{U}, \alpha, c, q_1, q_2, y); U\Big)$$

if

$$\mathsf{Agree}_\delta(\widehat{U}, U) = 1, \tag{1}$$
$$U d_\alpha = c, \tag{2}$$
$$q_1^\top U q_2 = y. \tag{3}$$

The witness U determines the multilinear polynomial extracted from the outer oracle. Equation (2) binds the commitment- phase DEEP values to its rows, while Equation (3) is the original evaluation claim.

Definition A.3 (Two-constraint WHIR relation)**.** The relation $\mathcal{R}^{(2)}_{\mathsf{WHIR}}$ contains

$$((\widehat{u}; a_0, s_0; a_1, s_1); u)$$

if

$$\Delta_L\big(\widehat{u}, \mathsf{Enc}_{\mathsf{row}}(u)\big) \leq \delta, \tag{4}$$
$$\langle u, a_0 \rangle = s_0, \tag{5}$$
$$\langle u, a_1 \rangle = s_1. \tag{6}$$

For a transcript prefix containing $\widehat{U}$, α, c, and v, the verifier samples $r \xleftarrow{\$} \mathbb{F}^m$ and defines

$$\widehat{U}_r := \sum_{i=1}^m r_i \widehat{U}_i, \tag{7}$$
$$x_{\mathsf{WHIR}}(r) := (\widehat{U}_r; q_2, \langle r, v \rangle; d_\alpha, \langle r, c \rangle). \tag{8}$$

For an honest prover, the witness is $U_r = r^\top U$.

For a word $w \in \mathbb{F}^N$, define its coefficient-vector decoding list by

$$\mathcal{L}^{\mathsf{coef}}_\delta(w) := \{u \in \mathbb{F}^k : \Delta_L(w, \mathsf{Enc}_{\mathsf{row}}(u)) \leq \delta\}.$$

Assumption A.4 (Single-constraint WHIR knowledge soundness)**.** Fix any transcript prefix before the WHIR invocation. For every constrained Reed–Solomon instance $(\widehat{u}; a, s)$ determined by this prefix and every malicious continuation that is accepted with probability p, there is an expected polynomial-time extractor that outputs $u \in \mathbb{F}^k$ satisfying

$$\Delta_L\big(\widehat{u}, \mathsf{Enc}_{\mathsf{row}}(u)\big) \leq \delta, \qquad \langle u, a \rangle = s$$

with probability at least $p - \kappa_{\mathsf{WHIR}}$. Let $T_{\mathsf{Ext},\mathsf{WHIR}}$ denote its expected running time.

This assumption is the single-constraint form of WHIR's round-by-round knowledge-soundness analysis for constrained Reed–Solomon codes [3].

Lemma A.5 (Extraction for two batched constraints)**.** *Fix $(\widehat{u}; a_0, s_0; a_1, s_1)$ and suppose that $|\mathcal{L}^{\mathsf{coef}}_\delta(\widehat{u})| \leq B_{\mathsf{loc}}$. Sample $\gamma \xleftarrow{\$} \mathbb{F}$ and run WHIR on the single constraint*

$$\langle u, a_0 + \gamma a_1 \rangle = s_0 + \gamma s_1.$$

If this batched execution accepts with probability p, then an extractor outputs $u \in \mathcal{L}^{\mathsf{coef}}_\delta(\widehat{u})$ satisfying both $\langle u, a_0 \rangle = s_0$ and $\langle u, a_1 \rangle = s_1$ with probability at least

$$p - \kappa_{\mathsf{WHIR}} - \frac{B_{\mathsf{loc}}}{|\mathbb{F}|}.$$

Proof. Invoke the extractor from Assumption A.4 for the constraint fixed by γ. Averaging over γ, it returns a nearby

vector satisfying the batched constraint with probability at least $p - \kappa_{\mathsf{WHIR}}$.

For every $u \in \mathcal{L}_\delta^{\mathsf{coef}}(\widehat{u})$ that does not satisfy both original constraints, the polynomial

$$\big(\langle u, a_0\rangle - s_0\big) + \gamma\big(\langle u, a_1\rangle - s_1\big)$$

is nonzero and has degree at most one in γ. It therefore vanishes for at most one field element. A union bound over the decoding list shows that an extracted witness can satisfy the batched constraint without satisfying both original constraints with probability at most $B_{\mathsf{loc}}/|\mathbb{F}|$. Discarding such outputs proves the claim. This is the two-constraint specialization of WHIR's batching analysis [3, Theorem 5.6]. □

For the LiftWHIR instance, set

$$a_0 = q_2, \quad s_0 = \langle r, v\rangle, \qquad a_1 = d_\alpha, \quad s_1 = \langle r, c\rangle.$$

Thus an output of Lemma A.5 satisfies

$$\Delta_L\big(\widehat{U}_r, \mathsf{Enc}_{\mathsf{row}}(U_r)\big) \le \delta, \tag{9}$$

$$\langle U_r, q_2\rangle = \langle r, v\rangle, \tag{10}$$

$$\langle U_r, d_\alpha\rangle = \langle r, c\rangle. \tag{11}$$

A.2 Extraction from correlated folds

Define the joint decoding list

$$\mathcal{J}_\delta(\widehat{U}) := \{U \in \mathbb{F}^{m\times k} : \mathsf{Agree}_\delta(\widehat{U}, U) = 1\},$$

Let B be an upper bound on $|\mathcal{J}_\delta(\widehat{U})|$, and let $T_{\mathsf{iRS}}(m, N, k, \delta)$ denote the time required to enumerate this joint list.

Lemma A.6 (Consistency of the folded list)**.** *Fix $0 < \delta < (N-k+1)/N$. Suppose the identity generator has mutual correlated agreement for $\mathsf{RS}_{L,k}$ at radius δ, with error $\varepsilon_{\mathsf{MCA}}$. Then, except with probability $\varepsilon_{\mathsf{MCA}}$ over $r \xleftarrow{\$} \mathbb{F}^m$,*

$$\mathcal{L}_\delta^{\mathsf{coef}}(\widehat{U}_r) = \{r^\top U : U \in \mathcal{J}_\delta(\widehat{U})\}.$$

Proof. This is the list-commutativity consequence of mutual correlated agreement [3, Lemma 4.13]. A codeword in the interleaved decoding list of the rows of $\widehat{U}$ is exactly a matrix in $\mathcal{J}_\delta(\widehat{U})$, and linearity maps its row fold to $r^\top U$. □

Lemma A.7 (Selection by the DEEP value)**.** *Let $|\mathcal{J}_\delta(\widehat{U})| \le B$, and sample $\alpha \xleftarrow{\$} \mathbb{F} \setminus L$. Except with probability*

$$\varepsilon_{\mathsf{DEEP}} := \binom{B}{2}\frac{k-1}{|\mathbb{F}| - N},$$

the vectors Ud_α are distinct over all $U \in \mathcal{J}_\delta(\widehat{U})$.

Fix this event and any $c \in \mathbb{F}^m$. If Lemma A.6 holds for a random r, then, except with probability at most $B/|\mathbb{F}|$ over r, every $U_r \in \mathcal{L}_\delta^{\mathsf{coef}}(\widehat{U}_r)$ satisfying

$$\langle U_r, d_\alpha\rangle = \langle r, c\rangle$$

equals $r^\top U^\star$ for the unique $U^\star \in \mathcal{J}_\delta(\widehat{U})$ satisfying $U^\star d_\alpha = c$. If no such candidate exists, no U_r satisfies the displayed equality outside the same error event.

Proof. For distinct $U, U' \in \mathcal{J}_\delta(\widehat{U})$, choose a row on which they differ. The difference of the corresponding row polynomials is nonzero and has degree less than k, so it vanishes at a random $\alpha \in \mathbb{F} \setminus L$ with probability at most $(k-1)/(|\mathbb{F}| - N)$. A union bound over candidate pairs proves the first claim.

By Lemma A.6, every nearby U_r has the form $r^\top U$ for some $U \in \mathcal{J}_\delta(\widehat{U})$. Its DEEP check is equivalent to

$$\langle r, Ud_\alpha - c\rangle = 0.$$

For each candidate with $Ud_\alpha \ne c$, this nontrivial linear equation holds with probability $1/|\mathbb{F}|$. A union bound over at most B candidates proves the second claim. □

A.3 Perfect completeness and knowledge soundness

Lemma A.8 (Perfect completeness)**.** *If WHIR is perfectly complete, then the interactive oracle protocol in Section 3.1 is perfectly complete.*

Proof. For an honest prover, linearity of the row encoder gives $\widehat{U}_r = \mathsf{Enc}_{\mathsf{row}}(U_r)$. The identities $v = Uq_2$, $c = Ud_\alpha$, and $U_r = r^\top U$ satisfy both linear claims and therefore their batched form for every γ. The outer evaluation check also holds because $\langle q_1, v\rangle = y$. Perfect completeness of WHIR then implies that every verifier check accepts. □

Theorem A.9 (LiftWHIR knowledge soundness)**.** *Assume WHIR knowledge soundness as stated in Assumption A.4. Suppose that every joint list $\mathcal{J}_\delta(\widehat{U})$ has size at most B and can be enumerated in time $T_{\mathsf{iRS}}(m, N, k, \delta)$. Suppose also that the identity generator satisfies the mutual correlated agreement condition in Lemma A.6. Then the LiftWHIR IOP is knowledge sound for $\mathcal{R}_{\mathsf{Lift}}$ with knowledge error at most*

$$\kappa_{\mathsf{Lift}} \le \kappa_{\mathsf{WHIR}} + \varepsilon_{\mathsf{MCA}} + \varepsilon_{\mathsf{DEEP}} + \frac{2B+1}{|\mathbb{F}|},$$

where $\varepsilon_{\mathsf{DEEP}}$ is defined in Lemma A.7. The two terms $B/|\mathbb{F}|$ account for constraint batching and DEEP selection. The remaining $1/|\mathbb{F}|$ accounts for the folded evaluation constraint. The extractor runs in expected time

$$T_{\mathsf{Ext}} = T_{\mathsf{iRS}}(m, N, k, \delta) + T_{\mathsf{Ext,WHIR}} + O(Bmk).$$

Proof. Fix a malicious LiftWHIR prover $\widetilde{\mathcal{P}}$, and let p_{acc} be its acceptance probability. The outer oracle $\widehat{U}$ is fixed before α is sampled, and c and v are fixed before the verifier samples r. For each challenge r, define a malicious WHIR prover by continuing $\widetilde{\mathcal{P}}$ from that transcript prefix. When WHIR queries

coordinate j of its virtual initial oracle, the reduction answers with

$$\widehat{U}_{r,j} = \sum_{i=1}^{m} r_i \widehat{U}_{i,j}.$$

Except with probability $\varepsilon_{\mathsf{MCA}}$, Lemma A.6 identifies the decoding list of $\widehat{U}_r$ with the folds of the joint list and therefore bounds its size by B. Conditioned on this event, Lemma A.5, applied to the continuation after r, returns a vector U_r satisfying Equations (9)–(11) with probability at least the continuation's acceptance probability minus $\kappa_{\mathsf{WHIR}} + B/|\mathbb{F}|$. Averaging over the uniform challenge r and accounting for the excluded event, such a vector is extracted with probability at least

$$p_{\mathsf{acc}} - \varepsilon_{\mathsf{MCA}} - \kappa_{\mathsf{WHIR}} - \frac{B}{|\mathbb{F}|}.$$

The vector U_r extracted from WHIR is an intermediate value in the analysis. The LiftWHIR extractor enumerates $\mathcal{J}_\delta(\widehat{U})$, computes Ud_α for every matrix in the list, and finds the unique $U^\star$ with $U^\star d_\alpha = c$. It outputs $U^\star$ if $U^\star q_2 = v$. The DEEP collision event has probability at most $\varepsilon_{\mathsf{DEEP}}$.

It remains to bound the probability that the verifier accepts but the LiftWHIR extractor does not output a witness. Conditioned on the conclusion of Lemma A.6, the extracted vector has the form $U_r = r^\top U$ for some $U \in \mathcal{J}_\delta(\widehat{U})$. The extracted DEEP constraint becomes

$$\langle r, Ud_\alpha - c \rangle = 0.$$

Lemma A.7 shows that, except with probability $B/|\mathbb{F}|$ over r, this equality implies that $U^\star$ exists and $U_r = r^\top U^\star$.

The extracted evaluation constraint now gives

$$\langle r, U^\star q_2 - v \rangle = 0.$$

If $U^\star q_2 \neq v$, this nontrivial linear equation holds for a uniform r with probability $1/|\mathbb{F}|$. Hence, outside the stated error, $U^\star q_2 = v$. The verifier also checks $\langle q_1, v \rangle = y$, and therefore

$$q_1^\top U^\star q_2 = \langle q_1, v \rangle = y.$$

$U^\star$ belongs to $\mathcal{J}_\delta(\widehat{U})$, satisfies $U^\star d_\alpha = c$, and satisfies the claimed evaluation. It therefore satisfies every condition of $\mathcal{R}_{\mathsf{Lift}}$, which proves the claimed knowledge error.

Enumerating the joint list costs $T_{\mathsf{iRS}}(m,N,k,\delta)$, the WHIR extraction costs $T_{\mathsf{Ext,WHIR}}$, and evaluating Ud_α for all candidates costs $O(Bmk)$ field operations. This proves the running-time bound. □

For $\delta = 1 - \sqrt{\rho} - \eta$ with fixed $\eta > 0$, polynomial-time Reed–Solomon list decoding and the mutual correlated agreement theorem of Bordage et al. [12,28] satisfy the hypotheses above within the Johnson radius. Under Assumption B.6, the theorem also applies in the conjectured capacity regime with the associated list-size bound and mutual correlated agreement error.

Corollary A.10 (Knowledge soundness of the compiled PCS)**.** Apply a Merkle-based IOP compiler that extracts an outer oracle consistent with the commitment, and apply the Fiat–Shamir transformation in the model required by that compiler. Under the conditions of Lemma A.8 and Theorem A.9, the compiled LiftWHIR evaluation argument is perfectly complete and knowledge sound. Its knowledge error is bounded by

$$\kappa_{\mathsf{PCS}} \leq \kappa_{\mathsf{Lift}} + \kappa_{\mathsf{comp}} + \mathsf{Adv}_H^{\mathsf{coll}}(\lambda),$$

where κ_{comp} is the compiler's knowledge error and $\mathsf{Adv}_H^{\mathsf{coll}}$ is the collision-finding advantage against the Merkle hash function.

Remark A.11 (Exact opening relation). The Merkle-based compiler first extracts an outer oracle $\widehat{U}$ consistent with the commitment. Theorem A.9 then extracts the polynomial represented by the unique matrix $U^\star$ in the joint list of $\widehat{U}$ that is selected by the DEEP values. As in the extractability analysis of Brakedown [26], the extracted oracle need not equal the encoding of the extracted polynomial. Instead, the proximity test binds the oracle to nearby codewords, and the DEEP constraint selects a unique matrix and hence a unique polynomial. This polynomial opens the commitment in the decoded opening relation and satisfies the exact evaluation relation $\mathcal{R}_{\mathsf{Eval,pp}}$ from Section 2.4.

B Security of the Standalone Variant

This appendix proves completeness and knowledge soundness for the standalone variant from Section 4. We first analyze its interactive oracle form. We then apply the same Merkle commitment and Fiat–Shamir compilation used for LiftWHIR.

B.1 Main Theorem

Recall that the row code is $\mathsf{RS}_{L,k} \subseteq \mathbb{F}^N$, where $|L| = N$, and let $\widehat{U} \in \mathbb{F}^{m \times N}$ be the committed oracle. Define its joint decoding list by

$$\mathcal{J}_\delta(\widehat{U}) := \{U \in \mathbb{F}^{m \times k} : \mathsf{Agree}_\delta(\widehat{U}, U) = 1\},$$

where common agreement is defined in Definition A.1. For a word $w \in \mathbb{F}^N$, also write

$$\mathcal{L}_\delta^{\mathsf{coef}}(w) := \{u \in \mathbb{F}^k : \Delta_L(w, \mathsf{Enc}_{\mathsf{row}}(u)) \leq \delta\}.$$

Let B be an upper bound on $|\mathcal{J}_\delta(\widehat{U})|$, and let $T_{\mathsf{iRS}}(m,N,k,\delta)$ denote the running time of an interleaved Reed–Solomon list decoder that enumerates this joint list. Such a decoder can be obtained by viewing each column as a symbol over an extension field and applying Reed–Solomon list decoding [3, 28].

Lemma B.1 (Perfect completeness)**.** *The interactive oracle protocol in Section 4.1 is perfectly complete.*

Proof. For an honest prover, the encoded row fold agrees with every authenticated column opening. Since $U_r = r^\top U$, $v = Uq_2$, and $c = Ud_\alpha$, both linear checks hold, and $\langle q_1, v\rangle = y$ verifies the claimed evaluation. Hence every verifier check accepts. □

Theorem B.2 (Standalone knowledge soundness)**.** *Fix a decoding radius δ and suppose that:*

1. *every committed oracle $\widehat{U}$ satisfies $|\mathcal{J}_\delta(\widehat{U})| \le B$, and its joint list can be enumerated in time $T_{\mathsf{iRS}}(m,N,k,\delta)$; and*
2. *the identity generator has mutual correlated agreement at radius δ with error $\varepsilon_{\mathsf{MCA}}$.*

Then the interactive oracle protocol in Section 4.1 is knowledge sound for $\mathcal{R}_{\mathsf{Lift}}$ from Definition A.2, with knowledge error

$$\kappa_{\mathsf{stand}} = (1-\delta)^\ell + \varepsilon_{\mathsf{MCA}} + \binom{B}{2}\frac{k-1}{|\mathbb{F}|-N} + \frac{B+1}{|\mathbb{F}|}.$$

Its oracle extractor runs in time

$$T_{\mathsf{Ext}} = T_{\mathsf{iRS}}(m,N,k,\delta) + O(Bmk)$$

.

B.2 Proof Ingredients

For $r \in \mathbb{F}^m$, define the folded word $\widehat{U}_r := \sum_{i=1}^m r_i \widehat{U}_i$. Mutual correlated agreement gives the following list consistency property.

Lemma B.3 (Consistency of the folded list)**.** *Suppose the identity generator has mutual correlated agreement for $\mathsf{RS}_{L,k}$ at radius δ, with error $\varepsilon_{\mathsf{MCA}}$. Then, except with probability $\varepsilon_{\mathsf{MCA}}$ over $r \xleftarrow{\$} \mathbb{F}^m$,*

$$\mathcal{L}_\delta^{\mathsf{coef}}(\widehat{U}_r) = \{r^\top U : U \in \mathcal{J}_\delta(\widehat{U})\}.$$

For every fixed slack below the Johnson radius, this property holds with an explicit error bound [12]; see also the list commutativity formulation used by WHIR [3, Lemma 4.13].

Lemma B.4 (Soundness of the sampled column test)**.** *Fix r and a vector $u_r \in \mathbb{F}^k$. If*

$$\Delta_L\big(\widehat{U}_r, \mathsf{Enc}_{\mathsf{row}}(u_r)\big) > \delta,$$

then the sampled column test with ℓ coordinates accepts with probability at most $(1-\delta)^\ell$.

Proof. The two words disagree on more than δN coordinates. A uniformly sampled set of ℓ coordinates, chosen without replacement, lies entirely in their agreement set with probability at most $(1-\delta)^\ell$. □

B.3 Selection by the DEEP Value

For $U \in \mathcal{J}_\delta(\widehat{U})$ and $\alpha \in \mathbb{F}\setminus L$, define $c_\alpha(U) := Ud_\alpha \in \mathbb{F}^m$. The next lemma captures the role of the commitment-phase DEEP values.

Lemma B.5 (Selection by an out-of-domain evaluation)**.** *Let $|\mathcal{J}_\delta(\widehat{U})| \le B$, and sample $\alpha \xleftarrow{\$} \mathbb{F}\setminus L$. Except with probability*

$$\varepsilon_{\mathsf{DEEP}} := \binom{B}{2}\frac{k-1}{|\mathbb{F}|-N},$$

the vectors $c_\alpha(U)$ are distinct over all $U \in \mathcal{J}_\delta(\widehat{U})$.

Fix this good event and any $c \in \mathbb{F}^m$. Suppose Lemma B.3 holds for a random r. Except with probability at most $B/|\mathbb{F}|$ over r, every $u_r \in \mathcal{L}_\delta^{\mathsf{coef}}(\widehat{U}_r)$ satisfying

$$\langle u_r, d_\alpha\rangle = \langle r, c\rangle$$

equals $r^\top U^\star$ for the unique candidate $U^\star \in \mathcal{J}_\delta(\widehat{U})$ satisfying $U^\star d_\alpha = c$. If no such candidate exists, no u_r satisfies the displayed equality outside the same error event.

Proof. For distinct $U, U' \in \mathcal{J}_\delta(\widehat{U})$, choose a row on which they differ. The difference of the corresponding row polynomials is nonzero and has degree less than k. It therefore has at most $k-1$ roots. A union bound over candidate pairs proves the first claim.

Under the conclusion of Lemma B.3, every nearby u_r has the form $r^\top U$ for a joint candidate U. Its DEEP check is equivalent to

$$\langle r, Ud_\alpha - c\rangle = 0.$$

For every candidate with $Ud_\alpha \ne c$, this is a nontrivial linear equation in a uniform r and holds with probability $1/|\mathbb{F}|$. The union bound over at most B candidates proves the second claim. The first part of the lemma ensures that at most one candidate satisfies $Ud_\alpha = c$. □

B.4 Proof of Theorem B.2

Proof. The committed oracle $\widehat{U}$ is fixed before α is sampled. The extractor enumerates $\mathcal{J}_\delta(\widehat{U})$, computes Ud_α for every candidate, and outputs the unique $U^\star$ satisfying $U^\star d_\alpha = c$, if it exists. List enumeration costs T_{iRS}, and evaluating all row polynomials costs $O(Bmk)$.

It remains to show that an accepting transcript yields the required relation, except with the stated error. If the vector U_r sent by the prover is farther than δ from $\widehat{U}_r$, the sampled column test accepts with probability at most $(1-\delta)^\ell$ by Lemma B.4. Otherwise, failure of folded-list consistency contributes at most $\varepsilon_{\mathsf{MCA}}$. Conditioned on neither event, $U_r = r^\top U$ for some $U \in \mathcal{J}_\delta(\widehat{U})$.

The DEEP collision event contributes at most $\binom{B}{2}(k-1)/(|\mathbb{F}|-N)$. Outside this event, Lemma B.5 shows, with additional error at most $B/|\mathbb{F}|$, that the DEEP check selects the unique candidate $U^\star$ with $U^\star d_\alpha = c$ and that $U_r = r^\top U^\star$.

The prover sends v before seeing r. If $v \neq U^\star q_2$, then

$$\langle U_r, q_2 \rangle = \langle r, v \rangle$$

is equivalent to the nontrivial equation $\langle r, U^\star q_2 - v \rangle = 0$, which holds with probability $1/|\mathbb{F}|$. Thus an accepting transcript outside the accumulated error satisfies $v = U^\star q_2$. The verifier also checks $\langle q_1, v \rangle = y$, so

$$q_1^\top U^\star q_2 = y.$$

Together with common agreement and $U^\star d_\alpha = c$, this proves that $((\widehat{U}, \alpha, c, q_1, q_2, y); U^\star)$ belongs to $\mathcal{R}_{\mathsf{Lift}}$. Summing the five error terms completes the proof. □

The theorem has an unconditional instantiation up to the Johnson radius with fixed slack. If the row code has rate ρ and $\delta = 1 - \sqrt{\rho} - \eta$ for fixed $\eta > 0$, then the interleaved list size satisfies

$$B \leq \frac{1}{2\eta\sqrt{\rho}},$$

and the list can be enumerated in polynomial time by Reed–Solomon list decoding [3, 28]. The mutual correlated agreement theorem of Bordage et al. [12] supplies $\varepsilon_{\mathsf{MCA}}$ for the same range with fixed slack.

Assumption B.6 (Efficient mutual correlated agreement at capacity)**.** Let $\mathbb{F} = \mathbb{F}_p$, let the row code have rate $\rho = k/N$, and fix $\eta > 0$ and δ such that

$$\rho + \frac{1}{N} + \eta \leq C_p(\delta).$$

For every $\widehat{U} \in \mathbb{F}^{m \times N}$, its joint decoding list has size at most B_{cap} and can be enumerated in time T_{cap}. Both quantities are polynomial in N, $1/\eta$, and $\log p$. Moreover, for $r \xleftarrow{\$} \mathbb{F}^m$,

$$\mathcal{L}_\delta^{\mathsf{coef}}(\widehat{U}_r) = \{r^\top U : U \in \mathcal{J}_\delta(\widehat{U})\}$$

except with probability $\varepsilon_{\mathsf{MCA,cap}} \leq \varepsilon_{\mathsf{cap}}$, where $\varepsilon_{\mathsf{cap}}$ is defined in Conjecture 2.4.

Assumption B.6 is the efficient mutual form of Conjecture 2.4. Under this assumption, Theorem B.2 applies to the conjectured capacity regime with $B = B_{\mathsf{cap}}$, $T_{\mathsf{iRS}} = T_{\mathsf{cap}}$, and $\varepsilon_{\mathsf{MCA}} = \varepsilon_{\mathsf{MCA,cap}}$.

Corollary B.7 (Compiled standalone PCS)**.** Apply a knowledge-preserving IOP compiler to the protocol from Theorem B.2, instantiate the oracle by a binding Merkle commitment, and apply Fiat–Shamir in the random-oracle model. Then the resulting standalone evaluation argument is perfectly complete and knowledge sound. Its knowledge error is at most

$$\kappa_{\mathsf{stand}} + \kappa_{\mathsf{comp}} + \mathsf{Adv}_H^{\mathsf{coll}}(\lambda),$$

where κ_{comp} is the compiler error and $\mathsf{Adv}_H^{\mathsf{coll}}$ is the collision-finding advantage against the Merkle hash function. This is the standard IOP compilation used by code-based polynomial commitments [9, 23].