SNARKs over Small Prime Fields without Extension Field Multiplication

Yuanju Wei[2,3], Kaixuan Wang[4], Binwu Xiang[5], Xinxuan Zhang[2,3], Yi Deng[1(✉)], Xudong Zhu[6], Hailong Wang[7], Li Lin[7], and Lei Wang[4]

[1] School of Cryptology, Xidian University, Xi'an, China
[2] State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[4] Shanghai Jiao Tong University
[5] East China Normal University
[6] Development and Strategy Department, China Mobile Yunnan Company
[7] Digital Technologies, Ant Group
{weiyuanju, zhangxinxuan}@iie.ac.cn, {wangkaixuan, wanglei_hb}@sjtu.edu.cn,bwxiang@sc.ecnu.edu.cn,ydeng.cas@gmail.com, zhuxudong@yn.chinamobile.com, whl383799@antgroup.com, felix.ll@alibaba-inc.com

Abstract. Sumcheck-based transparent SNARKs over small prime fields encounter a basic soundness bottleneck. When instantiated over $\mathbb{F}_p$, the round-by-round soundness error is $O(d/p)$, where d is the degree parameter in the sumcheck instance. This is non-negligible for small p. A common remedy is to work over an extension field, which departs from native $\mathbb{F}_p$ arithmetic and requires implementing general $\mathbb{F}_{p^k}$ arithmetic, in particular multiplication. A natural alternative is to amplify soundness by parallel repetition. However, naive k-parallel repetition does not achieve a proportional k-fold increase in round-by-round soundness for multi-round sumcheck.

In this work, we propose the *packed sumcheck protocol*, which amplifies round-by-round soundness over $\mathbb{F}_p$ by combining repetition with folding, while avoiding extension field multiplication. The core idea is to couple k independent instances within each round using random matrix challenges from the matrix ring $\mathbb{F}_p^{k\times k}$, so that every folding step mixes all k coordinates. For degree-d claims, we obtain round-by-round soundness error $\big(((2k-1)d)/p\big)^k$. The prover uses $O((kd^2+k^2d)N)$ base field multiplications and the verifier uses $O(k^2d\mu)$ base field operations, with communication $O(kd\mu)$ base field elements, where $N=2^\mu$ is the size of the μ-dimensional Boolean hypercube.

To compile the packed sumcheck protocol into SNARKs, we introduce *matrix-weighted multilinear extension (MW-MLE) commitments*, capturing the matrix-weighted linear forms induced by packed sumcheck, and instantiate them by adapting Brakedown (CRYPTO 2023) and Basefold (CRYPTO 2024). The resulting components are integrated into the HyperPlonk (EUROCRYPT 2023) framework by replacing its sumcheck and polynomial commitment modules with packed variants, obtaining

transparent SNARK instantiations without extension field multiplication in the Random Oracle Model. We further apply our packed techniques to construct a TFHE-friendly SNARK; we evaluate its prover performance for a single bootstrapping, obtaining a prover time of 2.2 seconds.

Keywords: SNARKs · sumcheck · small prime fields.

1 Introduction

Succinct Non-interactive Arguments of Knowledge (SNARKs) [13, 36, 39, 44, 50] have become a standard tool for verifiable computation and related applications. Their constructions are typically obtained by combining a *Polynomial Interactive Oracle Proof* (PIOP) with a *Polynomial Commitment Scheme* (PCS), and then applying the Fiat-Shamir transformation to remove interactions [8–10, 37, 45, 54, 59–61]. In this paradigm, a PIOP reduces an algebraic statement to a small number of polynomial openings at verifier-chosen points, while the PCS enables the verifier to obtain these evaluations succinctly.

For Fiat-Shamir compiled SNARKs, a central security requirement is *round-by-round soundness* [19]: for any false statement, no cheating prover can extend any prefix history to a complete accepting transcript, except with negligible probability. For the PIOPs based on sumcheck protocols [48], round-by-round soundness is ultimately bounded by the Schwartz-Zippel lemma: a nonzero polynomial of degree d over a field $\mathbb{F}$ vanishes at a uniformly random point with probability at most $d/|\mathbb{F}|$. Thus, to achieve negligible soundness error, the field size $|\mathbb{F}|$ needs to be significantly larger than the degree d of the polynomials involved in the PIOP.

Motivation. Recently, there has been increasing research interest in SNARKs over small prime fields (e.g., 32-bit or 64-bit prime fields, such as BabyBear and Goldilocks). This is driven by the deployability of native small prime field arithmetic: modular addition and multiplication fit well into machine-word operations, admit SIMD parallelism, and integrate cleanly with NTT-friendly evaluation domains and code-based proximity machinery used in many transparent SNARK designs [2, 3, 8–10, 37, 60, 61] . Moreover, several practical applications naturally operate over small prime fields, including verifiable CPU computation and verifiable fully homomorphic encryption (vFHE) [4, 20, 35, 46, 55, 56, 63]. However, small fields exacerbate the Schwartz-Zippel limitation: the field size $|\mathbb{F}_p| = p$ is not substantially larger than typical polynomial degrees d in PIOPs, leading to weak soundness guarantees.

A standard resolution is to execute the PIOP over an extension field $\mathbb{F}_{p^k}$ (with $p^k \approx 2^\lambda$ for λ-bit security), while keeping the witness representation and commitments over the base field $\mathbb{F}_p$ [30, 31]. This restores soundness, but introduces a persistent performance conflict: multiplication in $\mathbb{F}_{p^k}$ is substantially more expensive than multiplication in $\mathbb{F}_p$ and is often harder to optimize on commodity hardware. Recent works have sought to mitigate the extension field

overhead from both sides: on the PIOP side, algebraic techniques can replace some extension multiplications by structured collections of base field multiplications [6, 7, 27, 28, 40]; on the PCS side, highly efficient polynomial commitment schemes over small prime fields that support extension field openings have been developed [17, 30, 31]. Nevertheless, these approaches still rely on extension field computations (particularly multiplications) to achieve soundness, and fully avoiding extension field computations without incurring prohibitive overhead remains challenging.

A natural alternative is to amplify soundness via *parallel repetition*. However, as pointed out in [7], a k-fold parallel repetition of the μ-round sumcheck protocol does not achieve round-by-round soundness error $O\big((d/p)^k\big)$, but only $O\big((d/p)^{\lceil k/\mu \rceil}\big)$. Consequently, to obtain negligible soundness error, one needs an extremely large number of repetitions k (since μ is typically ≥ 20), which is impractical. This motivates the following question:

Can we design a practical SNARK over small prime fields without relying on field extension multiplication to achieve round-by-round soundness?

1.1 Our Contributions

We answer this question positively. Our key idea is a new *folding-with-repetition* mechanism that amplifies round-by-round soundness *within each round* over $\mathbb{F}_p$, avoiding extension field multiplications.

- **Packed sumcheck over small prime fields.** We propose *packed sumcheck*, a new sumcheck protocol based on folding with repetition. Given k independent degree-d sumcheck statements over $\mathbb{F}_p$ on $\log N$ variables, our protocol packs them into a single instance over the matrix ring $\mathbb{F}_p^{k\times k}$ and achieves round-by-round soundness error $O\left(\left(\frac{d(2k-1)}{p}\right)^k\right)$. Crucially, the prover performs *only* $\mathbb{F}_p$ operations, with complexity

$$O\big((kd^2 + k^2 d)\, N\big) \cdot \mathsf{mul} \;+\; O\big(k^2 d^2 N\big) \cdot \mathsf{add},$$

  where $\mathsf{mul}, \mathsf{add}$ denote operations in $\mathbb{F}_p$. When instantiated over BabyBear, packed sumcheck achieves a 3.25×–4.03× speedup for $k = 4$ and a 2.78×–3.36× speedup for $k = 5$ over their fastest BabyBear baseline in [7], which evaluates the PIOP over a degree-4 extension field to obtain comparable soundness.
- **Commitments for matrix-weighted multilinear extensions.** Instantiating packed sumcheck requires polynomial commitments that support *matrix-weighted* evaluations of multilinear extensions (MLEs). We provide two instantiations derived from the Brakedown [37] and Basefold [61] frameworks. Both instantiations keep the prover in $\mathbb{F}_p$ while supporting matrix-weighted openings. The additional $k \times k$ matrix weighting is amortized over the k packed instances, so the asymptotic cost per instance matches that of the corresponding small field schemes supporting extension field openings.

Additionally, we benchmark both instantiations over BabyBear and compare them with the corresponding small field schemes supporting extension field openings to show that they achieve nearly the same concrete performance.

We demonstrate the versatility of these components by instantiating the HyperPlonk [21] PIOP with packed variants of sumcheck and ZeroCheck, together with our matrix-weighted MLE commitments, obtaining a SNARK over small prime fields without extension field computations. We further specialize packed sumcheck and our Brakedown-based matrix-weighted MLE commitment to a TFHE-friendly SNARK. We evaluate its prover performance for a single TFHE bootstrapping, obtaining a prover time of 2.2 seconds.

1.2 Technical Overview

This overview presents the main ideas of our construction. We first revisit the limitation of standard parallel repetition, then introduce packed sumcheck and the supporting commitment machinery.

Parallel repetition sumcheck revisited. Let $N = 2^\mu$, write $B_\mu := \{0,1\}^\mu$, and consider k parallel repetitions of a μ-round sumcheck instance over a small prime field $\mathbb{F}_p$. Throughout this overview, a degree-d sumcheck instance refers to the structured case

$$F_{0,j} = \prod_{s=0}^{d-1} f_{0,j}^{(s)},$$

where each factor $f_{0,j}^{(s)}$ is multilinear; hence $F_{0,j}$ has individual degree at most d. In standard parallel repetition, the same set of factors is used across j. Before the protocol, we have k claims of the form

$$\sum_{b\in\{0,1\}^\mu} F_{0,j}(b) = H_{0,j}, \qquad j \in [0, k-1].$$

In each round i, the verifier's running claim is a vector $H_i = (H_{i,0}, \ldots, H_{i,k-1}) \in \mathbb{F}_p^k$ satisfying

$$\sum_{b\in\{0,1\}^{\mu-i}} F_{i,j}(b) = H_{i,j}, \qquad j \in [0, k-1],$$

the prover responds with a vector of degree-d univariate polynomials $g_i(X) = (g_{i,0}(X), ..., g_{i,k-1}(X))$. The verifier checks $g_i(0) + g_i(1) = H_i$ and updates $H_{i+1} = (g_{i,0}(r_{i,0}), ..., g_{i,k-1}(r_{i,k-1}))$ for random challenges $(r_{i,0}, ..., r_{i,k-1}) \leftarrow \mathbb{F}_p^k$.

The key obstacle is *coordinate-wise cheating* [5,7]. The protocol has μ rounds, so the prover can distribute the k inconsistent coordinates across the rounds instead of fixing them all at once. By averaging, there exists a round in which it needs to "fix" at most $\lceil k/\mu \rceil$ coordinates. Fixing one coordinate in a round succeeds only if the forged polynomial happens to agree with the verifier's random point for that coordinate, which occurs with probability about d/p. Thus, the probability of successfully fixing $\lceil k/\mu \rceil$ coordinates handled in that round is about $(d/p)^{\lceil k/\mu \rceil}$.

Packed sumcheck: coupling coordinates via matrix mixing. The above attack suggests that the real weakness is that the k coordinates are treated independently within each round. To strengthen security, we want to treat the k coordinates as a single object, so that a prover cannot "fix a few coordinates now and the rest later."

The first attempt is to regard $\mathbb{F}_p^k$ as a direct-product ring $(\mathbb{F}_p^k, +, \odot)$, where addition is componentwise and multiplication $\odot$ is the Hadamard product. Running sumcheck over this ring is the same as running k independent sumcheck protocols over $\mathbb{F}_p$ in parallel: each ring element is a k-tuple, each ring polynomial is a k-tuple of polynomials, and every ring check decomposes into k coordinate checks. This attempt does not change the protocol; however, it provides insight into the next attempt, where we introduce matrix ring to couple coordinates round by round.

We change the update rule by sampling *matrix-valued* challenges from a matrix ring $(\mathbb{F}_p^{k\times k}, +, \cdot)$, where $+$ and $\cdot$ are standard matrix addition and multiplication. The matrix ring acts on $\mathbb{F}_p^k$ by left multiplication (viewing vectors as columns), so the update mixes coordinates. Next, we describe one round of the protocol and show how to achieve soundness amplification via matrix mixing.

At the beginning of round i, the j-th coordinate of the folded state claims

$$\sum_{X_i,\ldots,X_{\mu-1}\in\{0,1\}} \prod_{s=0}^{d-1} f_{i,j}^{(s)}(X_i,\ldots,X_{\mu-1}) = H_{i,j}, \qquad j \in [0,k-1].$$

Although the preceding parallel-repetition discussion used the standard case where the factors are shared across the k copies, the packed protocol itself does not rely on this restriction. The polynomials $f_{i,j}^{(s)}$ may therefore differ across j, allowing the protocol to pack k distinct sumcheck instances in a single execution.

Let $u \in \{0,1\}^{\mu-i-1}$ be an assignment to the suffix variables $(X_{i+1},\ldots,X_{\mu-1})$. Inspired by the SumFold technique introduced in NeutronNova [45], we pack these sumcheck instances into a single instance. In our adaptation, the interpolation is expressed in the Lagrange basis. Specifically, for each factor $s \in [0, d-1]$, we define a univariate polynomial $\psi_{i,u}^{(s)}(Y)$ of degree at most $2k-1$ that encodes the two Boolean-point evaluations of each of the k instances at the $2k$ interpolation points $\{0,1,\ldots,2k-1\}$. We assume that these interpolation points are pairwise distinct in $\mathbb{F}_p$. Let $\{\mathcal{L}_t(Y)\}_{t=0}^{2k-1}$ denote the Lagrange basis associated with $\{0,1,\ldots,2k-1\}$; that is, $\mathcal{L}_t(t) = 1$ and $\mathcal{L}_t(t') = 0$ for all $t' \neq t$ in this set. Define the $\leq 2k-1$-degree univariate polynomial:

$$\psi_{i,u}^{(s)}(Y) := \sum_{j=0}^{k-1} \Big(\mathcal{L}_{2j}(Y)\, f_{i,j}^{(s)}(0,u) \;+\; \mathcal{L}_{2j+1}(Y)\, f_{i,j}^{(s)}(1,u) \Big),$$

so that $\psi_{i,u}^{(s)}(2j+b) = f_{i,j}^{(s)}(b,u)$ for all $j \in [0,k-1]$ and $b \in \{0,1\}$.

The prover's message in round i is the polynomial

$$P_i(Y) \;:=\; \sum_{u\in\{0,1\}^{\mu-i-1}} \prod_{s=0}^{d-1} \psi_{i,u}^{(s)}(Y),$$

which has degree at most $(2k-1)d$ and satisfies

$$P_i(2j+b) = \sum_{u\in\{0,1\}^{\mu-i-1}} \prod_{s=0}^{d-1} f_{i,j}^{(s)}(b,u) \qquad \text{for all } j\in[0,k-1],\ b\in\{0,1\}.$$

Upon receiving $P_i(Y)$, the verifier checks all k boundary constraints at once by verifying $\big(P_i(0)+P_i(1),\ldots,P_i(2k-2)+P_i(2k-1)\big) = H_i \in \mathbb{F}_p^k$. It then samples $(r_{i,0},\ldots,r_{i,k-1}) \leftarrow \mathbb{F}_p^k$ and sets $H_{i+1} = (P_i(r_{i,0}),\ldots,P_i(r_{i,k-1}))$.

The update $H_{i+1} = (P_i(r_{i,0}),\ldots,P_i(r_{i,k-1}))$ induces two $k\times k$ *challenge matrices* $M_i^{(0)}, M_i^{(1)} \in \mathbb{F}_p^{k\times k}$. For $b\in\{0,1\}$, define

$$M_i^{(b)}[a,c] \ :=\ \mathcal{L}_{2c+b}(r_{i,a}), \qquad a,c\in[0,k-1].$$

Pack the evaluation tables of $f_{i,j}^{(s)}$ on $B_{\mu-i}$ into a vector $F_i^{(s)} \in (\mathbb{F}_p^k)^{2^{\mu-i}}$ as:

$$F_i^{(s)}(b_i,\ldots,b_{\mu-1}) = (f_{i,0}^{(s)}(b_i,\ldots,b_{\mu-1}),\ldots,f_{i,k-1}^{(s)}(b_i,\ldots,b_{\mu-1})),$$

then the transition to the next round is exactly a matrix-mixing fold:

$$F_{i+1}^{(s)}(u) = M_i^{(0)}\cdot F_i^{(s)}(0,u) + M_i^{(1)}\cdot F_i^{(s)}(1,u), \qquad u\in\{0,1\}^{\mu-i-1}.$$

This is the central departure from the direct-product view: since $P_i(r_{i,a})$ is a non-trivial linear combination of *all* $2k$ boundary values $\{f_{i,j}^{(s)}(b,u)\}_{j,b}$ via the Lagrange weights $\mathcal{L}_{2j+b}(r_{i,a})$, *every next-round coordinate depends on all k previous coordinates, with coefficients determined by fresh randomness.* As a result, the prover can no longer isolate a single coordinate and repair it independently.

For soundness in round i, suppose the prover sends $\tilde{P}_i \neq P_i$ of degree $\le (2k-1)d$ that still passes the boundary check. Then $\Delta(Y) := \tilde{P}_i(Y) - P_i(Y)$ is a nonzero polynomial of degree $\le (2k-1)d$. For the next state to be consistent, the prover must satisfy $\Delta(r_{i,0}) = \cdots = \Delta(r_{i,k-1}) = 0$.

By Schwartz–Zippel and independence of $r_{i,j}$,

$$\Pr[\Delta(r_{i,0}) = \cdots = \Delta(r_{i,k-1}) = 0] \ \le\ \left(\frac{(2k-1)d}{p}\right)^k.$$

Thus, a deviation in round i must make Δ vanish on k independently sampled points, giving error at most $\big((2k-1)d/p\big)^k$.

Matrix-Weighted multilinear extensions. The folding process of classical sumcheck is closely related to the folding evaluation process of multilinear extensions (MLEs). Since the folding process of packed sumcheck introduces matrix mixing, the traditional algebraic structure of MLEs is no longer sufficient to describe the modified sumcheck protocol. We need a new algebraic structure to describe the folding evaluation process of packed sumcheck.

Packed sumcheck provides a clear structure for the folding process:

$$F_{i+1}^{(s)}(u) = M_i^{(0)} \cdot F_i^{(s)}(0,u) + M_i^{(1)} \cdot F_i^{(s)}(1,u), \qquad u \in \{0,1\}^{\mu-i-1}.$$

The next round coordinates are linear combinations of the previous round coordinates, with coefficients determined by the random challenge matrices $M_i^{(0)}, M_i^{(1)}$. We continue this process until the last round, and by unfolding the folding structure, we obtain a final folded evaluation:

$$F_\mu^{(s)}(\emptyset) = \sum_{b\in\{0,1\}^\mu} W_b \cdot F_0^{(s)}(b),$$

where $W_b = \prod_{i=\mu-1}^{0} \left((1-b_i)M_i^{(0)} + b_i M_i^{(1)}\right) \in \mathbb{F}_p^{k\times k}$.

Because matrix multiplication is non-commutative, we must define W_b in the order from $\mu-1$ to 0. This final folded evaluation is a matrix-weighted linear combination, where each weight W_b is a matrix product of the random challenge matrices, and each weighted value $F_0^{(s)}(b)$ is a vector. The definition and computation of this weight matrix W_b are analogous to those of the classical MLE, except that the scalar, commutative weights are replaced by ordered matrix products. In classical MLE, each variable chooses $1-r_i$ or r_i as the weight based on the random challenge r_i, and then takes the product of all variable weights to get the final weight. In the new structure, each variable chooses $M_i^{(0)}$ or $M_i^{(1)}$ as the weight based on b_i, and then takes the ordered matrix product of all variable weights to get the final weight. We call this new algebraic structure *matrix-weighted multilinear extension* (MW-MLE):

$$E^{(s)}(X_0^{(0)}, X_0^{(1)}, \ldots, X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = \sum_{b\in\{0,1\}^\mu} \left(\prod_{i=\mu-1}^{0} \left((1-b_i)X_i^{(0)} + b_i X_i^{(1)}\right) \right) \cdot F^{(s)}(b),$$

Here each variable $X_i^{(b)}$ ranges over $\mathbb{F}_p^{k\times k}$, and the output lies in $\mathbb{F}_p^k$ via matrix–vector multiplication. The computations in $E^{(s)}$ are all in $\mathbb{F}_p$. Let $B_\mu^{(\mathsf{mat})} := \{L,R\}^\mu$ denote the matrix-pair hypercube, where $L := (I, \mathbf{0})$ and $R := (\mathbf{0}, I)$ are correlated pairs of matrices, and I and $\mathbf{0}$ are the $k \times k$ identity and zero matrices, respectively. With these definitions, the statement to be proved can be rewritten as follows:

$$\sum_{(x_0^{(0)},x_0^{(1)}),\ldots,(x_{\mu-1}^{(0)},x_{\mu-1}^{(1)})\in B_\mu^{(\mathsf{mat})}} \bigodot_{s=0}^{d-1} E^{(s)}(x_0^{(0)}, x_0^{(1)}, \ldots, x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)}) \;=\; H \in \mathbb{F}_p^k,$$

One point worth noting is that the new statement is not a pure sumcheck statement over MW-MLEs, because the outermost operation is the coordinate-wise Hadamard product in the output module, rather than multiplication in the matrix ring. However, from the perspective of matrix rings, we can clearly describe the structure and properties of MW-MLEs and their role in packed sumcheck.

Commitments for matrix-weighted multilinear extensions. Packed sum-check reduces the final verification step to opening MW-MLEs at a transcript-defined point:

$$E^{(s)}(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}) = \sum_{b\in\{0,1\}^\mu} W_b \cdot F^{(s)}(b).$$

We need a PCS that certifies such structured linear combinations efficiently. We present two instantiations based on Brakedown [37] and Basefold PIOPs [61], respectively.

A coding ingredient: block Reed–Solomon without extension multiplication. Both of our instantiations build on code-based PCS frameworks, so we need an encoding procedure that avoids general $\mathbb{F}_{p^k}$ multiplication. We adopt a block-level [30] Reed–Solomon encoding: we group an $\mathbb{F}_p$-vector into blocks of size k and view each block as one symbol in $\mathbb{F}_{p^k}$. We then encode by evaluating the corresponding polynomial on an NTT-friendly evaluation set $W \subset \mathbb{F}_p$. Since all evaluation points lie in the base field, the evaluation step uses only multiplications of the form "$\mathbb{F}_{p^k}$-symbol $\times$ $\mathbb{F}_p$-scalar"; each such multiplication expands into k independent multiplications in $\mathbb{F}_p$. Viewed as a Reed–Solomon code over $\mathbb{F}_{p^k}$, the resulting code has rate α (i.e., message length $\alpha|W|$ over block length $|W|$) and relative distance $1-\alpha$.

Brakedown-style commitment. Write $\mu = 2m$ for simplicity. Reshape the hypercube table $F^{(s)} \in (\mathbb{F}_p^k)^{2^\mu}$ into a matrix

$$U \in (\mathbb{F}_p^k)^{2^m \times 2^m}, \qquad U_{b_L,b_R} := F^{(s)}(b_L, b_R),$$

where $b_L, b_R \in \{0,1\}^m$ denote the low-index and high-index halves of b.

Now split the matrix weights accordingly. Define

$$W_{b_L}^L := \prod_{i=m-1}^{0} \left((1-b_{L,i})M_i^{(0)} + b_{L,i}M_i^{(1)}\right) \in \mathbb{F}_p^{k\times k},$$

$$W_{b_R}^R := \prod_{i=2m-1}^{m} \left((1-b_{R,i-m})M_i^{(0)} + b_{R,i-m}M_i^{(1)}\right) \in \mathbb{F}_p^{k\times k}.$$

Because the product defining W_b is ordered from $\mu-1$ down to 0, we get the exact factorization $W_{(b_L,b_R)} = W_{b_R}^R \cdot W_{b_L}^L$. Substituting into the desired opening gives a two-stage form:

$$\begin{aligned} E^{(s)}(\mathbf{M}) &= \sum_{b_R\in\{0,1\}^m} \sum_{b_L\in\{0,1\}^m} \left(W_{b_R}^R W_{b_L}^L\right) \cdot U_{b_L,b_R} \\ &= \sum_{b_R\in\{0,1\}^m} W_{b_R}^R \cdot \underbrace{\left(\sum_{b_L\in\{0,1\}^m} W_{b_L}^L \cdot U_{b_L,b_R}\right)}_{=:\ A_{b_R}\in\mathbb{F}_p^k}. \end{aligned}$$

Thus it suffices to (i) certify the intermediate vector $A \in (\mathbb{F}_p^k)^{2^m}$ with entries A_{b_R}, and then (ii) let the verifier compute $\sum_{b_R} W_{b_R}^R \cdot A_{b_R}$ locally.

This matches the kind of linear claim supported by Brakedown: the prover commits to a matrix and then proves a verifier-specified linear aggregation of its rows. Define the block row vector (block matrix) that encodes the left-half weights as $S_0 := \left[W_{b_L}^L\right]_{b_L \in \{0,1\}^m} \in \mathbb{F}_p^{k \times (k \cdot 2^m)}$. If we expand U into base field coordinates by viewing each $\mathbb{F}_p^k$ entry as a length-k block, then U becomes a matrix in $\mathbb{F}_p^{(k \cdot 2^m) \times 2^m}$, and the intermediate aggregation is exactly the matrix product $A = S_0 \cdot U \in \mathbb{F}_p^{k \times 2^m}$, which we parse back into $A_{b_R} \in \mathbb{F}_p^k$ per column. Brakedown's evaluation phase certifies such linear claims $A = S_0 U$ using a linear-code encoding and spot checks. Then the verifier computes the final opening as $E^{(s)}(\mathbf{M}) = \sum_{b_R \in \{0,1\}^m} W_{b_R}^R \cdot A_{b_R}$.

Encoding rows is performed with a block Reed–Solomon encoding, so no extension multiplication is needed in the commit phase. And all matrix multiplications are over $\mathbb{F}_p$, so no extension multiplication is needed in the evaluation phase either.

For the efficiency analysis of MW-Brakedown, we provide a theoretical analysis in Section 4.1 and Appendix C.1, showing that its complexity matches that of Binius (Brakedown with block-level encoding) [30]. In Section 6.1, we evaluate the concrete efficiency of the two schemes over the BabyBear field, and our results show that MW-Brakedown incurs only minor overhead compared to the block-level encoding version of Brakedown.

Basefold-style commitment. Basefold provides an IOP of proximity for foldable linear codes and runs it in lockstep with sumcheck by reusing the same folding challenges [61]. Packed sumcheck has an explicit folding trajectory (the matrices $(M_i^{(0)}, M_i^{(1)})$), so we adapt Basefold to certify MW-MLE openings by making the code folding mirror the matrix folding.

At a high level, the commitment is to a matrix

$$U^{(\mu)} \in \mathbb{F}_p^{k \times 2^\mu}, \qquad U_{\cdot,j}^{(\mu)} = F^{(s)}(\text{the } j\text{-th Boolean index}),$$

i.e., columns are the packed $\mathbb{F}_p^k$ values. We pack each row into $\mathbb{F}_{p^k}$-symbols and encode each packed row with a foldable code family $\{C_i\}_{i=0}^{\mu}$, committing by Merkle trees of the codeword columns.

The algebraic heart is the foldable-code recursion:

$$G_i = \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1} \cdot T_{i-1} & G_{i-1} \cdot T'_{i-1} \end{bmatrix},$$

where $T_{i-1}, T'_{i-1} \in \mathbb{F}_{p^k}^{n_{i-1} \times n_{i-1}}$ are diagonal matrices satisfying $\mathsf{diag}(T_{i-1})[j] \neq \mathsf{diag}(T'_{i-1})[j]$ for all $j \in [n_{i-1}]$. This structure implies that any codeword at level i can be seen as concatenation of two "children" that share a common pair of latent components. Concretely, for each column index j at level $i-1$, the two parent columns $a, b \in \mathbb{F}_{p^k}^k$ satisfy $a = x + ty, b = x + t'y$ for uniquely determined

$(x, y) \in (\mathbb{F}_{p^k}^k)^2$ and known (t, t') (the corresponding diagonal entries of T, T'). This allows the verifier to test folding consistency by checking that the claimed folded column equals a linear combination of (x, y).

Now we align this with packed sumcheck. In round i, the packed sumcheck transcript defines matrices $(M_i^{(0)}, M_i^{(1)}) \in (\mathbb{F}_p^{k\times k})^2$. The Basefold-style fold sets the next-level column to $\hat{U}_{\cdot,j}^{(\mu-i)} := M_i^{(0)} \cdot x + M_i^{(1)} \cdot y$, where (x, y) are recovered from the two parent columns (a, b) as above. Because $M_i^{(0)}, M_i^{(1)}$ lie in $\mathbb{F}_p^{k\times k} \subset \mathbb{F}_{p^k}^{k\times k}$, this folding uses only base field defined coefficients, and can be executed efficiently.

The verifier performs $Q = \Theta(\lambda)$ random column queries per round: it opens the folded column and its two parents with Merkle paths, recomputes (x, y) from (a, b) and (t, t'), and checks the folding equation

$$\hat{U}_{\cdot,j}^{(\mu-i)} \stackrel{?}{=} M_i^{(0)} x + M_i^{(1)} y.$$

A direct recovery formula is $y = (a - b) \cdot (t_j - t'_j)^{-1}, x = a - t_j\, y$. This would be expensive if $(t_j - t'_j)^{-1}$ required arithmetic in $\mathbb{F}_{p^k}$. We avoid this by ensuring that $t_j, t'_j \in \mathbb{F}_p$. This holds in our block-level RS encoding, since the evaluation points lie in $W \subseteq \mathbb{F}_p$ and the induced folding scalars lie in $\mathbb{F}_p$ as well.

To argue soundness, we must handle k correlated rows, since packed sumcheck couples the k coordinates. We therefore view the committed object as an interleaving of k codewords and measure distance column-wise. For a matrix $A \in \mathbb{F}_p^{k\times n}$ and a code $C \subseteq \mathbb{F}_p^n$, define

$$\Delta_{\mathrm{blk}}(A, C^{\times k}) := \min_{C' \in C^{\times k}} \frac{1}{n} \cdot \left|\left\{j \in [n] : A_{\cdot,j} \neq C'_{\cdot,j}\right\}\right|.$$

This distance counts the fraction of column positions where at least one row is inconsistent. Using this metric, we prove that the Basefold proximity checks remain sound for the interleaving needed by packed sumcheck, and that the committed table stays consistent with the folding trajectory determined by the packed transcript.

For the efficiency analysis of MW-Basefold, we provide a theoretical analysis in Appendix D.5, showing that its complexity matches that of FRI-Binius (Basefold with block-level encoding) [31]. In Section 6.1, we evaluate the efficiency of the two schemes over the BabyBear field, and our results show that MW-Basefold incurs only minor overhead compared to the block-level encoding version of Basefold.

SNARKs over Small Prime Fields. We combine packed sumcheck with MW-MLE commitments to obtain SNARKs over small prime fields, without requiring extension field multiplication in the prover. Our construction follows the HyperPlonk [21] PIOP framework, and replaces its sumcheck components by packed variants.

Packed ZeroCheck. In HyperPlonk, a central subclaim is a ZeroCheck statement asserting that a low-degree polynomial vanishes on the Boolean hypercube. We provide a packed ZeroCheck protocol by reducing k such vanishing claims to a single packed sumcheck instance on the same domain. This reduction preserves the k-fold round-by-round soundness amplification over $\mathbb{F}_p$.

HyperPlonk instantiation. We keep the standard HyperPlonk structure (gate constraints and permutation constraints), but instantiate its sumcheck and ZeroCheck parts with packed sumcheck and packed ZeroCheck, and use MW-MLE commitments for the resulting openings. For the permutation constraints, HyperPlonk offers two variants. We use the variant whose core check is sumcheck-based, since it is directly compatible with packed sumcheck and avoids an extra multiplicative layer that would increase the degree and weaken soundness over small fields. The resulting SNARK avoids extension field multiplication.

TFHE-friendly constructions. We defer a TFHE [24]-friendly SNARK to Appendix H. We use TFHE only as a concrete evaluation target, to show that our techniques have practical impact on verifiable FHE.

1.3 Related Work

SNARKs over Small Fields. A growing line of work studies SNARKs over small prime fields, motivated by fast native arithmetic and by the role of small field IOPs in transparent SNARKs. Early theoretical progress includes multi-sumcheck protocols based on tensor codes and code-switching by Ron-Zewi and Rothblum [53] and Holmgren and Rothblum [42].

Alongside these theoretical advances, a growing body of work targets practicality. On the PCS side, Binius [30], FRI-Binius [31], and Blaze [17] provide polynomial commitments over small fields with openings in extension fields, supporting the common approach of running sumcheck over an extension field to obtain negligible soundness error while keeping the witness over the base field. On the PIOP side, several works then optimize sumcheck and its special case ZeroCheck in this setting: Gruen [40] accelerates extension field sumcheck via univariate skip, and Bagad et al. [7] reduce extension field multiplications by replacing some of them with structured base field computations. A separate line optimizes ZeroCheck using the structure of the equality function [27, 28]. In a more specialized direction, Hu et al. [43] and Xie and Liu [58] present systems tailored to parallel Boolean circuits.

More broadly, Dao et al. [25] study general sumcheck prover optimizations, including a monomial-basis formulation of the protocol and related implementation improvements. This direction is largely orthogonal to our focus on round-by-round soundness amplification over small prime fields.

Concurrent work. Recently, a *concurrent work* by Liu and Zhang [47] offered a helpful refinement by reducing the number of extension field multiplications

through optimized challenge design, thereby improving prover efficiency in settings where the extension factor is a power of two. Another concurrent work by Bagad, Dao, Domb, and Thaler [6] further optimizes their previous works [7,28] and presents an efficient ZeroCheck protocol that reduces the number of extension field multiplications. Continuing this line of work, Dao, DeStefano, Bagad, Domb, and Thaler [26] further optimize sumcheck proving by exploiting small-value arithmetic and structural properties such as high-degree product structure and decomposable equality-polynomial factors, reducing the cost of expensive large-field operations in sumcheck and ZeroCheck.

In contrast, we amplify round-by-round soundness directly over the base field via folding with repetition, without moving sumcheck to an extension field. As a result, the prover keeps all witness arithmetic in $\mathbb{F}_p$ and avoids extension field multiplication. This gives a complementary and modular approach to round-by-round soundness over small fields.

Verifiable FHE. Recent advances in SNARKs have made them a central tool for constructing verifiable FHE [9,10,17,23,34,37,54,59–61]. Several works have explored this direction. Fiore et al. [33] and Bois et al. [15] use commit-and-prove SNARKs to prove basic FHE ciphertext operations. Viand et al. [56] benchmark mainstream SNARKs for proving the BGV scheme [16]. Other works propose proof systems specialized to the algebraic structure of FHE. Atapoor et al. [4] employ double-CRT arithmetic and a lattice-based SNARK, while Ganesh et al. [35] use Rinocchio, a SNARK designed for ring arithmetic; both constructions are based on the Linear PCP model and therefore require a trusted setup. In a different direction, Cascudo et al. [20] propose modifying the ciphertext structure to enable decomposition into SNARK-friendly Galois fields. Complementary to this SNARK-based line of work, Zhou et al. [63] demonstrate proofs of BGV [16] and CKKS [22] using VOLE-based zero-knowledge protocols. None of these works specifically addresses the verification of TFHE bootstrapping, and thus their focus is largely orthogonal to ours.

The problem of verifying TFHE bootstrapping [24] has recently received attention. Thibault and Walter [55] present a complete proof for TFHE bootstrapping using the Plonky2 system, instantiating both the FHE and SNARK components over the 64-bit Goldilocks field. Their construction relies on IVC to recursively prove the blind rotation, at the CMUX level. More recently, Liu et al. [46] propose Hasteboots, which constructs a proof system for bit-wise bootstrapping using the Brakedown polynomial commitment scheme together with the sumcheck protocol. Their scheme operates over the Babybear field and requires a 4-fold field extension to achieve soundness.

Our work differs from these approaches in two fundamental respects. First, we introduce the packed sumcheck protocol, which allows all computations to be performed directly in the base field. Second, we simplify the structure of TFHE bootstrapping through parameter selection and reduce the proof of bootstrapping to proving vector relations.

Outline-and-batch Protocol. At a high level, our folding-with-repetition approach is conceptually reminiscent of the "outline-and-batch" paradigm of Bitansky et al. [14]. Their work is developed in the context of showing that PPAD is as hard as LWE, whereas our setting and techniques are different. We developed our construction independently of their work.

2 Preliminaries

We denote a finite field by $\mathbb{F}$, the security parameter by λ, and a negligible function in λ by $\mathsf{negl}(\lambda)$. We use PPT to denote probabilistic polynomial time. For any field $\mathbb{F}$ and integer n, we write $\mathbb{F}[n]$ for the set of polynomials in n variables with coefficients in $\mathbb{F}$. For any vector $\mathbf{a}$, we use $\mathbf{a}[i]$ to denote its i-th entry. We use $\circ$ to denote the Hadamard product. We use $[0,k]$ to denote the set $\{0,1,\ldots,k\}$ and $[0,k)$ to denote the set $\{0,1,\ldots,k-1\}$. For a vector v, we let $v_{[a,b]}$ denote the subvector $(v_a,\ldots,v_b)$.

2.1 Round-by-Round Soundness

Definition 1 (Round-by-Round Soundness [19]). *Let $\Pi = (P,V)$ be a $2r$-message public-coin interactive proof system for a language L. For any $x \in \{0,1\}^*$ and any transcript prefix τ, let $V(x,\tau)$ denote the distribution of the next message (or output) of V when the transcript so far is τ and V is executed on input x.*

We say that Π has **round-by-round soundness** *error $\epsilon(\lambda)$ if there exists a deterministic (not necessarily efficiently computable) function* State *that takes as input an instance x and a transcript prefix τ and outputs either* accept *or* reject *such that the following properties hold:*

1. *If $x \notin L$, then* $\mathrm{State}(x,\emptyset) = \mathrm{reject}$*, where $\emptyset$ denotes the empty transcript.*
2. *If* $\mathrm{State}(x,\tau) = \mathrm{reject}$ *for a transcript prefix τ, for every potential prover message α, it holds that* $\Pr_{\beta \leftarrow V(x,\tau|\alpha)}\left[\mathrm{State}(x,\tau|\alpha|\beta) = \mathrm{accept}\right] \le \epsilon(\lambda)$.
3. *For every full transcript τ, if* $\mathrm{State}(x,\tau) = \mathrm{reject}$ *then* $V(x,\tau) = 0$.

We say that Π is round-by-round sound *if it has round-by-round soundness error $\epsilon(\lambda)$ for some $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.*

2.2 SNARKs

Let $\mathcal{R}$ be an efficiently computable binary relation which consists of pairs of the form (x,w) where x is a statement and w is a witness. Let $\mathbf{L}$ be the language associated with the relation $\mathcal{R}$, i.e.$\mathcal{L} = \{x | \exists w, \text{s.t.} \mathcal{R}(x,w) = 1\}$.

A proof or argument system for R consists in a triple of PPT algorithms $\Pi = (\mathbf{Setup}, \mathbf{Prove}, \mathbf{Verify})$ defined as follows:

- $\mathrm{Setup}(1^\lambda) \to (\sigma, \mathrm{vk})$: take a security parameter λ and outputs a common reference string σ together with private verification information vk.

– $\text{Prove}(\sigma, x, w) \to \pi$: on input σ, a statement x and witness w, outputs an argument π.
– $\text{Verify}(\sigma, \text{vk}, x, \pi) \to 1/0$: on input σ, the private verification key vk, a statement x and a proof π, it outputs either 1 indicating accepting the argument or 0 for rejecting it.

Definition 2 (SNARK). *A triple of PPT algorithms* (**Setup, Prove**, **Verify**) *is a SNARK for an NP relation $\mathcal{R}$, if the following properties are satisfied:*

- *Completeness. For all $(x, w) \in \mathcal{R}$, the following holds:*

$$Pr\left[\mathbf{Verify}(\sigma, vk, x, \pi) = 1 \;\middle|\; \begin{array}{l}(\sigma, vk) \leftarrow \mathbf{Setup}(1^\lambda) \\ \pi \leftarrow \mathbf{Prove}(\sigma, x, w)\end{array}\right] = 1.$$

- *Knowledge Soundness. For any PPT adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{E}_\mathcal{A}$ such that the following probability is negligible in λ:*

$$Pr\left[\begin{array}{c}\mathbf{Verify}(\sigma, vk, \tilde{x}, \tilde{\pi}) = 1 \\ \wedge \mathcal{R}(\tilde{x}, w') = 0\end{array} \;\middle|\; \begin{array}{l}(\sigma, vk) \leftarrow \mathbf{Setup}(1^\lambda) \\ ((\tilde{x}, \tilde{\pi}); w') \leftarrow \mathcal{A}|\mathcal{E}_\mathcal{A}(\sigma)\end{array}\right]$$

- *A non-interactive argument of knowledge satisfies knowledge is succinct if the size of proof π and the time to* **Verify** *are sublinear in the size of the statement proven.*

If a SNARK does not require a private verification key vk, then the SNARK scheme is referred to as a public-key SNARK. If a SNARK does not require a CRS σ, then the SNARK scheme is referred to as transparent.

3 Packed Sumcheck

This section introduces *Packed Sumcheck*, a sumcheck variant that amplifies *round-by-round* soundness over a small prime field $\mathbb{F}_p$ without resorting to extension field arithmetic. We proceed in two steps. In Section 3.1, we revisit k-fold parallel repetition of sumcheck and reformulate it as a single sumcheck over the direct-product ring $(\mathbb{F}_p^k, +, \odot)$. This viewpoint makes the failure of round-by-round soundness transparent: the verifier's checks remain coordinate-wise, enabling *coordinate-wise repair* attacks. In Section 3.2, we remedy this by folding and *mixing* the k coordinates in every round using random matrices.

3.1 Revised parallel repetition sumcheck over small fields

Let p be a prime and let $\mu \geq 1$. Write $B_\mu := \{0,1\}^\mu$ and $N := 2^\mu$. Fix integers $d \geq 1$ and $k \geq 1$. For each coordinate (repetition) $j \in [0, k-1]$, the prover claims

$$\sum_{b \in B_\mu} \prod_{s=0}^{d-1} f_j^{(s)}(b) \;=\; h_j, \tag{1}$$

where each $f_j^{(s)} : \mathbb{F}_p^\mu \to \mathbb{F}_p$ is multilinear. In standard k-parallel repetition one typically has $f_0^{(s)} = \cdots = f_{k-1}^{(s)}$ for all s, but we keep (1) general since our packed protocol will naturally support k distinct instances.

A direct-product ring packing view $(\mathbb{F}_p^k, +, \odot)$ is a commutative ring, where $+$ is the vector addition and $\odot$ is the Hadamard product. We can pack the k-parallel instances into a single sumcheck instance over the direct-product ring $\mathbb{F}_p^k$.

For each $s \in \{0, ..., d-1\}$ and $b \in B_\mu$, define the packed evaluation:

$$F^{(s)}(b) = (f_0^{(s)}(b), f_1^{(s)}(b), ..., f_{k-1}^{(s)}(b)) \in \mathbb{F}_p^k.$$

Next, we define a multilinear polynomial over $\mathbb{F}_p^k$, denoted as $G^{(s)}$, as follows:

$$G^{(s)}(X_0, \cdots, X_{\mu-1}) := \sum_{b \in B_\mu} \bigodot_{j=0}^{\mu-1} \Big((\mathbf{1} - \mathbf{b_j}) \odot (\mathbf{1} - X_j) + \mathbf{b_j} \odot X_j \Big) \odot F^{(s)}(b)$$

where $\mathbf{0}$ and $\mathbf{1}$ are the all-zero and all-one vectors in $\mathbb{F}_p^k$ and $\mathbf{b_j}$ denotes $\mathbf{1}$ if $b_j = 1$ and $\mathbf{0}$ if $b_j = 0$. Define the hypercube over $\mathbb{F}_p^k$ as $B_\mu^{(\mathsf{vec})} = \{\mathbf{0}, \mathbf{1}\}^\mu$ and let $G(X) = \odot_{s=0}^{d-1} G^{(s)}(X)$. With these definitions, the k-parallel repetition sumcheck can be compactly expressed as a single sumcheck over the hypercube in $\mathbb{F}_p^k$: $\sum_{X \in B_\mu^{(\mathsf{vec})}} G(X) = H$.

We formalize the description of the sumcheck protocol over $\mathbb{F}_p^k$ as follows:

Sumcheck over $\mathbb{F}_p^k$:

- In each round $i \in [0, \mu - 2]$, the prover sends the polynomial over $\mathbb{F}_p^k$:

$$P_i(X_i) = \sum_{x_{i+1}, \cdots, x_{\mu-1} \in \{\mathbf{0}, \mathbf{1}\}} \bigodot_{s=0}^{d-1} G^{(s)}(r_0, r_1, ..., r_{i-1}, X_i, x_{i+1}, \cdots, x_{\mu-1})$$

- The verifier checks that $P_i(\mathbf{0}) + P_i(\mathbf{1}) = H_i$, where $H_0 = H$ and $H_i = P_{i-1}(r_{i-1})$ for $i \in [1, \mu - 2]$. Then the verifier samples a random challenge $r_i \in \mathbb{F}_p^k$ and sends it to the prover. And the verifier updates $H_{i+1} = P_i(r_i)$.
- In the last round $i = \mu - 1$, the prover sends:

$$P_{\mu-1}(X_{\mu-1}) = \bigodot_{s=0}^{d-1} G^{(s)}(r_0, r_1, ..., r_{\mu-2}, X_{\mu-1})$$

- The verifier checks that $P_{\mu-1}(\mathbf{0}) + P_{\mu-1}(\mathbf{1}) = H_{\mu-1}$. Then the verifier samples a random challenge $r_{\mu-1} \in \mathbb{F}_p^k$ and updates $H_\mu = P_{\mu-1}(r_{\mu-1})$. Then the verifier checks that $H_\mu = G(r_0, \cdots, r_{\mu-1})$.

In each round of the protocol, the polynomial $P_i(X_i)$ sent by the prover is defined over the direct-product ring $\mathbb{F}_p^k$, which can be understood as having k components corresponding to the polynomials sent by the prover in each sub-instance of the parallel repetition sumcheck protocol. The challenge r_i sent by the verifier is also an element defined over the direct-product ring $\mathbb{F}_p^k$, which can be understood as having k components corresponding to the challenges sent by the verifier in each sub-instance of the parallel repetition sumcheck protocol.

Failure of round-by-round soundness. We now explain why the above sumcheck protocol over $\mathbb{F}_p^k$ fails to achieve the ideal round-by-round soundness. For each round $i \in [0, \mu-1]$, we define the state predicate function $\mathsf{State}_i(G, r_0, ..., r_{i-1}, H_i)$ as follows:

$$\mathsf{State}_i(G, r_0, ..., r_{i-1}, H_i) := \begin{cases} 1, & \text{if } \sum_{x_i,...,x_{\mu-1} \in \{\mathbf{0},\mathbf{1}\}} G(r_0, ..., r_{i-1}, x_i, ..., x_{\mu-1}) = H_i \\ 0, & \text{otherwise} \end{cases}$$

When $\mathsf{State}_i(r_0, ..., r_{i-1}, H_i) = 0$, let $S_i \subset \{0, \ldots, k-1\}$ be the set of inconsistent coordinates:

$$S_i = \left\{ j \in [0, k-1] : \sum_{x_i,...,x_{\mu-1} \in \{\mathbf{0},\mathbf{1}\}} G(r_0, ..., r_{i-1}, x_i, ..., x_{\mu-1})_j \neq H_{i,j} \right\}.$$

where $H_{i,j}$ is the j-th component of H_i and $G(r_0, ..., r_{i-1}, x_i, ..., x_{\mu-1})_j$ is the j-th component of $G(r_0, ..., r_{i-1}, x_i, ..., x_{\mu-1})$.

Consider any round i with $S_i \neq \emptyset$, a cheating prover does not need to repair all the inconsistent components in one round. Instead, it can try to repair a subset $T \subset S_i$ in the current round while keeping the other inconsistent components for later rounds. Fix any such target subset T. For each $j \in T$, let $P_{i,j}(X_i)$ be the honest degree-$\leq d$ polynomial in that coordinate. Since $P_{i,j}(0) + P_{i,j}(1) \neq H_{i,j}$, a cheating prover can instead send a forged polynomial $\tilde{P}_{i,j}(X_i) \in \mathbb{F}_p[X]$ of degree at most d such that $\tilde{P}_{i,j}(0) + \tilde{P}_{i,j}(1) = H_{i,j}$. The verifier will then sample a random challenge $r_{i,j} \in \mathbb{F}_p$ as the j-th component of r_i and update $H_{i+1,j} = \tilde{P}_{i,j}(r_{i,j})$. Since $\tilde{P}_{i,j^*}(X_{i,j^*}) \neq P_{i,j^*}(X_{i,j^*})$ and both have degree at most d, they can agree on at most d points. Hence

$$\Pr_{r_{i,j^*} \in \mathbb{F}_p}[\tilde{P}_{i,j^*}(r_{i,j^*}) = P_{i,j^*}(r_{i,j^*})] \leq \frac{d}{p}.$$

If this event occurs, then the verifier's update is consistent with the honest polynomial for the j^*-th component, and thus prover can potentially fix the inconsistency in the j^*-th component.

Since the verifier samples the k coordinates of its challenge independently, the events that the prover successfully matching the honest update across different coordinates are independent. Hence, the probability that the prover successfully fixes all the inconsistent components in T is at most $\left(\frac{d}{p}\right)^{|T|}$.

So the prover can attempt to shrink S_i gradually by repairing a few coordinates per round, and each such partial repair has a $\left(\frac{d}{p}\right)^{|T|}$ probability of success. For k-parallel repetition, $S_0 = \{0, \ldots, k-1\}$, so the prover can fix at most $\lceil k/\mu \rceil$ coordinates per round. Hence the round-by-round soundness error is $\left(\frac{d}{p}\right)^{\lceil k/\mu \rceil}$.

3.2 Packed Sumcheck: Achieving Round-by-round Soundness by Repetition and Folding

The failure above stems from *coordinate separation*: the verifier's random challenges do not mix the k coordinates, so a prover can attempt coordinate-wise repair. To obtain k-fold round-by-round amplification, we want a one-round property of the following form: if the prover's message differs from the honest one, then the induced difference polynomial must vanish at *all* k verifier challenges simultaneously. Achieving this requires *mixing* coordinates in every round.

The direct-product ring $\mathbb{F}_p^k$ cannot express such mixing, because its multiplication is Hadamard. We therefore upgrade the verifier's challenges from vectors to matrices by working over the matrix ring $(\mathbb{F}_p^{k\times k}, +, \cdot)$, where addition is defined as matrix addition and multiplication is defined as matrix multiplication. We view $\mathbb{F}_p^k$ as a left R_{mat}-module under the natural matrix–vector action (vectors are column vectors). Our packed polynomials will have coefficients in $\mathbb{F}_p^k$ and take matrix inputs in $\mathbb{F}_p^{k\times k}$. We denote the product of matrices by $\prod_{i=n-1}^{0} M_i := M_{n-1}M_{n-2}\cdots M_0$. For each $s \in [0, d-1]$, we define a new packed polynomial:

$$E^{(s)}(X_0^{(0)}, X_0^{(1)}, ..., X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = \sum_{b\in B_\mu} \prod_{i=\mu-1}^{0} \Big((1-b_i)X_i^{(0)} + b_i X_i^{(1)}\Big) \cdot F^{(s)}(b).$$

Here, b_i denotes the i-th bit of b, and $F^{(s)}(b)$ represents the previously defined packed evaluation over $\mathbb{F}_p^k$. Intuitively, this is a matrix-valued multilinear polynomial, extension of the packed evaluations. For each index b, the bit b_i selects either $X_i^{(0)}$ or $X_i^{(1)}$ as the weight for that index. These selected matrices are then multiplied in the prescribed order, and the resulting matrix acts on the vector $F^{(s)}(b)$.

Define the matrix-pair hypercube $B_\mu^{(\mathsf{mat})} := \{L, R\}^\mu$, where $L := (I, \mathbf{0})$ and $R := (\mathbf{0}, I)$ are correlated pairs of matrices (with I and $\mathbf{0}$ the $k \times k$ identity and zero matrices). An element of $B_\mu^{(\mathsf{mat})}$ assigns to each position t either $(X_t^{(0)}, X_t^{(1)}) = (I, 0)$ or $(0, I)$.

A direct check shows that for every $b \in B_\mu$, if we choose $(X_t^{(0)}, X_t^{(1)}) = (I, 0)$ when $b_t = 0$ and $(0, I)$ when $b_t = 1$, then $E^{(s)}$ selects exactly the b-th table entry:

$$E^{(s)}\big((1-b_0)I, b_0 I, \ldots, (1-b_{\mu-1})I, b_{\mu-1}I\big) = F^{(s)}(b).$$

Therefore the k sumcheck claims can be written as the single packed identity

$$\sum_{(x_0^{(0)},x_0^{(1)}),\ldots,(x_{\mu-1}^{(0)},x_{\mu-1}^{(1)})\in B_\mu^{(\mathsf{mat})}} \bigodot_{s=0}^{d-1} E^{(s)}(x_0^{(0)}, x_0^{(1)}, \ldots, x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)}) \;=\; H \in \mathbb{F}_p^k,$$

where $\odot$ is the Hadamard product in $\mathbb{F}_p^k$. We can now describe the new packed sumcheck protocol.

Packed Sumcheck over $\mathbb{F}_p^{k\times k}$:

Parameters: We embed the kd independent multilinear polynomials $f_i^{(s)} : \mathbb{F}_p^{\mu} \to \mathbb{F}_p, s \in [0, d-1], i \in [0, k-1]$ into d multilinear polynomials with coefficients in $\mathbb{F}_p^k$ and variables taking values in $\mathbb{F}_p^{k \times k}$:

$$E^{(s)}(X_0^{(0)}, X_0^{(1)}, ..., X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = \sum_{b \in B_\mu} \prod_{i=\mu-1}^{0} \Big((1-b_i)X_i^{(0)} + b_i X_i^{(1)}\Big) \cdot F^{(s)}(b).$$

The parameters p, d, k need to satisfy $p > (2k-1)d+1$. The prover claims that:

$$\sum_{(x_0^{(0)}, x_0^{(1)}), ..., (x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)}) \in B_\mu^{\mathsf{mat}}} \bigodot_{s=0}^{d-1} E^{(s)}(x_0^{(0)}, x_0^{(1)}, ..., x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)}) = H$$

where $H = (h_0, h_1, ..., h_{k-1}) \in \mathbb{F}_p^k$. In this protocol, we do not require that the k polynomials $f_i^{(s)}$ for each $s \in [0, d-1]$ are identical, allowing us to handle k different sumcheck instances simultaneously. Let $W = \{0, \ldots, 2k-1\} \subset \mathbb{F}_p$ and define a set of Lagrange interpolation basis functions $\mathcal{L}_i$ over W, satisfying $\mathcal{L}_i(j) = 1$ if $i = j$, else 0.

Protocol:

- Initialize $H_0 = H$ and $E_0^{(s)} = E^{(s)}$ for each $s \in [0, d-1]$. For each round $i \in [0, \mu-1]$:
  1. View $E_i^{(s)}$ as a function over the remaining matrix-pair hypercube $B_{\mu-i}^{\mathsf{mat}} = \{L, R\}^{\mu-i}$. Its evaluation table is a vector of length $2^{\mu-i}$ over $\mathbb{F}_p^k$. For each coordinate $j \in [0, k-1]$, let $f_{i,j}^{(s)} : \{0,1\}^{\mu-i} \to \mathbb{F}_p$ denote the multilinear polynomial corresponding to the j-th coordinate of this table.
     The prover computes a univariate polynomial $P_i(Y_i) \in \mathbb{F}_p[Y_i]$ of degree at most $(2k-1)d$:

$$P_i(Y_i) = \sum_{b_{[i+1,\mu-1]} \in \{0,1\}^{\mu-1-i}} \prod_{s=0}^{d-1} \Big(\sum_{j=0}^{k-1} \mathcal{L}_{2j}(Y_i) f_{i,j}^{(s)}(0, b_{[i+1,\mu-1]}) + \mathcal{L}_{2j+1}(Y_i) f_{i,j}^{(s)}(1, b_{[i+1,\mu-1]}) \Big)$$

     The prover sends $P_i(Y_i)$ to the verifier.
  2. The verifier consistency with the running claim:

$$(P_i(0) + P_i(1), P_i(2) + P_i(3), ..., P_i(2k-2) + P_i(2k-1)) = H_i^{\top}$$

     If the check passes, the verifier samples a random challenge $r_i = (r_{i,0}, \ldots, r_{i,k-1}) \in \mathbb{F}_p^k$ and sends it to the prover. The verifier then updates

$$H_{i+1} = (P_i(r_{i,0}), P_i(r_{i,1}), ..., P_i(r_{i,k-1}))^{\top}$$

     .

3. If $i \le \mu - 2$, define the folding matrices $M_i^{(0)}, M_i^{(1)} \in \mathbb{F}_p^{k\times k}$ by

$$M_i^{(0)}[a,c] := \mathcal{L}_{2c}(r_{i,a}), \quad M_i^{(1)}[a,c] := \mathcal{L}_{2c+1}(r_{i,a}) \quad \text{for } a,c \in [0,k-1],$$

the prover updates the remaining polynomial by partial evaluation:

$$E_{i+1}^{(s)}(X_{i+1}^{(0)}, X_{i+1}^{(1)}, ..., X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = E_i^{(s)}(M_i^{(0)}, M_i^{(1)}, X_{i+1}^{(0)}, ..., X_{\mu-1}^{(1)}).$$

- After round $\mu - 1$, the verifier performs the final check:

$$H_\mu = \bigodot_{s=0}^{d-1} E^{(s)}(M_0^{(0)}, M_0^{(1)}, ..., M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}).$$

where $M_{\mu-1}^{(0)}$ and $M_{\mu-1}^{(1)}$ are defined from the last challenge $r_{\mu-1}$ using the same rule above.

Theorem 1. *The packed sumcheck protocol over $\mathbb{F}_p^{k\times k}$ satisfies the following properties:*

- *Completeness holds perfectly;*
- *Round-by-round soundness is guaranteed with error at most $\left(\frac{(2k-1)d}{p}\right)^k$;*
- *Prover complexity is $O((kd^2+k^2d)N)\cdot \mathsf{mul} + O(k^2d^2N)\cdot \mathsf{add}$, where $N = 2^\mu$;*
- *Verifier complexity is $O(k^2d\mu)$ field operations;*
- *Communication complexity is $O(kd\mu)$ field elements in $\mathbb{F}_p$.*

The proof of Theorem 1 is given in Appendix B.1.

3.3 Complexity comparison with other sumcheck protocols.

To our knowledge, packed sumcheck is the first sumcheck protocol that obtains k-fold round-by-round soundness amplification over a small prime base field using only base field arithmetic, without relying on extension field multiplication.

For completeness, we provide a complexity comparison with other SOTA sumcheck protocols over small fields. Let $N = 2^\mu$ be the size of the sumcheck instance, d be the degree of the underlying polynomial, and k be the round-by-round soundness amplification factor. We summarize the complexities in Table 1.

The complexities in the table 1 are calculated by counting one $\mathbb{F}_{p^k}$ multiplication as $O(k^2)$ $\mathbb{F}_p$-operations using the naive algorithm. In some common extension structures (e.g., tower extensions), extension field multiplication can be optimized to $O(k^{\log_2 3}) \approx O(k^{1.585})$ using Karatsuba's algorithm. In such cases, we provide the optimized complexities in the table 2.

From Tables 1 and 2, we see that even when Karatsuba-style optimization is applicable to prior extension-field-based protocols, the packed sumcheck tends to be more favorable in the practically relevant regime of small k. This makes it especially suitable for applications over small prime fields of roughly 20–30 bits, such as BabyBear, where taking $k = 4$ or $k = 5$ already suffices to meet common security targets.

Table 1: Complexity comparison of sumcheck protocols over small fields

Protocol	Prover	Verifier	Size	RBR Soundness
Naive k-extension	$O(k^2d^2N)$	$O(k^2d\log N)$	$O(kd\log N)$	d/p^k
Naive $k\log N$-parallel	$O(kd^2N\log N)$	$O(kd\log^2 N)$	$O(kd\log^2 N)$	$(d/p)^k$
Gruen Algorithm [40]	$O(kd^2N)$	$O(k^2d\log N)$	$O(kd\log N)$	$(dk-1)/p^k$
[LZ25] [47]	$O(kd^2N)$	$O(k^2d\log N)$	$O(kd\log N)$	d/p^k
[BDT24] [7]	See pp. 11, 25 of [BDT24] [7]	$O(k^2d\log N)$	$O(kd\log N)$	d/p^k
[BDDT25] [6]	See Thm. 4.1 of [BDDT25] [6]	$O(k^2d\log N)$	$O(kd\log N)$	d/p^k
[DDBDT26] [26]	$O((kd)^{2-2/\delta}N)$, $\delta=\log_2(d+1)$	$O(k^2d\log N)$	$O(kd\log N)$	d/p^k
packed sumcheck	$O((d^2+kd)N)\mathsf{mul}+O(kd^2N)\mathsf{add}$	$O(dk\log N)$	$O(d\log N)$	$((2k-1)d/p)^k$

Notes:

1. All complexities are measured in $\mathbb{F}_p$-operations and one $\mathbb{F}_{p^k}$ multiplication is counted as $O(k^2)$ $\mathbb{F}_p$-operations.
2. Except for our packed sumcheck, all protocols have the same-order multiplication and addition complexities.
3. The complexities in the table are amortized, so the complexity of our packed sumcheck is reduced by a factor of k compared to Theorem 1.
4. The prover complexity of the protocols in [6,7] involve elaborate expressions so we refer to the original papers.

Table 2: Complexity comparison of sumcheck protocols with Karatsuba

Protocol	Prover	Verifier	Size	RBR Soundness
Naive k-extension	$O(k^{1.585}d^2N)$	$O(k^{1.585}d\log N)$	$O(kd\log N)$	d/p^k
Naive $k\log N$-parallel	$O(kd^2N\log N)$	$O(kd\log^2 N)$	$O(kd\log^2 N)$	$(d/p)^k$
Gruen Algorithm [40]	$O(k^{0.585}d^2N)$	$O(k^{1.585}d\log N)$	$O(kd\log N)$	$(dk-1)/p^k$
[LZ25] [47]	$O(k^{0.585}d^2N)$	$O(k^{1.585}d\log N)$	$O(kd\log N)$	d/p^k
[BDT24] [7]	See pp. 11, 25 of [BDT24] [7]	$O(k^{1.585}d\log N)$	$O(kd\log N)$	d/p^k
[BDDT25] [6]	See Thm. 4.1 of [BDDT25] [6]	$O(k^{1.585}d\log N)$	$O(kd\log N)$	d/p^k
[DDBDT26] [26]	$O(k^{1.585(1-1/\delta)}d^{2-2/\delta}N)$, $\delta=\log_2(d+1)$	$O(k^{1.585}d\log N)$	$O(kd\log N)$	d/p^k
packed sumcheck	$O((d^2+k^{0.807}d)N)\mathsf{mul}+O(kd^2N)\mathsf{add}$	$O(dk\log N)$	$O(d\log N)$	$((2k-1)d/p)^k$

Notes:

1. Our packed sumcheck does not involve extension field multiplications, so it cannot benefit from Karatsuba optimization. However, the matrix multiplication process in our packed sumcheck can be optimized using Strassen's algorithm (see Appendix B.1 and Remark 2), which can reduce our complexity to $O((d^2+k^{0.807}d)N)\cdot\mathsf{mul}+O(kd^2N)\cdot\mathsf{add}$.

For the concrete performance comparison in Section 6.1, we use Bagad et al. [7] as the baseline. Gruen [40] and Liu and Zhang [47] provide only asymptotic analyses without implementations. The works [6, 7, 26] belong to the same line of sumcheck prover optimizations, and their small-characteristic optimizations build on Algorithm 4 of [7]. The later works [6] and [26] are instantiated over binary-tower fields and BN254, rather than small prime fields. In contrast, Bagad et al. [7] include an implementation over the BabyBear field, making it the closest reproducible baseline for our comparison.

Remark 1 (Round-by-round soundness error comparison). As shown in Table 1, the packed sumcheck has a somewhat larger round-by-round soundness error

than extension-field-based protocols. This mainly comes from the use of repetition for soundness amplification: under such an amplification mechanism, the numerator of the soundness error also grows exponentially with the repetition parameter. This is not specific to our protocol, but rather an intrinsic cost of repetition-based amplification.

Nevertheless, in practical parameter regimes, this gap does not lead to worse parameter choices. For example, over the BabyBear field and for degrees $d \in \{2, \ldots, 5\}$, achieving 100 bits of round-by-round soundness requires both the repetition parameter and the extension parameter to be 4, while achieving 128 bits requires both of them to be 5. Therefore, under these common security targets, our packed sumcheck does not lose out in parameter selection and remains comparable to extension-field-based protocols in concrete round-by-round security.

4 Commitments for Matrix-Weighted Multilinear Extensions

In the packed sumcheck protocol, the verifier's final check reduces to obtaining

$$E^{(s)}\big(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}\big) \in \mathbb{F}_p^k,$$

where the matrix pairs $(M_i^{(0)}, M_i^{(1)}) \in \mathbb{F}_p^{k\times k} \times \mathbb{F}_p^{k\times k}$ are determined by the protocol transcript. Once these matrices are fixed, the required value is a *structured* linear transformation of the initial hypercube evaluation table

$$F^{(s)} \in \big(\mathbb{F}_p^k\big)^{2^\mu}, \qquad F^{(s)}(b) \in \mathbb{F}_p^k \text{ for } b \in \{0,1\}^\mu.$$

Concretely, for each $b = (b_0, \ldots, b_{\mu-1}) \in \{0,1\}^\mu$, define the matrix weight

$$W_b \;:=\; \prod_{i=\mu-1}^{0} \Big((1-b_i)M_i^{(0)} + b_i M_i^{(1)}\Big) \;\in\; \mathbb{F}_p^{k\times k},$$

where the product is taken in the stated order. Then we have the decomposition

$$E^{(s)}\big(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}\big) \;=\; \sum_{b\in\{0,1\}^\mu} W_b \cdot F^{(s)}(b),$$

where "$\cdot$" denotes matrix–vector multiplication (vectors are column vectors).

This is analogous to a standard multilinear extension opening, where evaluation at $r \in \mathbb{F}_p^\mu$ is a linear combination of the hypercube table with tensor-product coefficients $\bigotimes_i (1-r_i, r_i)$. Here the coefficients are instead matrix-valued and induced by the packed-sumcheck transcript, mixing the k packed coordinates. Thus, packed sumcheck requires a commitment primitive for such matrix-weighted structured linear combinations.

Definition 3 (Structured MW-MLE commitment). *Fix a prime field $\mathbb{F}_p$ and integers $k, \mu \geq 1$. Let $F : \{0,1\}^\mu \to \mathbb{F}_p^k$ be a vector-valued hypercube table (vectors are column vectors). A* structured MW-MLE query *is a sequence of matrix pairs $\mathcal{M} = \{(M_i^{(0)}, M_i^{(1)})\}_{i=0}^{\mu-1}$, where $M_i^{(0)}, M_i^{(1)} \in \mathbb{F}_p^{k\times k}$. It induces, for each $b = (b_0, \ldots, b_{\mu-1}) \in \{0,1\}^\mu$, the matrix weight*

$$W_{\mathcal{M}}(b) \;:=\; \prod_{i=\mu-1}^{0} \left((1-b_i)M_i^{(0)} + b_i M_i^{(1)}\right) \in \mathbb{F}_p^{k\times k},$$

where the product is taken in the stated order, and defines the target value

$$\mathsf{Eval}(F, \mathcal{M}) \;:=\; \sum_{b\in\{0,1\}^\mu} W_{\mathcal{M}}(b) \cdot F(b) \;\in\; \mathbb{F}_p^k.$$

A commitment scheme supporting structured matrix-weighted MLE queries (consistently abbreviated as an MW-MLE commitment*) consists of PPT algorithms* $\mathsf{Commit}, \mathsf{Prove}, \mathsf{Verify}$ *and satisfies the properties below.*

- *Commit(F): takes $F \in (\mathbb{F}^k)^N$ and outputs a commitment C and auxiliary prover state σ: $(C, \sigma) \leftarrow \mathsf{Commit}(F)$.*
- *$\mathsf{Prove}(F, C, \sigma, \mathcal{M})$: outputs an answer/proof pair (z, π): $(z, \pi) \leftarrow \mathsf{Prove}(C, \sigma, \mathcal{M})$.*
- *$\mathsf{Verify}(C, \mathcal{M}, z, \pi)$: outputs a bit in $\{0,1\}$: $b \leftarrow \mathsf{Verify}(C, \mathcal{M}, z, \pi)$.*

The scheme satisfies the following properties.

- ***Completeness.*** *For all $F \in \mathcal{F}$ and all structured queries $\mathcal{M}$, if $(C, \sigma) \leftarrow \mathsf{Commit}(F)$ and $(z, \pi) \leftarrow \mathsf{Prove}(C, \sigma, \mathcal{M})$, then*

$$\Pr\left[\mathsf{Verify}(C, \mathcal{M}, z, \pi) = 1\right] = 1,$$

and moreover the accepted value equals the target evaluation, i.e., $z = \mathsf{Eval}(F, \mathcal{M})$.
- ***Binding.*** *For all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ outputs $(C, \mathcal{M}, z, \pi, z', \pi')$ with $z \neq z'$ such that $\mathsf{Verify}(C, \mathcal{M}, z, \pi) = 1$ and $\mathsf{Verify}(C, \mathcal{M}, z', \pi') = 1$ is negligible in the security parameter.*
- ***Knowledge soundness.*** *There exists a PPT extractor $\mathcal{E}$ such that for every PPT prover $\mathcal{P}^*$ there is a negligible function $\mathsf{negl}(\cdot)$ with the following: for any structured query $\mathcal{M}$, if*

$$(C, \sigma) \leftarrow \mathsf{Commit}(F) \text{ for some } F \in \mathcal{F}, \quad (z, \pi) \leftarrow \mathcal{P}^*(C, \mathcal{M}),$$

and $\mathsf{Verify}(C, \mathcal{M}, z, \pi) = 1$, then with probability at least $1 - \mathsf{negl}(\cdot)$ the extractor $\mathcal{E}^{\mathcal{P}^*}(C)$ outputs a table $\widetilde{F} \in \mathcal{F}$ such that C is a commitment to $\widetilde{F}$ and $z = \mathsf{Eval}(\widetilde{F}, \mathcal{M})$.

This section presents two instantiations of the structured MW-MLE commitment defined above. One is based on Brakedown polynomial commitments [37], and the other is based on Basefold PIOPs [61]. Both of our instantiations build on code-based PCS frameworks, so we need an encoding procedure that avoids general $\mathbb{F}_{p^k}$ multiplication. We adopt a block-level [30] Reed–Solomon encoding. The details of this encoding are given in Appendix E.

4.1 Structured MW-MLE commitment based on Brakedown

Brakedown commits to a table by encoding rows with a linear code and Merkle-committing to the encoded columns. It supports verification of arbitrary linear tests of the form $u = R \cdot U$ using spot checks and the code distance. Since our packed opening is a structured linear functional of the table, Brakedown can certify it by choosing a linear test matrix whose coefficients are derived from the packed sumcheck transcript.

Structured MW-MLE based on Brakedown Construction.

• **Parameters:** Let p be a prime and k satisfy $p^k = O(2^\lambda)$. Let E be the packed polynomial with coefficients in $\mathbb{F}_p^k$ and matrix pair variables over $\mathbb{F}_p^{k\times k}$, defined over the hypercube $B_\mu^{\mathsf{mat}} = \{L, R\}^\mu$. Let $F \in (\mathbb{F}_p^k)^{2^\mu}$ be the evaluations of E over B_μ^{mat} (identifying $L \mapsto 0$ and $R \mapsto 1$). Let Enc be a linear code over $\mathbb{F}_{p^k}$ with rate α and relative distance γ, H be a collision-resistant hash function, and let $Q = \Theta(\lambda)$ be the repetition parameter.

• **Committing phase.**

1. The prover reshapes F into a matrix $U^F \in (\mathbb{F}_p^k)^{2^{\mu/2}\times 2^{\mu/2}}$ via $U^F_{i,j} = F_{(i,j)}$ for $i, j \in \{0,1\}^{\mu/2}$. Viewing each entry of $\mathbb{F}_p^k$ as a length-k vector over $\mathbb{F}_p$, the prover further represents U^F as a matrix $U \in \mathbb{F}_p^{(k\cdot 2^{\mu/2})\times 2^{\mu/2}}$.
2. The prover performs block-level encoding [30]: In each row of U, every k consecutive field elements are treated as a single symbol in an extension field $\mathbb{F}_{p^k}$, and the prover encodes each row using Enc to obtain $\widehat{U} \in \mathbb{F}_{p^k}^{(k\cdot 2^{\mu/2})\times(2^{\mu/2}/(\alpha k))}$. The prover hashes each column of $\widehat{U}$ and commits by sending the Merkle root.

• **Testing phase.**

1. The verifier samples a random matrix $R \in \mathbb{F}_p^{k\times(k\cdot 2^{\mu/2})}$ and sends it to the prover.
2. The prover returns $V = R \cdot U \in \mathbb{F}_p^{k\times 2^{\mu/2}}$.
3. The verifier samples $S \subset [0, 2^{\mu/2}/(\alpha k))$ with $|S| = Q$. For each $j \in S$:
   - The prover opens the j-th column of $\widehat{U}$ with a Merkle proof; the verifier checks the proof.
   - For each row $V_{a,\cdot}$, the verifier checks consistency of the linear test under encoding:
   $$\mathsf{Enc}(V_{a,\cdot})_j \stackrel{?}{=} \sum_{\ell=0}^{k\cdot 2^{\mu/2}-1} R_{a,\ell} \cdot \widehat{U}_{\ell,j}.$$
   If all checks pass, the verifier accepts the linear-test consistency of V with the committed table.

• **Evaluation phase (opening for packed sumcheck).**

1. The verifier forms the structured linear-test coefficients induced by the packed-sumcheck transcript. Specifically, it computes the tensor product
   $$(M^{(0)}_{\mu/2-1}, M^{(1)}_{\mu/2-1}) \otimes \cdots \otimes (M^{(0)}_0, M^{(1)}_0),$$
   flattens it into a matrix $S_0 \in \mathbb{F}_p^{k\times(k\cdot 2^{\mu/2})}$, and sends S_0 to the prover.

2. The prover returns $W = S_0 \cdot U \in \mathbb{F}_p^{k \times 2^{\mu/2}}$.
3. The verifier repeats spot-checking procedure as in the testing phase to certify that W is consistent with the committed $\widehat{U}$ under the linear test S_0.
4. The verifier computes a length-$2^{\mu/2}$ vector $\mathcal{T}$ of $k \times k$ matrices

$$\mathcal{T} = (M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}) \otimes \cdots \otimes (M_{\mu/2}^{(0)}, M_{\mu/2}^{(1)}),$$

parses W into a length-$2^{\mu/2}$ vector $\mathcal{W}$ of elements in $\mathbb{F}_p^k$, and outputs $Z = \sum_{i=0}^{2^{\mu/2}-1} \mathcal{T}_i \cdot \mathcal{W}_i \in \mathbb{F}_p^k$. By construction of $\mathcal{T}$ and S_0 from the transcript, the output Z equals the opening value $E^{(s)}(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)})$.

Theorem 2. *For any packed-sumcheck transcript inducing the matrices* $(M_0^{(0)}, M_0^{(1)}), \ldots, (M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)})$*, the Brakedown-style commitment scheme above correctly opens the committed table to the value*

$$E^{(s)}(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}) \in \mathbb{F}_p^k$$

under honest computation. In the random oracle model, the scheme has negligible evaluation binding error and knowledge soundness error for these transcript-induced structured openings.

The proof of Theorem 2 and the complexity analysis are provided in Appendix B.2 and Appendix C.1, respectively.

4.2 Structured MW-MLE commitment based on Basefold

We defer the Basefold-based instantiation, including the construction, security theorem statements, proofs, and complexity analysis, to Appendix D.

5 A SNARK over Fields of Small Characteristic

This section describes a SNARK whose arithmetic is performed over a small prime field $\mathbb{F}_p$, while soundness is amplified by *packing* k parallel checks and enforcing round-by-round soundness via our packed sumcheck protocol (Section 3).

5.1 ZeroCheck Protocol

In this subsection we present a packed ZeroCheck protocol built on top of our packed sumcheck. The goal is to prove that k low-degree multivariate polynomials vanish identically on the Boolean hypercube $\{0,1\}^\mu$. The packed ZeroCheck protocol serves as a fundamental building block for our SNARK construction.

Definition 4 (Packed ZeroCheck Relation). *The relation* $\mathcal{R}_{ZeroCheck}$ *is the set of all tuples* $(\mathbb{x}; \mathbb{w}) = (\{[[f_i]]\}_{i\in[0,k)}; \{f_i\}_{i\in[0,k)})$ *where each* $f_i \in \mathbb{F}_p^{\leq d}[X_0, \ldots, X_{\mu-1}]$ *and* $f_i(b) = 0$ *for all* $b \in \{0,1\}^\mu$ *and* $i \in [0,k)$.

Packed ZeroCheck.

• **Parameters and packing model:**

Assume that each f_i is given as a product of d multilinear factors,

$$f_i \;=\; \prod_{s=0}^{d-1} f_i^{(s)}, \qquad \text{where each } f_i^{(s)} \text{ is multilinear.}$$

For each fixed s, we pack $\{f_i^{(s)}\}_{i\in[0,k)}$ into a single multilinear polynomial $E^{(s)}$ over the matrix ring $\mathbb{F}_p^{k\times k}$ using the matrix-pair hypercube variables from Section 3. We initialize $E_0^{(s)} := E^{(s)}$ for all $s \in [0, d)$.

We assume $p^k = \Theta(2^\lambda)$ and that $p > (2k-1)(d+1)+1$, so that the degree bounds used below are valid in $\mathbb{F}_p$.

Protocol.

1. The verifier samples k random points $\{\rho_i\}_{i\in[0,k)} \in \mathbb{F}_p^{\mu-1}$ and sends them to the prover.
2. Let $b' = b_{[1,\mu-1]}$ denote the last $\mu-1$ variables and $t = (t_0, \ldots, t_{k-1})$, prover computes the polynomial:

$$\Phi(t,y) = \sum_{b'\in\{0,1\}^{\mu-1}} \Big((\sum_{j=0}^{k-1} t_j \cdot \mathsf{eq}(\rho_j, b')) \cdot \prod_{s=0}^{d-1} \Big(\sum_{i=0}^{k-1} \mathcal{L}_{2i}(y) f_i^{(s)}(0,b') + \mathcal{L}_{2i+1}(y) f_i^{(s)}(1,b') \Big) \Big).$$

   and sends the polynomial Φ to the verifier. Here $\mathcal{L}_j(y)$ is the Lagrange basis polynomial for the set $\{0, 1, \ldots, 2k-1\}$.
3. Let $t^{(i)}$ denote the i-th standard basis vector with a 1 in the i-th coordinate and 0 elsewhere. For every $i, j \in [0, k)$, the verifier checks both

$$\Phi(t^{(i)}, 2j) = 0 \qquad \text{and} \qquad \Phi(t^{(i)}, 2j+1) = 0.$$

   If any check fails, it rejects. It then samples random challenges $(c, r_0) \in \mathbb{F}_p^{k\times k} \times \mathbb{F}_p^k$ and sends them to the prover. It computes

$$H_1 = (\Phi(c_{0,\cdot}, r_{0,0}), \ldots, \Phi(c_{k-1,\cdot}, r_{0,k-1}))^\top$$

   where $c_{i,\cdot}$ denotes the i-th row of matrix c.
4. Define a multilinear polynomial over $\mathbb{F}_p^{k\times k}$:

$$C(X_1^{(0)}, X_1^{(1)}, \ldots, X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = \sum_{b'\in B_{\mu-1}} \prod_{j=\mu-1}^{1} \Big((1-b'_j) X_j^{(0)} + b'_j X_j^{(1)} \Big) \cdot EQ(b')$$

   where $EQ(b') = \Big[\sum_{i=0}^{k-1} c_{0,i} \cdot \mathsf{eq}(\rho_i, b'), \ldots, \sum_{i=0}^{k-1} c_{k-1,i} \cdot \mathsf{eq}(\rho_i, b') \Big]^\top$ The prover computes

$$E_1^{(s)}(X_1^{(0)}, X_1^{(1)}, \ldots, X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = E_0^{(s)}(M_0^{(0)}, M_0^{(1)}, X_1^{(0)}, X_1^{(1)} \ldots, X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)})$$

where $M_0^{(0)}, M_0^{(1)}$ are the challenge matrices derived from r_0 in the packed sumcheck protocol.

5. Let $x' = (x_1^{(0)}, x_1^{(1)}, \ldots, x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)})$, both parties run a packed sumcheck protocol to prove that $\sum_{x' \in B_{\mu-1}^{\mathsf{mat}}} \left(C(x') \odot \left(\bigodot_{s=0}^{d-1} E_1^{(s)}(x') \right) \right) = H_1$

Theorem 3. *The packed ZeroCheck protocol satisfies the following properties:*

- *Completeness holds perfectly.*
- *Round-by-round soundness error is at most* $\max\left(\frac{\mu^k}{p^k}, \frac{(d(2k-1)+1)^k}{p^k}, \frac{((2k-1)(d+1))^k}{p^k}\right)$.
- *Prover complexity is* $O\left((kd^2 + k^2 d) \cdot 2^\mu\right) \cdot \mathsf{mul} + O\left(k^2 d^2 2^\mu\right) \cdot \mathsf{add}$.
- *Verifier complexity is* $O(k^3 \mu)$.
- *Communication complexity is* $O\left(k^2 d + kd \cdot \mu\right)$ *field elements.*

The proof of Theorem 3 is given in Appendix B.3.

5.2 SNARK for Small Prime Fields

We present a SNARK construction over small prime fields $\mathbb{F}_p$ based on the HyperPlonk framework [21]. By replacing the sumcheck and ZeroCheck protocols in HyperPlonk with our packed sumcheck and packed ZeroCheck protocols, respectively, and substituting the polynomial commitment scheme with our matrix-weighted MLE commitment scheme, we obtain a SNARK suitable for small prime fields. A detailed description and its knowledge soundness proof of this SNARK construction is provided in Appendix F.

5.3 Overview of the TFHE Arithmetization

We outline the arithmetization underlying our TFHE-friendly SNARK, with the complete construction given in Appendix H.

Given an LWE ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$, TFHE bootstrapping consists of modulus switching, blind rotation, sample extraction, and a combined modulus- and key-switching step. We instantiate $N_R = n = 1024$ and choose the NTT-friendly prime $q = Q = p = 2^{31} - 2^{27} + 1$. Aligning the LWE, RLWE, and SNARK moduli eliminates the modulus- and key-switching steps after blind rotation. Moreover, sample extraction is a public linear transformation and can be performed by the verifier. Therefore, only modulus switching and blind rotation need to be arithmetized. We express their computations as vector relations over $\mathbb{F}_p$ and prove them using vector lookup, Hadamard-product, and NTT checks based on our packed sumcheck.

Modulus switching. Modulus switching rescales and rounds each ciphertext coefficient from modulus q to modulus $2N_R$. Since rounding is not directly captured by arithmetic constraints, we partition the input domain into intervals corresponding to the possible outputs and use vector lookup checks to prove that each input coefficient lies in the interval associated with its claimed output.

Blind rotation. We divide the arithmetization of blind rotation into initialization and repeated CMUX evaluation. During initialization, lookup and Hadamard-product checks certify the monomial embeddings in the NTT representation. Each CMUX iteration is then reduced to gadget decomposition, NTT, and component-wise multiplication, which are certified using vector lookup, NTT, and Hadamard-product checks, respectively. We aggregate the corresponding relations across all CMUX iterations and prove them together. The detailed relations and protocols are given in Appendix H.2.

6 Implementation and Evaluation

6.1 Packed Sumcheck and MW-MLE Performance

We conducted single-threaded experiments on an Apple M2 Pro processor, evaluating performance over the BabyBear field $\mathbb{F}_p$, where $p = 2^{31} - 2^{27} + 1$. Our experimental evaluation aims to validate the efficiency of our packed sumcheck protocol and the low overhead of our MW-MLE schemes, demonstrating the practical value of our techniques in concrete implementations.

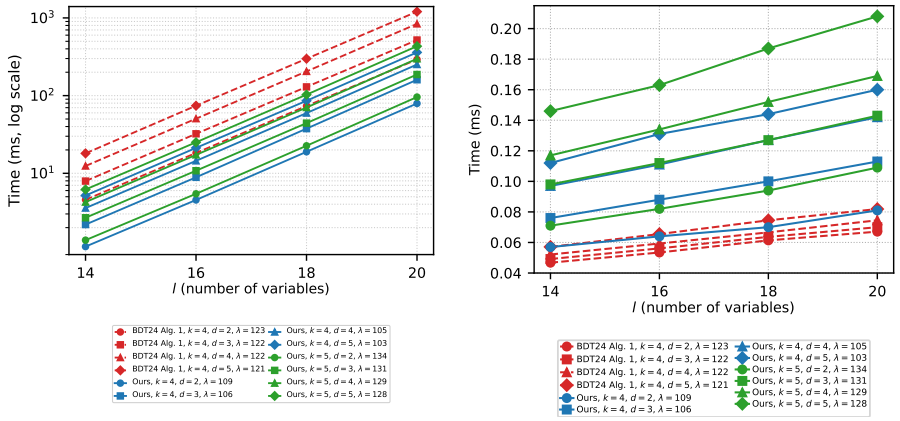
Performance for packed sumcheck. For packed sumcheck, we set the expansion factors to $k = 4$ (100-bit round-by-round soundness) and $k = 5$ (128-bit round-by-round soundness). We also benchmark Algorithm 1[8] from [7] in the BabyBear field using a 4-fold extension (121-bit round-by-round soundness).[9].

Figure 1 reports the prover and verifier results, where λ denotes the round-by-round soundness level in bits. Compared with Algorithm 1 at $k = 4$, our prover achieves 3.25×–4.03× speedups for $k = 4$ and 2.78×–3.36× for $k = 5$. However, our verifier does not outperform the baseline in practice, and the asymptotic advantage shown in Table 2 does not translate into lower measured verification time. This is because both verifiers are already highly efficient, operating at the microsecond scale, so fixed overheads and constant factors dominate. Moreover, sumcheck verification is substantially cheaper than proving; related works [6, 7, 26, 40, 47] and our work focus on improving prover efficiency.

Performance for MW-MLE. Our MW-MLE PCS have a structure and asymptotic prover complexity similar to those of small-field MLE PCS based on block-level encoding, the technique used in Binius [30] and FRI-Binius [31] (see Section 4.1, Appendix C.1, and Appendix D.5). We evaluated both MW-MLE instantiations over BabyBear under a unified setting (same finite-field library, hash functions, and block-level RS encoding) on instances of $2^{14} - 2^{20}$ field elements

[8] Algorithm 4 of [7] is effective over binary-tower fields, while providing smaller gains over BabyBear. We ran the code provided in [7] under the BabyBear setting with $d = \{2, 3, 4, 5\}$. Since Algorithm 1 of [7] is roughly the fastest under this setting, consistent with the results reported in their paper, we compare only with Algorithm 1.

[9] Both our implementation and the baseline from [7] use the BabyBear field implementation from ark-ff, which employs Karatsuba for extension-field multiplication. We benchmark both under the same environment, and also report Intel i9-13900H results in Appendix I to confirm cross-platform consistency.



(a) Prover time (b) Verifier time

Fig. 1: Sumcheck prover and verifier performance over the BabyBear field.

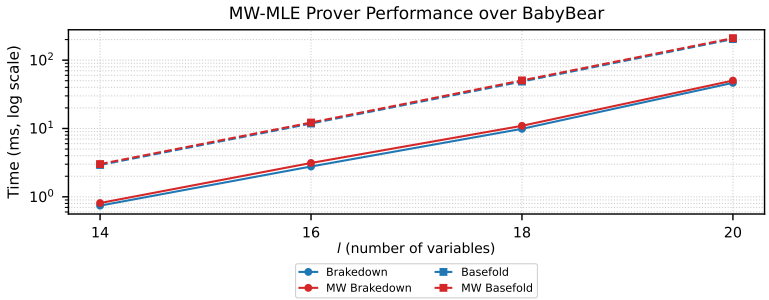


Fig. 2: MW-MLE prover performance over the BabyBear field.

with repetition factor $k = 4$. Figure 2 shows that the prover overhead is small: MW-Brakedown is within 12% of Brakedown with block-level encoding, and MW-Basefold is within 4% of Basefold. Thus matrix weighting adds only minor concrete cost.

Combining packed sumcheck with MW-MLE. To validate the practical effect of combining packed sumcheck with MW-MLE, we evaluate their composed prover costs over the BabyBear field and compare them with baselines built from prior components. Since [7] focuses on optimizing the sumcheck algorithm itself and provides performance data and source code only for the sumcheck component, we evaluate the baseline built by combining Algorithm 1 from [7] with Brakedown and Basefold schemes using block-level encoding technique. Figure 3 reports the results for repetition factor and extension factor both set to 4: our construction achieves 1.9×–2.3× speedups in the Brakedown-type setting and 1.3×–1.5× in the Basefold-type setting.

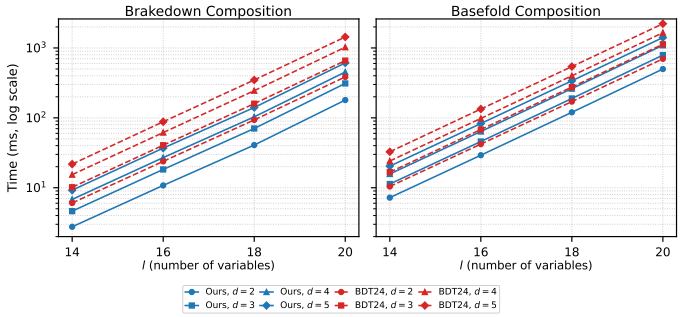


Fig. 3: Performance comparison after composing sumcheck with MW-MLE.

6.2 Verifiable TFHE Bootstrapping

Verifiable FHE is an important problem, and TFHE ciphertexts naturally live in small prime fields, making it a well-suited application for packed sumcheck. We present a TFHE-friendly SNARK using packed sumcheck in Appendix H, potentially of independent interest. In this subsection, we evaluate the performance of our TFHE-friendly SNARK, demonstrating its practical efficiency and the positive impact of our techniques on real-world applications.

TFHE is instantiated with LWE parameters $n = 1024$, $q = 2^{31} - 2^{27} + 1$, RLWE parameters $N_R = 1024$, $Q = 2^{31} - 2^{27} + 1$, and gadget decomposition parameter $D_d = 4$ [1]. We target $\lambda = 100$ bits of security, as in recent works [41, 62]. Since the base field has size 2^{31}, we set the expansion factor to $k = 4$. For the vector commitment, we use a block RS code of rate 1/2. By Lemma 4, the verifier opens 378 columns because $\left(1 - \frac{1}{3} \cdot \frac{1}{2}\right)^{378} = \left(\frac{5}{6}\right)^{378} < 2^{-100}$. For the vector lookup IOP, we set $s = 10$, which gives $\left(\frac{2^{20}}{2^{31}}\right)^{10} < 2^{-100}$ by Lemma 8.

All VFHE workload measurements were conducted on a server equipped with two AMD EPYC 9474F 48-core processors and 1 TB of RAM. Following the TFHE-friendly SNARK construction described in Appendix H, we evaluate the prover workloads of its individual components under the target parameters. Table 3 reports the resulting cost breakdown.[10] Table 4 compares our system with Thibault–Walter (TW25) [55] and Hasteboots [46].[11]

Acknowledgments. We would like to thank the anonymous reviewers for their valuable suggestions, and we also thank Zhelei Zhou, Lang Qin, and others for

[10] The modulus-switching phase operates only on vectors of size approximately 2^{10}, whereas blind-rotation initialization and blind rotation contain instances of size approximately 2^{20}. Its contribution to the overall proving cost is therefore negligible, and we omit it from the evaluation.

We reorganized the implementation, refined the experimental evaluation procedure, and reran the experiments under the same parameter settings. The updated mea-

Table 3: Cost Breakdown of our TFHE-friendly SNARK

	Proving Time	**Verification Time**	**Proof Size**
Vector Commitment	651.69ms	48.51ms	132.54MB
Blind Rotation Initialization	151.69ms	14.56ms	0.015MB
NTT/INTT	19.86ms	36.65ms	0.328MB
Gadget Decomposition	861.37ms	12.77ms	0.059MB
Hadamard Product	531.97ms	2.59ms	0.01MB
TFHE-friendly SNARK	2.2s	115.07ms	132.957MB

Table 4: Verifiable TFHE Bootstrapping Performance Comparison.

	Machine	Prover Time	Verifier Time	Proof Size
TW25 [55]	2×AMD EPYC 9474F 48-Core Processor	17min59s	7.3ms	192.48KB
Hasteboots [46]	2×AMD EPYC 9474F 48-Core Processor	7.52s	364.11ms	135.45MB
Our work	2×AMD EPYC 9474F 48-Core Processor	2.2s	115.07ms	132.957MB

helpful discussions. ChatGPT was used to improve the clarity and presentation of the manuscript and to assist with the implementation used in the experimental evaluation. All implementations and experimental results were verified by the authors.

We are supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB0690200), the National Key Research and Development Program of China (Grant No. 2023YFB4503203), the National Natural Science Foundation of China (Grant Nos. 62372447 and 62502513), the Postdoctoral Fellowship Program of China Postdoctoral Science Foundation (Grant No. GZC20252181), the Fundamental Research Funds for the Central Universities and Science (40500-20104-222609) and Technology Commission of Shanghai Municipality (Grant No. 23511101100).

References

1. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Math. Cryptol. **9**(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml

surements report a larger proof size than previously reported. These revisions do not affect the protocol, its security analysis, or the main technical contributions.

[11] We ran their implementations on the same machine; all three implementations support multithreading. Hasteboots provides multiple parameter and PCS configurations. We use the configuration closest to ours, which instantiates Brakedown over the BabyBear field. Its other configurations trade higher prover time for lower verifier time and smaller proofs.

2. Arnon, G., Chiesa, A., Fenzi, G., Yogev, E.: STIR: reed-solomon proximity testing with fewer queries. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X. Lecture Notes in Computer Science, vol. 14929, pp. 380–413. Springer (2024). https://doi.org/10.1007/978-3-031-68403-6_12, https://doi.org/10.1007/978-3-031-68403-6_12
3. Arnon, G., Chiesa, A., Fenzi, G., Yogev, E.: WHIR: reed-solomon proximity testing with super-fast verification. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 15604, pp. 214–243. Springer (2025). https://doi.org/10.1007/978-3-031-91134-7_8, https://doi.org/10.1007/978-3-031-91134-7_8
4. Atapoor, S., Baghery, K., Pereira, H.V.L., Spiessens, J.: Verifiable FHE via lattice-based snarks. IACR Commun. Cryptol. **1**(1), 24 (2024). https://doi.org/10.62056/A6KSDKP10, https://doi.org/10.62056/a6ksdkp10
5. Attema, T., Fehr, S., Klooß, M.: Fiat-shamir transformation of multi-round interactive proofs. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13747, pp. 113–142. Springer (2022). https://doi.org/10.1007/978-3-031-22318-1_5, https://doi.org/10.1007/978-3-031-22318-1_5
6. Bagad, S., Dao, Q., Domb, Y., Thaler, J.: Speeding up sum-check proving. IACR Cryptol. ePrint Arch. p. 1117 (2025), https://eprint.iacr.org/2025/1117
7. Bagad, S., Domb, Y., Thaler, J.: The sum-check protocol over fields of small characteristic. IACR Cryptol. ePrint Arch. p. 1046 (2024), https://eprint.iacr.org/2024/1046
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018), https://doi.org/10.4230/LIPIcs.ICALP.2018.14
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptol. ePrint Arch. p. 46 (2018), http://eprint.iacr.org/2018/046
10. Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S., Saraf, S.: Proximity gaps for reed-solomon codes. In: Irani, S. (ed.) 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020. pp. 900–909. IEEE (2020), https://doi.org/10.1109/FOCS46700.2020.00088
11. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A.D. (eds.) Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9986, pp. 31–60 (2016). https://doi.org/10.1007/978-3-662-53644-5_2, https://doi.org/10.1007/978-3-662-53644-5_2
12. Ben-Sasson, E., Goldberg, L., Kopparty, S., Saraf, S.: DEEP-FRI: sampling outside the box improves soundness. In: Vidick, T. (ed.) 11th Innovations in Theoretical Computer Science Conference, ITCS 2020, Seattle, Washington, USA, January 12-14, 2020. LIPIcs, vol. 151, pp. 5:1–5:32. Schloss Dagstuhl - Leibniz-Zentrum für

Informatik (2020). https://doi.org/10.4230/LIPICS.ITCS.2020.5, https://doi.org/10.4230/LIPIcs.ITCS.2020.5
13. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 326–349. ACM (2012), https://doi.org/10.1145/2090236.2090263
14. Bitansky, N., Choudhuri, A.R., Holmgren, J., Kamath, C., Lombardi, A., Paneth, O., Rothblum, R.D.: PPAD is as hard as LWE and iterated squaring. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13748, pp. 593–622. Springer (2022). https://doi.org/10.1007/978-3-031-22365-5_21, https://doi.org/10.1007/978-3-031-22365-5_21
15. Bois, A., Cascudo, I., Fiore, D., Kim, D.: Flexible and efficient verifiable computation on encrypted data. In: Garay, J.A. (ed.) Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12711, pp. 528–558. Springer (2021). https://doi.org/10.1007/978-3-030-75248-4_19, https://doi.org/10.1007/978-3-030-75248-4_19
16. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014), https://doi.org/10.1145/2633600
17. Brehm, M., Chen, B., Fisch, B., Resch, N., Rothblum, R.D., Zeilberger, H.: Blaze: Fast snarks from interleaved RAA codes. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 15604, pp. 123–152. Springer (2025). https://doi.org/10.1007/978-3-031-91134-7_5, https://doi.org/10.1007/978-3-031-91134-7_5
18. Bünz, B., Chen, J., DeStefano, Z., Chen, B.: Almost linear-time permutation check. Cryptology ePrint Archive, Paper 2025/1850 (2025), https://eprint.iacr.org/2025/1850
19. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019. pp. 1082–1090. ACM (2019). https://doi.org/10.1145/3313276.3316380, https://doi.org/10.1145/3313276.3316380
20. Cascudo, I., Costache, A., Cozzo, D., Fiore, D., Guimarães, A., Soria-Vazquez, E.: Verifiable computation for approximate homomorphic encryption schemes. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 16006, pp. 643–677. Springer (2025). https://doi.org/10.1007/978-3-032-01907-3_21, https://doi.org/10.1007/978-3-032-01907-3_21
21. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the

Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 499–530. Springer (2023). https://doi.org/10.1007/978-3-031-30617-4_17, https://doi.org/10.1007/978-3-031-30617-4_17
22. Cheon, J.H., Kim, A., Kim, M., Song, Y.S.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 409–437. Springer, Cham, Switzerland, Hong Kong, China (Dec 3–7, 2017). https://doi.org/10.1007/978-3-319-70694-8_15
23. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zksnarks with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 738–768. Springer (2020), https://doi.org/10.1007/978-3-030-45721-1_26
24. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020). https://doi.org/10.1007/S00145-019-09319-X, https://doi.org/10.1007/s00145-019-09319-x
25. Dao, Q., Biswas, A., Eagen, L., Milson, A., Papini, S., Thaler, J.: The sum-check protocol over the monomial basis, and other optimizations. Cryptology ePrint Archive (2026)
26. Dao, Q., DeStefano, Z., Bagad, S., Domb, Y., Thaler, J.: Speeding up sum-check proving (extended version). Cryptology ePrint Archive (2026)
27. Dao, Q., Thaler, J.: Constraint-packing and the sum-check protocol over binary tower fields. IACR Cryptol. ePrint Arch. p. 1038 (2024), https://eprint.iacr.org/2024/1038
28. Dao, Q., Thaler, J.: More optimizations to sum-check proving. IACR Cryptol. ePrint Arch. p. 1210 (2024), https://eprint.iacr.org/2024/1210
29. Diamond, B.E., Posen, J.: Proximity testing with logarithmic randomness. IACR Communications in Cryptology **1**(1) (2024). https://doi.org/10.62056/aksdkp10, https://doi.org/10.62056/aksdkp10
30. Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. In: Fehr, S., Fouque, P. (eds.) Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 15604, pp. 93–122. Springer (2025). https://doi.org/10.1007/978-3-031-91134-7_4, https://doi.org/10.1007/978-3-031-91134-7_4
31. Diamond, B.E., Posen, J.: Polylogarithmic proofs for multilinears over binary towers. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 16547, pp. 3–32. Springer (2026). https://doi.org/10.1007/978-3-032-25336-1_1, https://doi.org/10.1007/978-3-032-25336-1_1
32. Eagen, L., Fiore, D., Gabizon, A.: cq: Cached quotients for fast lookups. IACR Cryptol. ePrint Arch. p. 1763 (2022), https://eprint.iacr.org/2022/1763
33. Fiore, D., Nitulescu, A., Pointcheval, D.: Boosting verifiable computation on encrypted data. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020,

Proceedings, Part II. Lecture Notes in Computer Science, vol. 12111, pp. 124–154. Springer (2020). https://doi.org/10.1007/978-3-030-45388-6_5, https://doi.org/10.1007/978-3-030-45388-6_5
34. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. p. 953 (2019), https://eprint.iacr.org/2019/953
35. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: Snarks for ring arithmetic. J. Cryptol. **36**(4), 41 (2023), https://doi.org/10.1007/s00145-023-09481-3
36. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011. pp. 99–108. ACM (2011), https://doi.org/10.1145/1993636.1993651
37. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14082, pp. 193–226. Springer (2023). https://doi.org/10.1007/978-3-031-38545-2_7, https://doi.org/10.1007/978-3-031-38545-2_7
38. Graham, R.L.: Concrete mathematics: a foundation for computer science. Pearson Education India (1994)
39. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_11, https://doi.org/10.1007/978-3-662-49896-5_11
40. Gruen, A.: Some improvements for the PIOP for zerocheck. IACR Cryptol. ePrint Arch. p. 108 (2024), https://eprint.iacr.org/2024/108
41. Guo, Y., Liu, X., Huang, K., Qu, W., Tao, T., Zhang, J.: DeepFold: Efficient multilinear polynomial commitment from Reed-Solomon code and its application to zero-knowledge proofs. In: 34th USENIX Security Symposium (USENIX Security 25). pp. 3497–3516. USENIX Association, Seattle, WA (Aug 2025), https://www.usenix.org/conference/usenixsecurity25/presentation/guo-yanpei
42. Holmgren, J., Rothblum, R.D.: Faster sounder succinct arguments and IOPs. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. Lecture Notes in Computer Science, vol. 13507, pp. 474–503. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15802-5_17, https://doi.org/10.1007/978-3-031-15802-5_17
43. Hu, Y., Li, C., Qiu, Z., Xie, T., Ying, Y., Zhang, J., Zhang, Z.: GKR for boolean circuits with sub-linear RAM operations. IACR Cryptol. ePrint Arch. p. 717 (2025), https://eprint.iacr.org/2025/717
44. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada. pp. 723–732. ACM (1992), https://doi.org/10.1145/129712.129782
45. Kothapalli, A., Setty, S.: NeutronNova: Group-based folding done right. In: Heninger, N., Rosulek, M. (eds.) Advances in Cryptology – CRYPTO 2026. Lecture

Notes in Computer Science, vol. 16808, pp. 524–556. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-35424-2_18, https://doi.org/10.1007/978-3-032-35424-2_18
46. Liu, F., Liang, H., Xie, X., Yu, Y., Zheng, W., Hu, Y.: HasteBoots: Proving TFHE programmable bootstrapping in seconds. In: 35th USENIX Security Symposium (USENIX Security 26). USENIX Association, Baltimore, MD (Aug 2026), https://www.usenix.org/conference/usenixsecurity26/presentation/liu-fengrun
47. Liu, T., Zhang, Y.: Efficient snarks for boolean circuits via sumcheck over tower fields. IACR Cryptol. ePrint Arch. p. 594 (2025), https://eprint.iacr.org/2025/594
48. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. In: 31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I. pp. 2–10. IEEE Computer Society (1990). https://doi.org/10.1109/FSCS.1990.89518, https://doi.org/10.1109/FSCS.1990.89518
49. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6110, pp. 1–23. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_1, https://doi.org/10.1007/978-3-642-13190-5_1
50. Micali, S.: CS proofs (extended abstracts). In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 436–453. IEEE Computer Society (1994), https://doi.org/10.1109/SFCS.1994.365746
51. Micciancio, D., Polyakov, Y.: Bootstrapping in fhew-like cryptosystems. In: WAHC '21: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Virtual Event, Korea, 15 November 2021. pp. 17–28. WAHC@ACM (2021). https://doi.org/10.1145/3474366.3486924, https://doi.org/10.1145/3474366.3486924
52. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93. ACM (2005). https://doi.org/10.1145/1060590.1060603, https://doi.org/10.1145/1060590.1060603
53. Ron-Zewi, N., Rothblum, R.D.: Proving as fast as computing: succinct arguments with constant prover overhead. In: Leonardi, S., Gupta, A. (eds.) STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022. pp. 1353–1363. ACM (2022). https://doi.org/10.1145/3519935.3519956, https://doi.org/10.1145/3519935.3519956
54. Setty, S.T.V.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 704–737. Springer (2020), https://doi.org/10.1007/978-3-030-56877-1_25
55. Thibault, L.T., Walter, M.: Towards verifiable FHE in practice: Proving correct execution of tfhe's bootstrapping using plonky2. In: Huang, C., Chen, J., Shieh, S., Lie, D., Cortier, V. (eds.) Proceedings of the 2025 ACM SIGSAC Conference

on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025. pp. 1113–1126. ACM (2025). https://doi.org/10.1145/3719027.3765097, https://doi.org/10.1145/3719027.3765097
56. Viand, A., Knabenhans, C., Hithnawi, A.: Verifiable fully homomorphic encryption. CoRR **abs/2301.07041** (2023). https://doi.org/10.48550/ARXIV.2301.07041, https://doi.org/10.48550/arXiv.2301.07041
57. Wei, Y., Zhang, X., Deng, Y.: Transparent snarks over galois rings. In: Jager, T., Pan, J. (eds.) Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12-15, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 15674, pp. 418–451. Springer (2025). https://doi.org/10.1007/978-3-031-91820-9_14, https://doi.org/10.1007/978-3-031-91820-9_14
58. Xie, T., Liu, T.: Almost optimal succinct arguments for boolean circuit on RAM. IACR Cryptol. ePrint Arch. p. 839 (2024), https://eprint.iacr.org/2024/839
59. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 733–764. Springer (2019), https://doi.org/10.1007/978-3-030-26954-8_24
60. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13510, pp. 299–328. Springer (2022), https://doi.org/10.1007/978-3-031-15985-5_11
61. Zeilberger, H., Chen, B., Fisch, B.: Basefold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X. Lecture Notes in Computer Science, vol. 14929, pp. 138–169. Springer (2024). https://doi.org/10.1007/978-3-031-68403-6_5, https://doi.org/10.1007/978-3-031-68403-6_5
62. Zhang, X., Wang, R., Liu, Z., Xiang, B., Deng, Y., Fisch, B., Lu, X.: Phalanx: An fhe-friendly SNARK for verifiable computation on encrypted data. In: Huang, C., Chen, J., Shieh, S., Lie, D., Cortier, V. (eds.) Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025. pp. 4035–4048. ACM (2025). https://doi.org/10.1145/3719027.3765226, https://doi.org/10.1145/3719027.3765226
63. Zhou, Z., Li, Y., Wang, Y., Yang, Z., Zhang, B., Hong, C., Wei, T., Chen, W.: ZHE: efficient zero-knowledge proofs for HE evaluations. In: Blanton, M., Enck, W., Nita-Rotaru, C. (eds.) IEEE Symposium on Security and Privacy, SP 2025, San Francisco, CA, USA, May 12-15, 2025. pp. 3328–3346. IEEE (2025). https://doi.org/10.1109/SP61157.2025.00199, https://doi.org/10.1109/SP61157.2025.00199

Appendix

A Forward Difference

Definition 5. *The forward difference operator for a function $f(x)$ over $\mathbb{F}_p$ at discrete, equally spaced points $x_k = x_0 + kh$ (where h is the step size) is expressed as:*

$$\Delta f(x_k) = f(x_{k+1}) - f(x_k)$$

Higher-order differences are defined recursively. The n-th order forward difference is:

$$\Delta^n f(x_k) = \Delta(\Delta^{n-1} f(x_k)) = \Delta^{n-1} f(x_{k+1}) - \Delta^{n-1} f(x_k)$$

When the function f is clear from the context, we may denote $\Delta_k^n := \Delta^n f(x_k)$.

Fact 1 *[38] For a polynomial f of degree at most k over $\mathbb{F}_p$, its k-th finite difference, $\Delta^k f(x)$, is constant.*

Algorithm 1: Extrapolating f via Forward Differences

Input: Evaluations $f(0), \ldots, f(k-1)$ and an integer m
Output: Evaluations $f(k), \ldots, f(k+m-1)$

```
// Construct initial difference table
```
for $i \in [0, k-1]$ **do**
  Set $\Delta_i^0 = f(i)$;
for $i \leftarrow 1$ ***to*** $k-1$ **do**
  for $j \leftarrow 0$ ***to*** $k-1-i$ **do**
    $\Delta_j^i \leftarrow \Delta_{j+1}^{i-1} - \Delta_j^{i-1}$;
```
// Store the diagonal
```
for $i \in [0, k-1]$ **do**
  Set $D_i \leftarrow \Delta_{k-1-i}^i$;
```
// Extrapolate m new values
```
for $i \leftarrow 1$ ***to*** m **do**
  for $j \leftarrow k-2$ ***down to*** 0 **do**
    $D_j \leftarrow D_j + D_{j+1}$;
  Set $f(k-1+i) \leftarrow D_0$;
return $f(k), \ldots, f(k+m-1)$

Lemma 1. *Let f be a univariate polynomial of degree at most $k-1$ over $\mathbb{F}_p$, where $p > k$. Given the initial values $f(0), \ldots, f(k-1)$, for any integer m such that $p > k+m-1$, Algorithm 1 computes the evaluations $f(k), \ldots, f(k+m-1)$ using $O(k^2 + km)$ additions in $\mathbb{F}_p$.*

Proof. **Correctness.** The correctness of the algorithm follows from the fact that forward-difference extrapolation is equivalent to Lagrange polynomial interpolation. A univariate polynomial f of degree at most $k-1$ is uniquely determined by its evaluations at k distinct points, say $f(s), \ldots, f(s+k-1)$. The Lagrange interpolation formula gives the evaluation at a new point $s+k$ as:

$$f(s+k) = \sum_{i=0}^{k-1} c_i \cdot f(s+i), \quad \text{where} \quad c_i = \prod_{\substack{j=0 \\ j \neq i}}^{k-1} \frac{k-j}{i-j}.$$

Algorithm 1 computes the same value $f(s+k)$ by iteratively updating a diagonal of difference terms. This process is a linear transformation of the initial values $f(s), \ldots, f(s+k-1)$. A direct expansion of the forward-difference formulas reveals that the coefficient of each $f(s+i)$ in the final computation of $f(s+k)$ is precisely the Lagrange coefficient c_i defined above. This equivalence holds for any sequence of k consecutive points, thus proving the algorithm's correctness.
Complexity Analysis. The computational cost is dominated by two phases:

- Initialization: The construction of the initial difference table involves computing Δ_j^i for $i \in [1, k-1]$ and $j \in [0, k-1-i]$. The total number of subtractions is $\sum_{i=1}^{k-1}(k-i) = \frac{k(k-1)}{2}$, which is $O(k^2)$.
- Extrapolation: The algorithm computes m new values. Each new value is obtained by updating the diagonal, which requires $k-1$ additions. This phase costs $m(k-1)$ additions, resulting in a complexity of $O(km)$.

Combining both phases, the total complexity is $O(k^2 + km)$ additions in $\mathbb{F}_p$. □

B Deferred Proofs

B.1 Proof of Theorem 1

Theorem 4 (Theorem 1). *The packed sumcheck protocol over $\mathbb{F}_p^{k \times k}$ satisfies the following properties:*

- *Completeness holds perfectly;*
- *Round-by-round soundness is guaranteed with error at most $\left(\frac{(2k-1)d}{p}\right)^k$;*
- *Prover complexity is*

$$O((kd^2 + k^2 d)N) \cdot \mathsf{mul} + O(k^2 d^2 N) \cdot \mathsf{add},$$

  where $N = 2^\mu$ and mul and add denote the costs of a field multiplication and addition in $\mathbb{F}_p$ respectively;
- *Verifier complexity is $O(k^2 d\mu)$ field operations;*
- *Communication complexity is $O(kd\mu)$ field elements in $\mathbb{F}_p$.*

Proof. We define a state predicate that captures the running claim of the protocol. For each round $i \in [0, \mu]$, define $\mathsf{State}_i(\{E_i^{(s)}\}_{s\in[d-1]}, r_0, ..., r_{i-1}, H_i) = 1$ iff

$$\sum_{(x_i^{(0)},x_i^{(1)}),...,(x_{\mu-1}^{(0)},x_{\mu-1}^{(1)})\in B_{\mu-i}^{\mathsf{mat}}} \bigodot_{s=0}^{d-1} E_i^{(s)}(x_i^{(0)}, x_i^{(1)}, ..., x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)}) = H_i.$$

where $r_j = (r_{j,0}, ..., r_{j,k-1}) \in \mathbb{F}_p^k$ is the verifier's challenge in round j and $B_l^{\mathsf{mat}} = \{L, R\}^l$ denotes the l-dimensional matrix-pair hypercube. By construction, the verifier accepts at the end of the protocol if and only if $\mathsf{State}_\mu(\{E_\mu^{(s)}, \}_{s\in[d-1]}, r_0, ..., r_{\mu-1}, H_\mu) = 1$. We prove completeness and round-by-round soundness by showing a one-step transition property of State_i.

Completeness: Fix any $i \in [0, \mu-1]$. Suppose $\mathsf{State}_i(\{E_i^{(s)}\}_{s\in[d-1]}, r_0, ..., r_{i-1}, H_i) = 1$. We show that under an honest prover, after executing round i the verifier's check passes and $\mathsf{State}_{i+1}(\{E_{i+1}^{(s)}\}_{s\in[d-1]}, r_0, ..., r_i, H_{i+1}) = 1$.

For each $s \in [0, d-1]$, view the evaluation table of $E_i^{(s)}$ over $B_{\mu-i}^{\mathsf{mat}}$ as a length-$2^{\mu-i}$ vector over $\mathbb{F}_p^k$. For each coordinate $j \in [0, k-1]$, let $f_{i,j}^{(s)} : \{0,1\}^{\mu-i} \to \mathbb{F}_p$ denote the multilinear polynomial corresponding to the j-th coordinate of this evaluation table.

Let $\{\mathcal{L}_u(Y)\}_{u=0}^{2k-1}$ be the Lagrange basis polynomials over the set $W = \{0, 1, ..., 2k-1\}$. The honest prover compute the univariate polynomial $P_i(Y_i)$ of degree at most $(2k-1)d$:

$$P_i(Y) = \sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-1-i}} \prod_{s=0}^{d-1} \Big(\sum_{j=0}^{k-1} \mathcal{L}_{2j}(Y) f_{i,j}^{(s)}(0, b_{[i+1,\mu-1]}) + \mathcal{L}_{2j+1}(Y) f_{i,j}^{(s)}(1, b_{[i+1,\mu-1]}) \Big).$$

We first verify the verifier's check. Fix any $t \in [0, k-1]$, we have:

$$\begin{aligned}
&P_i(2t) + P_i(2t+1) = \\
&\sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-1-i}} \prod_{s=0}^{d-1} \Big(\sum_{j=0}^{k-1} \mathcal{L}_{2j}(2t)\, f_{i,j}^{(s)}(0, b_{[i+1,\mu-1]}) + \mathcal{L}_{2j+1}(2t)\, f_{i,j}^{(s)}(1, b_{[i+1,\mu-1]}) \Big) + \\
&\sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-1-i}} \prod_{s=0}^{d-1} \Big(\sum_{j=0}^{k-1} \mathcal{L}_{2j}(2t+1)\, f_{i,j}^{(s)}(0, b_{[i+1,\mu-1]}) + \mathcal{L}_{2j+1}(2t+1)\, f_{i,j}^{(s)}(1, b_{[i+1,\mu-1]}) \Big) \\
&= \sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-1-i}} \prod_{s=0}^{d-1} \Big(f_{i,t}^{(s)}(0, b_{[i+1,\mu-1]}) + f_{i,t}^{(s)}(1, b_{[i+1,\mu-1]}) \Big) \\
&= \sum_{b_{[i,\mu-1]}\in\{0,1\}^{\mu-i}} \prod_{s=0}^{d-1} f_{i,t}^{(s)}(b_{[i,\mu-1]}).
\end{aligned}$$

which is exactly the t-th coordinate of the left-hand side in the state equality defining State_i. Since $\mathsf{State}_i = 1$ by assumption, this equals $H_{i,t}$ for every t, so the verifier's check passes.

Next, the verifier samples a random challenge $r_i = (r_{i,0}, ..., r_{i,k-1}) \in \mathbb{F}_p^k$ and updates

$$H_{i+1} = (P_i(r_{i,0}), P_i(r_{i,1}), ..., P_i(r_{i,k-1}))^\top.$$

According to the definition of $H_{(i+1)}$ in the protocol, for any $t \in [0, k-1]$, we have:

$$H_{i+1,t} = P_i(r_{i,t}) = \sum_{b_{[i+1,\mu-1]} \in \{0,1\}^{\mu-1-i}} \bigodot_{s=0}^{d-1} \Big(\sum_{j=0}^{k-1} \mathcal{L}_{2j}(r_{i,t}) f_{i,j}^{(s)}(0, b_{[i+1,\mu-1]}) + \mathcal{L}_{2j+1}(r_{i,t}) f_{i,j}^{(s)}(1, b_{[i+1,\mu-1]}) \Big)$$

Combining all $H_{i+1,t}$ for $t \in [0, k-1]$ into vector form and using the definitions $M_i^{(0)}[a, c] = \mathcal{L}_{2c}(r_{i,a})$ and $M_i^{(1)}[a, c] = \mathcal{L}_{2c+1}(r_{i,a})$, we obtain:

$$\begin{aligned}
&\begin{bmatrix} H_{i+1,0} & H_{i+1,1} & \dots & H_{i+1,k-1} \end{bmatrix}^\top \\
&= \sum_{b_{[i+1,\mu-1]} \in \{0,1\}^{\mu-1-i}} \bigodot_{s=0}^{d-1} \left(M_i^{(0)} \begin{bmatrix} f_{i,0}^{(s)}(0, b_{[i+1,\mu-1]}) \\ f_{i,1}^{(s)}(0, b_{[i+1,\mu-1]}) \\ \vdots \\ f_{i,k-1}^{(s)}(0, b_{[i+1,\mu-1]}) \end{bmatrix} + M_i^{(1)} \begin{bmatrix} f_{i,0}^{(s)}(1, b_{[i+1,\mu-1]}) \\ f_{i,1}^{(s)}(1, b_{[i+1,\mu-1]}) \\ \vdots \\ f_{i,k-1}^{(s)}(1, b_{[i+1,\mu-1]}) \end{bmatrix} \right) \\
&= \sum_{b_{[i+1,\mu-1]} \in \{0,1\}^{\mu-1-i}} \bigodot_{s=0}^{d-1} \left(\sum_{b_i \in \{0,1\}} M_i^{(b_i)} \begin{bmatrix} f_{i,0}^{(s)}(b_i, b_{[i+1,\mu-1]}) \\ f_{i,1}^{(s)}(b_i, b_{[i+1,\mu-1]}) \\ \vdots \\ f_{i,k-1}^{(s)}(b_i, b_{[i+1,\mu-1]}) \end{bmatrix} \right)
\end{aligned}$$

According to the definition of $E_{i+1}^{(s)}$, its evaluation over $\{L, R\}^{\mu-i-1}$ is:

$$E_i^{(s)}((1-b_i)I, b_i I, \dots, (1-b_{\mu-1})I, b_{\mu-1}I) = \sum_{b_{[0,i-1]} \in \{0,1\}^i} \prod_{j=i-1}^{0} M_j^{(b_j)} F(b_0, \cdots, b_{\mu-1}).$$

Thus, we can conclude that:

$$
\begin{aligned}
&\left[H_{i+1,0}\ H_{i+1,1}\ \dots\ H_{i+1,k-1}\right]^\top \\
&= \sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-1-i}} \bigodot_{s=0}^{d-1} \left(\sum_{b_i\in\{0,1\}} M_i^{(b_i)} E_i^{(s)}((1-b_i)I, b_iI, \dots, (1-b_{\mu-1})I, b_{\mu-1}I) \right) \\
&= \sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-i-1}} \bigodot_{s=0}^{d-1} \left(\sum_{b_i\in\{0,1\}} M_i^{(b_i)} \sum_{b_{[0,i-1]}\in\{0,1\}^i} \prod_{j=i-1}^{0} M_j^{(b_j)} F(b_0, \cdots, b_{\mu-1}) \right) \\
&= \sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-i-1}} \bigodot_{s=0}^{d-1} \left(\sum_{b_{[0,i]}\in\{0,1\}^{i+1}} \prod_{j=i-1}^{0} M_j^{(b_j)} F(b_0, \cdots, b_{\mu-1}) \right) \\
&= \sum_{b_{[i+1,\mu-1]}\in\{0,1\}^{\mu-i-1}} \bigodot_{s=0}^{d-1} E_{i+1}^{(s)}((1-b_{i+1})I, b_{i+1}I, \dots, (1-b_{\mu-1})I, b_{\mu-1}I) \\
&= \sum_{(x_{i+1}^{(0)},x_{i+1}^{(1)}),\dots,(x_{\mu-1}^{(0)},x_{\mu-1}^{(1)})\in\{L,R\}} \bigodot_{s=0}^{d-1} E_{i+1}^{(s)}(x_{i+1}^{(0)}, x_{i+1}^{(1)}, ..., x_{\mu-1}^{(0)}, x_{\mu-1}^{(1)})
\end{aligned}
$$

Therefore, $\mathsf{State}_{i+1}(\{E_{i+1}^{(s)}\}_{s\in[d-1]}, r_0, \dots, r_i, H_{i+1}) = 1$, completing the proof of completeness.

Round-by-round Soundness: Fix any $i \in [0, \mu-1]$ and assume $\mathsf{State}_i(\{E_i^{(s)}\}_{s\in[d-1]}, r_0, ..., r_{i-1}, H_i) = 0$. Let $P_i(Y_i)$ denote the honest polynomial determined by the true evaluations of $E_i^{(s)}$ via the definition above. Since the state equality fails, there exists some coordinate $t^* \in [0, k-1]$ such that $P(2t^*) + P(2t^*+1) \neq H_{i,t^*}$.

To pass the verifier's check in round i, a cheating prover must send a forged polynomial $\tilde{P}_i(Y_i)$ such that: $\forall t \in [0, k-1]$, $\tilde{P}_i(2t) + \tilde{P}_i(2t+1) = H_{i,t}$. Consider the difference polynomial $\Delta(Y_i) = \tilde{P}_i(Y_i) - P_i(Y_i)$. From the inequality above, we get that

$$\Delta(2t^*) + \Delta(2t^*+1) = H_{i,t^*} - (P(2t^*) + P(2t^*+1)) \neq 0,$$

so $\Delta(Y_i)$ is a non-zero polynomial and the degree of $\Delta(Y_i)$ is at most $(2k-1)d$.

The verifier samples a random challenge $r_i = (r_{i,0}, ..., r_{i,k-1}) \in \mathbb{F}_p^k$ and sets $H_{i+1} = (\tilde{P}_i(r_{i,0}), ..., \tilde{P}_i(r_{i,k-1}))^\top$. Let $H_{i+1}^{\mathsf{hon}} = (P_i(r_{i,0}), ..., P_i(r_{i,k-1}))^\top$ be the value consistent with the true polynomials. If $\Delta(r_{i,t}) \neq 0$ for some $t \in [0, k-1]$, then $H_{i+1} \neq H_{i+1}^{\mathsf{hon}}$. Since the left-hand side of the next-round state equality is fixed by the true polynomial $E^{(s),(i+1)}$, this implies that $\mathsf{State}_{i+1}(r_0, ..., r_i, H_{i+1}) = 1$ cannot hold. Therefore, for a cheating prover to make the next state equality hold, it necessarily that

$$\Delta(r_{i,0}) = \Delta(r_{i,1}) = ... = \Delta(r_{i,k-1}) = 0.$$

By Schwartz-Zippel lemma over $\mathbb{F}_p$, for uniform $r \in \mathbb{F}_p$, we have:

$$\Pr_{r\in\mathbb{F}_p}[\Delta(r) = 0] \leq \frac{\deg(\Delta)}{p} \leq \frac{(2k-1)d}{p}.$$

Since the challenges $r_{i,0}, ..., r_{i,k-1}$ are chosen independently, We conclude

$$\Pr[\mathsf{State}_{i+1}(\{E_{i+1}^{(s)}\}_{s\in[d-1]}, r_0, ..., r_i, H_{i+1}) = 1] \leq \left(\frac{(2k-1)d}{p}\right)^k.$$

Equivalently, with probability at least $1 - \left(\frac{(2k-1)d}{p}\right)^k$, the protocol preserves falsity of the running claim from round i to round $i+1$. So the round-by-round soundness error of the packed sumcheck protocol is at most $\left(\frac{(2k-1)d}{p}\right)^k$.

Prover Complexity:

- Computation of $P_i(Y_i)$:
  The polynomial $P_i(Y_i)$ is the product of d terms, each a univariate polynomial of degree at most $2k-1$, so $\deg(P_i) \leq d(2k-1)$. While $P_i(Y_i)$ could be computed by interpolation using $d(2k-1)+1$ evaluations.
  1. Define

     $$\psi^{(s)}_{b_{[i+1,\mu-1]}}(Y_i) = \sum_{j=0}^{k-1} \mathcal{L}_{2j}(Y_i) f^{(s)}_{i,j}(0, b_{[i+1,\mu-1]}) + \mathcal{L}_{2j+1}(Y_i) f^{(s)}_{i,j}(1, b_{[i+1,\mu-1]})$$

     for each $s \in [0, d-1]$ and $b_{[i+1,\mu-1]} \in B_{\mu-1-i}$ with $\deg(\psi^{(s)}_{b_{[i+1,\mu-1]}}) \leq 2k-1$. And for any $l \in [0, 2k-1]$, we have $\psi^{(s)}_{b_{[i+1,\mu-1]}}(l) = f^{(s)}_{i,j}(c, b_{[i+1,\mu-1]})$, where $l = 2j + c$ for $j \in [0, k-1]$ and $c \in \{0, 1\}$. The remaining $d(2k-1) + 1 - (2k-1)$ evaluations are obtained via forward differences algorithm 1, requiring only $O(k^2d^2)$ additions.
  2. For each $b_{[i+1,\mu-1]}$, compute $\psi_{b_{[i+1,\mu-1]}}(Y_i) = \prod_{s=0}^{d-1} \psi^{(s)}_{b_{[i+1,\mu-1]}}(Y_i)$. at all points in $\{0, \ldots, d(2k-1)\}$. Using the precomputed factors from Step 1, each $b_{[i+1,\mu-1]}$ requires $d-1$ multiplications to form $\psi_{b_{[i+1,\mu-1]}}(Y_i)$.

  Overall, computing $P_i(Y_i)$ requires $d(2k-1)(d-1) \cdot 2^{\mu-1-i} = O(kd^2) \cdot 2^{\mu-1-i}$ multiplications.
- Computation of $E^{(s)}_{i+1}$: After receiving the verifier's challenge r_i, the prover updates the polynomials $E^{(s)}_i$ to $E^{(s)}_{i+1}$, for each $s \in [0, d-1]$. A direct implementation requires $2k^2d \cdot 2^{\mu-1-i}$ operations.

Thus, the i-th round costs

$$\big(d(2k-1)(d-1) + 2k^2d\big) \cdot 2^{\mu-1-i} \cdot \mathsf{mul} + O(k^2d^2) \cdot 2^{\mu-1-i} \cdot \mathsf{add}.$$

So the total prover complexity across all rounds is

$$O\big((kd^2 + k^2d)N\big) \cdot \mathsf{mul} + O(k^2d^2N) \cdot \mathsf{add}.$$

Remark 2. For polynomial updates, Strassen's algorithm can further reduce complexity, even when k is small. By partitioning each length-2^{l-1} vector into k-sized blocks, these combinations reduce to two $k \times k$ matrix multiplications per round. Using Strassen's algorithm (requiring $O(k^{2.807})$ multiplications and additions), the total cost becomes

$$O\big((kd^2 + k^{1.807}d)N\big) \cdot \mathsf{mul} + O(k^2d^2N) \cdot \mathsf{add}.$$

Verifier Complexity: In each round, the verifier needs to evaluate the polynomial P_i at $O(k)$ points and update H_{i+1}. Evaluating each point requires $O(kd)$ time, so the complexity per round is $O(k^2 d)$. Therefore, the total verifier complexity is $O(k^2 d\mu)$ field operations.
Communication Complexity: In each round, the prover sends a univariate polynomial $P_i(Y_i)$ of degree at most $(2k-1)d$, which can be represented using $O(kd)$ field elements. Over μ rounds, the total communication complexity is $O(kd\mu)$ field elements in $\mathbb{F}_p$. □

B.2 Proof of Theorem 2

Theorem 5 (Theorem 2). *For any packed-sumcheck transcript inducing the matrices $(M_0^{(0)}, M_0^{(1)}), \ldots, (M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)})$, the Brakedown-style commitment scheme above correctly opens the committed table to the value*

$$E^{(s)}(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}) \in \mathbb{F}_p^k$$

under honest computation. In the random oracle model, the scheme has negligible evaluation binding error and knowledge soundness error for these transcript-induced structured openings.

Proof. **Completeness:** Fix a packed sumcheck transcript, and hence fix the matrices $\{M_i^{(0)}, M_i^{(1)}\}_{i=0}^{\mu-1}$. Recall that for each $b = (b_0, \ldots, b_{\mu-1}) \in \{0,1\}^\mu$, the weight matrix is defined as

$$W_b = \prod_{i=\mu-1}^{0} \left((1-b_i)M_i^{(0)} + b_i M_i^{(1)}\right) \in \mathbb{F}_p^{k\times k}.$$

By the definition of the packed polynomial $E^{(s)}$, the desired opening value satisfies

$$E^{(s)}(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}) = \sum_{b\in\{0,1\}^\mu} W_b \cdot F^{(s)}(b)$$

where $F^{(s)}(b) \in \mathbb{F}_p^k$ is the hypercube evaluation table. To relate this to the Brakedown evaluation phase, we use the standard reshaping $F^{(s)} \mapsto U$: index $b \in \{0,1\}^\mu$ as $b = (b^{\mathrm{L}}, b^{\mathrm{R}})$ with $b^{\mathrm{L}}, b^{\mathrm{R}} \in \{0,1\}^{\mu/2}$, and set $U_{b^{\mathrm{L}}, b^{\mathrm{R}}} = F^{(s)}(b^{\mathrm{L}}, b^{\mathrm{R}})$. View each entry of U (an element of $\mathbb{F}_p^k$) as a length-k block, so that U can be regarded as a matrix in $\mathbb{F}_p^{(k\cdot 2^{\mu/2})\times 2^{\mu/2}}$.

Define the "left-half" products

$$W_{b^{\mathrm{L}}}^{\mathrm{L}} := \prod_{i=\mu/2-1}^{0} \left((1-b_i^{\mathrm{L}})M_i^{(0)} + b_i^{\mathrm{L}} M_i^{(1)}\right) \in \mathbb{F}_p^{k\times k},$$

and let $S_0 \in \mathbb{F}_p^{k\times(k\cdot 2^{\mu/2})}$ be the block matrix obtained by horizontally concatenating these$k \times k$ blocks in the order of b^{L}, $S_0 = \left[W_{b^{\mathrm{L}}}^{\mathrm{L}}\right]_{b^{\mathrm{L}}\in\{0,1\}^{\mu/2}}$. In the

evaluation phase, the honest prover returns $W = S_0 \cdot U$. Unpacking this block matrix product, for each fixed column index $b^{\mathrm{R}} \in \{0,1\}^{\mu/2}$, the corresponding output column (parsed as an element of $\mathbb{F}_p^k$) equals

$$\mathcal{W}_{b^{\mathrm{R}}} = \sum_{b^{\mathrm{L}} \in \{0,1\}^{\mu/2}} W^{\mathrm{L}}_{b^{\mathrm{L}}} \cdot F^{(s)}(b^{\mathrm{L}}, b^{\mathrm{R}}).$$

That is, $W = S_0 U$ computes exactly the prover-side linear aggregation of the table $F^{(s)}$ using the left-half weights.

Similarly, define the "right-half" products

$$W^{\mathrm{R}}_{b^{\mathrm{R}}} := \prod_{i=\mu-1}^{\mu/2} \left((1 - b^{\mathrm{R}}_i) M^{(0)}_i + b^{\mathrm{R}}_i M^{(1)}_i\right) \in \mathbb{F}_p^{k \times k},$$

and let the verifier's vector $\mathcal{T}$ be indexed so that $\mathcal{T}_{b^{\mathrm{R}}} = W^{\mathrm{R}}_{b^{\mathrm{R}}}$. The verifier outputs

$$Z = \sum_{b^{\mathrm{R}} \in \{0,1\}^{\mu/2}} \mathcal{T}_{b^{\mathrm{R}}} \cdot \mathcal{W}_{b^{\mathrm{R}}}.$$

Substituting the expression for $\mathcal{W}_{b^{\mathrm{R}}}$ and we gets

$$\begin{aligned} Z &= \sum_{b^{\mathrm{R}}} \sum_{b^{\mathrm{L}}} \left(W^{\mathrm{R}}_{b^{\mathrm{R}}} \cdot W^{\mathrm{L}}_{b^{\mathrm{L}}}\right) \cdot F^{(s)}(b^{\mathrm{L}}, b^{\mathrm{R}}) \\ &= \sum_{b \in \{0,1\}^{\mu}} W_b \cdot F^{(s)}(b) = E^{(s)}(M^{(0)}_0, M^{(1)}_0, \ldots, M^{(0)}_{\mu-1}, M^{(1)}_{\mu-1}), \end{aligned}$$

where we use that the overall weight W_b factors as the product of the right-half and left-half prefixes in the prescribed multiplication order. Therefore, the value Z computed by the verifier is exactly the correct opening of $E^{(s)}$ at the transcript-induced point.

Evaluation Binding and Knowledge Soundness: The proofs of evaluation binding and knowledge soundness follow mutatis mutandis from [57], which establishes these properties for a Brakedown-style commitment over arbitrary Galois rings $\mathrm{GR}(p^s, r)$ using a $k\times$ matrix challenge drawn from $\mathrm{GR}(p^s, r)$. Our setting is the direct specialization $s = 1, r = 1$ (so $\mathrm{GR}(p, 1) = \mathbb{F}_p$), under which the matrix challenge structure and the proof of [57] apply verbatim. □

B.3 Proof of Theorem 3

Theorem 6 (Theorem 3). *The packed ZeroCheck protocol satisfies the following properties:*

- *Completeness holds perfectly.*
- *Round-by-round soundness error is at most* $\max\left(\frac{\mu^k}{p^k}, \frac{(d(2k-1)+1)^k}{p^k}, \frac{((2k-1)(d+1))^k}{p^k}\right)$.
- *Prover complexity is* $O\left((kd^2 + k^2 d) \cdot 2^{\mu}\right) \cdot \mathsf{mul} + O\left(k^2 d^2 2^{\mu}\right) \cdot \mathsf{add}$.

– *Verifier complexity is* $O(k^3\mu)$.
– *Communication complexity is* $O\left(k^2 d + kd \cdot \mu\right)$ *field elements.*

Proof. **Completeness:** Assume $(\{[[f_i]]\}_{i\in[0,k)}; \{f_i\}_{i\in[0,k)}) \in \mathcal{R}_{\mathsf{ZeroCheck}}$, i.e., for every $i \in [0,k)$ and every $b \in \{0,1\}^\mu$ we have $f_i(b) = 0$. Then for every $b' \in \{0,1\}^{\mu-1}$ and every $i \in [0,k)$ the two boundary products satisfy

$$\prod_{s=0}^{d-1} f_i^{(s)}(0,b') = f_i(0,b') = 0, \qquad \prod_{s=0}^{d-1} f_i^{(s)}(1,b') = f_i(1,b') = 0.$$

By the definition of Φ, the checks in Step 3 evaluate the two boundary slices separately:

$$\Phi(t^{(i)}, 2j) = \sum_{b'\in\{0,1\}^{\mu-1}} \mathsf{eq}(\rho_i, b')\, f_j(0,b') = 0,$$

and

$$\Phi(t^{(i)}, 2j+1) = \sum_{b'\in\{0,1\}^{\mu-1}} \mathsf{eq}(\rho_i, b')\, f_j(1,b') = 0.$$

Hence all checks in Step 3 pass for an honest prover.

Conditioned on this, the challenges (c, r_0) are fresh randomness and H_1 is computed from the honest polynomial Φ. It remains to argue that the packed sumcheck statement in Step 5 holds under honest computation. The completeness analysis for this round is the same as that of the packed sumcheck protocol, with the only difference being that here $C(\cdot)$ is folded using a linear combination instead of Lagrange interpolation basis. By construction of $C(\cdot)$ and of the first-round fold $E_1^{(s)}$, the left-hand side of Step 5 is exactly the verifier's linear combination specified by (c, r_0) of the k multilinear slice-sums aggregated by Φ. Therefore an honest prover satisfies the packed sumcheck relation, and the packed sumcheck protocol accepts. This establishes perfect completeness.

Round-by-round soundness: Assume there exists $i^\star \in [0,k)$ such that $f_{i^\star}$ is not identically zero on $\{0,1\}^\mu$. Then there exists $b = (x_0, b') \in \{0,1\} \times \{0,1\}^{\mu-1}$ such that $f_{i^\star}(x_0, b') \neq 0$. In particular, at least one of the two $(\mu-1)$-variate polynomials $b' \mapsto f_{i^\star}(0,b')$ and $b' \mapsto f_{i^\star}(1,b')$ is nonzero. WLOG assume $b' \mapsto f_{i^\star}(0,b')$ is nonzero.

(A) Catching a false instance via random equality tests. For $\rho \in \mathbb{F}_p^{\mu-1}$ define

$$Z_{i^\star,0}(\rho) \;:=\; \sum_{b'\in\{0,1\}^{\mu-1}} \mathsf{eq}(\rho, b') \cdot f_{i^\star}(0,b').$$

As a polynomial in ρ, $Z_{i^\star,0}$ is not identically zero and has total degree at most $\mu-1$. By Schwartz–Zippel, for uniformly random $\rho \in \mathbb{F}_p^{\mu-1}$, $\Pr\left[Z_{i^\star,0}(\rho) = 0\right] \le \frac{\mu-1}{p}$. The verifier samples k independent points $\rho_0, \ldots, \rho_{k-1}$, so the probability that all of them are zeros of $Z_{i^\star,0}$ is at most $\left(\frac{\mu-1}{p}\right)^k \le \frac{\mu^k}{p^k}$. Equivalently, except with probability $\le \frac{\mu^k}{p^k}$, there exists $j \in [0,k)$ such that the slice claim $\sum_{b'} \mathsf{eq}(\rho_j, b') \cdot f_{i^\star}(0,b') = 0$ is false.

(B) Packing slice instances into a single Φ. $\Phi(t_0, \ldots, t_{k-1}, y)$ in Step 2 folds the $2k^2$ slice sumcheck instances $\sum_{b' \in \{0,1\}^{\mu-1}} \mathsf{eq}(\rho_j, b') \cdot f_i(0, b') = 0$ and $\sum_{b' \in \{0,1\}^{\mu-1}} \mathsf{eq}(\rho_j, b') \cdot f_i(1, b') = 0$ together. Thus, if any slice instance is false, then the honest Φ (determined by the witness) cannot satisfy all the individual boundary equalities $\Phi(t^{(i)}, 2j) = 0$ and $\Phi(t^{(i)}, 2j+1) = 0$.

Let $\widetilde{\Phi}$ be the polynomial sent by the prover and define $\Delta(t, y) := \widetilde{\Phi}(t, y) - \Phi(t, y)$. As a polynomial in t, y, $\deg_y(\Phi) \le d(2k-1)$, and including the linear dependence on t gives $\deg(\Delta) \le d(2k-1)+1$. If the prover passes Step 3 while some slice instance is false, then $\widetilde{\Phi}$ must differ from the honest Φ at at least one boundary point; hence Δ is nonzero. For the resulting claim to remain consistent with the packed sumcheck instance in Step 5, Δ must vanish at all k fresh challenge points. By the Schwartz–Zippel lemma, this occurs with probability at most $\frac{(d(2k-1)+1)^k}{p^k}$.

(C) Soundness of the packed sumcheck stage. Conditioned on Step 3 passing and on the challenges (c, r_0), the remaining claim is exactly the packed sumcheck instance in Step 5. By the round-by-round soundness of packed sumcheck, the additional failure probability is at most $\frac{((2k-1)(d+1))^k}{p^k}$.

Combining (A)–(C) and taking the maximum yields the stated round-by-round soundness error:

$$\max\left\{\frac{\mu^k}{p^k},\ \frac{(d(2k-1)+1)^k}{p^k},\ \frac{((2k-1)(d+1))^k}{p^k}\right\}.$$

Prover complexity: Let $N = 2^\mu$. Consider $\Phi(t_0, \ldots, t_{k-1}, y)$:

$$\Phi(t_0, \ldots, t_{k-1}, y) = \sum_{b' \in \{0,1\}^{\mu-1}} \Big((\sum_{j=0}^{k-1} t_j \cdot \mathsf{eq}(\rho_j, b')) \cdot \prod_{s=0}^{d-1} \Big(\sum_{i=0}^{k-1} \mathcal{L}_{2i}(y) f_i^{(s)}(0, b') + \mathcal{L}_{2i+1}(y) f_i^{(s)}(1, b') \Big) \Big)$$

$$= \sum_{j=0}^{k-1} t_j \cdot \sum_{b' \in \{0,1\}^{\mu-1}} \mathsf{eq}(\rho_j, b') \prod_{s=0}^{d-1} \Big(\sum_{i=0}^{k-1} \mathcal{L}_{2i}(y) f_i^{(s)}(0, b') + \mathcal{L}_{2i+1}(y) f_i^{(s)}(1, b') \Big)$$

Define

$$g_{\rho_j}(y) = \sum_{b' \in \{0,1\}^{\mu-1}} \mathsf{eq}(\rho_j, b') \prod_{s=0}^{d-1} \Big(\sum_{i=0}^{k-1} \mathcal{L}_{2i}(y) f_i^{(s)}(0, b') + \mathcal{L}_{2i+1}(y) f_i^{(s)}(1, b') \Big).$$

The degree of $g_{\rho_j}(y)$ is at most $d(2k-1)$. From the expression of $g_{\rho_i}(y)$, if we can obtain $g_{\rho_0}, \ldots, g_{rho_{k-1}}$, then we can derive the expression of $\Phi(t_0, \ldots, t_{k-1}, y)$. We next evaluate these polynomials over $[0, (2k-1)d]$.

1. Define the polynomial components:

$$\phi_{b'}^{(s)}(y) = \sum_{i=0}^{k-1} \mathcal{L}_{2i}(y) f_i^{(s)}(0, b') + \mathcal{L}_{2i+1}(y) f_i^{(s)}(1, b'), \quad \phi_{b'}(y) = \prod_{s=0}^{d-1} \phi_{b'}^{(s)}(y).$$

Here $\deg_y(\phi_{b'}^{(s)}) \le 2k-1$. And for any $j \in [0, 2k-1]$, we have $\psi^{(s)}_{b_{[i+1,\mu-1]}}(j) = f^{(s)}_{\lfloor j/2 \rfloor}(j \pmod 2, b_{[i+1,\mu-1]})$. The remaining $d(2k-1)+1-(2k-1)$ evaluations are obtained via forward differences algorithm 1, requiring only $O(k^2d^2)$ additions. Compute the evaluations $\phi_{b'}(y)$ over $[0, d(2k-1)]$ need $(d(2k-1)+1)(d-1)$ multiplications.

2. For each ρ_j,

$$g_{\rho_j}(y) = \sum_{b' \in \{0,1\}^{\mu-1}} \mathsf{eq}(\rho_j, b') \prod_{s=0}^{d-1} \Big(\sum_{i=0}^{k-1} \mathcal{L}_{2i}(y) f_i^{(s)}(0, b') + \mathcal{L}_{2i+1}(y) f_i^{(s)}(1, b') \Big).$$

where $\mathsf{eq}(\rho_j, b')$ is a multilinear polynomial evaluation at the point ρ_j. Computing $\{g_{\rho_j}(y)\}_{j \in [0,k-1]}$ requires $O(k^2dN)$ multiplications and additions, where $N = 2^\mu$.

The prover then computes C and $E_1^{(s)}$ for each $s \in [0, d)$, which costs $O(k^2dN)$ multiplications and additions.

Therefore, before executing the packed sumcheck, the total complexity is $O((kd^2 + k^2d)N) \cdot \mathsf{mul} + O(k^2d^2N) \cdot \mathsf{add}$, which matches the complexity of the packed sumcheck itself. Hence, the overall prover complexity for ZeroCheck is

$$O((kd^2 + k^2d)N) \cdot \mathsf{mul} + O(k^2d^2N) \cdot \mathsf{add}.$$

Verifier complexity: The verifier's complexity mainly arises from evaluating $C(\cdot)$ during the packed sumcheck protocol. Although C is defined as a sum over all $b' \in \{0,1\}^{\mu-1}$, the verifier never evaluates it by enumerating the hypercube. Instead, it exploits the linear form of $EQ(b')$ and the tensor product structure of multilinear openings.

Let $\{\rho_i\}_{i=0}^{k-1} \subseteq \mathbb{F}_p^{\mu-1}$ be the k random points sampled by the verifier in Step 1, and write $\rho_i = (\rho_{i,1}, \ldots, \rho_{i,\mu-1})$. Recall that

$$C\Big(X_1^{(0)}, X_1^{(1)}, \ldots, X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}\Big) = \sum_{b' \in \{0,1\}^{\mu-1}} \left(\prod_{j=1}^{\mu-1} \big((1-b'_j)X_j^{(0)} + b'_j X_j^{(1)}\big) \right) \cdot EQ(b'),$$

and that $EQ(b') \in \mathbb{F}_p^k$ is a linear combination of equality polynomials:

$$EQ(b') = \sum_{i=0}^{k-1} t_{\cdot,i} \cdot \mathsf{eq}(\rho_i, b'),$$

where $t_{\cdot,i} \in \mathbb{F}_p^k$ denotes the i-th column of the coefficient matrix (so the u-th entry of $t_{\cdot,i}$ is $t_{u,i}$).

By linearity we may swap the order of summation and write

$$C(X_1^{(0)}, X_1^{(1)}, \ldots, X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)}) = \sum_{i=0}^{k-1} v_i,$$

$$v_i \;:=\; \sum_{b' \in \{0,1\}^{\mu-1}} \left(\prod_{j=1}^{\mu-1} \big((1-b'_j)X_j^{(0)} + b'_j X_j^{(1)}\big) \right) \cdot \mathsf{eq}(\rho_i, b') \cdot t_{\cdot,i}.$$

The key point is that v_i can be computed *without* enumerating b': it is exactly a matrix-weighted multilinear opening at the point ρ_i, analogous to standard MLE evaluation whose coefficients factor as a tensor product. Concretely, define the interpolated matrices

$$A_{i,j} := (1 - \rho_{i,j})X_j^{(0)} + \rho_{i,j}X_j^{(1)} \in \mathbb{F}_p^{k\times k},$$

and compute v_i via the recursion

$$v_i^{(0)} := t_{\cdot,i}, \qquad v_i^{(j)} := A_{i,j}\, v_i^{(j-1)} \ \ (j = 1, \ldots, \mu - 1), \qquad v_i := v_i^{(\mu-1)}.$$

Thus the verifier evaluates $C(\cdot)$ by computing all v_i and summing them. Each update is one $k \times k$ matrix–vector multiplication ($O(k^2)$ field operations), performed $(\mu - 1)$ times for each of the k indices i. Therefore the total verifier time is $O(\mu k^3)$.

Communication complexity: The communication complexity mainly comes from the polynomial Φ, is of size $O(k(2k-1)d) = O(k^2 d)$. The packed sumcheck protocol requires $O(kd\mu)$ field elements. Therefore, the total communication complexity is $O(k^2 d + kd\mu)$ field elements. □

Lemma 2. *The permutation argument based on the packed sumcheck protocol satisfies the following properties:*

- *Completeness holds perfectly.*
- *Round-by-round soundness error is at most*

$$\frac{\mu^k}{p^k} + \frac{((2k-1)(\mu+1)+1)^k}{p^k}.$$

- *Prover complexity is*

$$O((k\mu^2 + k^2\mu)N) \cdot \mathsf{mul} + O(k^2\mu^2 N) \cdot \mathsf{add},$$

  where $N = 2^\mu$ and mul and add denote the costs of a field multiplication and addition in $\mathbb{F}_p$ respectively;
- *Verifier complexity is $O(k^2\mu^2)$ field operations;*
- *Communication complexity is $O(k\mu^2)$ field elements in $\mathbb{F}_p$.*

Proof. The completeness is straightforward. We next analyze the round-by-round soundness. If the constraint is violated, then the two multilinear polynomials differ: $f(x) \neq g(\sigma(x))$. Then the multilinear polynomial $g(x)$ and $h(x) = \sum_{y\in B_\mu} f(y)\cdot \mathsf{eq}(\tilde{\sigma}(y), x)$ are not identical. So their evaluations at k random points $\{\rho_i\}_{i\in[0,k)}$ are all equal with probability at most $\frac{\mu^k}{p^k}$.

If there exists i^* such that the equality $h(\rho_{i^*}) \neq g(\rho_{i^*})$ holds, then according to the soundness analysis of the packed sumcheck protocol, the round-by-round soundness error does not exceed $\frac{((2k-1)(\mu+1)+1)^k}{p^k}$, because the degree of sumcheck instances is $\mu + 1$.

And the prover complexity, verifier complexity, and communication complexity can be analyzed similarly to the packed sumcheck protocol. The only difference is that the degree of the polynomial in this case is $\mu + 1$. Therefore, we omit the detailed analysis here. □

C MW-MLE Complexity Analysis

C.1 Brakedown MW-MLE Complexity Analysis

- **Prover commit time.** The prover performs RS evaluation on an evaluation set $W \subseteq \mathbb{F}_p$. Because evaluation points lie in the base field, the encoding can be implemented as base field scalar multiplications applied to the expanded kN entries. Hence the commit-time arithmetic cost is

$$\tilde{O}\left(\frac{kN}{\alpha}\right) \quad \text{base field multiplications in } \mathbb{F}_p,$$

  Building the Merkle tree requires $O(k^2\sqrt{N} \cdot m) = O(kN/\alpha)$ hashing work.
- **Prover opening/evaluation time.** In both the testing and the structured opening, the prover computes linear aggregations of the form $V = R \cdot U$ and $W = S_0 \cdot U$, where $R, S_0 \in \mathbb{F}_p^{k \times (k\sqrt{N})}$ and $U \in \mathbb{F}_p^{(k\sqrt{N}) \times \sqrt{N}}$. Each multiplication costs

$$k \cdot (k\sqrt{N}) \cdot \sqrt{N} \;=\; k^2 N \quad \text{base field multiplications,}$$

  so computing both V and W costs $O(k^2 N)$ base field operations. To answer the Q spot checks, the prover reveals Q columns of $\widehat{U}$ and their authentication paths. This requires transmitting $O(Qk\sqrt{N})$ symbols over $\mathbb{F}_{p^k}$ and $O(Q \log m)$ hash values; the corresponding linear consistency and Merkle-path checks are performed by the verifier.
- **Proof size.** Opening one column reveals $k\sqrt{N}$ symbols over $\mathbb{F}_{p^k}$ plus a Merkle path containing $O(\log m)$ hash values. Therefore the total proof size is

$$O(Qk\sqrt{N}) \text{ symbols over } \mathbb{F}_{p^k} \quad \text{and} \quad O(Q \log m) \text{ hash values.}$$

  Equivalently, the field-element contribution is $O(Qk^2\sqrt{N})$ elements in $\mathbb{F}_p$.
- **Verifier time.** The verifier performs Q spot checks, each involving Merkle verification and linear checks for the opened rows. These checks touch k rows, each with a length-$(k\sqrt{N})$ linear relation over $\mathbb{F}_{p^k}$ with base field scalars, which costs $O(k^3\sqrt{N})$ base field operations per check. Hence total verifier time is

$$O(Qk^3\sqrt{N}) \quad \text{base field operations.}$$

  Merkle verification additionally requires $O\big(Q(k^2\sqrt{N} + \log m)\big)$ hashing work, including hashing the opened columns and checking their authentication paths.

Relation to Binius. Binius packs k length-N tables over $\mathbb{F}_p$ into one length-N table over $\mathbb{F}_{p^k}$ and commits over $\mathbb{F}_{p^k}$. Under the standard model where one $\mathbb{F}_{p^k}$ multiplication costs $\Theta(k^2)$ multiplications in $\mathbb{F}_p$, its commit phase costs $\tilde{O}(kN/\alpha)$ base field multiplications and its opening phase costs $O(k^2 N)$ base field multiplications. These are the same two dominant complexity scales that arise in our instantiation.

Remark 3 (Eliminating the separate testing phase). The separate testing phase of MW-Brakedown can be eliminated by extending the optimization of Diamond and Posen [29]. In the resulting variant, the structured challenges used to verify the evaluation claim also provide the randomness for proximity testing.

To justify this extension, recall that the argument of Diamond and Posen uses the tensor-product evaluation coefficients to form random combinations of the committed words. In MW-Brakedown, these scalar coefficients are replaced by matrix-product coefficients. Once the prover has fixed the committed words, each row of a fresh challenge matrix induces an independent low-degree polynomial combination of their interleaved Reed–Solomon words. The Reed–Solomon proximity-gap result of Ben-Sasson et al. [10] therefore applies to each row. Let $d_{\mathcal{C}}$ and $n_{\mathcal{C}}$ denote the minimum distance and block length of the code, respectively, and fix $0 \le e \le \lfloor (d_{\mathcal{C}} - 1)/3 \rfloor$. The false-witness probability for an affine combination is at most $(e+1)/p$. Since each row of our structured aggregation is a univariate polynomial combination of degree at most $2k-1$, the single-parameter Reed–Solomon bound increases this probability by at most a factor of $2k-1$. Thus, one row fails to preserve the distance with probability at most $\frac{(2k-1)(e+1)}{p}$.

The k rows use independent challenges, so the probability that they all fail at one aggregation level is at most the k-th power of this quantity. A union bound over the $\mu/2$ levels, followed by the Q evaluation-phase spot checks, gives the overall proximity-testing error

$$\frac{\mu}{2}\left(\frac{(2k-1)(e+1)}{p}\right)^{k} + \left(1 - \frac{e+1}{n_{\mathcal{C}}}\right)^{Q} + \varepsilon_{\mathsf{hash}},$$

where $\varepsilon_{\mathsf{hash}}$ is the hash-collision probability.

This bound is weaker than the bound obtained from the fully random matrices in the separate testing phase, because the matrix-product coefficients give low-degree polynomial, rather than affine, combinations. We therefore retain the separate testing phase in the construction analyzed above, although it may be omitted when the parameters make the weaker bound negligible.

D Basefold-based MW-MLE Commitment

D.1 Structured MW-MLE commitment based on Basefold

Basefold introduces a FRI-like IOPP for *foldable* codes, i.e., an interactive oracle proof that a given oracle is close to the code induced by a recursive foldable encoding. A central feature of Basefold is that this IOPP is executed in parallel with a sumcheck protocol: the verifier's folding challenges used by the IOPP are reused as the sumcheck randomness, so that the proximity test and the sumcheck reduction proceed along the same folding trajectory and ultimately certify the required evaluation.

Our packed sumcheck protocol fits the same template. The multilinear polynomial $E^{(s)}$ is progressively folded round by round, and the verifier transcript

determines a sequence of folding matrices $\{(M_i^{(0)}, M_i^{(1)})\}_{i=0}^{\mu-1}$ that specifies the evaluation to be checked at the end. Moreover, packed sumcheck achieves soundness via repetition across k coordinates, which naturally corresponds to reasoning about an interleaving of k codewords together with a block-wise distance measure. This motivates extending the Basefold IOPP from foldable codes to interleaved foldable codes, in a way that preserves the correlated-agreement principles underlying Basefold's soundness analysis.

In our instantiation, we commit to the evaluation table of $E^{(s)}$ using an interleaved foldable code and then run a Basefold-style IOPP in lockstep with packed sumcheck, reusing the packed-sumcheck challenges as the folding challenges in the IOPP. At the end of the interaction, the verifier obtains exactly the evaluation required by packed sumcheck, $E^{(s)}\big(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-1}^{(0)}, M_{\mu-1}^{(1)}\big)$, without introducing an additional, independent randomness stream or an extra folding schedule beyond what packed sumcheck already induces.

Definition 6 (Interleaved Code and Block-wise relative distance). *Let $\mathcal{C} \subseteq \mathbb{F}^n$ be a linear $[n, \alpha n, \gamma n]$ code over a field $\mathbb{F}$. The k-fold interleaving of $\mathcal{C}$ is*

$$\mathcal{C}^{\times k} \;:=\; \{(c_0, \ldots, c_{k-1}) \in (\mathbb{F}^n)^k \mid c_i \in \mathcal{C} \text{ for all } i \in [k]\},$$

which we view as the set of matrices in $\mathbb{F}^{k\times n}$ whose i-th row is a codeword in $\mathcal{C}$. For a matrix $A \in \mathbb{F}^{k\times n}$, we define the block-wise relative distance to $\mathcal{C}^{\times k}$ as

$$\Delta_{\mathrm{blk}}(A, \mathcal{C}^{\times k}) \;:=\; \min_{C\in\mathcal{C}^{\times k}} \frac{1}{n} \cdot \Big|\big\{j \in [n] \;:\; A_{\cdot,j} \neq C_{\cdot,j}\big\}\Big|,$$

where $A_{\cdot,j} \in \mathbb{F}^k$ denotes the j-th column of A (and similarly for C). Equivalently, Δ_{blk} measures the fraction of column positions where at least one row disagrees.

Definition 7 (Foldable Linear Code over $\mathbb{F}_{p^k}$). *A foldable linear code family $\{\mathcal{C}_i\}_{i=0}^{\mu}$ over $\mathbb{F}_{p^k}$ is specified by generator matrices $(G_0, \ldots, G_\mu)$ with $G_i \in \mathbb{F}_{p^k}^{\alpha_i n_i \times n_i}$, such that for every $i \in [\mu]$,*

$$G_i \;=\; \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1}\cdot T_{i-1} & G_{i-1}\cdot T'_{i-1} \end{bmatrix},$$

where $T_{i-1}, T'_{i-1} \in \mathbb{F}_{p^k}^{n_{i-1}\times n_{i-1}}$ are diagonal matrices satisfying $\mathsf{diag}(T_{i-1})[j] \neq \mathsf{diag}(T'_{i-1})[j]$ for all $j \in [n_{i-1}]$. We write $\mathsf{Enc}_i : \mathbb{F}_{p^k}^{\alpha_i n_i} \to \mathbb{F}_{p^k}^{n_i}$ for the encoding map induced by G_i.

For block-level Reed–Solomon codes built from an evaluation domain $W = \{w^1, \ldots, w^n\} \subseteq \mathbb{F}_p$ (NTT subgroup), the diagonal entries are precisely the paired evaluation points. Concretely, if we pair indices $(j, j + n/2)$ then

$$t = \mathrm{diag}(T)[j] = w^j, \qquad t' = \mathrm{diag}(T')[j] = w^{j+n/2} = w^j \cdot w^{n/2} = -w^j, \quad (2)$$

using $w^{n/2} = -1$ in an NTT subgroup of even order. In particular, $t, t' \in \mathbb{F}_p$. This is the crucial reason the "inverse step" in folding does not require extension field arithmetic.

Definition 8 (Coset relative minimum distance). *Let n be an even integer and let $\mathcal{C} \subseteq \mathbb{F}^n$ be a linear code. For a word $\mathbf{v} \in \mathbb{F}^n$ and a codeword $\mathbf{c} \in \mathcal{C}$, the* coset relative distance *between* $\mathbf{v}$ *and* $\mathbf{c}$ *is*

$$\Delta^*(\mathbf{v}, \mathbf{c}) := \frac{2}{n} \cdot \Big| \{ j \in [n/2] \ : \ \mathbf{v}[j] \neq \mathbf{c}[j] \ \ \textit{or} \ \ \mathbf{v}[j + n/2] \neq \mathbf{c}[j + n/2] \} \Big|.$$

The coset relative minimum distance *of* $\mathbf{v}$ *to the code* C *is*

$$\Delta^*(\mathbf{v}, C) := \min_{\mathbf{c} \in C} \Delta^*(\mathbf{v}, \mathbf{c}).$$

For a interleaved codeword $C \in \mathbb{F}^{k \times n}$, the coset block-wise relative distance *between a matrix $A \in \mathbb{F}^{k \times n}$ and C is*

$$\Delta^*_{\mathrm{blk}}(A, C) := \frac{2}{n} \cdot \Big| \{ j \in [n/2] \ : \ A_{\cdot,j} \neq C_{\cdot,j} \ \textit{or} \ A_{\cdot,j+n/2} \neq C_{\cdot,j+n/2} \} \Big|.$$

The coset relative minimum distance *of* $\mathbf{A}$ *to the code* $\mathcal{C}^{\times k}$ *is*

$$\Delta^*_{\mathrm{blk}}(A, \mathcal{C}^{\times k}) := \min_{C \in \mathcal{C}^{\times k}} \Delta^*_{\mathrm{blk}}(A, C).$$

Structured MW-MLE based on Basefold Construction.
Parameters.

Let p be a prime and $k \geq 1$ such that $p^k = \Theta(2^\lambda)$. Let E be a multilinear polynomial whose evaluation table over the hypercube $B_\mu^{\mathsf{mat}} = \{L, R\}^\mu$ is denoted by $F \in (\mathbb{F}_p^k)^{2^\mu}$. Let $\mathcal{C}_\mu$ be a foldable code over $\mathbb{F}_{p^k}$ with encoding algorithm Enc as in definition 7. Let $Q = \Theta(\lambda)$ be the repetition parameter.

Committing Phase.

The prover view F as a matrix $U \in \mathbb{F}_p^{k \times 2^\mu}$ whose j-th column is the length-k vector $F_{\cdot,j} \in \mathbb{F}_p^k$. The prover treats every k adjacent elements in each row of the matrix U as a single element in $\mathbb{F}_{p^k}$, obtaining a packed row in $\mathbb{F}_{p^k}^{2^\mu/k}$. It then encodes each packed row using Enc to obtain the matrix $\hat{U} \in \mathbb{F}_{p^k}^{k \times n_\mu}$ (where n_μ is the block length of $\mathcal{C}_\mu$). The prover commits $\hat{U}$ by Merkle Tree and sends the Merkle root π_0.

Evaluation phase in parallel with packed sumcheck. For rounds $i = 1, 2, \ldots, \mu - \log k$:

1. **Prover folding and commitment.**
   (a) Upon receiving the packed-sumcheck challenges $r_{i-1} \in \mathbb{F}_p^k$, the prover derives the folding matrices $M_{i-1}^{(0)}, M_{i-1}^{(1)} \in \mathbb{F}_p^{k \times k}$ (as defined by packed sumcheck) and views them as matrices over $\mathbb{F}_{p^k}$ via the canonical embedding $\mathbb{F}_p \mapsto \mathbb{F}_{p^k}$.
   (b) For each $j \in [n_{\mu-i} - 1]$, let $\mathbf{a}, \mathbf{b} \in \mathbb{F}_{p^k}^k$ be the j-th and $(n_{\mu-i} + j)$-th columns of the current matrix $\hat{U}^{(\mu-i+1)}$. Let $t := \mathsf{diag}(T_{\mu-i})[j]$ and $t' := \mathsf{diag}(T'_{\mu-i})[j]$. Because $t \neq t'$, so the prover computes $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{p^k}^k$ as the unique solution to $\mathbf{a} = \mathbf{x} + t\mathbf{y}$, $\mathbf{b} = \mathbf{x} + t'\mathbf{y}$, and sets the j-th column of the folded matrix $\hat{U}^{(\mu-i)}$ to $\hat{U}^{(\mu-i)}_{\cdot,j} := M_{i-1}^{(0)}\mathbf{x} + M_{i-1}^{(1)}\mathbf{y}$.

(c) The prover commits to $\hat{U}^{(\mu-i)}$ by Merkle Tree and sends the Merkle root π_i.

2. **Verifier queries.** The verifier samples a random query set $S_i \subseteq [n_{\mu-i}]$ of size $|S_i| = Q$ and sends S_i to the prover.
3. **Openings.** For every $j \in S_i$, the prover sends:
   - the column $\hat{U}^{(\mu-i)}_{\cdot,j}$ and its Merkle path of π_i,
   - the two parent columns $\hat{U}^{(\mu-i+1)}_{\cdot,j}$ and $\hat{U}^{(\mu-i+1)}_{\cdot,n_{\mu-i}+j}$ and their Merkle paths of π_{i-1}.
4. **Verifier checks.** For each $j \in S_i$, the verifier:
   (a) verifies the Merkle proofs;
   (b) recomputes $\mathbf{x}, \mathbf{y}$ from $\mathbf{a}, \mathbf{b}$ using t, t', where $(\mathbf{a}, \mathbf{b}) = (\hat{U}^{(\mu-i+1)}_{\cdot,j}, \hat{U}^{(\mu-i+1)}_{\cdot,n_{\mu-i}+j})$;
   (c) checks $\hat{U}^{(\mu-i)}_{\cdot,j} \stackrel{?}{=} M^{(0)}_{i-1}\mathbf{x} + M^{(1)}_{i-1}\mathbf{y}$.

Output. After $\mu - \log k$ rounds, the prover sends $\hat{U}^{(\log k)}$ to the verifier . It first checks that $\hat{U}^{(\log k)}$ is a valid interleaving of codewords in $\mathcal{C}_{\log k}$ (i.e., each row lies in $\mathcal{C}_{\log k}$). It then decodes $\hat{U}^{(\log k)}$ row-wise and unpacks the resulting $\mathbb{F}_{p^k}$-symbols back into $\mathbb{F}_p$, obtaining the partial evaluation table for the remaining $\log k$ variables, corresponding to

$$E^{(s)}\big(M^{(0)}_0, M^{(1)}_0, \ldots, M^{(1)}_{\mu-\log k-1}, (X^{(0)}_{\mu-\log k}, X^{(1)}_{\mu-\log k}), \ldots, (X^{(0)}_{\mu-1}, X^{(1)}_{\mu-1})\big),$$

after which the verifier completes the last $\log k$ packed-sumcheck rounds locally.

Theorem 7. *The structured MW-MLE commitment based on Basefold described above is complete.*

Theorem 8. *Let $\mathcal{C}_\mu$ be an $(n_\mu, \alpha_\mu n_\mu, \gamma_\mu n_\mu)$-foldable code over $\mathbb{F}_{p^k}$ with generator matrices $(G_0, \ldots, G_\mu)$. For each $i \in \{0, \ldots, \mu\}$, let $\mathcal{C}_i$ denote the code generated by G_i, and assume $\gamma_i \geq \gamma_{i+1}$. Fix $\epsilon \in (0, 1/3)$ and set*

$$\delta \;:=\; \min\Big\{\Delta^*_{\mathrm{blk}}\big(\hat{U}^{(\mu)}, \mathcal{C}^{\times k}_\mu\big),\;\; 1-(1-\gamma_\mu+\epsilon)^{1/3}\Big\},$$

where $\Delta^*_{\mathrm{blk}}(\cdot,\cdot)$ is the coset block-wise relative distance. Then, except with probability at most $\frac{2\mu}{\epsilon^2 p^k}$ over the verifier's randomness, the following holds: for any prover strategy producing Merkle commitments $\pi_0, \ldots, \pi_{\mu-\log k}$, if the verifier accepts the evaluation phase and the final codeword check with probability at least $(1-\delta+\epsilon\mu)^Q$, then the verifier's final decoded table equals the partial evaluation table corresponding to

$$E^{(s)}\big(M^{(0)}_0, M^{(1)}_0, \ldots, M^{(0)}_{\mu-\log k-1}, M^{(1)}_{\mu-\log k-1}, (X^{(0)}_{\mu-\log k}, X^{(1)}_{\mu-\log k}), \ldots, (X^{(0)}_{\mu-1}, X^{(1)}_{\mu-1})\big).$$

Theorem 9. *Assume the Merkle hash is modeled as a random oracle. Then the Basefold-style commitment for packed sumcheck described above is knowledge-sound: for any PPT prover $\mathcal{P}^*$, if the verifier accepts the evaluation phase and the final codeword check with non-negligible probability and outputs a value z, there exists a PPT extractor $\mathcal{E}^{\mathcal{P}^*}$ that outputs a table $\widetilde{F}$ consistent with the initial commitment such that $z = \mathsf{Eval}(\widetilde{F}, \mathcal{M})$ for the structured query $\mathcal{M}$ induced by the packed-sumcheck transcript.*

D.2 Proof of Theorem 7

Theorem 10 (Theorem 7). *The Basefold commitment scheme for packed sum-check described above is complete.*

Proof. If the prover follows the protocol honestly, then the verifier's per-round checks in the evaluation phase Step 4 always pass by construction. It remains to show that the final matrix $\hat{U}^{(\log k)}$ is a valid interleaving at level $\log k$, namely $\hat{U}^{(\log k)} \in \mathcal{C}^{\times k}_{\log k}$, so that the verifier's final codeword check succeeds.

Let $U^{(\mu)} := U$. For each $i \in [1, \mu - \log k]$, write

$$U^{(\mu-i+1)} \;=\; \big(U_l^{(\mu-i+1)} \,\|\, U_r^{(\mu-i+1)}\big),$$

where the split is by columns into two equal halves, and define

$$U^{(\mu-i)} \;:=\; M_{i-1}^{(0)}\, U_l^{(\mu-i+1)} \;+\; M_{i-1}^{(1)}\, U_r^{(\mu-i+1)}.$$

We prove by induction on i that $\hat{U}^{(\mu-i)} = \mathsf{Enc}_{\mu-i}(U^{(\mu-i)})$; in particular, $\hat{U}^{(\mu-i)} \in \mathcal{C}^{\times k}_{\mu-i}$. The base case $i = 0$ holds since the prover computes $\hat{U}^{(\mu)}$ by encoding $U^{(\mu)}$ in the commit phase.

For the inductive step, assume $\hat{U}^{(\mu-i+1)} = \mathsf{Enc}_{\mu-i+1}(U^{(\mu-i+1)})$. By the foldable-code structure of the generator matrices,

$$G_{\mu-i+1} = \begin{bmatrix} G_{\mu-i} & G_{\mu-i} \\ G_{\mu-i}T_{\mu-i} & G_{\mu-i}T'_{\mu-i} \end{bmatrix},$$

with $T_{\mu-i}, T'_{\mu-i}$ diagonal. Expanding the encoding of the concatenation $U^{(\mu-i+1)} = (U_l \| U_r)$ gives that for every column index $j \in [n_{\mu-i}]$,

$$\hat{U}^{(\mu-i+1)}_{\cdot,j} \;=\; \mathsf{Enc}_{\mu-i}(U_l)_{\cdot,j} \;+\; \mathsf{diag}(T_{\mu-i})[j] \cdot \mathsf{Enc}_{\mu-i}(U_r)_{\cdot,j},$$

Since $\mathsf{diag}(T_{\mu-i})[j] \neq \mathsf{diag}(T'_{\mu-i})[j]$, these two columns uniquely determine the pair

$$\mathbf{x}_j := \mathsf{Enc}_{\mu-i}(U_l)_{\cdot,j}, \qquad \mathbf{y}_j := \mathsf{Enc}_{\mu-i}(U_r)_{\cdot,j},$$

as the (unique) solution to $\mathbf{a} = \mathbf{x}_j + t\mathbf{y}_j$ and $\mathbf{b} = \mathbf{x}_j + t'\mathbf{y}_j$, where $\mathbf{a}, \mathbf{b}$ are the two parent columns of $\hat{U}^{\mu-i+1}$ and t, t' are the corresponding diagonal entries. This is exactly the computation performed by the honest prover in the folding step.

The prover then sets the folded column to

$$\hat{U}^{(\mu-i)}_{\cdot,j} \;:=\; M_i^{(0)}\mathbf{x}_j \;+\; M_i^{(1)}\mathbf{y}_j.$$

Using linearity of $\mathsf{Enc}_{\mu-i}$ and the fact that $M_i^{(0)}, M_i^{(1)}$ are over $\mathbb{F}_p \subseteq \mathbb{F}_{p^k}$, we obtain

$$\begin{aligned} \hat{U}^{(\mu-i)}_{\cdot,j} &= M_i^{(0)}\mathsf{Enc}_{\mu-i}(U_l)_{\cdot,j} + M_i^{(1)}\mathsf{Enc}_{\mu-i}(U_r)_{\cdot,j} \\ &= \mathsf{Enc}_{\mu-i}\big(M_i^{(0)}U_l + M_i^{(1)}U_r\big)_{\cdot,j} = \mathsf{Enc}_{\mu-i}(U^{(\mu-i)})_{\cdot,j}. \end{aligned}$$

Thus $\hat{U}^{(\mu-i)} = \mathsf{Enc}_{\mu-i}(U^{(\mu-i)})$, completing the induction.

Taking $i = \mu - \log k$, we conclude that $\hat{U}^{(\log k)} = \mathsf{Enc}_{\log k}(U^{(\log k)})$ and hence $\hat{U}^{(\log k)} \in \mathcal{C}_{\log k}^{\times k}$. Therefore the verifier's final codeword check succeeds, and completeness holds. □

D.3 Proof of Theorem 8

Theorem 11 (Theorem 8). *Let $\mathcal{C}_\mu$ be an $(n_\mu, \alpha_\mu n_\mu, \gamma_\mu n_\mu)$-foldable code over $\mathbb{F}_{p^k}$ with generator matrices $(G_0, \ldots, G_\mu)$. For each $i \in \{0, \ldots, \mu\}$, let $\mathcal{C}_i$ denote the code generated by G_i, and assume $\gamma_i \geq \gamma_{i+1}$. Fix $\epsilon \in (0, 1/3)$ and set*

$$\delta \;:=\; \min\Big\{\Delta^*_{\mathrm{blk}}\big(\hat{U}^{(\mu)}, \mathcal{C}_\mu^{\times k}\big),\quad 1-(1-\gamma_\mu+\epsilon)^{1/3}\Big\},$$

where $\Delta^*_{\mathrm{blk}}(\cdot,\cdot)$ is the coset block-wise relative distance. Then, except with probability at most $\frac{2\mu}{\epsilon^2 p^k}$ over the verifier's randomness, the following holds: for any prover strategy producing Merkle commitments $\pi_0, \ldots, \pi_{\mu-\log k}$, if the verifier accepts the evaluation phase and the final codeword check with probability at least $(1-\delta+\epsilon\mu)^Q$, then the verifier's final decoded table equals the partial evaluation table corresponding to

$$E^{(s)}\big(M_0^{(0)}, M_0^{(1)}, \ldots, M_{\mu-\log k-1}^{(0)}, M_{\mu-\log k-1}^{(1)}, (X_{\mu-\log k}^{(0)}, X_{\mu-\log k}^{(1)}), \ldots, (X_{\mu-1}^{(0)}, X_{\mu-1}^{(1)})\big).$$

Proof. We first define a bad event B: Let $(r_0, \ldots, r_{\mu-\log k-1})$ be the folding challenges output by the verifier and let $\pi_0, \ldots, \pi_{\mu-\log k}$ be the prover oracles. Intuitively, the bad event B occurs if there exists some $i \in [0, \mu - \log k - 1]$, the "folding" of oracle π_i with challenge r_i has significantly smaller block-wise distance to $\mathcal{C}_{\mu-i-1}^{\times k}$ compare to the block-wise distance of oracle π_i to $\mathcal{C}_{\mu-i}^{\times k}$. More formally, the bad event B happens if there exists some round $i \in [0, \mu-\log k-1]$ such that

$$\Delta_{\mathrm{blk}}(\mathsf{fold}_{r_i}(\pi_i), \mathcal{C}_{\mu-i-1}^{\times k}) \leq \min(\Delta^*_{\mathrm{blk}}(\pi_i, \mathcal{C}_{\mu-i}^{\times k}), 1-(1-\gamma_\mu+\epsilon)^{1/3}) - \epsilon.$$

where Δ^*_{blk} is the coset block-wise relative distance and $\mathsf{fold}_{r_i}(\pi_i)$ is defined as follows: let $V, V' \in \mathbb{F}_{p^k}^{k \times n_{\mu-i-1}}$ be the unque interpolated matrices such that

$$\pi_i = (V + \mathsf{diag}(T_{\mu-i-1}) \circ V') \| \; (V + \mathsf{diag}(T'_{\mu-i-1}) \circ V').$$

Then $\mathsf{fold}_{r_i}(\pi_i)$ is set to

$$\mathsf{fold}_{r_i}(\pi_i) = M_i^{(0)} V + M_i^{(1)} V'.$$

Next, we prove that the bad event B happens with probability at most $\frac{2\mu}{\epsilon^2 p^k}$.

Corollary 1 (Adapted from Corollary 1 for [61]). *Fix any $i \in [0, \mu-\log k-1]$ and any $\epsilon, \delta > 0$ such that $\delta \leq 1-(1-\gamma_\mu+\epsilon)^{1/3}$. Then if $\Delta^*_{\mathrm{blk}}(W, \mathcal{C}_{\mu-i}^{\times k}) > \delta$, then*

$$\Pr_{r_i}[\Delta_{\mathsf{blk}}(\mathit{fold}_{r_i}(W), \mathcal{C}_{\mu-i-1}^{\times k}) \leq \delta - \epsilon] \leq \frac{2}{\epsilon^2 p^k}.$$

Proof. Let $V, V' \in \mathbb{F}_{p^k}^{k \times n_{\mu-i-1}}$ be the unique interpolated matrices such that $\mathsf{fold} r_i(W) = M_i^{(0)} V + M_i^{(1)} V'$. Let $\mathcal{W} = \{M_i^{(0)} V + M_i^{(1)} V'\}$ and let $\tilde{W}$ be the set of elements in $\mathcal{W}$ that have block-wise distance less than $\delta - \epsilon$ from $\mathcal{C}_{\mu-i-1}^{\times k}$. Assume for contradiction that $|\tilde{W}| > \frac{2(2k-1)^k}{\epsilon^2}$. Then by lemma 3, there exist codewords $C, C' \in \mathcal{C}_{\mu-i-1}^{\times k}$ and a set $S \subseteq [n_{\mu-i-1}]$ such that $|S| \geq (1-\delta)|n_{\mu-i-1}|$ and for every $j \in S$, we have $W_{\cdot,j} = C_{\cdot,j}$ and $W'_{\cdot,j} = C'_{\cdot,j}$. Since C, C' are codewords in $\mathcal{C}_{\mu-i-1}^{\times k}$, by the foldable code structure, the following is a codeword in $\mathcal{C}_{\mu-i}^{\times k}$:

$$C_W = (C + \mathsf{diag}(T_{\mu-i-1}) \circ C') \| \ (C + \mathsf{diag}(T'_{\mu-i-1}) \circ C').$$

Therefore, for each $j \in S$, we have C_W agree with W on column j and column $j + n_{\mu-i-1}$. Therefore $\Delta^*_{\mathrm{blk}}(W, \mathcal{C}_{\mu-i}^{\times k}) \leq \delta$, which contradicts with our assumption that $\Delta^*_{\mathrm{blk}}(W, \mathcal{C}_{\mu-i}^{\times k}) > \delta$.

By a union bound over $i \in [0, \mu - \log k - 1]$, we have that the bad event B happens with probability at most $\frac{2\mu}{\epsilon^2 p^k}$. So except with probability at most $\frac{2\mu}{\epsilon^2 p^k}$, we can fix challenges $(r_0, \ldots, r_{\mu-1})$ such that $\neg B$ holds. Consider a weaker "path-query" verifier that runs Q independent checks as follows: it samples a uniform block index $j \in [n_{\log k}]$, follows the ancestor indices of j across the folding tree, and in each round i opens the corresponding triple of columns to test the round-i folding equation. This path-query verifier performs a subset of the checks made by our original verifier. Hence, for any fixed prover strategy, the acceptance probability of the path-query verifier is at least the acceptance probability of the original verifier (over the verifier's randomness).

For each round $i \in \{0, \ldots, \mu - \log k - 1\}$, define

$$\beta_i \ := \ \Delta_{\mathrm{blk}}\big(\pi_{i+1}, \mathsf{fold}_{r_i}(\pi_i)\big),$$

i.e., the fraction of blocks on which the folding constraint relating π_i and π_{i+1} fails. A single path query rejects if it encounters any failing block at some round, and therefore

$$\Pr[\text{accept in one path query}] \ \leq \ 1 - \sum_{i=0}^{\mu-\log k-1} \beta_i,$$

$$\Pr[\text{accept in all } Q \text{ path queries}] \ \leq \ \Big(1 - \sum_{i=0}^{\mu-\log k-1} \beta_i\Big)^Q.$$

Claim (Adapted from Claim 2 for [61]). For each $i \in \{0, \ldots, \mu - \log k - 1\}$, define $\delta^{(i)} = \min(\Delta^*_{\mathrm{blk}}(\pi_i, \mathcal{C}_{\mu-i}^{\times k}), 1 - (1 - \gamma_\mu + \epsilon)^{1/3})$. Then, we have

$$\beta_i \geq \delta^{(i)} - \delta^{(i+1)} - \epsilon.$$

Proof. By the condition that $\neg B$ holds, we have for each $i \in \{0, \ldots, \mu - \log k - 1\}$,

$$\Delta_{\mathrm{blk}}(\mathsf{fold}_{r_i}(\pi_i), \mathcal{C}_{\mu-i-1}^{\times k}) > \delta^{(i)} - \epsilon.$$

On the other hand, by triangle inequality, we have

$$\begin{aligned}\Delta_{\mathrm{blk}}(\mathsf{fold}_{r_i}(\pi_i), \mathcal{C}^{\times k}_{\mu-i-1}) &\le \Delta_{\mathrm{blk}}(\mathsf{fold}_{r_i}(\pi_i), \pi_{i+1}) + \Delta_{\mathrm{blk}}(\pi_{i+1}, \mathcal{C}^{\times k}_{\mu-i-1}) \\ &\le \beta_i + \Delta^*_{\mathsf{blk}}(\pi_{i+1}, \mathcal{C}^{\times k}_{\mu-i-1}).\end{aligned}$$

Combining the two inequalities above gives the desired result, we have

$$\beta_i \ge \delta^{(i)} - \Delta^*_{\mathrm{blk}}(\pi_{i+1}, \mathcal{C}^{\times k}_{\mu-i-1}) - \epsilon \ge \delta^{(i)} - \delta^{(i+1)} - \epsilon.$$

WLOG we assume that $\delta^{(i+1)} < \delta^{(i)} - \epsilon$, otherwise the claim trivially holds. This implies that $\delta^{(i+1)} < \delta^{(i)} \le 1 - (1 - \gamma_\mu + \epsilon)^{1/3}$ and thus $\delta^{(i+1)} = \Delta^*_{\mathrm{blk}}(\pi_i, \mathcal{C}^{\times k}_{\mu-i})$. So the claim holds. □

Recall that $\delta^{(0)} = \delta$ and $\delta^{(\mu-\log k)} = \Delta^*_{\mathrm{blk}}(\pi_{\mu-\log k}, \mathcal{C}^{\times k}_{\log k}) = 0$. So $\delta^{(0)} - \delta^{(\mu-\log k)} = \delta$. By the claim above, we have

$$\begin{aligned}\sum_{i=0}^{\mu-\log k-1} \beta_i &\ge \sum_{i=0}^{\mu-\log k-1} (\delta^{(i)} - \delta^{(i+1)} - \epsilon) \\ &= \delta^{(0)} - \delta^{(\mu-\log k)} - \epsilon(\mu - \log k) \\ &= \delta - \epsilon(\mu - \log k).\end{aligned}$$

Therefore, the acceptance probability of the path-query verifier is at most

$$(1 - \sum_{i=0}^{\mu-\log k-1} \beta_i)^Q \le (1 - \delta + \epsilon(\mu - \log k))^Q \le (1 - \delta + \epsilon\mu)^Q.$$

Since the acceptance probability of the original verifier is at most that of the path-query verifier, we conclude that the acceptance probability of the original verifier is at most $(1 - \delta + \epsilon\mu)^Q$. Finally, we show that if the verifier accepts, then the final decoded table must be the evaluation table of U. Suppose there exists another table U', such that the final decoded table is also the evaluation table of U'. We have $\Delta^*_{\mathrm{blk}}(\hat{U}, \mathsf{Enc}_\mu(U)) < \gamma_\mu/2$, and $\Delta^*_{\mathrm{blk}}(\hat{U'}, \mathsf{Enc}_\mu(U')) < \gamma_\mu/2$. By triangle inequality,

$$\begin{aligned}\gamma_\mu &\le \Delta_{\mathrm{blk}}(\mathsf{Enc}_\mu(U), \mathsf{Enc}_\mu(U')) \\ &\le \Delta_{\mathrm{blk}}(\mathsf{Enc}_\mu(U), \hat{U}) + \Delta_{\mathrm{blk}}(\hat{U'}, \mathsf{Enc}_\mu(U')) \\ &< \Delta^*_{\mathrm{blk}}(\hat{U}, \mathsf{Enc}_\mu(U)) + \Delta^*_{\mathrm{blk}}(\hat{U'}, \mathsf{Enc}_\mu(U')) \\ &< \gamma_\mu/2 + \gamma_\mu/2 = \gamma_\mu.\end{aligned}$$

where $\hat{U} = \hat{U'}$ unless the adversary finds a hash collision. This is a contradiction. Therefore, the final decoded table must be the evaluation table of U. This completes the proof. □

Lemma 3 (Repetition version of lemma 3.2 [12]). *Let $\mathcal{C} \subseteq \mathbb{F}_{p^k}^D$ be a linear code of relative distance γ. Let $\epsilon, \delta > 0$ with $\epsilon < 1/3$ and $\delta < 1 - (1 - \gamma + \epsilon)^{1/3}$.*

Let $U, U^* \in \mathbb{F}_{p^k}^{k \times D}$ (viewed as matrices with columns indexed by D) satisfy

$$\Pr_{\mathbf{x} \leftarrow \mathbb{F}_p^k} \left[\Delta_{\mathrm{blk}}\big(M_0(\mathbf{x})U^* + M_1(\mathbf{x})U,\ \mathcal{C}^{\times k}\big) < \delta - \epsilon \right] \ \geq\ \frac{2(2k-1)^k}{\epsilon^2 p^k},$$

where $M_0(\mathbf{x}), M_1(\mathbf{x}) \in \mathbb{F}_p^{k \times k}$ are defined by $M_0(\mathbf{x})[a,b] = \mathcal{L}_{2b}(x_a)$ and $M_1(\mathbf{x})[a,b] = \mathcal{L}_{2b+1}(x_a)$ for a fixed Lagrange basis $\{\mathcal{L}_i\}_{i=0}^{2k-1}$. Then there exist codewords $V, V^* \in \mathcal{C}^{\times k}$ and a subset $C \subseteq D$ such that

$$|C| \geq (1-\delta)|D|, \qquad U|_{(\cdot,C)} = V|_{(\cdot,C)}, \qquad U^*|_{(\cdot,C)} = V^*|_{(\cdot,C)}.$$

Proof. Let $U_{\mathbf{x}} = M_0(\mathbf{x})U^* + M_1(\mathbf{x})U$. Let

$$A = \{\mathbf{x} | \Delta(M_0(\mathbf{x})U^* + M_1(\mathbf{x})U, \mathcal{C}^k) < \delta - \epsilon\}$$

For each $\mathbf{x} \in A$, let $V_{\mathbf{x}} \in \mathcal{C}^k$ be the closest codeword to $U_{\mathbf{x}}$, and let $S_{\mathbf{x}} \subset D$ be the agreement set of $U_{\mathbf{x}}$ and $V_{\mathbf{x}}$, defined as $S_{\mathbf{x}} = \{y \in D | U_{\mathbf{x}}(\cdot, y) = V_{\mathbf{x}}(\cdot, y)\}$.

For $\mathbf{x}, \alpha, \beta$ picked uniformly at random from A and y picked uniformly at random from D, we have:

$$\begin{aligned}
\mathbf{E}_{\mathbf{x},\alpha,\beta}[|S_{\mathbf{x}} \cap S_\alpha \cap S_\beta|/n] &= \mathbf{E}_{\mathbf{x},\alpha,\beta,y}[1_{y \in S_{\mathbf{x}} \cap S_\alpha \cap S_\beta}] \\
&= \mathbf{E}_y[\mathbf{E}_{\mathbf{x}}[1_{y \in S_{\mathbf{x}}}]^3] \\
&\geq (\mathbf{E}_{y,\mathbf{x}}[1_{y \in S_{\mathbf{x}}}])^3 \\
&\geq (1 - \delta + \epsilon)^3 \\
&> 1 - \gamma + \epsilon.
\end{aligned}$$

The second equality above follows from the independence of the events $y \in S_{\mathbf{x}}, y \in S_\alpha, y \in S_\beta$ given $y \in D$. The first inequality follows from Jensen's inequality, and the last inequality is by assumption on δ, γ, ϵ.

Thus

$$\Pr_{\mathbf{x},\alpha,\beta,y}[y \in S_{\mathbf{x}} \cap S_\alpha \cap S_\beta] > 1 - \gamma + \epsilon.$$

Note that $\Pr_{\mathbf{x},\alpha,\beta,y}[\mathbf{x}, \alpha, \beta \text{ are not all distinct}] < \frac{3}{A}$. Since $|A| \geq \frac{2(2k-1)^k}{\epsilon^2} > \frac{6(2k-1)^k}{\epsilon}$, we have that $3/|A| < \epsilon/2$ and hence $\mathbf{x}, \alpha, \beta$ are all distinct with probability at least $1 - \epsilon/2$. Thus with probability at least $\epsilon/2$ over the choice of $\mathbf{x}, \alpha, \beta$, we have that $\mathbf{x}, \alpha, \beta$ are all distinct and $|S_{\mathbf{x}} \cap S_\alpha \cap S_\beta| \geq (1-\gamma)|D|$.

This means that there are distinct $\mathbf{x}_0, \alpha_0$ such that

$$\Pr_\beta[|S_{\mathbf{x}_0} \cap S_{\alpha_0} \cap S_\beta| > (1-\gamma)|D|] \geq \epsilon/2.$$

Fix a β where this happens. Let $S = S_{\mathbf{x}_0} \cap S_{\alpha_0} \cap S_\beta$. We have

$$\begin{aligned}
U_{\mathbf{x}_0}|_{(\cdot,S)} &= M_0(\mathbf{x}_0)U^*|_{(\cdot,S)} + M_1(\mathbf{x}_0)U|_{(\cdot,S)} = V_{\mathbf{x}_0}|_{(\cdot,S)}, \\
U_{\alpha_0}|_{(\cdot,S)} &= M_0(\alpha_0)U^*|_{(\cdot,S)} + M_1(\alpha_0)U|_{(\cdot,S)} = V_{\alpha_0}|_{(\cdot,S)}, \\
U_\beta|_{(\cdot,S)} &= M_0(\beta)U^*|_{(\cdot,S)} + M_1(\beta)U|_{(\cdot,S)} = V_\beta|_{(\cdot,S)}.
\end{aligned}$$

By definition of S, we get that

$$\begin{aligned} V_{\mathbf{x}_0}|_{(\cdot,S)} &= V_{\mathbf{x}_0}|_{(\cdot,S)}, \\ V_{\alpha_0}|_{(\cdot,S)} &= V_{\alpha_0}|_{(\cdot,S)}, \\ V_{\beta}|_{(\cdot,S)} &= V_{\beta}|_{(\cdot,S)}. \end{aligned}$$

Since $|S| > (1-\gamma)|D|$, by the distance property of the code $\mathcal{C}^k$, we have that V_β is determined by $V_\gamma|_{(\cdot,S)}$ via a unique linear transformation. This means that

$$\begin{aligned} V_{\mathbf{x}_0} &= M_0(\mathbf{x}_0)V^* + M_1(\mathbf{x}_0)V, \\ V_{\alpha_0} &= M_0(\alpha_0)V^* + M_1(\alpha_0)V, \\ V_{\beta} &= M_0(\beta)V^* + M_1(\beta)V. \end{aligned}$$

Thus, $\epsilon/2$ fraction of $\beta \in A$ have the "good" property that $V_\beta = M_0(\beta)V^* + M_1(\beta)V$, where V^* and V are consistent with $V_{\mathbf{x}_0}$ and V_{α_0}. Let A' denote the set of such "good" β's, recording that $|A'| \geq \epsilon|A|/2 \geq (2k-1)^k/\epsilon$.

Thus for $\mathbf{x} \in A', \Delta(M_0(\mathbf{x})U^* + M_1(\mathbf{x})U, M_0(\mathbf{x})V^* + M_1(\mathbf{x})V) < \delta - \epsilon$.

Consider the set $C \subset D$ defined by

$$C = \{y \in D | U^*(\cdot, y) = V^*(\cdot, y) \text{ AND } U(\cdot, y) = V(\cdot, y)\}$$

For each $y \notin C$, there exists at most $(2k-1)^k$ values of $\mathbf{x} \in \mathbb{F}_p^k$ such that $M_0(\mathbf{x})U^*(\cdot, y) + M_1(\mathbf{x})U(\cdot, y) = M_0(\mathbf{x})V^*(\cdot, y) + M_1(\mathbf{x})V(\cdot, y)$.

This is because if $y \notin C$, then for each component j of the $M_0(\mathbf{x})U^*(\cdot, y) + M_1(\mathbf{x})U(\cdot, y) - (M_0(\mathbf{x})V^*(\cdot, y) + M_1(\mathbf{x})V(\cdot, y))$, we have:

$$f_j(x_j) = \sum_{i=0}^{k-1} \mathcal{L}_{2i}(x_j)(U^*(i, y) - V^*(i, y)) + \sum_{i=0}^{k-1} \mathcal{L}_{2i+1}(x_j)(U(i, y) - V(i, y))$$

Since $y \notin C$, f_j is a non-zero polynomial in x_j of degree at most $2k-1$, and thus has at most $2k-1$ roots. Since there are k components, there are at most $(2k-1)^k$ solutions in total.

This implies that

$$\begin{aligned} \delta - \epsilon \geq & \mathbf{E}_{\mathbf{x} \in A'}[\Delta(M_0(\mathbf{x})U^* + M_1(\mathbf{x})U, M_0(\mathbf{x})V^* + M_1(\mathbf{x})V)] \\ \geq & \frac{|D \setminus C|}{|D|} \cdot \left(1 - \frac{(2k-1)^k}{|A'|}\right) \\ \geq & \left(1 - \frac{|C|}{|D|}\right) \cdot (1 - \epsilon) \geq 1 - \frac{|C|}{|D|} - \epsilon. \end{aligned}$$

Rearraning, we have $\frac{|C|}{|D|} \geq 1 - \delta$ and this completes the proof. □

D.4 Proof of Theorem 9

Theorem 12 (Theorem 9). *Assume the Merkle hash is modeled as a random oracle. Then the Basefold-style commitment for packed sumcheck described above is knowledge-sound: for any PPT prover $\mathcal{P}^*$, if the verifier accepts the evaluation phase and the final codeword check with non-negligible probability and outputs a value z, there exists a PPT extractor $\mathcal{E}^{\mathcal{P}^*}$ that outputs a table $\widetilde{F}$ consistent with the initial commitment such that $z = \mathsf{Eval}(\widetilde{F}, \mathcal{M})$ for the structured query $\mathcal{M}$ induced by the packed-sumcheck transcript.*

Proof. Let $\pi_0, \ldots, \pi_{\mu-\log k}$ be the Merkle roots output by $\mathcal{P}^*$. The root π_0 commits to the initial encoded matrix $\hat{U}^{(\mu)}$ and each π_i commits to the folded encoded matrix $\hat{U}^{(\mu-i)}$. As discussed in [61] and existing IOP-to-succinct-argument transformations [11], in the random oracle model any prover that makes the Fiat–Shamir verifier accept with non-negligible probability admits an efficient extractor that obtains the values opened by the prover at all verifier query positions. Concretely, the extractor runs $\mathcal{P}^*$ and uses standard random-oracle rewinding to recover, for every round i and every queried index $j \in S_i$, the opened columns together with valid Merkle authentication paths to the claimed roots π_i and π_{i-1}. Therefore the extractor obtains a set of openings that are consistent with the committed oracles at those queried positions.

We now reconstruct a witness table from these openings. Fix any level $\ell \in \{\mu, \mu-1, \ldots, \log k\}$. The extracted openings at level ℓ determine the values of the oracle $\hat{U}^{(\ell)}$ on a polynomial-size set of column indices. Since the verifier also performs the final codeword check at level $\log k$ and accepts with non-negligible probability, these openings are consistent with an interleaving of valid codewords except with negligible probability. Using the efficient unique decoding algorithm for the block-level Reed–Solomon code under the same metric used by the protocol, the extractor decodes each row and recovers a codeword matrix $\widehat{U}^{(\ell)} \in \mathcal{C}_\ell^{\times k}$ that matches the extracted openings. If for some transcript the openings do not determine a unique decoding, then the verifier acceptance probability for that transcript is negligible by code distance and proximity soundness, and the extractor may fail on that transcript without affecting the theorem.

Let $U^{(\ell)}$ be the decoded message matrix over $\mathbb{F}_{p^k}$ underlying $\widehat{U}^{(\ell)}$. Unpacking the block representation of $U^{(\mu)}$ back into $\mathbb{F}_p$ produces a base-field table $\widetilde{F}$. By construction, $\widetilde{F}$ is consistent with the initial commitment π_0.

It remains to show that the verifier output z equals $\mathsf{Eval}(\widetilde{F}, \mathcal{M})$ for the structured query $\mathcal{M}$ induced by the packed-sumcheck transcript. Fix the transcript, hence fix the folding challenges $r_0, \ldots, r_{\mu-\log k-1}$ and the corresponding matrices $\{(M_i^{(0)}, M_i^{(1)})\}_{i=0}^{\mu-\log k-1}$. In every round i and every queried index $j \in S_i$, the verifier checks that the opened parent columns $\mathbf{a} = \hat{U}_{\cdot,j}^{(\mu-i+1)}$ and $\mathbf{b} = \hat{U}_{\cdot,n_{\mu-i}+j}^{(\mu-i+1)}$ satisfy the inverse step defined by the diagonal pair t, t' at position j, and that the opened child column $\hat{U}_{\cdot,j}^{(\mu-i)}$ satisfies the folding equation

$$\hat{U}_{\cdot,j}^{(\mu-i)} \;=\; M_{i-1}^{(0)}\mathbf{x} + M_{i-1}^{(1)}\mathbf{y},$$

where $\mathbf{x}, \mathbf{y}$ are the unique vectors such that $\mathbf{a} = \mathbf{x} + t\mathbf{y}$ and $\mathbf{b} = \mathbf{x} + t'\mathbf{y}$.

Merkle binding in the random oracle model guarantees that the extracted openings at each level are consistent with a single oracle committed by π_i. Therefore all of the above local constraints hold for the extracted values at all queried positions. By the proximity soundness guarantee proved for the Basefold-style evaluation phase, this implies that the globally decoded matrices $U^{(\mu)}, U^{(\mu-1)}, \ldots, U^{(\log k)}$ coincide with the honest folding trajectory obtained by applying the transcript matrices $\{(M_i^{(0)}, M_i^{(1)})\}$ to the initial table $\widetilde{F}$, except with negligible probability.

Hence the table decoded by the verifier at level $\log k$ is exactly the partial evaluation table of $E^{(s)}$ at the transcript-induced point, and the verifier completes the last $\log k$ packed-sumcheck rounds locally to output $z = \mathsf{Eval}(\widetilde{F}, \mathcal{M})$. This proves knowledge soundness in the random oracle model. □

D.5 Basefold MW-MLE Complexity Analysis

We measure costs in base field operations over $\mathbb{F}_p$ and represent each $\mathbb{F}_{p^k}$ symbol as a length-k block over $\mathbb{F}_p$. Let Enc_μ be the block-level RS encoding with rate α_μ and code length $n_\mu = N/(\alpha_\mu k)$ over $\mathbb{F}_{p^k}$. Let $\hat{U} \in \mathbb{F}_{p^k}^{k \times n_\mu}$ be the committed matrix, and let $Q = \Theta(\lambda)$.

- **Prover time.** The commit stage encodes k rows of length n_μ over $\mathbb{F}_{p^k}$. For RS-style encodings over an evaluation domain $W \subseteq \mathbb{F}_p$, the arithmetic consists of multiplications of $\mathbb{F}_{p^k}$ symbols by base field scalars, which under the block representation costs k base field multiplications per symbol. Thus the encoding costs $\tilde{O}(k^2 n_\mu) = \tilde{O}(kN)$ base field multiplications. In the online phase, the prover performs $\mu - \log k$ rounds of folding. In each round, for each of the $n_{\mu-i}$ columns, the prover performs two $k \times k$ matrix over $\mathbb{F}_p$ products with k-dimensional vectors over $\mathbb{F}_{p^k}$, which costs $O(k^3)$ base field multiplications. Because the number of columns $n_{\mu-i} = O(n_\mu/2^i)$ halves each round, and $n_\mu = O(N/(\alpha_\mu k))$, the total cost across all rounds is $O(k^3 n_\mu) = O(k^2 N)$ base field multiplications.
  In total, the prover runs in $\tilde{O}(kN) + O(k^2 N)$ base field operations.
- **Proof size.** In each round and for each query, the prover opens one child column and its two parent columns, i.e., 3 columns in $\mathbb{F}_{p^k}^k$, plus Merkle paths. Every Merkle path has length $O(\log n_\mu) = O(\log N)$, and the opened columns contribute $O(k^2)$ base field elements per query. In each round, the prover opens Q queries, and there are $\mu - \log k$ rounds. Thus the total proof size is
  $$O((\mu - \log k)\, Q\, (k^2 + \log N))$$
  base field elements, which is $O(\log^2 N)$ when k is treated as a constant and $Q = \Theta(\lambda)$.
- **Verifier time.** For each opened triple, the verifier recomputes the inverse step (using $t, t' \in \mathbb{F}_p$) and checks one folding equation, dominated by two $k \times k$ matrix over $\mathbb{F}_p$ products with k-dimensional vectors over $\mathbb{F}_{p^k}$. This costs

$O(k^3)$ base field operations per query per round. And the verifier needs to check the Merkle paths, whose length is $O(\log N)$. So the total verifier time across all rounds is $O((\mu - \log k)\, Q\, (k^3 + \log N)) = O(\log^2 N + k^3 \log N)$.

Relation to FRI-Binius. It is useful to relate these operations to FRI-Binius [31]. FRI-Binius also starts from k length-N polynomials over $\mathbb{F}_p$. It packs them into one length-N polynomial over $\mathbb{F}_{p^k}$ and then runs a FRI-style commitment over $\mathbb{F}_{p^k}$. The initial commitment work corresponds to $\tilde{O}(k \cdot kn_\mu) = \tilde{O}(kN)$ base field multiplications. In the query and folding checks, it performs $O(N)$ multiplications in $\mathbb{F}_{p^k}$, which corresponds to $O(k^2 N)$ base field multiplications under the standard model. So the same two cost scales, $\tilde{O}(k^2 n_\mu) = \tilde{O}(kN)$ and $O(k^2 N)$, appear in our scheme as well.

E Block Reed-Solomon Code for Packed Sumcheck

Brakedown and Basefold are both commitment schemes based on linear codes. Before introducing these two schemes, we first present a linear code over $\mathbb{F}_p$. We adopt a Reed-Solomon encoding method that incorporates block-level encoding [30]. Let n be a multiple of k and let $\alpha \in (0, 1]$ be the target rate. We view $\mathrm{GF}(p, k)$ as a degree-k extension of $\mathbb{F}_p$, but we do not invoke its multiplication as an algebraic operation. Instead, we represent each symbol in $\mathrm{GF}(p, k)$ as a length-k vector over $\mathbb{F}_p$ in a fixed basis.

Let $W = \{w_0, \ldots, w_{m-1}\} \subseteq \mathbb{F}_p$ be an evaluation set of size $m := n/(\alpha k)$ (typically an NTT subgroup or coset). Given $\mathbf{x} \in \mathbb{F}_p^n$, partition $\mathbf{x}$ into blocks of size k, reinterpret each block as a symbol in $\mathrm{GF}(p, k)$, and encode via Reed–Solomon evaluation on W. Equivalently, the encoding can be implemented by applying RS evaluation coordinate-wise on the k extracted vectors over $\mathbb{F}_p$ and then repacking. Algorithm 2 summarizes this procedure.

Algorithm 2: Block RS code Enc Algorithm:

Input: $\mathbf{x} \in \mathbb{F}_p^n$

Output: $\mathbf{w}' \in \mathbb{F}_{p^k}^{n/\alpha k}$

1. Partition the input vector into groups of k elements. From each group, extract the elements at positions $0, \cdots, k-1$ to form vectors $\mathbf{x}_0, \cdots, \mathbf{x}_{k-1} \in \mathbb{F}_p^{n/k}$;
2. Select an NTT basis W of size $n/\alpha k$ over $\mathbb{F}_p$, and apply Reed–Solomon encoding with respect to W to each vector $\mathbf{x}_i$, obtaining the encoded vectors $\mathbf{w}_i \in \mathbb{F}_p^{n/\alpha k}$;
3. Recombine these encoded vectors into a single vector $\mathbf{w}$ of length n/α, defined by $\mathbf{w}[k \cdot j + i] = \mathbf{w}_i[j]$;
4. By packing every k consecutive entries of $\mathbf{w}$ into a single element over $\mathbb{F}_{p^k}$, we obtain the vector $\mathbf{w}' \in \mathbb{F}_{p^k}^{n/\alpha k}$, which is returned as the final codeword;

Lemma 4. *Over the field $\mathbb{F}_{p^k}$, the encoding in Alg. 2 achieves relative distance $1-\alpha$.*

Proof. The encoding method in Alg. 2 is inspired by the concept of block-level encoding. The essential idea is to treat every k consecutive elements over the base field $\mathbb{F}_p$ as a single element over the degree-k extension field $\mathbb{F}_{p^k}$. Accordingly, a vector $\mathbf{x} \in \mathbb{F}_p^n$ is reinterpreted as $\hat{\mathbf{x}} \in \mathbb{F}_{p^k}^{n/k}$. The vector $\hat{\mathbf{x}}$ is then viewed as the coefficient representation of a polynomial, which is evaluated on the NTT basis W. After encoding, $\hat{\mathbf{x}}$ achieves a relative code distance of $1-\alpha$. □

Remark 4. Our MW-Brakedown and MW-Basefold constructions are not restricted to block Reed–Solomon codes: they can also be instantiated with the native codes of Brakedown [37] and BaseFold [61], or with other suitable linear codes. We use block Reed–Solomon codes because their encoding requires no extension-field multiplication, whereas block-encoded variants of the native codes may introduce matrix multiplications over the extension field during encoding, which is inconsistent with our design goal of avoiding extension-field multiplication. If alternative codes further improve the complexity and concrete efficiency of the PCS, they would reduce the PCS fraction of the total runtime, making sumcheck efficiency comparatively more important.

F HyperPlonk for Small Prime Fields

In this section, we describe how to instantiate the HyperPlonk framework over small prime fields using our packed sumcheck, packed ZeroCheck, and MW-MLE commitment primitives. The purpose of this construction is to show compatibility with a general-purpose SNARK framework; applications that are naturally expressed as vector relations can also invoke the packed sumcheck protocols directly.

The constraint system consists of gate constraints and permutation constraints. For each gate in the circuit, let L_i, R_i, and O_i denote the left input, right input, and output wire values of the i-th gate. If the circuit has o public input gates, one output gate, and n remaining gates, assume $o+n+1=2^\mu$. Let $\langle i\rangle \in \{0,1\}^\mu$ be the binary representation of i, and define the multilinear witness polynomial M by

$$M(0,0,\langle i\rangle) = L_i, \qquad M(0,1,\langle i\rangle) = R_i, \qquad M(1,0,\langle i\rangle) = O_i.$$

The remaining point $(1,1,\cdot)$ is unused and can be set to zero. We use the same notation for the multilinear extension of these tables.

Let $k=2^\ell$. We write an index $b \in \{0,1\}^\mu$ as $b=(a,x)$, where $a \in \{0,1\}^\ell$ and $x \in \{0,1\}^{\mu-\ell}$. For each scalar wire table $f : \{0,1\}^\mu \to \mathbb{F}_p$, we use two packed encodings induced by the same underlying table:

$$F_{\mathsf{split}}(x) := \big(f(a,x)\big)_{a\in\{0,1\}^\ell} \in \mathbb{F}_p^k, \qquad F_{\mathsf{rep}}(a,x) := (f(a,x),\ldots,f(a,x)) \in \mathbb{F}_p^k.$$

The split-packed encoding is used by the gate checks, while the repetition-packed encoding is used by the permutation checks.[12] The prover commits to the corresponding protocol-level views using the MW-MLE commitment scheme. To ensure that they represent the same scalar witness, the prover also proves the consistency relations

$$F_{\mathsf{rep}}(a,x)_t = F_{\mathsf{split}}(x)_a \qquad \text{for all } a \in \{0,1\}^\ell,\ x \in \{0,1\}^{\mu-\ell},\ t \in [0,k).$$

Equivalently, with $\mathbf{1} = (1,\dots,1)^\top$,

$$F_{\mathsf{rep}}(a,x) - \mathbf{1} \cdot \sum_{u\in\{0,1\}^\ell} \mathsf{eq}(a,u)\, F_{\mathsf{split}}(x)_u = 0.$$

This is a vector-valued zero relation and is checked with packed ZeroCheck. The same consistency check is applied to the packed encodings of all wire tables used below.

F.1 Gate Constraints.

Let $S_1, S_2 : \{0,1\}^\mu \to \mathbb{F}_p$ be the selector tables, and let $I : \{0,1\}^\mu \to \mathbb{F}_p$ encode the public input/output values. The gate constraint is

$$\begin{aligned} 0 = {} & S_1(b)\big(M(0,0,b) + M(0,1,b)\big) + S_2(b)M(0,0,b)M(0,1,b) \\ & - M(1,0,b) + I(b), \qquad \text{for all } b \in \{0,1\}^\mu. \end{aligned}$$

For an addition gate, $S_1(\langle i\rangle) = 1$ and $S_2(\langle i\rangle) = 0$; for a multiplication gate, $S_1(\langle i\rangle) = 0$ and $S_2(\langle i\rangle) = 1$; for input and output gates, both selectors are zero and the value is constrained by I.

For the packed check, write each gate index as $b = (a,x)$, where $a \in \{0,1\}^\ell$ is the lane index and $x \in \{0,1\}^{\mu-\ell}$ is the remaining index. The scalar gate relation is equivalently the following vector-valued zero relation over $x \in \{0,1\}^{\mu-\ell}$, whose a-th coordinate is the gate constraint at (a,x):

$$\begin{aligned} 0 = {} & \big(S_1(a,x)\big(M(0,0,a,x) + M(0,1,a,x)\big)\big)_{a\in\{0,1\}^\ell} \\ & + \big(S_2(a,x)M(0,0,a,x)M(0,1,a,x)\big)_{a\in\{0,1\}^\ell} \\ & - \big(M(1,0,a,x)\big)_{a\in\{0,1\}^\ell} \\ & + \big(I(a,x)\big)_{a\in\{0,1\}^\ell}. \end{aligned}$$

Here $M(\tau_0,\tau_1,a,x)$ denotes $M(\tau_0,\tau_1,b)$ with $b = (a,x)$. The prover proves this relation by invoking packed ZeroCheck. The final openings required by this ZeroCheck are handled by the MW-MLE commitments to the corresponding split-packed tables.

[12] This protocol-level description treats F_{split} and F_{rep} as separate views. Remark 5 discusses how the concrete instantiation can commit only to F_{split}.

F.2 Permutation Constraints.

HyperPlonk enforces wiring consistency by proving a self-permutation relation for the global wire table. Equivalently, in the notation below one may view the two tables f and g as two views of the same committed wire table, and in the HyperPlonk instantiation we take $g = f$. We use the sumcheck-based permutation check, which avoids the grand-product degree loss and is well suited to packed sumcheck.

Let $\sigma : \{0,1\}^\mu \to \{0,1\}^\mu$ be a wiring permutation. We represent it by committed multilinear coordinate tables $\tilde{\sigma} = (\sigma_0, \ldots, \sigma_{\mu-1})$, where each $\sigma_r : \{0,1\}^\mu \to \mathbb{F}_p$ is Boolean on the hypercube. For two scalar wire tables $f, g : \{0,1\}^\mu \to \mathbb{F}_p$, the desired relation is

$$f(y) = g(\tilde{\sigma}(y)) \qquad \text{for all } y \in \{0,1\}^\mu.$$

For the table f, the prover also uses its repetition-packed encoding F_{rep}. Writing $y = (a, x)$, it proves the simple consistency relation

$$F_{\mathsf{rep}}(a, x)_i = f(a, x) \qquad \text{for all } a \in \{0,1\}^\ell,\ x \in \{0,1\}^{\mu-\ell},\ i \in [0, k).$$

This is again a vector-valued zero relation. At the protocol level, it can be checked by packed ZeroCheck against the ordinary-MLE view of f.

Define

$$\Psi(x) := \sum_{y \in B_\mu} f(y) \cdot \mathsf{eq}(\tilde{\sigma}(y), x).$$

Then the permutation relation is equivalent to $\Psi(x) = g(x)$ on the Boolean hypercube. To test this identity with k parallel challenges for soundness amplification, the verifier samples $\rho_0, \ldots, \rho_{k-1} \in \mathbb{F}_p^\mu$. We use the shorthand

$$Q_\rho(y) := \big(\mathsf{eq}(\tilde{\sigma}(y), \rho_0), \ldots, \mathsf{eq}(\tilde{\sigma}(y), \rho_{k-1})\big) \in \mathbb{F}_p^k.$$

This is not a separate committed oracle. It abbreviates the product expression computed from the committed permutation-coordinate tables $\sigma_0, \ldots, \sigma_{\mu-1}$ and the verifier's challenges. The prover then proves, by one packed sumcheck over the variable y, that

$$\sum_{y \in B_\mu} F_{\mathsf{rep}}(y) \odot Q_\rho(y) = \big(g(\rho_0), \ldots, g(\rho_{k-1})\big).$$

The right-hand side is obtained by opening the ordinary-MLE view of g at the k verifier challenges.[13] Thus the basic permutation check is one degree-$\mu + 1$ packed sumcheck, with committed factors F_{rep} and the permutation-coordinate tables $\sigma_0, \ldots, \sigma_{\mu-1}$; Q_ρ is only shorthand for their challenge-dependent product.

[13] At this protocol level, this uses both ordinary MLE PCS and MW-MLE PCS. Remark 5 explains how the ordinary MLE PCS can be omitted in the concrete instantiations.

Lemma 5. *The permutation IOP above, based on the packed sumcheck protocol, satisfies the following properties:*

- *Completeness holds perfectly.*
- *Round-by-round soundness error is at most*

$$\frac{\mu^k}{p^k} + \frac{((2k-1)(\mu+1)+1)^k}{p^k}.$$

- *Prover complexity is*

$$O((k\mu^2 + k^2\mu)N) \cdot \mathsf{mul} + O(k^2\mu^2 N) \cdot \mathsf{add},$$

  where $N = 2^\mu$.
- *Verifier complexity is* $O(k^2\mu^2)$ *field operations.*
- *Communication complexity is* $O(k\mu^2)$ *field elements in* $\mathbb{F}_p$.

Optimization for Permutation Constraints. Recently, Bünz et al. [18] further optimized this type of permutation argument by decomposing the permutation check into several lower-degree subchecks. The same idea can be combined with our packed sumcheck and packed ZeroCheck protocols.

Concretely, write $\mu = ml$, and decompose $\tilde{\sigma}$ into m blocks of l bits. For each challenge $\rho_i = (\rho_{i,0}, \ldots, \rho_{i,\mu-1})$, define auxiliary vector-valued tables $P_0, \ldots, P_{m-1}$, where for $j \in [0, m)$, $b \in \{0,1\}^\mu$, and $i \in [0, k)$,

$$P_j(b)_i = \prod_{r=0}^{l-1} \big((1 - \sigma_{jl+r}(b))(1 - \rho_{i,jl+r}) + \sigma_{jl+r}(b)\rho_{i,jl+r}\big).$$

Thus $P_j(b) \in \mathbb{F}_p^k$ contains, in its i-th coordinate, the contribution of the j-th block of permutation bits to $\mathsf{eq}(\tilde{\sigma}(b), \rho_i)$. The prover commits to the MW-MLE tables $P_0, \ldots, P_{m-1}$ and proves

$$\sum_{b \in B_\mu} F_{\mathsf{rep}}(b) \odot \bigodot_{j=0}^{m-1} P_j(b) = \big(g(\rho_0), \ldots, g(\rho_{k-1})\big)$$

by packed sumcheck. The defining relations of the P_j's are checked by packed ZeroCheck. This reduces the degree of the main packed sumcheck from $\mu + 1$ to $m + 1$, while the auxiliary checks have degree $l + 1$. Balancing the parameters with $l = O(\mu^{1/3})$ and $m = O(\mu^{2/3})$ yields prover multiplication complexity

$$O\big((k^2\mu + k\mu^{4/3})N\big),$$

matching the optimized decomposition while preserving the packed witness encodings described above.

Remark 5 (One MW-MLE commitment for multiple views). The protocol-level description above separates the split-packed table F_{split}, the repetition-packed table F_{rep}, and the ordinary MLE table g. In concrete instantiations, these are not committed independently. The prover commits once to the split-packed table F_{split}, and the two kinds of values used in the permutation check are obtained from this same committed table as follows.

Write a point of the original hypercube as $y = (a, x)$, where a selects one of the k lanes of the split-packed table, and define the expanded scalar table

$$\bar{f}(a, x) := F_{\mathsf{split}}(x)_a.$$

First consider the left-hand side, which is written in the protocol using F_{rep}. The repetition-packed view is the public linear view

$$F_{\mathsf{rep}}(a, x) = \mathbf{1} e_a^\top F_{\mathsf{split}}(x),$$

where e_a is the unit vector selecting lane a. Hence a MW-MLE query to F_{rep} of the form

$$z = \sum_{a,x} W_{\mathcal{M}}(a, x)\, F_{\mathsf{rep}}(a, x)$$

is answered from the committed F_{split} by first deriving the matrix weights

$$\widetilde{W}_{\mathcal{M}}(x) := \sum_a W_{\mathcal{M}}(a, x)\, \mathbf{1} e_a^\top,$$

and then opening the MW-MLE query

$$z = \sum_x \widetilde{W}_{\mathcal{M}}(x) F_{\mathsf{split}}(x).$$

The derived weights are determined by the packed-sumcheck transcript after the commitment to F_{split}; they are not fixed before the commitment. In the Brakedown instantiation, this means that the verifier's linear test $R \cdot U$ is formed using the coefficients induced by the derived weights $\widetilde{W}_{\mathcal{M}}(x)$. In the Basefold instantiation, the matrix-weighted folding is run with the same derived weights $\widetilde{W}_{\mathcal{M}}(x)$, while the Basefold folding challenges themselves are sampled in the usual order of the opening protocol. Thus F_{rep} is not opened as a separate committed table in either instantiation. The i-th coordinate of this value is

$$\left(\sum_{y \in B_\mu} F_{\mathsf{rep}}(y) \odot Q_\rho(y) \right)_i = \sum_{a,x} \bar{f}(a, x)\, \mathsf{eq}(\tilde{\sigma}(a, x), \rho_i).$$

Next consider the ordinary MLE value $g(\rho_i)$. The ordinary MLE table g is the scalar table $\bar{f}$ viewed as an ordinary multilinear extension. For $\rho_i = (\rho_{i,a}, \rho_{i,x})$, this value is the scalar linear functional

$$g(\rho_i) = \sum_x \mathsf{eq}(x, \rho_{i,x}) \left(\sum_a \mathsf{eq}(a, \rho_{i,a}) e_a^\top \right) F_{\mathsf{split}}(x).$$

Equivalently, the prover first forms the lane-linear scalar table

$$h_{\rho_{i,a}}(x) := \left(\sum_a \mathsf{eq}(a, \rho_{i,a}) e_a^\top \right) F_{\mathsf{split}}(x),$$

and then opens $h_{\rho_{i,a}}(\rho_{i,x})$ by the ordinary MLE opening procedure. In the Brakedown instantiation, this ordinary opening is certified by the linear test whose coefficients encode the scalar weights $\mathsf{eq}(x, \rho_{i,x})\mathsf{eq}(a, \rho_{i,a})$. In the Basefold instantiation, after ρ_i has been sampled, the prover first applies the lane-linear combination defining $h_{\rho_{i,a}}$, and then runs the usual Basefold MLE folding to open $h_{\rho_{i,a}}(\rho_{i,x})$ with the normal verifier challenges.

The permutation identity test compares the i-th coordinate obtained from the F_{rep}-side MW-MLE query with the ordinary MLE value $g(\rho_i)$. Both values are therefore opened from the same commitment to F_{split}, but they induce different derived queries and hence use the corresponding MW-MLE or ordinary MLE opening procedure.

F.3 Knowledge Soundness

Theorem 13 (Knowledge soundness of the compiled SNARK). *Assume that the parameters are chosen such that the round-by-round soundness errors of the constituent PIOPs are negligible. In the random oracle model, compiling the PIOP for the gate and permutation constraints described in Sections F.1 and F.2 with either of our knowledge-sound MW-MLE commitment schemes yields a knowledge-sound SNARK.*

Proof. The gate and permutation protocols described in Sections F.1 and F.2 together form a public-coin PIOP. Its witness oracles are the multilinear extensions of the wire tables and their packed views. The consistency checks ensure that these views encode the same underlying wire assignment, while the gate and permutation checks enforce circuit satisfiability and wiring consistency, respectively.

All these relations are reduced to invocations of packed ZeroCheck and packed sumcheck. By Theorems 1 and 3, each invocation has negligible round-by-round soundness error under the parameters of the theorem. The permutation protocol additionally satisfies the round-by-round soundness bound of Lemma 5. Since the complete PIOP contains only polynomially many such invocations, its soundness error remains negligible by a union bound.

Every witness component of this PIOP is represented by a polynomial oracle. Therefore, Lemma 2.3 of HyperPlonk [21] applies and shows that the PIOP is knowledge-sound with the same negligible error. The corresponding extractor recovers multilinear tables satisfying the consistency, gate, and permutation relations above.

In the compiled protocol, the prover commits to these oracle tables using MW-MLE commitments and realizes each oracle query through the corresponding structured opening protocol. The extractors from Theorems 2 and 9 recover tables consistent with the commitments and the accepted openings. Since

each commitment is fixed before its transcript-induced queries, the real interaction can be augmented with the tables recovered by these extractors without changing the transcript, except with negligible extraction failure. Thus, the two MW-MLE schemes provide the witness-extended emulation required for the structured queries in our PIOP. Theorem 2.4 of HyperPlonk [21] then gives a knowledge-sound interactive argument. Applying Fiat–Shamir to this public-coin protocol yields the claimed knowledge-sound SNARK in the random oracle model.

G Fully Homomorphic Encryption Preliminaries

Definition 9. *A fully homomorphic encryption scheme FHE defined over a message space $\mathcal{M}$ and a ciphertext space $\mathcal{C}$ is a tuple of PPT algorithms* (FHE.KeyGen,FHE.Enc,FHE.Eval,FHE.Dec) *where the algorithm are defined as follows:*

1 FHE.KeyGen(1^λ)*: Upon input the security parameter* λ, FHE.KeyGen *outputs a pair of keys* pk,sk *where* pk *and* sk *are the public key and secret resp.*
2 FHE.Enc(pk, x)*: Upon input the public key* pk *and a message* x, FHE.Enc *outputs a ciphertext* $c = \hat{x}$.
3 FHE.Eval$(pk, f, c_1, \cdots, c_n)$*: Upon input the public key* pk. *a circuit* $f : \mathcal{M}^n \to \mathcal{M}$ *and a set of ciphertexts* $(c_1, \cdots, c_n)$ *that encrypt the message* $x_s = (x_1, \cdots, x_n)$, FHE.Eval *outputs a ciphertext* $c = f(\hat{x}_s)$.
4 FHE.Dec(sk, c)*: Upon input the secret key* sk *and a ciphertext* $c = \hat{x}$, FHE.Dec *outputs a message* x.

– *Correctness: An FHE scheme is said to be correct if, for all* $m \in \mathcal{M}$, $c \in \mathcal{C}$, *and* f *that can be efficiently evaluated, it holds that:*

$$\Pr[\text{FHE.Dec}(sk, c_r) = f(x_1, \cdots, x_n)] \geq 1 - negl(\lambda).$$

where $c_r = \text{FHE.Eval}(pk, f, c_1, \cdots, c_n)$.
– *Security: An FHE scheme is said to be secure if, for all* $m \in \mathcal{M}, c \in \mathcal{C}$, *and a* PPT *adversary* $\mathcal{A}$, c *does not leak any information about its underlying message* m *to* $\mathcal{A}$ *beyond what is public know.*

Ciphertext Structure The security of TFHE bootstrapping is based on the hardness of the Learning With Errors (LWE) [52] and Ring-LWE (RLWE) [49], and typically involves four ciphertext types: LWE, RLWE, RLWE', and RGSW. Let $q, Q \in \mathbb{Z}_{>0}$, $R = \mathbb{Z}[X]/(X^{N_R} + 1)$, and $R_Q = R/QR$. Unless specified otherwise, all encryptions share a fixed secret key (for LWE, $\mathbf{s} \in \mathbb{Z}_q^n$; for RLWE, $s' \in R$).

For some error distribution χ over $\mathbb{Z}_q$ (typically a discrete Gaussian or a bounded uniform distribution) and its ring analogue χ_R over R_Q, an LWE encryption of $m \in \mathbb{Z}_q$ samples $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ uniformly and $e \leftarrow \chi$, and outputs

$$(\mathbf{a}, -\langle \mathbf{a}, \mathbf{s} \rangle + m + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

An RLWE encryption of $m \in R_Q$ samples $a \leftarrow R_Q$ uniformly and $e \leftarrow \chi_R$, and outputs

$$(a,\ -a \cdot s' + m + e) \in R_Q^2.$$

Let $\mathbf{g} = (g_0, \ldots, g_{D_d-1})$ be a *gadget vector* (e.g., $g_i = B^i$ with $B \geq 2$ and $D_d = \lceil \log_B Q \rceil$) such that every $t \in R_Q$ admits a *gadget decomposition* $t = \sum_{i=0}^{D_d-1} g_i\, t_i$ with small coefficients $t_i \in R$. We adopt the definitions of RLWE' and RGSW from [51]: for $m \in R_Q$ and secret $s' \in R$,

$$\mathsf{RLWE}'_{s'}(m) := \big(\mathsf{RLWE}_{s'}(g_0 m),\ \mathsf{RLWE}_{s'}(g_1 m),\ \ldots,\ \mathsf{RLWE}_{s'}(g'_{D_d-1} m)\big) \in R_Q^{2D_d},$$

$$\mathsf{RGSW}_{s'}(m) := \big(\mathsf{RLWE}'_{s'}(sm),\ \mathsf{RLWE}'_{s'}(m)\big) \in R_Q^{2\times 2D_d},$$

where each $\mathsf{RLWE}_{s'}(\cdot)$ component is instantiated with fresh, independent randomness and noise.

G.1 TFHE Bootstrapping

TFHE bootstrapping takes as input an LWE ciphertext $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e + \ m)$ and a function f. It returns a refreshed LWE ciphertext under the same key and modulus whose message is transformed to $f(m)$ and whose noise e' satisfies $|e'| < |e|$. The high-level overview of the bootstrapping framework is as follows:

$$\mathsf{LWE}_{q,\mathbf{s}}(m) \xrightarrow{\mathsf{MS}} \mathsf{LWE}_{2N_R,\mathbf{s}} \xrightarrow{\mathsf{BR}} \mathsf{RLWE}_{Q,s'} \xrightarrow{\mathsf{Ext}} \mathsf{LWE}_{Q,\mathbf{s}'} \xrightarrow{\mathsf{MS+KS}} \mathsf{LWE}_{q,\mathbf{s}}(f(m))$$

Step 1: Modulus Switching The modulus switching step maps an input LWE ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ to $(\mathbf{a}', b') \in \mathbb{Z}_{2N_R}^{n+1}$ by scaling and flooring each coefficient:

$$a'_i = \left\lfloor \tfrac{a_i \cdot 2N_R}{q} \right\rceil, \qquad b' = \left\lfloor \tfrac{b \cdot 2N_R}{q} \right\rceil.$$

This aligns the ciphertext with the subgroup $\langle X \rangle \subset R_{2N_R}$ for blind rotation.

Step 2: Blind Rotation We first apply the embedding $\mathcal{F} : \alpha \mapsto X^\alpha$ to the modulus-switched ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_{2N_R}^{n+1}$, obtaining $(X^{a_0}, \ldots, X^{a_{n-1}}, X^b)$. Then we initialize $c = (0, X^b \cdot tv(X)) \in R_Q^2$, where $tv(X)$ is a test polynomial For each $i \in [n]$, update $c \leftarrow (X^{a_i} - 1)c \odot C_i + c$, where $C_i = \mathsf{RGSW}_{s'}(s_i)$ and $\odot$ denotes the external product (involving gadget decomposition, NTT, Hadamard multiplication, and inverse NTT). The final ciphertext is $c = \mathsf{RLWE}_{Q,s'}\big(X^{b+\langle \mathbf{a},\mathbf{s} \rangle} \cdot tv(X)\big)$, whose constant term equals $f(m)$ by design of $tv(X)$.

Step 3: Sample Extraction Given $\mathbf{c} = (a, b) \in R_Q^2$ with $a = \sum_{j=0}^{N_R-1} \alpha_j X^j$ and secret s', define $\mathbf{a}' = (\alpha_0, -\alpha_{N_R-1}, \ldots, -\alpha_1), b' = b_0$. The extracted ciphertext $(\mathbf{a}', b') \in \mathbb{Z}_Q^{N_R+1}$ satisfies $b' - \langle \mathbf{a}', \mathbf{s}' \rangle = (b - a \cdot s')_0$, thus obtaining an $\mathsf{LWE}_{Q,\mathbf{s}'}(f(m))$ encryption.

Step 4: Modulus Switching and Key Switching Finally, the ciphertext under $\mathbf{s}'$ and modulus Q is converted back to the original key $\mathbf{s} \in \{0,1\}^n$ and q. One can refer to the details in [51].

H TFHE-friendly SNARK

In this section, we present a TFHE-friendly SNARK construction. We present an overview of a TFHE-friendly SNARK construction based on our packed sumcheck and Brakedown-style MW-MLE. The detailed protocol is provided in the Section H.1, H.2 and H.3. In Section H.1, we present vector relation checks including Hadamard product checks, NTT checks, and lookup checks. In Section H.2, we show how to use these tools to prove modulus switching, blind rotation initialization and blind rotation in TFHE bootstrapping. In Section H.3, we put all of this together to construct our TFHE-friendly SNARK.

Given an LWE ciphertext $(\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$, TFHE bootstrapping consists of four steps: (i) *Modulus switching.* Map (a_i, b) to $(\tilde{a}_i, \tilde{b}) \in \mathbb{Z}_{2N_R}^{n+1}$. (ii) *Blind rotation.* Apply the embedding $\mathcal{F} : \mathbb{Z}_{2N_R} \to R$, where $\mathcal{F}(\alpha) = X^\alpha$ for $\alpha \in \{\tilde{a}_0, \ldots, \tilde{a}_{n-1}, \tilde{b}\}$, and execute the refresh procedure to obtain an RLWE ciphertext with reduced noise. (iii) *Sample extraction.* Transform the RLWE output into a length-N_R LWE ciphertext over modulus Q. (iv) *Modulus/key switching.* Convert the modulus/keys back to (q, n).

We instantiate $N_R = n = 1024$ and set $q = Q = p = 2^{31} - 2^{27} + 1$, an NTT-friendly prime. In this way, the secret key and ciphertext moduli of LWE and RLWE can be kept consistent, incurring only a minor computational overhead for FHE. Thus, neither key switching nor modulus switching is required after blind rotation; and since sample extraction is essentially free, we delegate it to the verifier as in [55]. Thus, only steps (i)-(ii) require proofs.

All computations arising in step (i)-(ii) are expressed as a sequence of vector relations. Each vector relation is then certified using our packed sumcheck protocol. Below, we describe the how to prove the step (i)-(ii) one by one. An overview of our TFHE-friendly SNARK is illustrated in Figure 4.

Verifiable Modulus Switching. To prove modulus switching, it suffices to prove the rounding operation $\beta = \left\lfloor \frac{\alpha \cdot (2N_R)}{q} \right\rceil$. Because rounding is cumbersome to express with simple arithmetic constraints, we prove it via a table lookup. Assume $2N_R \mid (q-1)$, which is consistent with standard TFHE parameter choices. Under this assumption, $\mathbb{Z}_q$ is partitioned into $2N_R$ contiguous regions: $I_0 = \left[0, \frac{(q-1)}{2N_R}\right]$ and $I_t = \left[\frac{t(q-1)}{2N_R} + 1, \frac{(t+1)(q-1)}{2N_R}\right]$ for $t = 1, \ldots, 2N_R - 1$. The index t of the region containing α determines β after modulus switching (specifically, $\beta = t$). We therefore use a table lookup to identify the region of α and thereby compute β, which implements the modulus switch within our proof system.

Blind Rotation Initialization. Prior to blind rotation, the prover needs to prove the mapping $\mathcal{F}(\alpha) = X^\alpha$. X^α can be represented in an NTT basis:

$$X^\alpha := (\omega^\alpha, \omega^{3\alpha}, \ldots, \omega^{(2N_R-1)\alpha}),$$

where ω is a primitive $2N_R$-th root of unity in $\mathbb{Z}_Q$ and ω^i denotes its i-th power. We first verify ω^α via a lookup table argument showing that (α, ω^α) belongs to the precomputed table $\{(j, \omega^j)\}_{j=0}^{2N_R-1}$. The remaining $N_R - 1$ entries are then certified using the multiplicative recurrence

$$\omega^{i\alpha} = \omega^{(i-2)\alpha} \cdot (\omega^\alpha)^2 \qquad \text{for } i \in \{3, 5, \ldots, 2N_R - 1\}.$$

Because multiplicative constraints are cheaper than additional table lookups, this design reduces the prover's cost.

Blind Rotation. Blind rotation mainly includes four core operations: gadget decomposition, NTT, Hadamard product, and inverse NTT. Gadget decomposition is certified via a lookup table argument that enforces the value ranges of the decomposed components. Both the NTT and the inverse NTT are verified with a vector NTT check, while the Hadamard product is validated with a vector Hadamard product check.

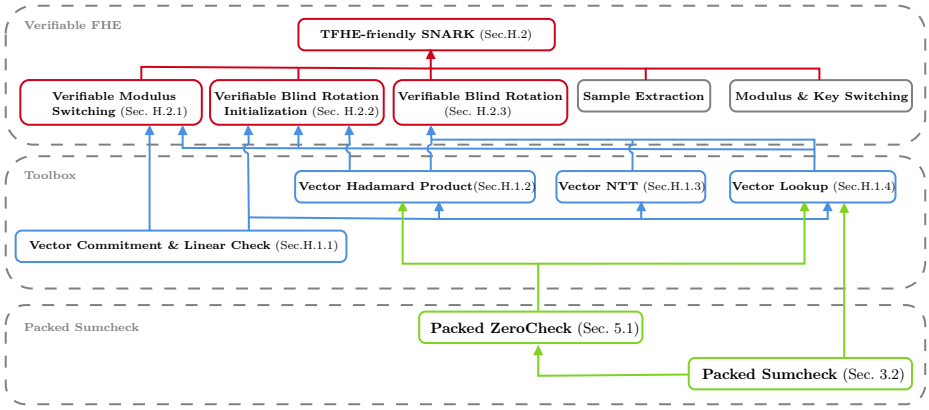


Fig. 4: The Workflow of Our TFHE-friendly SNARK

H.1 Vector Relation Proofs

In this subsection, we introduce several vector relation proof tools that serve as the foundational toolboxes for our TFHE-friendly SNARK construction, including vector Hadamard product relation, vector NTT relation, and vector lookup relation. All vectors are defined over the finite field $\mathbb{F}_p$, where the field size satisfies $p^k = O(2^\lambda)$, with λ denoting the security parameter. In the following proof protocols, these vectors will be treated as multilinear polynomials.

Vector Hadamard Product Relation. We consider the Hadamard product check over $\mathbb{F}_p$ for the relation $\mathbf{a} \circ \mathbf{b} = \mathbf{c}$, where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}_p^N$.

Definition 10 (Vector Hadamard Product Relation over $\mathbb{F}_p$). *The relation $\mathcal{R}_{VHPR}$ is the set of all tuples $(\mathbb{x};\mathbb{w}) = (([[a]],[[b]],[[c]]);(\mathbf{a},\mathbf{b},\mathbf{c}))$ such that $\mathbf{a}\circ\mathbf{b}=\mathbf{c}$.*

We can split the first $\log k$ variables of the multilinear polynomials $\mathbf{a},\mathbf{b},\mathbf{c}$ into k multilinear polynomials and k ZeroCheck instances, and then directly invoke the packed ZeroCheck protocol to verify the Hadamard product relation.

Vector NTT Check. We now turn to the Number Theoretic Transform (NTT) check over $\mathbb{F}_p$. In our TFHE-friendly SNARK, the relevant polynomials have degree N_R-1. Hence, we partition vectors into blocks of size N_R. Let $\mathbf{a},\mathbf{b}\in\mathbb{F}_p^M$ with $M = N_R\cdot m$. Partition $\mathbf{a}$ and $\mathbf{b}$ into m blocks $\mathbf{a}_i,\mathbf{b}_i$ for $i\in[0,m-1]$. The prover's goal is to prove that for all i, the relation $\mathbf{a}_i = H\cdot\mathbf{b}_i$ holds, where H is the $N_R\times N_R$ NTT matrix.

The prover invokes the vector commitment scheme with the vector length parameter set to $n_c = N_R$, and commits to all sub-vectors $\mathbf{a}_i$ and $\mathbf{b}_i$. Because the NTT relation is linear, random linear combinations can be used to fold the m individual NTT checks into a single aggregated check.

Definition 11 (Vector NTT Relation over $\mathbb{F}_p$). *The relation $\mathcal{R}_{VNR}$ is the set of all tuples $(\mathbb{x},\mathbb{w}) = ((Com(\mathbf{a}),Com(\mathbf{b}));(\mathbf{a},\mathbf{b}))$ such that $\mathbf{a}_i = H\mathbf{b}_i$ where $\mathbf{a},\mathbf{b}\in\mathbb{F}_p^M$ with $M = N_R\cdot m$, and both vectors are partitioned into m blocks $\mathbf{a}_i,\mathbf{b}_i$ for $i\in[0,m-1]$ and $H\in\mathbb{F}_p^{N_R\times N_R}$ denotes the NTT matrix.*

Vector NTT IOP Construction.

1. The verifier samples k random challenges $\mathbf{r}_0,\cdots,\mathbf{r}_{k-1}\in\mathbb{F}_p^m$ and sends them to the prover.
2. For each $s\in\{0,\cdots,k-1\}$, the prover computes the folded vectors:

$$\tilde{\mathbf{a}}_s = \sum_{i=0}^{m-1}\mathbf{r}_s[i]\cdot\mathbf{a}_i,\quad \tilde{\mathbf{b}}_s = \sum_{i=0}^{m-1}\mathbf{r}_s[i]\cdot\mathbf{b}_i.$$

   The prover sends commitments to $\{\tilde{\mathbf{a}}_s\}_{s\in[0,k-1]}$ and $\{\tilde{\mathbf{b}}_s\}_{s\in[0,k-1]}$ to the verifier.
3. The parties invoke the vector linear evaluation IOP to verify the correctness of these linear combinations.
4. The verifier locally checks that $\tilde{\mathbf{a}}_s = H\cdot\tilde{\mathbf{b}}_s$ holds for each $s\in\{0,\cdots,k-1\}$.

Remark 6. In the Vector NTT IOP, the verifier's computational complexity is $O(m + N_R\log N_R)$. When $m = O(N_R)$ (as is the case in Verifiable FHE; see Section H.2 for details), the verifier's complexity remains sublinear in the vector length M, thereby preserving succinctness.

Lemma 6. *The Vector NTT IOP achieves perfect completeness and has soundness error $\frac{1}{p^k}$.*

Proof. Perfect completeness follows directly. We now prove that the soundness error is at most $\frac{1}{p^k}$. For each row index i, define the polynomial

$$p_i(\mathbf{r}) = \left(\sum_{j=0}^{m-1} \mathbf{r}[j] \cdot \mathbf{a}_j\right)[i] - \left(H\left(\sum_{j=0}^{m-1} \mathbf{r}[j] \cdot \mathbf{b}_j\right)\right)[i],$$

where $\mathbf{r}$ is the random challenge vector. If the prover cheats, then there exists at least one index i^* such that p_{i^*} is a nonzero polynomial. Since $\deg(p_{i^*}) \leq 1$, the Schwartz–Zippel lemma implies that for a single random evaluation point, the probability that p_{i^*} vanishes is at most $\frac{1}{p}$. As the protocol samples k independent random points, the probability that all k evaluations vanish is at most $\frac{1}{p^k}$. Hence, the soundness error is bounded by $\frac{1}{p^k}$. □

Vector Lookup Check Finally, we describe the vector lookup argument over $\mathbb{F}_p$. For a vector $\mathbf{a} \in \mathbb{F}_p^N$ and a public table $\mathbf{t} \in \mathbb{F}_p^M$, the goal is to prove that every element of $\mathbf{a}$ is contained in $\mathbf{t}$. Let $N = n \cdot N_c$ and $M = m \cdot N_c$. The prover partitions the vectors $\mathbf{a}$ and $\mathbf{t}$ into blocks of length N_c and commits to each block using the vector commitment scheme. Our construction is based on the cq lookup protocol [32].

Definition 12 (Vector Lookup Relation over $\mathbb{F}_p$). *The relation $\mathcal{R}_{VLTR}$ is set of all tuples $(\mathbb{x}; \mathbb{w}) = ((Com(\mathbf{a}), \mathbf{t}); (\mathbf{a}, \mathbf{h}))$, where $\mathbf{a} \in \mathbb{F}_p^N$, $\mathbf{t} \in \mathbb{F}_p^M$, and $\mathbf{h} \in \mathbb{F}_p^M$ is a multiplicity vector such that every element of $\mathbf{a}$ appears in the table $\mathbf{t}$.*

The protocol builds on the following lemma, which reduces the lookup claim to a rational function identity.

Lemma 7 ([32]). *All elements of $\mathbf{a} \in \mathbb{F}_p^N$ belong to the table $\mathbf{t} \in \mathbb{F}_p^M$ if and only if there exists a multiplicity vector $\mathbf{h} \in \mathbb{F}_p^M$ satisfying:*

$$\sum_{j=0}^{N-1} \frac{1}{X + \mathbf{a}[j]} = \sum_{e=0}^{M-1} \frac{\mathbf{h}[e]}{X + \mathbf{t}[e]}.$$

Vector Lookup Proof Construction.

1. The verifier samples s random challenges $r_0, \cdots, r_{s-1} \in \mathbb{F}_p$ and sends them to the prover.
2. For each $i \in [0, s-1]$, the prover defines two new vectors:
   - $\mathbf{f}_i \in \mathbb{F}_p^N$, where $\mathbf{f}_i[j] = \frac{1}{r_i + \mathbf{a}[j]}$ for $j \in [0, N-1]$.
   - $\mathbf{g}_i \in \mathbb{F}_p^M$, where $\mathbf{g}_i[e] = \frac{\mathbf{h}_i[e]}{r_i + \mathbf{t}[e]}$ for $e \in [0, M-1]$.

   Using the vector length parameter N_c, the prover partitions and commits to $\{\mathbf{f}_i\}_{i \in [0,s-1]}$ and $\{\mathbf{g}_i\}_{i \in [0,s-1]}$.

3. The parties invoke a packed sumcheck protocol to prove that for each $i \in [0, s-1]$:
$$\sum_{j=0}^{N-1} \mathbf{f}_i[j] = \sum_{e=0}^{M-1} \mathbf{g}_i[e].$$
4. The parties invoke a packed ZeroCheck protocol to prove the following for each $i \in [0, s-1]$:
   - $\mathbf{f}_i[j] \cdot (r_i + \mathbf{a}[j]) = 1$ for all $j \in [0, N-1]$.
   - $\mathbf{g}_i[e] \cdot (r_i + \mathbf{t}[e]) = \mathbf{h}[e]$ for all $e \in [0, M-1]$.

Lemma 8. *Assume that the sumcheck protocol has soundness error ϵ_{SUM} and the ZeroCheck protocol has soundness error ϵ_{ZERO}. Then the Vector Lookup IOP achieves perfect completeness and has soundness error at most*
$$s\epsilon_{\mathsf{SUM}} + 2s\epsilon_{\mathsf{ZERO}} + \tfrac{(N+M-1)^s}{p^s}.$$

Proof. Let $LS(X) = \sum_{j=0}^{N-1} \frac{1}{X+\mathbf{a}[j]}$ and $RS(X) = \sum_{e=0}^{M-1} \frac{\mathbf{h}[e]}{X+\mathbf{t}[e]}$. By Lemma 7, if not all elements of $\mathbf{a}$ are contained in the table $\mathbf{t}$, then $LS(X) \neq RS(X)$. Suppose we sample a random point $\alpha \in \mathbb{F}_p$. If $LS(\alpha) = RS(\alpha)$, then we obtain
$$\prod_{z=0}^{N-1} (\alpha + \mathbf{a}[z]) \cdot \prod_{z=0}^{M-1} (\alpha + \mathbf{t}[z]) \cdot LS(\alpha) = \prod_{z=0}^{N-1} (\alpha + \mathbf{a}[z]) \cdot \prod_{z=0}^{M-1} (\alpha + \mathbf{t}[z]) \cdot RS(\alpha).$$
Both $\prod_{z=0}^{N-1}(X+\mathbf{a}[z]) \cdot \prod_{z=0}^{M-1}(X+\mathbf{t}[z]) \cdot LS(X)$ and $\prod_{z=0}^{N-1}(X+\mathbf{a}[z]) \cdot \prod_{z=0}^{M-1}(X+\mathbf{t}[z]) \cdot RS(X)$ are polynomials of degree at most $N+M-1$. By the Schwartz–Zippel lemma, the probability that they agree at a random point is at most $\frac{N+M-1}{p}$. Therefore, when sampling s independent random points, the probability that all evaluations agree is at most $\frac{(N+M-1)^s}{p^s}$.

Since the Vector Lookup IOP invokes s instances of sumcheck and $2s$ instances of ZeroCheck, its overall soundness error is bounded by the union bound:
$$s\epsilon_{\mathsf{SUM}} + 2s\epsilon_{\mathsf{ZERO}} + \tfrac{(N+M-1)^s}{p^s}.$$

□

Definition 13 (l-Vector Lookup Relation over $\mathbb{F}_p$.). *The relation $\mathcal{R}_{LVLTR}$ is defined over tuples*
$$\big((Com(\mathbf{a}_0), \ldots, Com(\mathbf{a}_{l-1}), \mathbf{t}_0, \ldots, \mathbf{t}_{l-1}), (\mathbf{a}_0, \ldots, \mathbf{a}_{l-1}, \mathbf{h})\big),$$
where each $\mathbf{a}_i \in \mathbb{F}_p^N$ and $\mathbf{t}_i \in \mathbb{F}_p^M$. The relation requires that for every index $i \in [0, N-1]$, there exists some $j \in [0, M-1]$ such that
$$(\mathbf{a}_0[i], \ldots, \mathbf{a}_{l-1}[i]) = (\mathbf{t}_0[j], \ldots, \mathbf{t}_{l-1}[j]).$$
Here, $\mathbf{h} \in \mathbb{F}_p^M$ denotes a multiplicity vector certifying that every element of $(\mathbf{a}_0, \ldots, \mathbf{a}_{l-1})$ is contained in the set defined by $(\mathbf{t}_0, \ldots, \mathbf{t}_{l-1})$.

Lemma 9. *All elements of* $(\mathbf{a}_0, \cdots, \mathbf{a}_{l-1}) \in \mathbb{F}_p^{l\times N}$ *belong to the table* $(\mathbf{t}_0, \cdots, \mathbf{t}_{l-1}) \in \mathbb{F}_p^{l\times M}$ *if and only if there exists a multiplicity vector* $\mathbf{h} \in \mathbb{F}_p^M$ *satisfying:*

$$\sum_{j=0}^{N-1} \frac{1}{X + \sum_{i=0}^{l-1}(\mathbf{a}_i[j] \cdot Y^i)} = \sum_{e=0}^{M-1} \frac{\mathbf{h}[e]}{X + \sum_{i=0}^{l-1}(\mathbf{t}_i[e] \cdot Y^i)}.$$

Proof. We define the polynomials

$$f_j(X,Y) = X + \sum_{i=0}^{l-1} \mathbf{a}_i[j] \cdot Y^i, \qquad g_j(X,Y) = X + \sum_{i=0}^{l-1} \mathbf{t}_i[j] \cdot Y^i.$$

Let

$$LS(X,Y) = \sum_{j=0}^{N-1} \frac{1}{f_j(X,Y)}, \qquad RS(X,Y) = \sum_{j=0}^{M-1} \frac{\mathbf{h}[j]}{g_j(X,Y)}.$$

If the l-Vector Lookup Relation holds, then each $f_i(X,Y)$ must appear in the set $\{g_j(X,Y)\}_{j\in[0,M-1]}$. In this case, $\mathbf{h}[j]$ denotes the multiplicity with which $g_j(X,Y)$ occurs among $\{f_i\}_{i\in[0,N-1]}$, and it follows immediately that $LS(X,Y) = RS(X,Y)$.

Conversely, suppose there exists a vector $\mathbf{h}$ such that $LS(X,Y) = RS(X,Y)$. Then

$$\sum_{j=0}^{N-1} \frac{1}{f_j(X,Y)} - \sum_{j=0}^{M-1} \frac{\mathbf{h}[j]}{g_j(X,Y)} = 0.$$

If some $f_i(X,Y)$ does not belong to $\{g_j(X,Y)\}_{j\in[0,M-1]}$, consider

$$S(X,Y) = \left(\prod_{j=0}^{N-1} f_j(X,Y)\right)\left(\prod_{j=0}^{M-1} g_j(X,Y)\right)\big(LS(X,Y) - RS(X,Y)\big).$$

In this case, $S(X,Y)$ contains a nonzero term that cannot cancel out. Since each f_j and g_j has degree at most $(l-1)$, the degree of $S(X,Y)$ is bounded by $(l-1)(N+M-1) < p$. Hence $S(X,Y)$ is not the zero polynomial over $\mathbb{F}_p$, contradicting the assumption. Therefore, if $LS(X,Y) = RS(X,Y)$ holds for some $\mathbf{h}$, the L-Vector Lookup Relation must also hold. □

Optimization for small public tables. Since the public lookup tables in our TFHE instantiation are small, the prover sends the multiplicity vector $\mathbf{h}$ directly to the verifier. The verifier then computes the table-side vectors $\mathbf{g}_i$ and their sums locally, so the prover only commits to $\mathbf{f}_i$ and proves the corresponding sumcheck and ZeroCheck relations.

H.2 Verifiable Computational Steps in Bootstrapping

In our TFHE-friendly SNARK, the prover commits to a sequence of vectors produced along the bootstrapping pipeline. The prover and verifier agree in

advance on the order of these vectors, allowing the verifier to associate each commitment with its designated role. We instantiate the vector commitments using MW-Brakedown. Unless stated otherwise, all packed ZeroCheck instances in our TFHE construction are instantiated with $d = 2$, as they enforce vector Hadamard product relations.

By carefully selecting parameters, we can align the secret key length and modulus of LWE with those of RLWE, thereby avoiding the need for subsequent modulus and key switching in Section G.1. The detailed parameter design is deferred to Section 6.2. Since sample extraction is essentially free, we delegate it to the verifier as in [55]

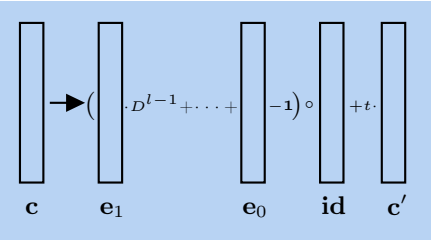


Fig. 5: Modulus Switching

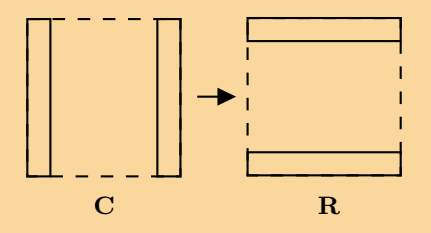


Fig. 6: Blind Rotation Initialization

Modulus Switching The modulus switching operation maps $\mathbf{c} = (\mathbf{a}, b) \in \mathbb{Z}_q^{n+1}$ to $\mathbf{c}' = (\mathbf{a}', b') \in \mathbb{Z}_{2N_R}^{n+1}$. To prove this relation, we rely on the following lemma:

Lemma 10. *Suppose* $2N_R \mid (q-1)$ *and let* $q = t\cdot(2N_R)+1$. *For any* $\alpha \in \mathbb{Z}_q \backslash \{0\}$, *there exists a unique decomposition* $\alpha = \beta \cdot t + \gamma$, *where* $\beta \in [0, 2N_R - 1]$ *and* $\gamma \in [1, t]$, *such that*

$$\left\lfloor \frac{\alpha \cdot (2N_R)}{q} \right\rceil = \beta.$$

Proof. We compute:

$$\begin{aligned} \left\lfloor \frac{\alpha \cdot (2N_R)}{q} \right\rceil &= \left\lfloor \frac{(\beta \cdot t + \gamma) \cdot 2N_R}{q} \right\rceil = \left\lfloor \frac{(\beta \cdot \frac{q-1}{2N_R} + \gamma) \cdot 2N_R}{q} \right\rceil \\ &= \left\lfloor \frac{\beta(q-1) + \gamma \cdot 2N_R}{q} \right\rceil = \beta + \left\lfloor \frac{\gamma \cdot 2N_R - \beta}{q} \right\rceil. \end{aligned}$$

Since $0 \le \beta \le 2N_R - 1$ and $\gamma \ge 1$, we have $(\gamma \cdot 2N_R - \beta) > 0$. Moreover, as $\gamma \le t = \frac{q-1}{2N_R}$, it follows that $(\gamma \cdot 2N_R - \beta) < q$. Hence, $\lfloor \frac{\gamma \cdot 2N_R - \beta}{q} \rceil = 0$, which yields $\lfloor \frac{\alpha \cdot (2N_R)}{q} \rceil = \beta$.

Lemma 11. *Let* $\mathbf{id}, \mathbf{r}, \mathbf{c}' \in \mathbb{Z}_q^{n+1}$ *such that every entry of* $\mathbf{id}$ *lies in* $\{0, 1\}$, *every entry of* $\mathbf{r}$ *lies in* $[1, t]$, *and every entry of* $\mathbf{c}'$ *lies in* $[0, 2N_R - 1]$. *If*

$$\mathbf{c} = (t \cdot \mathbf{c}' + \mathbf{r}) \circ \mathbf{id},$$

and

$$\mathbf{c}' \circ (\mathbf{1} - \mathbf{id}) = \mathbf{0},$$

then $\mathbf{c}'$ *is exactly the result of applying modulus switching to* $\mathbf{c}$.

Proof. It suffices to prove the claim coordinate-wise. Fix a coordinate i and write $c = \mathbf{c}[i]$, $c' = \mathbf{c}'[i]$, $r = \mathbf{r}[i]$, and $id = \mathbf{id}[i]$. By assumption,

$$id \in \{0, 1\}, \qquad r \in [1, t], \qquad c' \in [0, 2N_R - 1],$$

and

$$c = (tc' + r) \cdot id \pmod{q}, \qquad c'(1 - id) = 0.$$

We first consider the case $c \equiv 0 \pmod{q}$. If $id = 1$, then

$$c \equiv tc' + r \pmod{q}.$$

However, since $c' \in [0, 2N_R - 1]$ and $r \in [1, t]$, we have

$$1 \leq tc' + r \leq (2N_R - 1)t + t = 2N_R t = q - 1 < q.$$

Thus $tc' + r$ is a nonzero integer strictly smaller than q, and hence cannot be congruent to 0 modulo q. This contradicts $c \equiv 0 \pmod{q}$. Therefore $id = 0$. The additional constraint $c'(1 - id) = 0$ then gives $c' = 0$. This agrees with the modulus switching formula: $c' = 0 = \left\lfloor \frac{0 \cdot 2N_R}{q} \right\rceil$.

We now consider the case $c \not\equiv 0 \pmod{q}$. If $id = 0$, then

$$c = (tc' + r) \cdot 0 \equiv 0 \pmod{q},$$

contradicting $c \not\equiv 0 \pmod{q}$. Hence $id = 1$, and therefore

$$c \equiv tc' + r \pmod{q}.$$

As above, $1 \leq tc' + r \leq q - 1$. Thus the congruence is in fact an equality over integers $c = tc' + r$. By Lemma 10, this decomposition is unique with $c' \in [0, 2N_R - 1]$ and $r \in [1, t]$. Therefore

$$c' = \left\lfloor \frac{c \cdot 2N_R}{q} \right\rceil.$$

This proves the claim for every coordinate, and hence for the whole vector $\mathbf{c}'$. □

To facilitate proving that $\mathbf{r} \in [1, t]^{n+1}$, it suffices to show that $\mathbf{r} - \mathbf{1} \in [0, t - 1]^{n+1}$. To further reduce table size, we decompose $\mathbf{r} - \mathbf{1}$ into l blocks under a D-ary expansion. We present the proof framework in Figure 5, and the protocol details in Figure 7.

Overall, the modulus-switching proof commits to $l + 3 + s(l + 2)$ vectors of length $n+1$. It invokes $2+s(l+2)$ packed ZeroCheck instances of size $n+1$ and $s(l + 2)$ degree-one packed sumcheck instances of size $n + 1$. The commitments consist of $l+3$ original vectors, namely $\mathbf{c}$, $\mathbf{c}'$, $\mathbf{id}$, and the l digit vectors, together with $s(l + 2)$ reciprocal vectors generated by the $l + 2$ range checks. The two Hadamard-product relations account for two packed ZeroCheck instances, while the reciprocal relations account for the remaining $s(l + 2)$ packed ZeroCheck instances in lookup arguments. The $s(l + 2)$ packed sumcheck instances prove the corresponding lookup sum relations.

The Verifiable Modulus Switch

We define a vector $\mathbf{id} \in \mathbb{Z}_q^{n+1}$ such that $\mathbf{id}[i] = 0$ if $\mathbf{c}[i] = 0$ and $\mathbf{id}[i] = 1$ otherwise. The prover additionally commits to $\mathbf{id}$. Moreover, the prover sets the vector length parameter to $n_c = n + 1$ when committing to $\mathbf{c}, \mathbf{c}'$, and $\{\mathbf{e}_i\}_{i\in[0,l-1]}$.

- Using the vector Hadamard product IOP, the prover demonstrates the following two relations:

$$\mathbf{c} = \left(\mathbf{e}_{l-1} \cdot D^{l-1} + \cdots + \mathbf{e}_0 + \mathbf{1} + \mathbf{c}' \cdot t\right) \circ \mathbf{id}.$$

  and

$$\mathbf{c}' \circ (\mathbf{1} - \mathbf{id}) = \mathbf{0}.$$

- Using the vector lookup IOP, the prover proves that all entries of $\mathbf{id}$ lie in $\{0, 1\}$, each element of $\{\mathbf{e}_j\}_{j\in[0,l-2]}$ lies in $[0, D-1]$, each element of $\mathbf{e}_{l-1}$ lies in $[0, \lfloor \frac{t-1}{D^{l-1}} \rfloor]$, and each element of $\mathbf{c}'$ lies in $[0, 2N_R - 1]$.

Fig. 7: The Verifiable Modulus Switch

Blind Rotation Initialization Let $w_1, \ldots, w_{2N_R-1}$ denote the odd powers of a primitive $2N_R$-th root of unity in $\mathbb{Z}_Q$, forming an NTT basis of $\mathbb{Z}_Q[X]/(X^{N_R}+1)$. After modulus switching, the entries of $\mathbf{c}'$ are placed at these exponent indices, yielding an $(n+1) \times N_R$ matrix $\mathbf{C}$ whose rows encode $X^{a_i'}$ or $X^{b'}$ in the NTT basis. In verifiable TFHE bootstrapping, the prover commits to $\mathbf{C}$ column-wise, denoting the i-th column as $\mathbf{C}_{\cdot,i}$. First, by a lookup against the table $(\alpha, w_1^\alpha)_{\alpha\in[0,2N_R-1]}$, the verifier checks correctness of the first column $\mathbf{C}_{\cdot,0}$. For subsequent columns, observe that $(\mathbf{C}_{\cdot,0} \circ \mathbf{C}_{\cdot,0}) \circ \mathbf{C}_{\cdot,i} = \mathbf{C}_{\cdot,i+1}$, which is verified using packed ZeroCheck with $d = 3$. This certifies the validity of the entire matrix $\mathbf{C}$.

Since $\mathbf{C}$ is committed column-wise but each row corresponds to a polynomial ring element, we also construct a row-wise commitment matrix R. To check consistency between $\mathbf{C}$ and $\mathbf{R}$, the verifier samples random matrices $\mathbf{S}_0 \in \mathbb{Z}_p^{(n+1)\times k}$ and $\mathbf{S}_1 \in \mathbb{Z}_Q^{N_R\times k}$, requests the prover to compute $\mathbf{T}_0 = \mathbf{S}_0^\top C, \mathbf{T}_1 = R\,\mathbf{S}_1$, and verifies $\mathbf{T}_0^\top \mathbf{S}_1 = \mathbf{T}_1^\top \mathbf{S}_0$. We present the proof framework in Figure 6, and the protocol details in Figure 8.

Overall, the blind-rotation initialization commits to $N_R + s$ vectors of length $n+1$ and $n+1$ vectors of length N_R. It invokes s packed ZeroCheck instances of size $n+1$, one packed ZeroCheck instance of size $(n+1)(N_R-1)$, and s degree-one packed sumcheck instances of size $n+1$. The commitments consist of the N_R columns of $\mathbf{C}$, the $n+1$ rows of $\mathbf{R}$, and s reciprocal vectors introduced by the lookup for the first column. The reciprocal relations and their sum relations account for the s packed ZeroCheck and s packed sumcheck instances of size $n+1$, respectively. The $N_R - 1$ recurrence relations, each over vectors of length $n+1$, are concatenated and verified by a single packed ZeroCheck instance of size $(n+1)(N_R-1)$ with $d = 3$.

The Verifiable Blind Rotation Initialization

The prover commits to the columns of matrix $\mathbf{C}$ with parameter $n_c = n + 1$, and to the rows of matrix $\mathbf{R}$ with parameter $n_c = N$.

- By performing a vector lookup IOP against the table $\{(\alpha, w_1^{\alpha})\}_{\alpha \in [0, 2N_R - 1]}$, the verifier checks the correctness of the first column $\mathbf{C}_{\cdot,0}$. The remaining columns are then verified using the vector Hadamard product IOP by proving $(\mathbf{C}_{\cdot,0} \circ \mathbf{C}_{\cdot,0}) \circ \mathbf{C}_{\cdot,i} = \mathbf{C}_{\cdot,i+1}$.
- To enforce consistency between $\mathbf{C}$ and $\mathbf{R}$, the verifier samples random matrices/vectors $\mathbf{S}_0 \in \mathbb{Z}_p^{(n+1)\times k}$ and $\mathbf{S}_1 \in \mathbb{Z}_p^{N_R \times k}$, asks the prover to compute

$$\mathbf{T}_0 = \mathbf{S}_0^{\top}\mathbf{C}, \qquad \mathbf{T}_1 = \mathbf{R}\,\mathbf{S}_1,$$

  and verifies that $\mathbf{T}_0\mathbf{S}_1 = \mathbf{S}_0^{\top}\mathbf{T}_1$.

Fig. 8: The Verifiable Blind Rotation Initialization

Blind Rotation Blind rotation constitutes the primary computational cost in TFHE bootstrapping and requires n highly repetitive iterations (Figure 9). Each vector corresponds to an element of $\mathbb{Z}_p[X]/(X^{N_R}+1)$. In our verifiable protocol, the prover aggregates ciphertexts from the same position across iterations into length-nN vectors and proves the corresponding relations in a unified manner. By contrast, the approach of Thibault and Walter [55] recursively proves each iteration, which is less efficient. Leveraging the repetitive structure enables unified proofs and achieves significant performance improvements.

In Figure 9, the index i ranges over $\{1, \ldots, n\}$. During the initialization phase of blind rotation, the RLWE ciphertext is represented in interpolation form. Before Round 1, it is denoted $(\mathbf{ai}_0, \mathbf{bi}_0) \in \mathbb{Z}_Q^{2N_R}$. After an inverse NTT (INTT), the protocol enters an iterative loop of n CMUX gates, yielding $(\mathbf{ai}_n, \mathbf{bi}_n) \in \mathbb{Z}_Q^{2N_R}$. A final INTT produces the output RLWE ciphertext $(\mathbf{ac}_{n+1}, \mathbf{bc}_{n+1}) \in \mathbb{Z}_Q^{2N_R}$. The INTT step before and after the n CMUX iterations can be directly proven using the vector NTT check. We focus on the proof of the n CMUX gates, as illustrated in Figure 10.

Let $N_{\mathsf{BR}} := nN_R$ denote the length of an aggregated blind-rotation vector. Overall, blind rotation commits to $8D_d + 8 + 2D_d s$ vectors of length N_{BR}. It invokes $4D_d + 2 + 2D_d s$ packed ZeroCheck instances of size N_{BR} and $2D_d s$ degree-one packed sumcheck instances of size N_{BR}. The commitments consist of $2D_d$ gadget-decomposition vectors, $2D_d$ transformed vectors, four D_d-indexed vector families appearing in the Hadamard products, eight remaining input, output, and auxiliary vectors, and $2D_d s$ reciprocal vectors introduced by lookup. The intermediate products account for $4D_d$ packed ZeroCheck instances, the two accumulator updates account for two instances, and the reciprocal relations account for the remaining $2D_d s$ instances. The $2D_d s$ packed sumcheck instances prove the corresponding lookup sum relations.

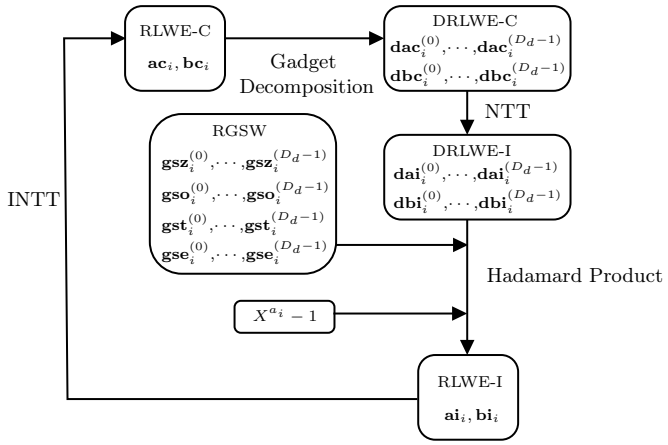


Fig. 9: Round-i CMUX Operation

Verifiable n CMUX Gates

The prover first commits to the relevant vectors using the vector length parameter $n_c = N_R$. For clarity, subscripts are omitted to indicate aggregated vectors. Specifically, we define the concatenated vectors $\mathbf{ai}' := \mathbf{ai}_0||\cdots||\mathbf{ai}_{n-1}$, $\mathbf{bi}' := \mathbf{bi}_0||\cdots||\mathbf{bi}_{n-1}$, $\mathbf{ai} := \mathbf{ai}_1||\cdots||\mathbf{ai}_n$, $\mathbf{bi} := \mathbf{bi}_1||\cdots||\mathbf{bi}_n$.

- **Gadget decomposition via vector lookup.**

$$\mathbf{ac} = \sum_{s\in[0,D_d-1]} \mathbf{dac}^{(s)} \cdot B^s, \quad \mathbf{bc} = \sum_{s\in[0,D_d-1]} \mathbf{dbc}^{(s)} \cdot B^s.$$

  The prover then uses vector lookup IOP to prove that entries of $\{\mathbf{dac}^{(i)}, \mathbf{dbc}^{(i)}\}_{i\in[0,D_d-2]}$ lie in $[0, B-1]$, while entries of $\mathbf{dac}^{(D_d-1)}, \mathbf{dbc}^{(D_d-1)}$ lie in $\left[0, \lfloor p/B^{(D_d-1)} \rfloor\right]$.
- **Vector NTT checks.** The prover proves that the $2D_d$ relevant pairs $\{(\mathbf{dac}^{(i)}, \mathbf{dai}^{(i)}), (\mathbf{dbc}^{(i)}, \mathbf{dbi}^{(i)})\}_{i\in[0,D_d-1]})$ satisfy the NTT relation.
- **Vector Hadamard product checks.** We define two auxiliary vectors:

$$\mathbf{at} = \sum_{s\in[0,D_d-1]} \left(\mathbf{gsz}^{(s)} \circ \mathbf{dai}^{(s)}\right) + \sum_{s\in[0,D_d-1]} \left(\mathbf{gst}^{(s)} \circ \mathbf{dbi}^{(s)}\right),$$
$$\mathbf{bt} = \sum_{s\in[0,D_d-1]} \left(\mathbf{gso}^{(s)} \circ \mathbf{dai}^{(s)}\right) + \sum_{s\in[0,D_d-1]} \left(\mathbf{gse}^{(s)} \circ \mathbf{dbi}^{(s)}\right).$$

  The prover prove the correctness of $\mathbf{at}$ and $\mathbf{bt}$ and then prove:

$$\mathbf{ai} = (X^{\mathbf{a}} - 1) \circ \mathbf{at} + \mathbf{ai}', \qquad \mathbf{bi} = (X^{\mathbf{a}} - 1) \circ \mathbf{bt} + \mathbf{bi}'.$$

  where $(X^{\mathbf{a}} - 1) := (X^{a_1} - 1||\cdots||X^{a_n} - 1)$
- **Final NTT checks.** The prover proves that $(\mathbf{ac}, \mathbf{ai})$ and $(\mathbf{bc}, \mathbf{bi})$ satisfy the NTT relation.

Fig. 10: The Verifiable Blind Rotation

H.3 Putting Everything Together

By combining the verifiable modulus switching, verifiable blind rotation initialization, and verifiable blind rotation, we obtain a SNARK specialized for TFHE bootstrapping. The overall circuit size of the bootstrapping procedure is $C = O(nN_R)$ and $n = N_R = O(\sqrt{C})$. (The detailed parameter design is deferred to Section 6.2.) The resulting asymptotic costs are summarized in Table 5.

Table 5: TFHE-friendly SNARK Complexity

	Prover	**Verifier**	**Proof Size**
Vector Commitment	$(nN_R \log N_R)$	$O(n + N_R \log N_R)$	$O(n + N_R)$
Vector Hadamard Product	$O(nN_R)$	$O(\log n + N_R)$	$O(\log n + N_R)$
Vector NTT	$O(nN_R)$	$O(n + N_R \log N_R)$	$O(N_R)$
Vector Lookup	$O(nN_R)$	$O(\log n + N_R)$	$O(\log n + N_R)$
SNARK	$O(C \log C)$	$O(\sqrt{C} \log C)$	$O(\sqrt{C})$

I Sumcheck Performance on Intel CPU

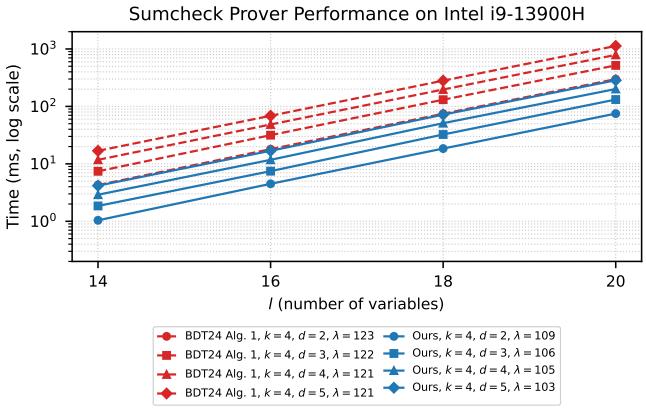


Fig. 11: Sumcheck prover performance on Intel i9-13900H. The data are generated from the CSV benchmark file corresponding to this appendix.

We show the prover performance of our packed sumcheck protocol on an Intel i9-13900H processor in Figure 11. Compared to Algorithm 1 from [7] with $k = 4$, our packed sumcheck achieves a speedup of 3.86×–4.23× on the overlapping benchmark points. This performance improvement is consistent with the results observed on the Apple M2 Pro processor.