Revisiting Sum-check-based Polynomial Commitment Schemes

Yuncong Zhang

School of Cyber Science and Technology, Shandong University
yuncong@sdu.edu.cn

Abstract. The sum-check protocol is a fundamental building block in succinct arguments. However, its security formalization is often tightly coupled with the larger protocol in which it is embedded, making modular design and analysis challenging. To address this limitation, we introduce *functional proof systems (FPS)*, generalizing interactive proof systems by viewing the verifier as a function parameterized by the prover, and defining security by asserting properties of this function. This novel perspective not only enables a clean, self-contained security definition for sum-check, but also opens possibility for designing and formalizing security of protocols that prove statements dynamically generated during the protocol—both tasks are difficult for traditional proof systems.
We develop a framework for composing multiple FPSs by executing them in parallel, and analyzing security of the composite protocol. We apply this framework to analyze existing protocols, including BaseFold, a popular polynomial commitment scheme, and BulletProofs, a well-known inner product argument, providing more modular and simpler security proofs than their original analysis. Particularly for BaseFold, our security proof avoids the need for introducing a new variant of correlated agreement theorem, thus building its security directly on the well-studied theorems of FRI. Finally, we construct a new transparent, pairing-free, doubly efficient, and homomorphically additive polynomial commitment scheme by composing existing protocols, demonstrating the practical utility of our framework for designing novel cryptographic schemes.

Keywords: Proof System · SNARK · Sum-check · Polynomial Commitment Scheme.

1 Introduction

The sum-check protocol, introduced by Lund et al. [LFKN90], is an interactive proof system designed to verify statements of the form $c = \sum_{\mathbf{x}\in\{0,1\}^\mu} f(\mathbf{x})$, where f represents a μ-variate polynomial over a finite field. This protocol serves as a fundamental building block in modern cryptographic systems such as succinct argument systems [GKR08], particularly SNARKs [CBBZ22, Set20, LZZ+25, STW23, KS23, XZZ+19, ZXZS20, ZCF23, ACFY24b]. Its importance in research and practice is further evidenced by the substantial body of work on optimizing its performance and efficiency [BDDT25, BCF+25, DT24, Gru24, SP25].

In its early applications, the sum-check protocol was typically deployed alongside polynomial commitment schemes (PCS) [ACFY24a,ZCF23,GPS25,ZXZS20, KT23], with both components serving complementary roles in SNARK constructions. However, recent developments have demonstrated that sum-check itself can facilitate more efficient PCS. Two prominent examples exemplify this evolution:

- **BaseFold** [ZCF23] stands as the first hash-based multilinear PCS to achieve both linear prover complexity and logarithmic verifier complexity. The scheme's design centers on the parallel execution of sum-check and the FRI protocol [BBHR18]. The subsequent work WHIR [ACFY24b] adopts a similar methodology by combining sum-check with STIR [ACFY24a].
- **BulletProofs** [BBB$^+$18] presents an inner-product argument (IPA) based on the discrete logarithm assumption. This work has inspired an extensive line of research in elliptic curve cryptography (ECC)-based SNARKs, including notable constructions such as Dory [Lee20], Halo [BDFG20], and Nova [KST21]. While its connection to sum-check was not immediately apparent, the relationship was later clarified by Bootle et al. [BCS21], who revealed that BulletProofs operates as a sum-check protocol in disguise. As demonstrated later in this work, BulletProofs can be understood as the parallel execution of multiple sum-check protocols, including sum-check for group-valued polynomials.

A crucial technical element underlying both examples is parallel protocol composition. Despite its importance, parallel composition lacks a systematic framework for analyzing its security in the existing literature. Current approaches treat composed protocols as monolithic constructions, failing to delegate their security guarantees to their individual components. This challenge is exemplified by BaseFold [ZCF23], which establishes its security independently rather than building upon the well-developed security analysis of the FRI protocol [BCI$^+$20]. While subsequent work by Haböck [Hab24] improves its security level, this work still necessitates new analysis, relying on a different variant (the subcode version) of the weighted agreement theorem used in FRI.

A fundamental obstacle to achieving modular protocol analysis lies in the incapability of traditional proof systems to capture security of the component protocols in isolation. One such component is the sum-check protocol. Despite its wide-spread use, sum-check has never seen a purely context-independent security definition. Current analyses typically demands polynomial oracles for capturing the verifier's ability to evaluate polynomials. Although this strategy is adequate for analyzing sequential protocol compositions, it fails to characterize the polynomial evaluation ability when the evaluation protocol is interleaved with sum-check itself, as in BaseFold. Another example is FRI. Although its security is well-defined as a low-degree test, FRI is additionally expected to verifiably evaluate the committed polynomial at the random point generated during the protocol execution, as required by BaseFold. This perspective of FRI has never been analyzed independently, as proof systems for such dynamically generated statements lack a formal treatment in the literature.

1.1 Our Contributions

To address this challenge, we introduce *functional proof systems (FPS)*, which extend traditional proof systems by viewing the verifier as a function parameterized by the prover. This new perspective of proof systems enables us to capture sum-check security by a self-contained and information-theoretic formation, provide modular analyses of existing protocols like BaseFold and BulletProofs, and finally construct a new efficient polynomial commitment scheme. In more detail:

- We introduce FPS, a generalization of interactive proof systems where the verifier is viewed as a function parameterized by the prover. For more expressiveness, we allow the FPS verifier to output values from a general set rather than being restricted to Boolean values as in traditional proof systems. We define the system's security in terms of the function's properties, rather than the traditional assertions about a single Boolean value. This new perspective on proof systems enables us to establish a generic composition theorem for modular security analysis of parallel protocol execution.
- We present the first clean, self-contained security definition for the sum-check protocol using FPS. Additionally, we offer novel perspectives on FRI by recasting it as FPS instance for evaluating committed multilinear polynomials at dynamically generated challenge points (denoted FriEval), and extracting another FPS from BulletProofs with the same functionality (denoted BulletEval). We prove their security under two distinct FPS variants: FPS with proximity (FPSP) and functional argument of knowledge (FAoK).
- To demonstrate using our framework for security analysis, we reconstruct BaseFold and BulletProofs by composing sum-check with FriEval or BulletEval, respectively, facilitating modular security analyses of both protocols. Our analysis yields a security proof for BaseFold that directly leverages FRI's established security guarantees, avoiding the weighted agreement theorem for subcodes required in [Hab24].
- To demonstrate using our framework for protocol design, we construct Bird, a new transparent, pairing-free, and doubly efficient polynomial commitment scheme by composing a group-based variant of FriEval with BulletEval and sum-check. The scheme achieves linear prover complexity and polylogarithmic verifier complexity and proof size. Table 1 shows Bird achieves the same state-of-the-art asymptotic complexity as the concurrent work of Eagen et al. [EG25], while Bird has better concrete performance. Compared to hash-based PCS, which are considered to have the best prover performance, Bird supports homomorphic linear combinations of arbitrarily many commitments, facilitating efficient batching techniques that significantly reduce proving, verification, and communication costs of SNARKs.

1.2 Technical Overview

For readers to better understand the main techniques developed in this paper, we provide a high-level and intuitive overview of our functional proof systems framework and its applications.

Formalizing sum-check soundness. Our investigation begins with the sum-check protocol, a foundational component of interactive proof systems. This protocol verifies claims of the form $\sum_{\mathbf{x}\in\{0,1\}^\mu} f(\mathbf{x}) = c$ for a given multilinear polynomial f. Despite its widespread adoption, its security has never been formalized in a self-contained manner, for reasons that will be explained below.

The protocol operates by reducing the original claim into a claim about a single point evaluation, i.e., $f(\mathbf{r}) = y$ for a randomly chosen $\mathbf{r}$ generated during the protocol. The security of sum-check thus relies on the verifier's ability to evaluate f at arbitrary points such that f cannot be tampered by the prover. This ability is provided to the verifier through a variety of strategies depending on the context in which sum-check is deployed. Formalizing sum-check security thus requires a rigorous definition of this evaluation ability.

This ability has traditionally been formalized as an evaluation oracle. Proving security in this model requires identifying a clearly delineated part of the protocol, and establishing that it securely instantiates the oracle. Although this approach is sufficient in analyzing sequential compositions of sum-check and other protocols, it is not well-suited for analyzing parallel compositions, in which the instantiation of the oracle is intertwined with sum-check itself.

To extract the core functionality of sum-check, we restrict our attention to the interaction procedure of this protocol, excluding the final evaluation check. We observe that when the prover is fixed to be any deterministic algorithm P, the value y computed by the honest verifier is completely decided by the claimed sum c and the random point $\mathbf{r}$, so we can formalize this dependence as a function $y = V_{\mathsf{P}}(c, \mathbf{r})$. Shifting the focus from the concrete value of y to the function V_{P} reveals a simple, concise, self-contained, and mathematically rigorous characterization of the security of sum-check.

For convenience, we introduce some notion for describing properties of functions. For two functions f, g with the same domain D, we say f is *far from* g if $|\{x \in D : f(x) = g(x)\}| < \varepsilon \cdot |D|$ for some negligible ε. Sum-check protocol then enjoys the following straightforward security guarantee:

- **Completeness**: For the honest prover interacting with the verifier with any polynomial f that has the correct sum c, the function $V_{\mathsf{P}}(c, \cdot)$ is equal to f.
- **Soundness**: For any polynomial f that has a wrong sum, and any prover P, the function $V_{\mathsf{P}}(c, \cdot)$ is far from f.

Critically, this definition eliminates the need for capturing the verifier's ability to evaluate f. Instead, it establishes a guarantee regarding the function V_{P} in response to any prover strategy P, and this guarantee is independent of any particular polynomial.

Functional proof systems. To formalize the aforementioned insight, we introduce *functional proof systems (FPS)*. An FPS generalizes traditional interactive proof systems by allowing the verifier to output a value from an extended domain rather than a binary decision (accept/reject). This value y depends only on the instance x, witness w, and verifier's randomness r for any fixed prover strategy P, and we capture this dependence by a function $y = V_{\mathsf{P}}(x, w, r)$. The

completeness property of an FPS requires that for the honest verifier and prover, this function V_{P} matches a target function TarF. Soundness is characterized by a predicate function, PredF, which describes the expected relationship between $x, w, r, V_{\mathsf{P}}(x, w, r)$ that holds with overwhelming probability for any x, w and uniformly distributed r, for any (potentially malicious) provers.

This new notion naturally accommodates the sum-check protocol. In this specific instantiation, the target function TarF is defined as $(c, f, \mathbf{r}) \mapsto f(\mathbf{r})$. The predicate function $\mathsf{PredF}(c, f, y, \mathbf{r})$ then encapsulates the core security property: either the claimed sum is correct, i.e., $\sum f(\mathbf{x}) = c$, or the verifier's output differs from the expected evaluation, i.e., $y \neq f(\mathbf{r})$.

A framework for composition. Building upon the FPS framework, we develop a composition mechanism that enables the construction of more complex proof systems from simpler components by parallel execution: the composed prover executes multiple individual provers simultaneously in each round, concatenating their messages before transmission to the verifier. Similarly, the composed verifier runs multiple individual verifiers in parallel, distributing the appropriate message components to each. Intuitively, for public-coin protocols, if the individual verifiers share the same random tape, then the composed protocol is secure if the individual protocols are secure. We prove this intuition in Theorem 1.

Revisiting BaseFold and BulletProofs. To demonstrate the practical utility of our composition framework, we provide more modular and simpler security proofs for two prominent protocols—BaseFold and BulletProofs. Both protocols can be decomposed into two components: a sum-check, and an FPS belonging to a particularly interesting category that we term *multilinear polynomial evaluation FPS*. Such systems ensure that the verifier outputs either the correct evaluation $f(\mathbf{r})$ or a failure symbol $\perp$, given a commitment to a multilinear polynomial f.

Both BaseFold and BulletProofs operate in the following paradigm: they execute in parallel a multilinear polynomial evaluation FPS, whose input is a committed polynomial $f(\mathbf{X})$, and a sum-check for statement $t = \sum f(\mathbf{x}) \cdot g(\mathbf{x})$, where $g(\mathbf{X})$ is a public polynomial. After the interaction, the verifiers of the two components respectively output y_{eval} and y_{sum}. By their respective predicates, $y_{eval} = f(\mathbf{r})$ (or $\perp$, which the verifier directly rejects) is guaranteed, while $y_{sum} = f(\mathbf{r}) \cdot g(\mathbf{r})$ only if $t = \sum f(\mathbf{x}) \cdot g(\mathbf{x})$. The verifier then computes $g(\mathbf{r})$ locally, and compares y_{sum} against $y_{eval} \cdot g(\mathbf{r})$ to decide whether to accept t or not.

BaseFold and BulletProofs differ in both the instantiation of the multilinear polynomial evaluation FPS and the choice of the polynomial $g(\mathbf{X})$. For BaseFold, the former comes from a new perspective on FRI, which we formalize as the $\mathsf{FriEval}$ FPS (relaxed to its proximity version to accommodate the proximity tolerance δ inherent to FRI-based systems), and the latter $g(\mathbf{X})$ is $\mathsf{eq}(\mathbf{z}, \mathbf{X})$, the Lagrange basis polynomial over the Boolean hypercube $\{0, 1\}^{\mu}$. For BulletProofs, the former is instantiated with a group-valued sum-check, as previously noted by Bootle et al. [BCS21], which we formalize as the $\mathsf{BulletEval}$ FPS (relaxed to its argument of knowledge version to accommodate the discrete logarithm hardness assumption), and the latter $g(\mathbf{X})$ can be any multilinear polynomial.

Constructing a new PCS. As a concrete application of our framework, we construct a new polynomial commitment scheme (PCS) by leveraging the modular components we have developed: FriEval, BulletEval, and sum-check. Our construction uses BulletProofs as the starting point, which offers several attractive properties for polynomial commitment schemes. Specifically, BulletProofs achieves transparency, linear prover complexity (measured in group scalar multiplications), homomorphic additions of commitments, and logarithmic proof size that is concretely smaller than hash-based constructions.

Despite these strengths, BulletProofs suffers from a critical limitation: linear verifier complexity, which makes it impractical for deployment in SNARKs. This bottleneck arises solely from the requirement for the verifier to evaluate $G(\mathbf{X})$, a dense group-valued polynomial, at the random point $\mathbf{r}$. Our solution to this bottleneck exploits the modular decomposition of BulletProofs as a parallel composition of BulletEval and sum-check. To eliminate the linear verifier cost, we introduce a third component to this composition: a group-valued variant of the FriEval protocol, which enables the verifier to output $G(\mathbf{r})$ without computing it directly, thereby reducing the linear cost to polylogarithmic.

1.3 Related Works

Sum-check. The sum-check protocol, introduced by Lund et al. [LFKN90], is an interactive proof system for verifying the sum of a multivariate polynomial over a Boolean hypercube. We have mentioned most related works at the beginning of this paper, so we omit them here. As a particularly relevant work to ours, the Sum-check Argument of Bootle et al. [BCS21] reveals the connection between BulletProofs and sum-check. Unlike [BCS21], which treats BulletProofs as a single sum-check for a three-dimensional polynomial (with coefficients from $\mathbb{G} \times \mathbb{G} \times \mathbb{F}$), we formalize BulletProofs as the parallel composition of BulletEval and a sum-check, simplifying the analysis and allowing more flexible compositions, e.g., BulletEval with other protocols than sum-check.

SNARKs. Succinct non-interactive arguments of knowledge (SNARKs) [BCCT12, PHGR13] prove correct execution of a program through short and efficiently verifiable proofs. This technology has experienced rapid development in recent years, driven by its generality and high performance [Gro16,MBKM19,CHM+20, GWC19,BCR+19,BFS20,CBBZ22,Set20,LZZ+25,STW23,KS23,XZZ+19,ZXZS20, ZCF23, ACFY24b, ZSZ+22]. SNARKs has widespread deployment in practical applications, particularly in blockchains for scalability and privacy [Scr22,Loo22, Azt22, zkS22], with potential for broader adoption in cloud computing [CTV15, WB15] and verifiable machine learning [QSL+25, SLZ24, LXZ21, W24].

The increasing complexity of SNARK constructions has motivated the development of modular design principles to improve clarity and reusability. A prominent example is the PIOP (Polynomial Interactive Oracle Proof) pipeline [BFS20], which was concurrently developed under different names including AHP (Algebraic Holographic Proof) [CHM+20] and polynomial protocols [GWC19], and has become the predominant methodology for SNARK construction. Further

modularization efforts include the work by Ràfols et al. [RZ21], which formalizes Checkable Subspace Sampling Arguments for pairing-based SNARKs, and Zhang et al. [ZSZ+22], who introduce the vector oracle model to further decompose PIOP constructions.

Polynomial commitment schemes. As a core component of the PIOP pipeline, PCS [KZG10] has attracted extensive research attention, resulting in numerous proposed constructions [BFS20, Lee20, XZS22, BBHR18, ZCF23, ZXZS20, ACFY24a, ACFY24b, BGKS19, GLS+21, GPS25]. Recently, the evolution of PCS research has shifted focus from early univariate polynomials [KZG10, BBHR18] toward multilinear polynomials, motivated by the pursuit of SNARKs with linear-time prover complexity, leading to dedicated multilinear PCS constructions [ACFY24a, ZCF23, GPS25, ZXZS20, KT23] as well as transformation techniques that adapt univariate schemes for multilinear use [PH23, BCHO22].

Concurrent with our work, Eagen et al. [EG25] proposed a PCS that combines BulletProofs with a group-valued variant of BaseFold, achieving the same asymptotic efficiency as our construction. However, our approach offers several advantages: we employ parallel rather than sequential composition, reducing the round complexity by half, and we directly compose BulletProofs with FRI rather than with BaseFold, avoiding the additional sum-check from BaseFold.

2 Preliminaries

2.1 Notations

Throughout this paper, λ denotes the security parameter. We work with a finite field $\mathbb{F}$ and a group $\mathbb{G}$, both of prime order $p \approx 2^\lambda$. For integers $m \leq n$, we define $[m..n] \triangleq \{m, \ldots, n\}$ and $[n] \triangleq \{0, \ldots, n-1\}$. We use capital letters (e.g., G) for group elements and lowercase letters (e.g., a, α) for field elements. Vectors are denoted in bold font, with $\mathbf{G} \in \mathbb{G}^n$ for group-valued vectors and $\mathbf{v} \in \mathbb{F}^n$ for field-valued vectors. For a vector $\mathbf{v}$, we denote its i-th element as $\mathbf{v}_{[i]}$ (indices start from zero) and the subvector from index m to n as $\mathbf{v}_{[m..n]}$. For even-sized vectors, we define $\mathbf{v}_{[\mathsf{left}]} = \mathbf{v}_{[0..n/2-1]}$, $\mathbf{v}_{[\mathsf{right}]} = \mathbf{v}_{[n/2..n-1]}$, $\mathbf{v}_{[\mathsf{even}]} = (\mathbf{v}_{[0]}, \mathbf{v}_{[2]}, \ldots, \mathbf{v}_{[n-2]})$, and $\mathbf{v}_{[\mathsf{odd}]} = (\mathbf{v}_{[1]}, \mathbf{v}_{[3]}, \ldots, \mathbf{v}_{[n-1]})$.

We use additive notation for group operations (e.g., $G + H$). Scalar multiplication of $\alpha \in \mathbb{F}$ and $G \in \mathbb{G}$ is denoted $\alpha \cdot G$. For $\mathbf{v} \in \mathbb{F}^n$ and $G \in \mathbb{G}$, the notation $\mathbf{v} \cdot G$ represents the vector $(\mathbf{v}_{[0]} \cdot G, \ldots, \mathbf{v}_{[n-1]} \cdot G)$. The inner product of $\mathbf{v} \in \mathbb{F}^n$ and $\mathbf{G} \in \mathbb{G}^n$ is $\langle \mathbf{v}, \mathbf{G} \rangle \triangleq \sum_{i=0}^{n-1} \mathbf{v}_{[i]} \cdot \mathbf{G}_{[i]} \in \mathbb{G}$. For an ordered subset $L = \{x_0, \ldots, x_{n-1}\} \subset \mathbb{F}$ and function $f(X)$, we denote $f(L) = (f(x_0), \ldots, f(x_{n-1}))$.

2.2 Multilinear Extension

Any function $f : \{0,1\}^\mu \to \mathbb{F}$ defined on the Boolean hypercube can be uniquely extended to a multilinear polynomial $\hat{f}(X_0, \ldots, X_{\mu-1}) \in \mathbb{F}[X_0, \ldots, X_{\mu-1}]$ that

agrees with f on $\{0,1\}^\mu$. This extension, called the multilinear extension (MLE), admits the explicit formula:

$$\hat{f}(\mathbf{X}) \triangleq \sum_{\mathbf{b}\in\{0,1\}^\mu} f(\mathbf{b}) \cdot \mathsf{eq}(\mathbf{X}, \mathbf{b})$$

where $\mathbf{X} = (X_0, \ldots, X_{\mu-1})$ and $\mathsf{eq}(\mathbf{X}, \mathbf{Y})$ denotes the Lagrange basis polynomial over the hypercube:

$$\mathsf{eq}(\mathbf{X}, \mathbf{Y}) \triangleq \prod_{i=0}^{\mu-1} (X_i Y_i + (1 - X_i)(1 - Y_i)).$$

This basis polynomial has the property that $\mathsf{eq}(\mathbf{X}, \mathbf{b})$ evaluates to 1 when $\mathbf{X} = \mathbf{b}$ and to 0 for all other points in the hypercube.

2.3 Proof System

Interactive proof systems provide the foundational framework for cryptographic protocols where a prover convinces a verifier of the validity of some statement through a sequence of message exchanges.

Definition 1 (Interactive Proof System). *An* interactive proof system *for a language $\mathcal{L}$ is a pair of algorithms (P, V), where P is a computationally unbounded deterministic prover and V is a probabilistic polynomial-time (PPT) verifier. For a common input x, the prover, holding a private witness w for certifying $x \in \mathcal{L}$, engages in an interactive protocol with the verifier. The verifier then outputs a decision bit $b \in \{0,1\}$. We denote the interaction by $b \leftarrow \langle \mathsf{P}(x, w), \mathsf{V}(x)\rangle$.*

The system must satisfy two properties:

- ***Completeness**: For any $x \in \mathcal{L}$ and a valid witness w:*

$$\Pr[b = 1 \mid b \leftarrow \langle \mathsf{P}(x, w), \mathsf{V}(x)\rangle] \geq 1 - \varepsilon_{\mathsf{cmplt}}$$

  where $\varepsilon_{\mathsf{cmplt}}$ is the completeness error*.*
- ***Soundness**: For any $x \notin \mathcal{L}$ and any (possibly unbounded) prover P^*:*

$$\Pr[b = 1 \mid b \leftarrow \langle \mathsf{P}^*(x), \mathsf{V}(x)\rangle] \leq \varepsilon_{\mathsf{snd}}$$

  where $\varepsilon_{\mathsf{snd}}$ is the soundness error*.*

In both properties, the probability is over the randomness used by the verifier. If the completeness error is zero, we say the system has perfect completeness*. Similarly, if the soundness error is zero, it has* perfect soundness*.*

An *interactive argument* relaxes the soundness requirement to hold only against computationally bounded provers. While proof systems provide stronger theoretical guarantees, practical implementations typically employ argument systems due to their reliance on cryptographic primitives that inherently assume computational limitations.

Our work primarily employs *arguments of knowledge*, which strengthen standard arguments by requiring not only that a valid witness exists, but that the prover actually possesses such knowledge. This distinction becomes crucial in cryptographic contexts where the mere existence of a witness may be trivial. In this work, we adopt the definition via individual extractability, where each prover admits a corresponding extractor that can recover the witness by exploiting the prover's internal state. We restrict attention to deterministic provers without loss of generality, since any probabilistic prover can be derandomized by fixing the random coins that maximize the success probability.

Definition 2 (Argument of Knowledge). *An interactive argument* (P, V) *for a relation* $\mathcal{R}$ *is an* argument of knowledge *with soundness error* $\varepsilon_{\mathsf{snd}}$ *if for any polynomial-time deterministic prover* P^**, there exists an expected polynomial-time extractor* Ext *such that for any instance* x *and auxiliary input* z*:*

$$\Pr\left[b = 1 \wedge (x, w) \notin \mathcal{R} \,\middle|\, \begin{array}{l} b \leftarrow \langle \mathsf{P}^*(x, z), \mathsf{V}(x) \rangle \\ w \leftarrow \mathsf{Ext}(x, z) \end{array}\right] \leq \varepsilon_{\mathsf{snd}},$$

where the probability is taken over the verifier's randomness.

Public-coin protocols. A protocol is *public-coin* if all verifier messages consist of uniformly random challenges drawn from tossing a coin publicly. This property simplifies security analysis and enables non-interactive variants through the Fiat-Shamir transformation. All protocols considered in this work are public-coin.

2.4 Sum-check

The sum-check protocol [LFKN90] enables a prover to convince a verifier that the sum of a multivariate polynomial $f \in \mathbb{F}[X_0, \ldots, X_{\mu-1}]$ over the boolean hypercube $\{0,1\}^\mu$ equals a claimed value $c \in \mathbb{F}$. A complete description of the protocol is given in Protocol 1. We assume f has degree at most m in each variable. The protocol setup presupposes that the prover knows the polynomial f in its entirety, while the verifier can evaluate f at any point.

The verifier accepts if all checks pass and rejects otherwise. The protocol achieves soundness through the Schwartz-Zippel lemma: if the claimed sum is incorrect, the verifier will reject with high probability. Specifically, any cheating prover succeeds with probability at most $\mu \cdot m/|\mathbb{F}|$.

2.5 Polynomial Commitment Scheme

Polynomial commitment schemes (PCS) [KZG10] provide a cryptographic primitive that enables a prover to commit to a polynomial f of bounded degree and later prove evaluations of that polynomial at specific points. This primitive plays a central role in modern SNARK constructions, particularly within the PIOP framework. We present the definition for the multilinear case adapted from BaseFold [ZCF23].

Protocol 1 (Sum-check) The prover receives input $c \in \mathbb{F}, f \in \mathbb{F}[X_0, \ldots, X_{\mu-1}]$ and the verifier receives input $c \in \mathbb{F}$ and evaluation access to f. The protocol proceeds in μ rounds as follows:

- **Round 1**: The prover sends to the verifier a univariate polynomial

$$h_1(X) = \sum_{\mathbf{b} \in \{0,1\}^{\mu-1}} f(X, \mathbf{b}).$$

  The verifier checks that $\deg(h_1) \leq m$ and that $h_1(0) + h_1(1) = c$. The verifier then samples a random challenge $r_1 \in \mathbb{F}$ and sends it to the prover.
- **Round** i **(**$2 \leq i \leq \mu$**)**: The prover sends a univariate polynomial

$$h_i(X) = \sum_{\mathbf{b} \in \{0,1\}^{\mu-i}} f(r_1, \ldots, r_{i-1}, X, \mathbf{b}).$$

  The verifier checks that $\deg(h_i) \leq m$ and that $h_i(0) + h_i(1) = h_{i-1}(r_{i-1})$. The verifier then samples challenge $r_i \in \mathbb{F}$ and sends it to the prover.
- **Final check**: After μ rounds, the verifier has challenges $r_1, \ldots, r_\mu$. The verifier computes $f(r_1, \ldots, r_\mu)$ and checks if it equals $h_\mu(r_\mu)$.

Definition 3 (Polynomial Commitment Scheme). *A polynomial commitment scheme* PC *consists of algorithms* $(\mathsf{Setup}, \mathsf{Derive}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *with the following functionality:*

- $\mathsf{Setup}(1^\lambda, D) \to \mathsf{pp}$ *generates public parameters given security parameter* λ *and polynomial size bound* $D = 2^\mu$ *for* μ*-variate multilinear polynomials.*
- $\mathsf{Derive}(\mathsf{pp}) \to \mathsf{vp}$ *derives verification parameters* vp *from the public parameters, potentially reducing the verifier's storage requirements.*
- $\mathsf{Commit}(\mathsf{pp}, f) \to \mathsf{cm}$ *produces a commitment* cm *to multilinear polynomial* f *with at most* μ *variables.*
- $\mathsf{Open}(\mathsf{pp}, \mathsf{cm}, f) \to b$ *verifies that polynomial* f *is a valid opening of commitment* cm*, outputting a decision bit* b*.*
- $\mathsf{Eval}(\mathsf{vp}, \mathsf{cm}, \mathbf{z}, y; f, \mathsf{pp}) \to b$ *implements an interactive protocol between prover* P *and verifier* V*. Both parties receive commitment* cm*, evaluation point* $\mathbf{z}$*, and claimed value* $y \in \mathbb{F}$*. The prover additionally possesses polynomial* f *and public parameters* pp*. The protocol enables* P *to convince* V *that* f *opens to* cm *and satisfies* $f(\mathbf{z}) = y$*.*

The scheme must satisfy three security properties:

Completeness: *For any polynomial* f *within the size bound and any evaluation point* $\mathbf{z}$*, honest execution succeeds:*

$$\Pr\left[\mathsf{Eval}(\mathsf{vp}, \mathsf{cm}, \mathbf{z}, f(\mathbf{z}); f, \mathsf{pp}) = 1 \;\middle|\; \begin{array}{l} \mathsf{Setup}(1^\lambda, D) \to \mathsf{pp} \\ \mathsf{Derive}(\mathsf{pp}) \to \mathsf{vp} \\ \mathsf{Commit}(\mathsf{pp}, f) \to \mathsf{cm} \end{array}\right] = 1.$$

Binding: *No efficient adversary can produce two distinct polynomials that both open to the same commitment:*

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

Knowledge soundness: *The evaluation protocol* Eval *constitutes an argument of knowledge for the relation*

$$\mathcal{R}_{\mathsf{Eval}} = \{((\mathsf{pp}, \mathsf{cm}, \mathbf{z}, y), f) \mid f(\mathbf{z}) = y \wedge \mathsf{Open}(\mathsf{pp}, \mathsf{cm}, f) = 1\},$$

ensuring that successful provers must possess a polynomial that both opens to the given commitment and evaluates to the claimed value.

2.6 Inner Product Argument

Inner product arguments (IPAs) enable a prover to demonstrate knowledge of a committed vector satisfying a specified inner product relation. These protocols, e.g., BulletProofs [BBB+18], can be easily adapted to construct PCS.

Definition 4 (Inner Product Argument). *An inner product argument scheme* IPA *consists of algorithms* (Setup, Commit, Eval) *with the following functionality:*

- $\mathsf{Setup}(1^\lambda, D) \to \mathsf{pp}$ *generates public parameters given security parameter* λ *and vector dimension* $D \in \mathbb{N}$.
- $\mathsf{Commit}(\mathsf{pp}, \mathbf{v}) \to \mathsf{cm}$ *produces a commitment to vector* $\mathbf{v} \in \mathbb{F}^d$ *using the public parameters, where* $d \leq D$.
- $\mathsf{Eval}(\mathsf{pp}, \mathsf{cm}, \mathbf{u}, t; \mathbf{v}) \to b$ *implements an interactive protocol between prover* P *and verifier* V. *Both parties receive public parameters* pp, *commitment* cm, *public vector* $\mathbf{u} \in \mathbb{F}^d$, *and claimed inner product value* $t \in \mathbb{F}$. *The prover additionally possesses the witness vector* $\mathbf{v}$. *The protocol enables* P *to convince* V *that the committed vector* $\mathbf{v}$ *satisfies* $\langle \mathbf{v}, \mathbf{u} \rangle = t$.

The scheme must satisfy two security properties:

Completeness: *For any* $d \leq D$, *vectors* $\mathbf{v}, \mathbf{u} \in \mathbb{F}^d$, *with* $t = \langle \mathbf{v}, \mathbf{u} \rangle$, *honest execution succeeds:*

$$\Pr\left[\mathsf{Eval}(\mathsf{pp}, \mathsf{cm}, \mathbf{u}, t; \mathbf{v}) = 1 \;\middle|\; \begin{array}{l} \mathsf{Setup}(1^\lambda, D) \to \mathsf{pp} \\ \mathsf{Commit}(\mathsf{pp}, \mathbf{v}) \to \mathsf{cm} \end{array}\right] = 1.$$

Knowledge Soundness: *The evaluation protocol* Eval *constitutes an argument of knowledge for the relation*

$$\mathcal{R}_{\mathsf{Eval}} = \{((\mathsf{pp}, \mathsf{cm}, \mathbf{u}, t), \mathbf{v}) \mid \langle \mathbf{v}, \mathbf{u} \rangle = t \wedge \mathsf{Commit}(\mathsf{pp}, \mathbf{v}) = \mathsf{cm}\},$$

ensuring that successful provers must possess a vector that both opens to the given commitment and yields the claimed inner product value.

2.7 Reed-Solomon Code and FRI

For integer n, subset $L \subset \mathbb{F}$ of size n, and rate parameter $\rho \in (0,1)$, the Reed-Solomon code $\mathsf{RS}[\mathbb{F}, L, \rho]$ is the linear subspace of $\mathbb{F}^n$ defined by $\mathsf{RS}[\mathbb{F}, L, \rho] = \{f(L) | f \in \mathbb{F}[x], \deg(f) < \rho \cdot n\}$. Each element in $\mathsf{RS}[\mathbb{F}, L, \rho]$ is called a Reed-Solomon codeword, where n denotes the codeword length and $d = \rho \cdot n$ represents the code dimension. For a Reed-Solomon code C of length n and dimension d, we define the encoding function: for any vector $\mathbf{v}$ of size d, $\mathsf{Encode}_C(\mathbf{v}) = f_{\mathbf{v}}(L)$ where $f_{\mathbf{v}}$ is the polynomial with coefficients given by $\mathbf{v}$.

The FRI protocol [BBHR18] relies on a specific structure called a foldable family of Reed-Solomon codes. We assume both n and $1/\rho$ (and thus $d = \rho \cdot n$) are powers of two, with $d = 2^\mu$. A foldable family is a sequence of codes $C^{(0)}, \cdots, C^{(\mu)}$ where each $C^{(i)} = \mathsf{RS}[\mathbb{F}, L^{(i)}, \rho]$ is defined over a domain $L^{(i)}$ that is half the size of the previous domain $L^{(i-1)}$. Specifically, $|L^{(i)}| = |L^{(i-1)}|/2$ and $L^{(i)} = (L^{(i-1)})^2 = \{x^2 | x \in L^{(i-1)}\}$. This squaring operation creates a natural hierarchy of evaluation domains. Typical choices for the initial domain $L^{(0)}$ are multiplicative subgroups of $\mathbb{F}^*$ with order $2^{\mu+B}$, where the final domain satisfies $|L^{(\mu)}| = 2^B$. The final code $C^{(\mu)}$ in this sequence is called the base code, which by construction consists of all 2^B-dimensional vectors with identical entries.

The Reed-Solomon code family $C^{(0)}, \cdots, C^{(\mu)}$ with codeword sizes $n^{(0)}, \cdots, n^{(\mu)}$ and dimensions $d^{(0)}, \cdots, d^{(\mu)}$ has the following property.

Claim. There exist a sequence of vector pairs $(\mathbf{t}_+^{(i)}, \mathbf{t}_-^{(i)})$, which we call the *twist vectors*, for i from 1 to μ, where $|\mathbf{t}_+^{(i)}| = |\mathbf{t}_-^{(i)}| = n^{(i)}$, such that for i from 1 to μ, for any $\mathbf{c} \in \mathbb{F}^{n^{(i-1)}}$, $\mathbf{c} \in C^{(i-1)}$ if and only if $(\mathbf{c}_{[\mathsf{left}]} + \mathbf{c}_{[\mathsf{right}]}) \circ \mathbf{t}_+^{(i)} \in C^{(i)}$ and $(\mathbf{c}_{[\mathsf{left}]} - \mathbf{c}_{[\mathsf{right}]}) \circ \mathbf{t}_-^{(i)} \in C^{(i)}$, where "$\circ$" stands for entry-wise product between vectors. Moreover, the encoded messages of these codewords are related as follows. Suppose $\mathbf{c} = \mathsf{Encode}_{C^{(i-1)}}(\mathbf{v})$, then

$$(\mathbf{c}_{[\mathsf{left}]} + \mathbf{c}_{[\mathsf{right}]}) \circ \mathbf{t}_+^{(i)} = \mathsf{Encode}_{C^{(i)}}(\mathbf{v}_{[\mathsf{even}]})$$

$$(\mathbf{c}_{[\mathsf{left}]} - \mathbf{c}_{[\mathsf{right}]}) \circ \mathbf{t}_-^{(i)} = \mathsf{Encode}_{C^{(i)}}(\mathbf{v}_{[\mathsf{odd}]}).$$

Moreover, each entry of the twist vectors can be computed efficiently: given any i and j, there is an algorithm $\mathsf{Twist}(i,j) = (\mathsf{Twist}_+(i,j), \mathsf{Twist}_-(i,j))$ that computes $\mathbf{t}_{+,[j]}^{(i)}$ and $\mathbf{t}_{-,[j]}^{(i)}$ with number of field operations logarithmic in $n^{(i)}$.

The key operation in FRI is "folding", which reduces the problem size by half in each round. We define folding for both messages and codewords:

- **Message folding:** For any vector $\mathbf{v} \in \mathbb{F}^{d^{(i-1)}}$ and folding coefficient $r \in \mathbb{F}$:

$$\mathsf{Fold}(\mathbf{v}, r) \triangleq (1-r) \cdot \mathbf{v}_{[\mathsf{even}]} + r \cdot \mathbf{v}_{[\mathsf{odd}]},$$

  which produces a vector of size $d^{(i)} = d^{(i-1)}/2$.

– **Codeword folding:** For a vector $\mathbf{c} \in \mathbb{F}^{n^{(i-1)}}$ (which has codeword size but is not necessarily a codeword of $C^{(i-1)}$) and folding coefficient $r \in \mathbb{F}$:

$$\mathsf{Fold}(\mathbf{c}, r) \triangleq (1-r) \cdot (\mathbf{c}_{[\mathsf{left}]} + \mathbf{c}_{[\mathsf{right}]}) \circ \mathbf{t}_{+}^{(i)} + r \cdot (\mathbf{c}_{[\mathsf{left}]} - \mathbf{c}_{[\mathsf{right}]}) \circ \mathbf{t}_{-}^{(i)},$$

resulting in a vector of size $n^{(i)} = n^{(i-1)}/2$.

A crucial property of foldable Reed-Solomon codes is that folding and encoding commute. For any message $\mathbf{v} \in \mathbb{F}^{d^{(i-1)}}$ and folding coefficient r:

$$\mathsf{Fold}(\mathsf{Encode}_{C^{(i-1)}}(\mathbf{v}), r) = \mathsf{Encode}_{C^{(i)}}(\mathsf{Fold}(\mathbf{v}, r)).$$

This commutativity enables the FRI protocol to maintain the message-codeword correspondence throughout the folding process.

Another notable property is that folding a message vector $\mathbf{v}$ by r is equivalent to substituting the first variable of $\hat{\mathbf{v}}(\mathbf{X})$ by r, so repeatedly folding a message vector $\mathbf{v} \in \mathbb{F}^{2^{\mu}}$ respectively by $r_1, \ldots, r_\mu$ produces the scalar $\hat{\mathbf{v}}(r_1, \ldots, r_\mu)$. This equivalence enables viewing FRI as a multilinear polynomial evaluating protocol.

The Fast Reed-Solomon IOP of Proximity (FRI) protocol allows a prover to convince a verifier that a given oracle string is close to some Reed-Solomon codeword. This proximity testing capability can be exploited to construct PCS. In the FRI setting, the prover possesses a message vector $\mathbf{v} \in \mathbb{F}^d$, while the verifier has oracle access to a string $\mathbf{c} \in \mathbb{F}^n$. The protocol's objective is to verify that $\mathbf{c}$ is δ-close to the Reed-Solomon code C, meaning:

$$\Delta(\mathbf{c}, C) \triangleq \min_{\mathbf{c}' \in C} \{\mathsf{HW}(\mathbf{c} - \mathbf{c}')\} < \delta \cdot n,$$

where $\delta \in (0, 1)$ is a proximity parameter and HW denotes Hamming weight. We initialize the protocol with $C^{(0)} = C$, $\mathbf{v}^{(0)} = \mathbf{v}$, and $\mathbf{c}^{(0)} = \mathbf{c}$. The complete FRI protocol is described in Protocol 2.

In practical implementations, the final round of the commit phase can be optimized: since all elements of $\mathbf{c}^{(\mu)}$ are identical by the base code property, the prover need only send this single unique element rather than the entire vector. Additionally, the original FRI protocol allows early termination of the commit phase after fewer than μ rounds, with the prover directly revealing the final codeword or message. For clarity, we omit these optimizations in our description.

3 Functional Proof Systems

We introduce functional proof systems (FPS) generalizing traditional interactive proof systems by allowing the verifier to output values from arbitrary domains rather than binary values, with the guarantee that this output maintains a specified functional relationship with the instance and witness for arbitrary provers.

Definition 5 (Functional Proof System). *Let* $\mathsf{S}_{\mathsf{ins}}$, $\mathsf{S}_{\mathsf{out}}$, $\mathsf{S}_{\mathsf{wit}}$ *be the instance, output, and witness sets, respectively. Let* $\mathsf{S}_{\mathsf{rand}}$ *be the space of the verifier's*

Protocol 2 (FRI) The prover receives $\mathbf{v} \in \mathbb{F}^d$ and the verifier receives oracle access to $\mathbf{c} \in \mathbb{F}^n$. The protocol proceeds in the following two phases:
Commit phase: The parties interact in μ rounds, where in each round i,

- The verifier sends uniformly sampled $r_i \in \mathbb{F}$ to the prover.
- If $i < \mu$, the prover computes $\mathbf{c}^{(i)} = \mathsf{Fold}(\mathbf{c}^{(i-1)}, r_i)$ and sends the oracle of $\mathbf{c}^{(i)}$ to the verifier. Otherwise (i.e., $i = \mu$), the prover sends the vector $\mathbf{c}^{(\mu)}$ directly to the verifier.
- The verifier checks that $\mathbf{c}^{(\mu)}$ is a valid codeword of $C^{(\mu)}$, which is equivalent to checking that all elements of $\mathbf{c}^{(\mu)}$ are identical.

Query phase: The verifier repeats the following process ℓ times:

- The verifier samples $j^{(0)} \in [n]$ uniformly at random.
- For i from 1 to μ, let $j^{(i)} = j^{(i-1)} \bmod n/2^i$, and let $j^{(i-1)}_{\mathsf{left}} = j^{(i)}$, $j^{(i-1)}_{\mathsf{right}} = j^{(i)} + n/2^i$. The verifier queries the oracles $\mathbf{c}^{(i-1)}$ and $\mathbf{c}^{(i)}$ for $\mathbf{c}^{(i-1)}_{[j^{(i-1)}_{\mathsf{left}}]}$, $\mathbf{c}^{(i-1)}_{[j^{(i-1)}_{\mathsf{right}}]}$ and $\mathbf{c}^{(i)}_{[j^{(i)}]}$, and checks that

$$\mathbf{c}^{(i)}_{[j^{(i)}]} = (1 - r_i) \cdot (\mathbf{c}^{(i-1)}_{[j^{(i-1)}_{\mathsf{left}}]} + \mathbf{c}^{(i-1)}_{[j^{(i-1)}_{\mathsf{right}}]}) \cdot \mathsf{Twist}_+(i, j^{(i)}) + r_i \cdot (\mathbf{c}^{(i-1)}_{[j^{(i-1)}_{\mathsf{left}}]} - \mathbf{c}^{(i-1)}_{[j^{(i-1)}_{\mathsf{right}}]}) \cdot \mathsf{Twist}_-(i, j^{(i)}).$$

random tapes. Let $\mathsf{TarF} : \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \mathsf{S}_{\mathsf{out}}$ *and* $\mathsf{PredF} : \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{out}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \{0,1\}$ *be two functions. Let the relation induced by* $\mathsf{TarF}, \mathsf{PredF}$ *be*

$$\mathcal{R}_{\mathsf{TarF},\mathsf{PredF}} = \{(x, w) \in \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \mid \forall r \in \mathsf{S}_{\mathsf{rand}}, \mathsf{PredF}(x, \mathsf{TarF}(x, w, r), w, r) = 1\}.$$

A pair of interactive algorithms (P, V) *is a* functional proof system (FPS) *for target function* TarF *and predicate function* PredF *if the following conditions hold:*

- ***Syntax****: The verifier* V *is a deterministic algorithm that takes an instance* $x \in \mathsf{S}_{\mathsf{ins}}$ *and a random tape* $r \in \mathsf{S}_{\mathsf{rand}}$ *as inputs. After interacting with the prover* P *(who takes* (x, w) *as inputs), the verifier outputs a value* $y \in \mathsf{S}_{\mathsf{out}}$*. We denote this by* $y \leftarrow \langle \mathsf{P}(x, w), \mathsf{V}(x, r) \rangle$*.*
- ***Completeness****: For any* $(x, w) \in \mathcal{R}_{\mathsf{TarF},\mathsf{PredF}}$*, the verifier's output matches the target function except with probability* $\varepsilon_{\mathsf{cmplt}}$*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} [y = \mathsf{TarF}(x, w, r) \mid y \leftarrow \langle \mathsf{P}(x, w), \mathsf{V}(x, r) \rangle] = 1 - \varepsilon_{\mathsf{cmplt}},$$

  where $\varepsilon_{\mathsf{cmplt}}$ *is the completeness error.*
- ***Soundness****: For any instance* $x \in \mathsf{S}_{\mathsf{ins}}$*, any (potentially malicious) prover* P^**, and any witness* $w \in \mathsf{S}_{\mathsf{wit}}$ *(which* P^* *might use or ignore), the output* y *produced by the verifier is guaranteed to satisfy the predicate* PredF *except with probability* $\varepsilon_{\mathsf{snd}}$*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} [\mathsf{PredF}(x, y, w, r) = 1 \mid y \leftarrow \langle \mathsf{P}^*(x, w), \mathsf{V}(x, r) \rangle] \geq 1 - \varepsilon_{\mathsf{snd}},$$

  where $\varepsilon_{\mathsf{snd}}$ *is the soundness error.*

We say the system has perfect completeness if $\varepsilon_{\mathsf{cmplt}} = 0$*, and perfect soundness if* $\varepsilon_{\mathsf{snd}} = 0$*.*

Traditional interactive proofs constitute a special case of FPS. For an NP language $\mathcal{L}$ with relation $\mathcal{R}$, we set $\mathsf{S}_{\mathsf{out}} = \{0,1\}$, $\mathsf{TarF}(x,w,r) = 1$, and

$$\mathsf{PredF}(x,y,w,r) = (x,w) \in \mathcal{R} \vee y = 0.$$

This yields $\mathcal{R}_{\mathsf{TarF},\mathsf{PredF}} = \{(x,w) \mid (x,w) \in \mathcal{R}\}$, recovering the original relation. Completeness ensures the verifier behaves as the constant one function TarF interacting with honest provers, while soundness prevents malicious provers from achieving $y = 1$ for invalid statements.

Some protocols exhibit gaps between completeness and soundness guarantees, where certain instances cannot be definitively classified with overwhelming probability. A prominent example is FRI (Fast Reed-Solomon IOP of Proximity), which allows vectors close to Reed-Solomon codewords to pass verification with non-negligible probability, even though they are not valid instances. To accommodate such proximity-based verification, we extend FPS to functional proof systems with proximity (FPSP). We present an informal definition below, with the complete definition deferred to Definition 11 in Appendix A.

Definition 6 (Functional Proof System with Proximity, Informal). *This is the same as Definition 5, with the following modifications: the protocol has an additional parameter* $\mathsf{RPredF} : \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{out}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \{0,1\}$ *called the relaxed predicate function, satisfying* $\mathsf{PredF} \leq \mathsf{RPredF}$ *for all inputs. The soundness is redefined as:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} [\mathsf{RPredF}(x,y,w,r) = 1 \mid y \leftarrow \langle \mathsf{P}^*(x,w), \mathsf{V}(x,r) \rangle] \geq 1 - \varepsilon_{\mathsf{snd}},$$

where $\varepsilon_{\mathsf{snd}}$ *is the soundness error.*

For practical cryptographic applications, we require a computational variant that relies on cryptographic assumptions. This necessitates a setup phase, since we can only guarantee that false witnesses are computationally hard to find rather than nonexistent. Without a randomized setup, adversarial provers could hardcode problematic witnesses. We provide an informal definition below, with the formal definition deferred to Definition 12 in Appendix A.

Definition 7 (Functional Argument System, Informal). *This is the same as Definition 5, with the following modifications: the protocol is additionally parameterized by the public parameter set* S_{pp}*. The target function* TarF *and predicate function* PredF *additionally takes* $\mathsf{pp} \in \mathsf{S}_{\mathsf{pp}}$ *as input. The relation becomes*

$$\mathcal{R}_{\mathsf{pp},\mathsf{TarF},\mathsf{PredF}} = \{(x,w) \in \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \mid \forall r \in \mathsf{S}_{\mathsf{rand}}, \mathsf{PredF}(\mathsf{pp}, x, \mathsf{TarF}(x,w,r), w, r) = 1\}.$$

The protocol additionally has a setup algorithm G *that receives the security parameter* 1^λ *and outputs a public parameter* $\mathsf{pp} \in \mathsf{S}_{\mathsf{pp}}$*. Finally, the soundness is redefined as follows:*

– ***Soundness****: For any pair of (potentially malicious) polynomial-time deterministic provers* $(\mathsf{P}_1^*, \mathsf{P}_2^*)$ *and any assistant input* z*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} \left[\mathsf{PredF}(\mathsf{pp}, x, y, w, r) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}, (x, w, \mathsf{st}) \leftarrow \mathsf{P}_1^*(\mathsf{pp}, z) \\ y \leftarrow \langle \mathsf{P}_2^*(\mathsf{pp}, x, \mathsf{st}, z), \mathsf{V}(\mathsf{pp}, x, r) \rangle \end{array} \right] \geq 1 - \varepsilon_{\mathsf{snd}},$$

where $\varepsilon_{\mathsf{snd}}$ *is the soundness error.*

For SNARK constructions, we further adapt the definition to incorporate knowledge soundness. We omit the full definition, which is the same as Definition 12 except the deployment of argument of knowledge.

Definition 8 (Functional Argument of Knowledge, FAoK). *This is the same as Definition 7, except the soundness is replaced by the following property.*

– ***Argument of knowledge****: For any pair of (potentially malicious) polynomial-time deterministic provers* $(\mathsf{P}_1^*, \mathsf{P}_2^*)$ *and any assistant input* z*, there exists an expected polynomial-time algorithm* Ext*, such that:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} \left[\mathsf{PredF}(\mathsf{pp}, x, y, w, r) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}, (x, \mathsf{st}) \leftarrow \mathsf{P}_1^*(\mathsf{pp}, z) \\ y \leftarrow \langle \mathsf{P}_2^*(\mathsf{pp}, x, \mathsf{st}, z), \mathsf{V}(\mathsf{pp}, x, r) \rangle, \\ w \leftarrow \mathsf{Ext}(\mathsf{pp}, x, \mathsf{st}, z) \end{array} \right] \geq 1 - \varepsilon_{\mathsf{snd}},$$

where $\varepsilon_{\mathsf{snd}}$ *is the soundness error.*

Next, we develop a framework for parallel FPS composition, enabling systematic combination of multiple protocol instances into a single FPS that asserts the conjunction of all component predicates.

3.1 FPS Composition Framework

The following definition specifies the composition of FPSs by parallel execution.

Definition 9 (Parallel FPS Composition). *Let* $(\mathsf{P}_i, \mathsf{V}_i)$ *for* $i \in [k]$ *be functional proof systems. The* parallel composition (P, V) *is defined as follows:*

– *The composed prover* P*, on input* $(x_0, \cdots, x_{k-1}, w_0, \cdots, w_{k-1})$*, runs all component provers* $\mathsf{P}_i(x_i, w_i)$ *in parallel. In each round,* P *sends the concatenation of all component prover messages as its message to the verifier.*
– *The composed verifier* V*, on input* $(x_0, \cdots, x_{k-1})$ *and random tapes* $(r_0, \cdots, r_{k-1})$*, runs all component verifiers* $\mathsf{V}_i(x_i, r_i)$ *in parallel. In each round,* V *parses the received message into component parts and forwards each part to the corresponding component verifier. The composed verifier outputs* $(y_0, \cdots, y_{k-1})$ *where* y_i *is the output of* V_i*.*

The composed system's security follows from the independence of component protocols, formalized in the following theorem, where the challenge derivation condition is automatically satisfied for public-coin protocols using shared randomness. FPSP, FAS, and FAoK also have corresponding composition theorems. We can also compose different FPS variants by e.g., viewing FPS as a special case of FPSP or FAS.

Theorem 1 (Composition Security). *Let* $(\mathsf{P}_i, \mathsf{V}_i)$ *for* $i \in [k]$ *be functional proof systems for* $\mathsf{S}^i_{\mathsf{ins}}, \mathsf{S}^i_{\mathsf{out}}, \mathsf{S}^i_{\mathsf{wit}}, \mathsf{S}^i_{\mathsf{rand}}, \mathsf{TarF}^i, \mathsf{PredF}^i$ *with completeness error* $\varepsilon^i_{\mathsf{cmplt}}$ *and soundness error* $\varepsilon^i_{\mathsf{snd}}$*, respectively. Suppose there is an efficiently computable function* ChallengeDerive *such that for any provers with any inputs, for the honest verifier with any inputs, for each round* j *the challenges* $\{c_i\}_{i\in[k]}$ *satisfy that for each* $i \in [k]$, $\{c^{i'}_j\}_{i'\neq i} = \mathsf{ChallengeDerive}(i, j, c^i_j)$*. Then the parallel composition* (P, V) *is a functional proof system for:*

$$\mathsf{S}_{\mathsf{ins}} = \mathsf{S}^0_{\mathsf{ins}} \times \cdots \times \mathsf{S}^{k-1}_{\mathsf{ins}} \qquad \mathsf{S}_{\mathsf{out}} = \mathsf{S}^0_{\mathsf{out}} \times \cdots \times \mathsf{S}^{k-1}_{\mathsf{out}}$$

$$\mathsf{S}_{\mathsf{wit}} = \mathsf{S}^0_{\mathsf{wit}} \times \cdots \times \mathsf{S}^{k-1}_{\mathsf{wit}} \qquad \mathsf{S}_{\mathsf{rand}} = \mathsf{S}^0_{\mathsf{rand}} \times \cdots \times \mathsf{S}^{k-1}_{\mathsf{rand}}$$

$$\mathsf{TarF}((x_i)_{i\in[k]}, (w_i)_{i\in[k]}, (r_i)_{i\in[k]}) = (\mathsf{TarF}^i(x_i, w_i, r_i))_{i\in[k]}$$

$$\mathsf{PredF}((x_i)_{i\in[k]}, (w_i)_{i\in[k]}, (y_i)_{i\in[k]}, (r_i)_{i\in[k]}) = (\mathsf{PredF}^i(x_i, w_i, y_i, r_i))_{i\in[k]},$$

with completeness error $\varepsilon_{\mathsf{cmplt}} = \sum_{i\in[k]} \varepsilon^i_{\mathsf{cmplt}}$ *and soundness error* $\varepsilon_{\mathsf{snd}} = \sum_{i\in[k]} \varepsilon^i_{\mathsf{snd}}$.

The proof is deferred to Appendix B. Intuitively, ChallengeDerive captures the requirement that for every i, $\{c^{i'}_j\}_{i'\neq i}$ contain no more information than c^i_j itself, thus do not leak information and interfere with the i-th component protocol. For public-coin protocols, ChallengeDerive simply returns $k-1$ copies of the received challenge c.

We now demonstrate that sum-check, BaseFold, and BulletProofs can be viewed as FPS variants, enabling modular security analysis of these protocols.

4 Revisiting Sum-check

We formalize the sum-check protocol as a functional proof system, which is a fundamental component in the formalization of BaseFold and BulletProofs.

Protocol 3 (Sum-check as FPS) We first define the following choice of sets and functions:

$$\mathsf{S}_{\mathsf{ins}} = \mathsf{S}_{\mathsf{out}} = \mathbb{F} \qquad \mathsf{S}_{\mathsf{wit}} = \mathbb{F}[X_0, X_1, \ldots, X_{\mu-1}] \qquad \mathsf{S}_{\mathsf{rand}} = \mathbb{F}^\mu$$

$$\mathsf{TarF}(c, f, \mathbf{r}) = f(\mathbf{r}) \quad \mathsf{PredF}(c, y, f, \mathbf{r}) = \left(\sum_{\mathbf{b}\in\{0,1\}^\mu} f(\mathbf{b}) = c\right) \vee (f(\mathbf{r}) \neq y).$$

The prover receives $c \in \mathbb{F}, f \in \mathbb{F}[X_0, X_1, \ldots, X_{\mu-1}]$, the verifier receives $c \in \mathbb{F}$ and random tape $\mathbf{r} \in \mathbb{F}^\mu = (r_1, \cdots, r_\mu)$.

- The parties execute Protocol 1, where in round i the verifier sends r_i as the challenge to the prover.
- Let the final message sent from the prover in the previous step be h_μ.
- The verifier outputs $y = h_\mu(r_\mu)$.

Note that by the choice of the sets and functions in Protocol 3, we have

$$\mathcal{R}_{\mathsf{TarF},\mathsf{PredF}} = \left\{ (c, f) \mid \forall \mathbf{r} \in \mathbb{F}^{\mu}, \sum_{\mathbf{b} \in \{0,1\}^{\mu}} f(\mathbf{b}) = c \vee f(\mathbf{r}) \neq y \right\}$$

which is exactly $\{(c, f) \mid \sum_{\mathbf{b} \in \{0,1\}^{\mu}} f(\mathbf{b}) = c\}$, since the second clause is impossible to hold for all $\mathbf{r}$. This is the natural choice of relation to prove for sum-check in an effort to formalize its security. However, under traditional perspective of proof systems, this is a trivial relation, as for any instance c, the witness f always exists and has abundant easy-to-find candidates.

Theorem 2. *Protocol 3 is a functional proof system with perfect completeness and soundness error $\frac{\mu \cdot m}{|\mathbb{F}|}$, where m is the maximum degree of f in any variable.*

The proof is a simple wrapper over the standard sum-check security. The detailed proof is deferred to Appendix C.

5 Revisiting BaseFold

Now we have sum-check captured as an FPS, we are ready to deconstruct existing protocols from the perspective of FPS. We start from BaseFold [ZCF23], decomposing it into two simpler functional proof systems: FriEval and sum-check.

5.1 FriEval

Following BaseFold's insight, we view FRI as a procedure for evaluating a committed polynomial at a random point. We capture this as FriEval.

Protocol 4 (FriEval) Let C be the Reed-Solomon code $\mathsf{RS}[\mathbb{F}, n, \rho]$ and let $d = \rho \cdot n = 2^{\mu}$. Let $\delta \in (0, 1)$ be a parameter. We write $\mathbf{c} \approx \mathbf{c}'$ if $\mathsf{HW}(\mathbf{c} - \mathbf{c}') < \delta \cdot n$. We define the following sets and functions:

$$\mathsf{S}_{\mathsf{ins}} = \mathbf{c} \in \mathbb{F}^n \qquad \mathsf{S}_{\mathsf{wit}} = \mathbb{F}^d \qquad \mathsf{S}_{\mathsf{rand}} = \mathbb{F}^{\mu} \qquad \mathsf{S}_{\mathsf{out}} = \mathbb{F} \cup \{\perp\}$$

$$\mathsf{TarF}(\mathbf{c}, \mathbf{v}, \mathbf{r}) = \hat{\mathbf{v}}(\mathbf{r})$$

$$\mathsf{PredF}(\mathbf{c}, y, \mathbf{v}, \mathbf{r}) = (y = \perp) \vee (\mathbf{c} = \mathsf{Encode}(\mathbf{v}) \wedge y = \hat{\mathbf{v}}(\mathbf{r}))$$

$$\mathsf{RPredF}(\mathbf{c}, y, \mathbf{v}, \mathbf{r}) = (y = \perp) \vee (\exists \mathbf{v}' \in \mathbb{F}^d \text{ s.t. } \mathbf{c} \approx \mathsf{Encode}(\mathbf{v}') \wedge y = \hat{\mathbf{v}}'(\mathbf{r}))$$

The prover receives $\mathbf{c} \in \mathbb{F}^n$ and $\mathbf{v} \in \mathbb{F}^d$. The verifier receives $\mathbf{c} \in \mathbb{F}^n$ in the form of string oracle, and random tape $\mathbf{r} \in \mathbb{F}^{\mu} = (r_1, \cdots, r_{\mu})$.

- The parties proceed exactly as in Protocol 2.
- Let the final message from the prover in the commit phase of Protocol 2 be $\mathbf{c}^{(\mu)}$. In the honest case, $\mathbf{c}^{(\mu)}$ should consist of identical elements.
- The verifier outputs the only element of $\mathbf{c}^{(\mu)}$ or $\perp$ if the query phase fails.

Note that the relaxed predicate ignores the input $\mathbf{v}$ and accepts if $\mathbf{c}$ can be decoded to any $\mathbf{v}'$ that evaluates to y, which is irrelevant to $\mathbf{v}$, because FPSP soundness requires the predicate function to hold for *any* $\mathbf{v}$, while we only need to ensure *the* $\mathbf{v}$ encoded in $\mathbf{c}$ exists and evaluates to y.

Remark 1. Throughout this paper, we treat the string oracle of $\mathbf{c}$ to be interchangeable with $\mathbf{c}$ itself, to avoid unnecessary complications, e.g., including $\mathbf{c}$ as a witness when the verifier has oracle access to $\mathbf{c}$. Readers may safely ignore the oracles, treating the string oracle of $\mathbf{c}$ as $\mathbf{c}$ itself, for understanding the definitions in a rigorous manner. The oracles are only necessary when analyzing the verifier's performance.

Theorem 3. *Let ρ be the rate of C and $m \geq 3$ be a small integer. Let $\theta = 1-(1+\frac{1}{2m})\cdot\sqrt{\rho}$. The* FriEval *protocol in Protocol 4 is a functional proof system with perfect completeness and soundness error*

$$\varepsilon_{\mathsf{snd}} < \sum_{i=1}^{\mu}\left(\frac{1}{|\mathbb{F}|} + \varepsilon(n^{(i)}, B, \theta)\right) + (1-\theta)^{\ell},$$

where

$$\varepsilon(n, B, \theta) = \frac{1}{|\mathbb{F}|}\cdot\frac{1+1/2m}{\sqrt{\rho}}\cdot\max\left(\frac{(1+1/2m)^6 m^7}{3\rho}\cdot n^2, 2\cdot(n\cdot B+1)\right).$$

The proof follows the same structure as in [BCI+20]. The detailed proof is deferred to Appendix D due to space constraints.

5.2 BaseFold from FPS

We now formalize BaseFold as an FPS. The protocol proves that a committed multilinear polynomial f evaluates to t at $\mathbf{z} \in \mathbb{F}^{\mu}$, via a sum-check for the polynomial $f(\mathbf{X})\cdot\mathsf{eq}(\mathbf{z},\mathbf{X})$. The final step is for the verifier to check if the output y equals $f(\mathbf{r})\cdot\mathsf{eq}(\mathbf{z},\mathbf{r})$. To enable this final verification, BaseFold runs FRI and sum-check protocols in parallel, leveraging the polynomial evaluation functionality captured by FriEval.

We first compose FriEval and sum-check into a new FPSP, which is not yet a PCS, and then wrap this FPS within the PCS definition to complete the construction. For the first step, let the sets and functions for the sum-check FPS be $\mathsf{S}_{\mathsf{ins}}^{sum}$, etc. and those for the FriEval FPSP be $\mathsf{S}_{\mathsf{ins}}^{fri}$, etc. defined in Protocol 3 and Protocol 4, respectively. Let $\mathsf{S}_{\mathsf{rand}} = \mathbb{F}^{\mu}$ be the shared randomness space. The composed protocol, denoted BaseFoldFPS, executes the sum-check and FriEval protocols in parallel, per Definition 9. The full description is omitted for brevity. For the second step, we present the complete description of BaseFold.

Due to the proximity relaxation, the above PCS does not satisfy Definition 3 strictly, as an adversary can open the same $\mathbf{c}$ to multiple (polynomially many) evaluations y at the same point $\mathbf{z}$, such relaxed PCS can still be used to construct SNARKs, as discussed in RedShift [KPV19], or by applying the DEEP technique [BGKS19].

Protocol 5 (BaseFold from FPS) *The BaseFold polynomial commitment scheme consists of the following algorithms.*

- ***Setup.*** *On input* 1^λ *and degree bound* D*, output* pp *containing parameters for the Reed-Solomon code family* $C^{(0)}, \cdots, C^{(\mu)}$ *with* $d^{(0)} = D$ *and code rate* $\rho = d^{(0)}/n^{(0)}$*.*
- ***Derive.*** *On input* pp*, output* $\mathsf{vp} = \mathsf{pp}$*.*
- ***Commit.*** *On input* pp *and* μ*-variate polynomial* f *such that* $2^\mu = D$*, output the oracle to* $\mathbf{c} = \mathsf{Encode}_{C^{(0)}}(f(\{0,1\}^\mu))$*.*
- ***Open.*** *On input* pp*,* $\mathbf{c}, f$*, checks if* $\mathbf{c} = \mathsf{Encode}_{C^{(0)}}(f(\{0,1\}^\mu))$*.*
- ***Eval.*** *The parties receive public inputs* $\mathsf{vp}, \mathbf{c}, \mathbf{z}, y$*; the prover is also given* pp *and* f*. They execute the* $\mathsf{BaseFoldFPS}$ *protocol with instance* $(\mathbf{c}, y)$ *and witness* $(f(\mathbf{X}), f(\mathbf{X}) \cdot \mathsf{eq}(\mathbf{z}, \mathbf{X}))$*. The verifier receives* (y_{sum}, y_{fri}) *and checks that* $y_{fri} \neq \perp$ *and* $y_{sum} = y_{fri} \cdot \mathsf{eq}(\mathbf{z}, \mathbf{r})$*.*

Theorem 4 (BaseFold Security via FPS). *Protocol 5 is a PCS (Definition 3) with perfect completeness, perfect binding, and knowledge soundness for relation* $\mathcal{R}'_{\mathsf{Eval}}$ *with error* $\varepsilon_{\mathsf{snd}} = \varepsilon^{fri}_{\mathsf{snd}} + \varepsilon^{sum}_{\mathsf{snd}}$*:*

$$\mathcal{R}'_{\mathsf{Eval}} = \{(\mathsf{pp}, (\mathbf{c}, \mathbf{z}, y), f) | f(\mathbf{z}) = y \wedge \exists \mathbf{c}', \mathsf{Open}(\mathsf{pp}, \mathbf{c}', f) = 1 \wedge \Delta(\mathbf{c}, \mathbf{c}') < \delta \cdot D\}.$$

The detailed proof is provided in Appendix E.

6 Revisiting BulletProofs

Similar to BaseFold, the BulletProofs IPA [BBB+18] can also be deconstructed into a sum-check and a multilinear polynomial evaluation FPS based on the Pedersen commitment. We name the latter component $\mathsf{BulletEval}$.

6.1 BulletEval

The $\mathsf{BulletEval}$ protocol proves the knowledge of a vector $\mathbf{v}$ committed as $C = \langle \mathbf{G}, \mathbf{v} \rangle$ with public bases $\mathbf{G} \in \mathbb{G}^n$, and simultaneously reveals the evaluation $\hat{\mathbf{v}}(\mathbf{r})$ at a random point $\mathbf{r}$. As observed in [BCS21], this is achieved using a group-valued sum-check on the polynomial $\hat{\mathbf{G}}(\mathbf{X}) \cdot \hat{\mathbf{v}}(\mathbf{X})$ with claimed sum C, as detailed in Protocol 6.

Definition 10. *Let* n *be any positive integer. We say* $\mathbb{G}$ *satisfies* n*-DLG assumption if for any polynomial-time probabilistic adversary* $\mathcal{A}$*:*

$$\Pr\left[\langle \mathbf{G}, \mathbf{v} \rangle = O \,\middle|\, \mathbf{G} \xleftarrow{\$} \mathbb{G}^n, \mathbf{v} \leftarrow \mathcal{A}\right] \leq \varepsilon_{n\text{-}DLG},$$

where $\varepsilon_{n\text{-}DLG}$ *is a negligible function. For convenience, we call* $\mathbf{v}$ *a collision for* $\mathbf{G}$ *if* $\langle \mathbf{G}, \mathbf{v} \rangle = O$*.*

Theorem 5. *Assume* $\mathbb{G}$ *satisfies* d*-DLG assumption. The* $\mathsf{BulletEval}$ *protocol in Protocol 6 is a FAoK with perfect completeness and negligible soundness error.*

Protocol 6 (BulletEval) Let μ be a positive integer and $d = 2^\mu$. We formalize $\mathsf{BulletEval}$ as a functional argument system for the following parameters:

$$\mathsf{S}_{\mathsf{pp}} = \mathbb{G}^d \quad \mathsf{S}_{\mathsf{ins}} = \mathbb{G} \quad \mathsf{S}_{\mathsf{wit}} = \mathbb{F}^d \quad \mathsf{S}_{\mathsf{rand}} = \mathbb{F}^\mu \quad \mathsf{S}_{\mathsf{out}} = \mathbb{F} \cup \{\perp\}$$

$$\mathsf{TarF}(C, \mathbf{G}, \mathbf{v}, \mathbf{r}) = \hat{\mathbf{v}}(\mathbf{r})$$

$$\mathsf{PredF}(C, \mathbf{G}, \mathbf{v}, \mathbf{r}, y) = (C = \langle \mathbf{G}, \mathbf{v} \rangle \wedge y = \hat{\mathbf{v}}(\mathbf{r})) \vee (y = \perp).$$

Setup. On receiving 1^λ, the setup algorithm G uniformly randomly samples $\mathbf{G}$ from $\mathbb{G}^d$ and outputs $\mathbf{G}$.
Interaction. The prover receives $\mathbf{G}$, the commitment C and the vector $\mathbf{v}$. The verifier receives $\mathbf{G}$, the commitment C, and the random tap $\mathbf{r}$. The parties interact as follows:

- **Round 1**: The prover sends a univariate polynomial $H_1(X) = \sum_{\mathbf{b} \in \{0,1\}^{\mu-1}} \hat{\mathbf{G}}(X, \mathbf{b}) \cdot \hat{\mathbf{v}}(\mathbf{b})$. The verifier checks that $\deg(H_1) \leq 2$ and that $H_1(0) + H_1(1) = C$. The verifier then samples a random challenge $r_1 \in \mathbb{F}$ and sends it to the prover.
- **Round** i **(**$2 \leq i \leq \mu$**)**: The prover sends a univariate polynomial $H_i(X) = \sum_{\mathbf{b} \in \{0,1\}^{\mu-i}} \hat{\mathbf{G}}(r_1, \ldots, r_{i-1}, X, \mathbf{b}) \cdot \hat{\mathbf{v}}(r_1, \ldots, r_{i-1}, X, \mathbf{b})$. The verifier checks that $\deg(H_i) \leq 2$ and that $H_i(0) + H_i(1) = H_{i-1}(r_{i-1})$. The verifier then samples a random challenge $r_i \in \mathbb{F}$ and sends it to the prover.
- **Final check**: After μ rounds, the verifier has challenges $r_1, \ldots, r_\mu$. The prover sends $y = \hat{\mathbf{v}}(\mathbf{r})$. The verifier computes $Y = H_\mu(r_\mu)$ and $G = \hat{\mathbf{G}}(\mathbf{r})$ and checks if $G \cdot y = Y$. If the check fails the verifier outputs $y = \perp$.

The proof is a simplification of the one for Theorem 5.1 in [EG25], since $\mathsf{BulletEval}$ uses a single group-valued sum-check, while the proof in [EG25] is for the entire BulletProofs. The detailed proof is deferred to Appendix F.

6.2 BulletProofs from FPS

We are now ready to reconstruct the BulletProofs IPA, which proves knowledge of a committed witness vector $\mathbf{v}$ such that its inner product with a vector $\mathbf{u}$ is a given value t. We focus on the case where $\mathbf{u}$ is public, as this leads to polynomial commitment schemes by setting $\mathbf{u} = \mathsf{eq}(\mathbf{z}, \{0,1\}^\mu)$.

The idea is to apply the sum-check protocol to the polynomial $f(\mathbf{X}) = \hat{\mathbf{v}}(\mathbf{X}) \cdot \hat{\mathbf{u}}(\mathbf{X})$ to prove that its sum over the hypercube is t. Sum-check reduces this claim to a single evaluation claim at a random point $\mathbf{r}$, i.e., $y = \hat{\mathbf{v}}(\mathbf{r}) \cdot \hat{\mathbf{u}}(\mathbf{r})$. Since $\mathbf{u}$ is public, the verifier can compute $\hat{\mathbf{u}}(\mathbf{r})$ itself. To obtain the evaluation $\hat{\mathbf{v}}(\mathbf{r})$, we compose the sum-check with the $\mathsf{BulletEval}$ protocol.

Following the same two-step approach used for BaseFold, we first construct a functional argument of knowledge (FAoK) for the inner product, which we call $\mathsf{BulletFAoK}$. This is not yet a full IPA. $\mathsf{BulletFAoK}$ is constructed by composing the sum-check FPS (Protocol 3) and the $\mathsf{BulletEval}$ protocol (Protocol 6) in parallel, as described in Definition 9, with shared randomness space $\mathsf{S}_{\mathsf{rand}} = \mathbb{F}^\mu$.

In the second step, we construct the full BulletProofs IPA by adding the commitment and opening procedures to the $\mathsf{BulletFAoK}$ protocol.

Protocol 7 (BulletProofs from FPS) The BulletProofs inner product argument consists of the following algorithms.

- **Setup.** On input 1^λ, degree bound D such that $D = 2^\mu$, run the setup algorithm of $\mathsf{BulletEval}$ to obtain $\mathsf{pp} = \mathbf{G} \in \mathbb{G}^D$.
- **Derive.** On input pp, output $\mathsf{vp} = \mathsf{pp}$.
- **Commit.** On input pp and vector $\mathbf{v} \in \mathbb{F}^D$, output $C = \langle \mathbf{G}, \mathbf{v} \rangle$.
- **Open.** On input pp, $C, \mathbf{v}$, checks if $C = \langle \mathbf{G}, \mathbf{v} \rangle$.
- **Eval.** Both parties are given vp, C, $\mathbf{u}$, and t. The prover is additionally given pp and $\mathbf{v}$. The parties execute the $\mathsf{BulletFAoK}$ protocol with public parameter pp, instance (C, t) and witness $(\mathbf{v}, \hat{\mathbf{u}}(\mathbf{X}) \cdot \hat{\mathbf{v}}(\mathbf{X}))$. At the end of the protocol, the verifier outputs (y_{sum}, y_{eval}). The verifier accepts if $y_{eval} \neq \perp$ and $y_{sum} = y_{eval} \cdot \hat{\mathbf{u}}(\mathbf{r})$.

Theorem 6 (BulletProofs Security via Composition). *Assuming the D-DLOG assumption holds in group $\mathbb{G}$, Protocol 7 is an IPA with perfect completeness and knowledge soundness with error $\varepsilon_{\mathsf{snd}}^{eval} + \varepsilon_{\mathsf{snd}}^{sum}$.*

The detailed proof is deferred to Appendix G due to space limitation.

It is worth noting the strong similarity between the reconstructed BulletProofs protocol (Protocol 7) and the BaseFold protocol (Protocol 5). The only differences are: (1) they use different evaluation FPS (FAoK) for the committed multilinear polynomial, and (2) the public polynomial is eq for BaseFold and $\hat{\mathbf{u}}$ for BulletProofs. The modularity allows us to interchange components. For instance, by using the Reed-Solomon based evaluation FPS from BaseFold in the BulletProofs construction, we can create an IPA based on Reed-Solomon codes. Conversely, by using the Pedersen-based evaluation FPS from BulletProofs in the BaseFold construction, we can create a polynomial commitment scheme based on elliptic curve cryptography.

7 Bird: Transparent and Doubly Efficient PCS

We now introduce the application of our FPS framework in constructing a novel polynomial commitment scheme, which we name Bird, that achieves several desirable properties simultaneously: it is transparent, doubly efficient (i.e., featuring a linear-time prover and a logarithmic-time verifier), pairing-free, and supports homomorphic operations on commitments.

Bird is constructed by composing three components prepared in previous sections: the $\mathsf{FriEval}$ protocol (Protocol 4), the $\mathsf{BulletEval}$ protocol (Protocol 6), and the sum-check protocol (Protocol 3), where the $\mathsf{FriEval}$ protocol is adapted to operate on group-valued vectors as detailed later in Section 7.1.

As demonstrated in Section 6.2, the combination of the $\mathsf{BulletEval}$ and sum-check protocols forms the basis of BulletProofs. By replacing the public vector $\mathbf{u}$ with the evaluation of the equality polynomial eq, BulletProofs can be transformed into a polynomial commitment scheme. However, a well-known drawback of BulletProofs is the verifier's complexity: it must perform a linear number of

group operations to evaluate $G = \hat{\mathbf{G}}(\mathbf{r})$. To address this, we introduce a group-valued version of FRI, which delegates this costly evaluation to the prover.

7.1 Group-Valued FriEval

We first extend the $\mathsf{FriEval}$ protocol to operate on vectors of group elements. The core observation is that the operations in FRI are linear, which is concurrently noted and exploited in [EG25]. The prover's main task is to compute the Fold operation on a codeword, which is a linear transformation. The verifier's tasks—mainly the query checks, are also linear checks of the form $a \cdot \mathbf{c}_{[i]} + b \cdot \mathbf{c}_{[j]} = c \cdot \mathbf{c}'_{[k]}$. These linear operations can be directly lifted to the group setting by replacing the field elements of the codeword with their corresponding group elements.

This extension results in a group-valued FRI evaluation protocol, which we call $\mathsf{FriEval}_{\mathbb{G}}$. Let $C = \mathsf{RS}[\mathbb{G}, n, \rho]$ be a Reed-Solomon code over a group $\mathbb{G}$ with message length n and rate ρ, and let the codeword length be $d = n/\rho = 2^\mu$. We claim that $\mathsf{FriEval}_{\mathbb{G}}$ is an FPSP for the following sets and functions:

$$\mathsf{S}_{\mathsf{ins}} = \mathbb{G}^n \qquad \mathsf{S}_{\mathsf{wit}} = \mathbb{G}^d \qquad \mathsf{S}_{\mathsf{rand}} = \mathbb{F}^\mu \qquad \mathsf{S}_{\mathsf{out}} = \mathbb{G} \cup \{\perp\}$$

$$\mathsf{TarF}(\mathbf{C}, \mathbf{V}, \mathbf{r}) = \hat{\mathbf{V}}(\mathbf{r})$$

$$\mathsf{PredF}(\mathbf{C}, Y, \mathbf{V}, \mathbf{r}) = (Y = \perp) \vee (\mathbf{C} = \mathsf{Encode}_C(\mathbf{V}) \wedge Y = \hat{\mathbf{V}}(\mathbf{r}))$$

$$\mathsf{RPredF}(\mathbf{C}, Y, \mathbf{V}, \mathbf{r}) = (Y = \perp) \vee (\exists \mathbf{V}' \in \mathbb{G}^d \text{ s.t. } \mathbf{C} \approx \mathsf{Encode}_C(\mathbf{V}') \wedge Y = \hat{\mathbf{V}}'(\mathbf{r})),$$

where the definition of $\approx$ is the same as in Protocol 4 for a parameter $\delta \in (0, 1)$.

7.2 The PCS Construction

We now construct the Bird polynomial commitment scheme. The construction is a two-step process. First, we compose the $\mathsf{FriEval}_{\mathbb{G}}$, $\mathsf{BulletEval}$, and sum-check protocols to create a single FAoK with proximity. Then, we build the full PCS by adding the commitment and opening procedures around this FAoK.

In the first step, we construct the FAoK with proximity, named $\mathsf{BirdFAoK}$, using our composition framework (Definition 9). This is achieved by running the sum-check, $\mathsf{BulletEval}$, and $\mathsf{FriEval}_{\mathbb{G}}$ protocols in parallel, with a shared randomness space $\mathsf{S}_{\mathsf{rand}} = \mathbb{F}^\mu$. The parameter sets for each protocol are as defined in their respective sections. For the sum-check and $\mathsf{BulletEval}$ protocols, the relaxed predicate is the same as the standard predicate.

In the second step, we define the algorithms of the Bird polynomial commitment scheme.

Theorem 7 (Bird Security). *The Bird protocol (in Protocol 8) is a PCS (Definition 3) with perfect completeness, binding error $\varepsilon_{D\text{-}DLG}$, and knowledge soundness with error $\varepsilon_{\mathsf{snd}} = \varepsilon_{\mathsf{snd}}^{fri} + \varepsilon_{\mathsf{snd}}^{sum} + \varepsilon_{\mathsf{snd}}^{eval}$.*

Protocol 8 (Bird) The Bird PCS consists of the following algorithms.

- **Setup.** On input 1^λ and a degree bound $D = 2^\mu$, run the setup of BulletEval to obtain $\mathbf{G} \in \mathbb{G}^D$. Let $\{C^{(i)}\}_{i=0}^{\mu}$ be a Reed-Solomon code family where $C^{(0)}$ has codeword length $d^{(0)} = D$ and rate $\rho = n^{(0)}/d^{(0)}$. Let $\mathbf{C} = \mathsf{Encode}_{C^{(0)}}(\mathbf{G})$. The public parameters are $\mathsf{pp} = (\mathbf{G}, \mathbf{C}, \{C^{(i)}\})$.
- **Derive.** On input pp, the verifier's parameters vp include the oracle of $\mathbf{C}$ and descriptions of $\{C^{(i)}\}_{i=0}^{\mu}$.
- **Commit.** On input pp and $\mathbf{v} \in \mathbb{F}^D$, the commitment is $C = \langle \mathbf{G}, \mathbf{v} \rangle$.
- **Open.** On input pp, C, and $\mathbf{v}$, check if $C = \langle \mathbf{G}, \mathbf{v} \rangle$.
- **Eval.** Both parties are given vp, a commitment $C \in \mathbb{G}$, a point $\mathbf{z} \in \mathbb{F}^\mu$, and an evaluation $y \in \mathbb{F}$. The prover is additionally given pp and the vector $\mathbf{v} \in \mathbb{F}^D$. The parties execute the BirdFAoK protocol with instance $(\mathbf{C}, C, y)$ and witness $(\mathbf{G}, \mathbf{v}, \hat{\mathbf{v}}(\mathbf{X}) \cdot \mathsf{eq}(\mathbf{z}, \mathbf{X}))$. A modification is made in the final check of the underlying BulletEval protocol: instead of computing $G = \hat{\mathbf{G}}(\mathbf{r})$, the verifier uses the output Y_{fri} of the $\mathsf{FriEval}_{\mathbb{G}}$ protocol for G. The verifier accepts if $y_{eval} \neq \perp$, $Y_{fri} \neq \perp$, and $y_{sum} = y_{eval} \cdot \mathsf{eq}(\mathbf{z}, \mathbf{r})$.

Proof. **Completeness.** The perfect completeness of Bird follows directly from the perfect completeness of its components ($\mathsf{FriEval}_{\mathbb{G}}$, BulletEval, and sum-check) and the composition theorem (Theorem 1). The modification in the "Eval" step does not affect completeness because the completeness of $\mathsf{FriEval}_{\mathbb{G}}$ guarantees that $Y_{fri} = \hat{\mathbf{G}}(\mathbf{r})$ with probability 1.

Binding. The binding property relies on the D-DLOG assumption. If an adversary can open a commitment C to two different vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, they can compute a non-zero vector $\mathbf{v}' = \mathbf{v}_1 - \mathbf{v}_2$ such that $\langle \mathbf{G}, \mathbf{v}' \rangle = 0$, which constitutes a collision for $\mathbf{G}$. The probability of this is at most $\varepsilon_{D\text{-}DLG}$.

Knowledge Soundness. We analyze the knowledge soundness of the modified protocol by relating it to an unmodified version where the verifier computes $G = \hat{\mathbf{G}}(\mathbf{r})$ directly.

1. For the unmodified protocol, the same argument as in the proof of Theorem 6 applies. The composition of BulletEval and sum-check ensures that an extractor can output a vector $\mathbf{v}$ such that $C = \langle \mathbf{G}, \mathbf{v} \rangle$ and $\langle \mathbf{v}, \mathsf{eq}(\mathbf{z}, \{0,1\}^\mu) \rangle = y$ (which is equivalent to $\hat{\mathbf{v}}(\mathbf{z}) = y$), except with probability $\varepsilon_{\mathsf{snd}}^{eval} + \varepsilon_{\mathsf{snd}}^{sum}$.
2. Now consider the modified protocol. The soundness of $\mathsf{FriEval}_{\mathbb{G}}$ guarantees that, except with probability $\varepsilon_{\mathsf{snd}}^{fri}$, the output Y_{fri} is either $\perp$ or equal to the true evaluation $\hat{\mathbf{G}}(\mathbf{r})$.
3. When the verifier accepts, $Y_{fri} \neq \perp$, so we have $Y_{fri} = \hat{\mathbf{G}}(\mathbf{r})$. The modified protocol is therefore identical to the unmodified one from the verifier's perspective.

By the union bound, the total soundness error is the sum of the errors of the three components. The extractor for the composed protocol can successfully extract a valid witness $\mathbf{v}$ except with probability $\varepsilon_{\mathsf{snd}}^{fri} + \varepsilon_{\mathsf{snd}}^{eval} + \varepsilon_{\mathsf{snd}}^{sum}$. □

7.3 Efficiencies and Properties

To make the protocol concrete, we instantiate the oracles of the codewords in FRI using Merkle trees. This allows us to analyze the efficiency and properties of the resulting PCS, which we summarize below.

Transparency. The setup phase requires no trusted setup as the public parameters consist only of random group elements.

Efficiency. The commit complexity and prover complexity are both optimal. The verifier complexity and communication size are both polylogarithmic.

- **Commit complexity**: $O(2^\mu)$ group operations for computing $\langle \mathbf{G}, \mathbf{v} \rangle$.
- **Prover complexity**: $O(2^\mu)$ group operations for $\mathsf{FriEval}_{\mathbb{G}}$ and $\mathsf{BulletEval}$ and $O(n)$ field operations for sum-check.
- **Verifier Complexity**: $O(\mu \cdot \ell)$ group operations for$\mathsf{FriEval}_{\mathbb{G}}$, $O(\mu^2 \cdot \ell)$ hashes in checking Merkle paths in $\mathsf{FriEval}_{\mathbb{G}}$, $O(\mu)$ group operations for $\mathsf{BulletEval}$, and $O(\mu)$ field operations for sum-check, where $\ell = O(\lambda)$ is the number of queries made in $\mathsf{FriEval}_{\mathbb{G}}$. Overall, the verifier complexity is dominated by $O(\lambda \cdot \mu^2)$ group operations.
- **Communication**: $O(\mu \cdot \ell)$ group elements in $\mathsf{FriEval}_{\mathbb{G}}$, $O(\mu^2 \cdot \ell)$ hash digests in Merkle paths, $O(\mu)$ group operations for $\mathsf{BulletEval}$, and $O(\mu)$ field elements for sum-check. Overall, the communication size is dominated by $O(\lambda \cdot \mu^2)$ hash digests.

Pairing-Free. The construction relies only on standard discrete logarithm assumptions in a single group, avoiding the complexity and efficiency overhead of bilinear pairings.

Homomorphic Properties. The commitment scheme naturally supports homomorphic addition. This property inherits from that of Pedersen commitments and can be beneficial to the overall performance of SNARKs that involve batch opening hundreds to thousands of polynomial commitments, e.g., zero-knowledge virtual machines (ZKVMs) for modern processors such as RISC-V.

Comparison. Table 1 compares the performance and properties of Bird with KZG [KZG10], BulletProofs [BBB+18] and FRI [BBHR18], the representatives of pairing-based, group-based and hash-based PCS, respectively, which still have the state-of-the-art asymptotic costs in their respective categories. We also compare with the concurrent work of Eagen et al. [EG25]. The table shows that Bird and [EG25] are the only PCS that achieve all the desired properties and asymptotic costs.

Compared to [EG25], the advantage of Bird is in the concrete level: (1) by composing BulletProofs directly with the group-valued version of FRI rather than with BaseFold, Bird saves the cost of one additional sum-check introduced from BaseFold; and (2) by parallel composition rather than sequential composition, Bird costs only μ rounds compared to 2μ rounds in [EG25].

Table 1. Comparing Bird with KZG, BulletProofs, FRI and [EG25] shows that Bird and [EG25] are the only PCS that achieve all the desired properties and asymptotic costs. Here λ is the security parameter. The running time is measured in the unit of field operations, where a 2-to-1 hash is considered to have the same level of cost as a field operation, a group scalar multiplication is considered to take $O(\lambda)$ field operations, and a multi-scalar multiplication of scale n takes $O(\lambda n/\log n)$ field operations. The proof size is measured in the unit of field elements. "Hm." stands for "Homomorphic". "Tr." stands for "Transparent". The *undesired* properties and asymptotic costs are marked with surrounding boxes.

Property	Committing	Open	Verify	Proof Size	Hm.	Tr.
KZG [KZG10]	$O\left(\frac{\lambda n}{\log n}\right)$	$O(\lambda n)$	$O(\lambda)$	$O(1)$	✓	$\boxed{\times}$
BulletProofs [BBB$^+$18]	$O\left(\frac{\lambda n}{\log n}\right)$	$O(\lambda n)$	$\boxed{O(\lambda n)}$	$O(\log n)$	✓	✓
FRI [BCR$^+$19]	$\boxed{O(n \log n)}$	$O(n)$	$O(\log^2 n)$	$O(\log^2 n)$	$\boxed{\times}$	✓
[EG25] (Concurrent work)	$O\left(\frac{\lambda n}{\log n}\right)$	$O(\lambda n)$	$O(\lambda \log n)$	$O(\log^2 n)$	✓	✓
Bird (This work)	$O\left(\frac{\lambda n}{\log n}\right)$	$O(\lambda n)$	$O(\lambda \log n)$	$O(\log^2 n)$	✓	✓

8 Conclusion

This work introduces the FPS framework, a new paradigm for analyzing and designing interactive proofs. By shifting the focus from a binary decision to the verifier viewed as a function, the FPS framework provides a more robust and modular way to reason about the security of fundamental protocols like sum-check, FRI, and BulletProofs. The primary impact of this approach is the simplification of security analysis for complex proof systems. Our composition theorem allows designers to build protocols from independently verified components, moving away from monolithic, error-prone proofs and enabling a more confident and agile development process, as demonstrated by our analyses of BaseFold and BulletProofs. The practical implications of this framework are highlighted by our construction of a novel PCS, which is transparent, doubly efficient, and supports homomorphic additions.

References

ACFY24a. Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed–Solomon Proximity Testing with Fewer Queries. https://eprint.iacr.org/2024/390, 2024.

ACFY24b. Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: Reed–Solomon Proximity Testing with Super-Fast Verification. https://eprint.iacr.org/2024/1586, 2024.

Azt22. Team Aztec. Aztec. https://zk.money/, 2022.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy,*

SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA, pages 315–334. IEEE Computer Society, 2018.

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. http://drops.dagstuhl.de/opus/volltexte/2018/9018.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference on - ITCS '12*, pages 326–349. ACM Press, 2012.

BCF+25. Anubhav Baweja, Alessandro Chiesa, Elisabetta Fedele, Giacomo Fenzi, Pratyush Mishra, Tushar Mopuri, and Andrew Zitek-Estrada. Time-space trade-offs for sumcheck. https://eprint.iacr.org/2025/1473, 2025.

BCHO22. Jonathan Bootle, Alessandro Chiesa, Yuncong Hu, and Michele Orrù. Gemini: Elastic SNARKs for Diverse Environments. Technical Report 420, 2022. http://eprint.iacr.org/2022/420.

BCI+20. Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed-Solomon Codes. https://eprint.iacr.org/2020/654, 2020.

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent Succinct Arguments for R1CS. In *Advances in Cryptology – EUROCRYPT 2019*, volume 11476, pages 103–128. Springer International Publishing, 2019.

BCS21. Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck Arguments and their Applications. Technical Report 333, 2021. http://eprint.iacr.org/2021/333.

BDDT25. Suyash Bagad, Quang Dao, Yuval Domb, and Justin Thaler. Speeding up sum-check proving. https://eprint.iacr.org/2025/1117, 2025.

BDFG20. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo Infinite: Recursive zk-SNARKs from any Additive Polynomial Commitment Scheme. Technical Report 1536, 2020.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK Compilers. In *Advances in Cryptology – EUROCRYPT 2020*, Lecture Notes in Computer Science, pages 677–706. Springer International Publishing, 2020.

BGKS19. Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling Outside the Box Improves Soundness. https://eprint.iacr.org/2019/336, 2019.

CBBZ22. Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates. https://eprint.iacr.org/2022/1355, 2022.

CHM+20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS. In *Advances in Cryptology – EUROCRYPT 2020*, Lecture Notes in Computer Science, pages 738–768. Springer International Publishing, 2020.

CTV15. Alessandro Chiesa, Eran Tromer, and Madars Virza. Cluster Computing in Zero Knowledge. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9057, pages 371–403. Springer Berlin Heidelberg, 2015. http://link.springer.com/10.1007/978-3-662-46803-6_13.

DT24. Quang Dao and Justin Thaler. More Optimizations to Sum-Check Proving. https://eprint.iacr.org/2024/1210, 2024.

EG25. Liam Eagen and Ariel Gabizon. Revisiting the IPA-sumcheck connection. https://eprint.iacr.org/2025/1325, 2025.

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the Fourtieth Annual ACM Symposium on Theory of Computing - STOC 08*, page 113. ACM Press, 2008. http://dl.acm.org/citation.cfm?doid=1374376.1374396.

GLS+21. Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and post-quantum SNARKs for R1CS, 2021.

GPS25. Chaya Ganesh, Sikhar Patranabis, and Nitin Singh. Samaritan: Linear-time prover SNARK from new multilinear polynomial commitments. https://eprint.iacr.org/2025/419, 2025.

Gro16. Jens Groth. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology – EUROCRYPT 2016*, volume 9666, pages 305–326. Springer Berlin Heidelberg, 2016.

Gru24. Angus Gruen. Some Improvements for the PIOP for ZeroCheck. https://eprint.iacr.org/2024/108, 2024.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. Technical Report 953, 2019.

Hab24. Ulrich Haböck. Basefold in the List Decoding Regime. https://eprint.iacr.org/2024/1571, 2024.

KPV19. Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from List Polynomial Commitment IOPs. Technical Report 1400, 2019. http://eprint.iacr.org/2019/1400.

KS23. Abhiram Kothapalli and Srinath Setty. HyperNova: Recursive arguments for customizable constraint systems. `https://eprint.iacr.org/2023/573`, 2023.

KST21. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive Zero-Knowledge Arguments from Folding Schemes. https://eprint.iacr.org/2021/370, 2021.

KT23. Tohru Kohrita and Patrick Towa. Zeromorph: Zero-Knowledge Multilinear-Evaluation Proofs from Homomorphic Univariate Commitments, 2023.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *Advances in Cryptology - ASIACRYPT 2010*, volume 6477, pages 177–194. Springer Berlin Heidelberg, 2010.

Lee20. Jonathan Lee. Dory: Efficient, Transparent arguments for Generalised Inner Products and Polynomial Commitments. Technical Report 1274, 2020.

LFKN90. Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. 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*, pages 2–10. IEEE Computer Society, 1990.

Loo22. Team Loopring. Loopring - zkRollup Layer2 for Trading and Payment. https://loopring.org/, 2022.

LXZ21. Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2968–2985. ACM, 2021.

LZZ+25. Tianyi Liu, Zhenfei Zhang, Yuncong Zhang, Wenqing Hu, and Ye Zhang. Ceno: Non-uniform, segment and parallel zero-knowledge virtual machine. *J. Cryptology*, 38(2):17, January 2025. https://doi.org/10.1007/s00145-024-09533-2.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2111–2128. ACM, 2019.

PH23. Shahar Papini and Ulrich Haböck. Improving logarithmic derivative lookups using GKR. https://eprint.iacr.org/2023/1284, 2023.

PHGR13. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, May 2013. http://ieeexplore.ieee.org/document/6547113/.

QSL+25. Wenjie Qu, Yijun Sun, Xuanming Liu, Tao Lu, Yanpei Guo, Kai Chen, and Jiaheng Zhang. zkGPT: An efficient non-interactive zero-knowledge proof framework for LLM inference. https://eprint.iacr.org/2025/1184, 2025.

RZ21. Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In *Annual International Cryptology Conference*, pages 774–804. Springer, 2021.

Scr22. Team Scroll. Scroll. https://scroll.io/, 2022.

Set20. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *Annual International Cryptology Conference*, pages 704–737. Springer, 2020.

SLZ24. Haochen Sun, Jason Li, and Hongyang Zhang. zkLLM: Zero knowledge proofs for large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 4405–4419. ACM, 2024.

SP25. Nitin Singh and Sikhar Patranabis. SubLogarithmic linear time SNARKs from compressed sum-check. https://eprint.iacr.org/2025/908, 2025.

STW23. Srinath Setty, Justin Thaler, and Riad Wahby. Unlocking the lookup singularity with Lasso. https://eprint.iacr.org/2023/1216, 2023.

W24. Wilbert W. Folding-based zkLLM. https://eprint.iacr.org/2024/480, 2024.

WB15. Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, January 2015. https://dl.acm.org/doi/10.1145/2641562.

XZS22. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero Knowledge Proof with Linear Prover Time. *Cryptology ePrint Archive*, 2022.

XZZ+19. Tiacheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation. In *Advances in Cryptology – CRYPTO 2019*, Lecture Notes in Computer Science, pages 733–764. Springer International Publishing, 2019.

ZCF23. Hadas Zeilberger, Binyi Chen, and Ben Fisch. BaseFold: Efficient Field-Agnostic Polynomial Commitment Schemes from Foldable Codes, 2023.
zkS22. Team zkSync. zkSync. https://zksync.io/, 2022.
ZSZ+22. Yuncong Zhang, Alan Szepeniec, Ren Zhang, Shi-Feng Sun, Geng Wang, and Dawu Gu. VOProof: Efficient zkSNARKs from Vector Oracle Compilers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, pages 3195–3208. Association for Computing Machinery, November 2022.
ZXZS20. Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 859–876. IEEE, 2020.

A Formal Definition of FPSP and FAoK

We present the formal definition of FPSP and FAoK, where the differences between the standard FPS are marked by blue color.

Definition 11 (Functional Proof System with Proximity). *Let* $\mathsf{S}_{\mathsf{ins}}$, $\mathsf{S}_{\mathsf{out}}$, $\mathsf{S}_{\mathsf{wit}}$ *be the instance, output, and witness sets, respectively. Let* $\mathsf{S}_{\mathsf{rand}}$ *be the space of the verifier's random tapes. Let* $\mathsf{TarF} : \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \mathsf{S}_{\mathsf{out}}, \mathsf{PredF}, \mathsf{RPredF} : \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{out}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \{0,1\}$ *be three functions such that* $\mathsf{PredF} \leq \mathsf{RPredF}$ *for all inputs. Let the relation induced by* $\mathsf{TarF}, \mathsf{PredF}$ *be defined as*

$$\mathcal{R}_{\mathsf{TarF},\mathsf{PredF}} = \{(x,w) \in \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \mid \forall r \in \mathsf{S}_{\mathsf{rand}}, \mathsf{PredF}(x, \mathsf{TarF}(x,w,r), w, r) = 1\}.$$

A pair of interactive algorithms (P, V) *is a* functional proof system with proximity (FPSP) *for target function* TarF*, predicate function* PredF*, and relaxed predicate function* RPredF *if the following conditions hold:*

- ***Syntax****: The verifier* V *is a deterministic algorithm that takes an instance* $x \in \mathsf{S}_{\mathsf{ins}}$ *and a random tape* $r \in \mathsf{S}_{\mathsf{rand}}$ *as input. After interacting with the prover* P *(who takes* (x,w) *as input), the verifier outputs a value* $y \in \mathsf{S}_{\mathsf{out}}$*. We denote this by* $y \leftarrow \langle \mathsf{P}(x,w), \mathsf{V}(x,r) \rangle$*.*
- ***Completeness****: For any* $(x,w) \in \mathcal{R}_{\mathsf{TarF},\mathsf{PredF}}$*, the verifier's output is the expected one except with probability* $\varepsilon_{\mathsf{cmplt}}$*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} [\langle \mathsf{P}(x,w), \mathsf{V}(x,r) \rangle = \mathsf{TarF}(x,w,r)] = 1 - \varepsilon_{\mathsf{cmplt}},$$

  where $\varepsilon_{\mathsf{cmplt}}$ *is the completeness error.*
- ***Soundness****: For any instance* $x \in \mathsf{S}_{\mathsf{ins}}$*, any (potentially malicious) prover* P^**, and any witness* $w \in \mathsf{S}_{\mathsf{wit}}$ *(which* P^* *might use or ignore), the output* y *produced by the verifier is guaranteed to satisfy the predicate* PredF *except with probability* $\varepsilon_{\mathsf{snd}}$*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} [\mathsf{RPredF}(x,y,w,r) = 1 \mid y \leftarrow \langle \mathsf{P}^*(x,w), \mathsf{V}(x,r) \rangle] \geq 1 - \varepsilon_{\mathsf{snd}},$$

  where $\varepsilon_{\mathsf{snd}}$ *is the soundness error.*

Definition 12 (Functional Argument System, FAS). *Let* $\mathsf{S}_{\mathsf{pp}}, \mathsf{S}_{\mathsf{ins}}, \mathsf{S}_{\mathsf{out}}, \mathsf{S}_{\mathsf{wit}}$ *be the public parameters, instance, output, and witness sets, respectively. Let* $\mathsf{S}_{\mathsf{rand}}$ *be the space of the verifier's random tapes. Let* $\mathsf{TarF} : \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \mathsf{S}_{\mathsf{out}}$ *and* $\mathsf{PredF} : \mathsf{S}_{\mathsf{pp}} \times \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{out}} \times \mathsf{S}_{\mathsf{wit}} \times \mathsf{S}_{\mathsf{rand}} \to \{0,1\}$ *be two functions. Let the relation induced by* $\mathsf{pp} \in \mathsf{S}_{\mathsf{pp}}, \mathsf{TarF}, \mathsf{PredF}$ *be defined as* $\mathcal{R}_{\mathsf{pp},\mathsf{TarF},\mathsf{PredF}} = \{(x,w) \in \mathsf{S}_{\mathsf{ins}} \times \mathsf{S}_{\mathsf{wit}} \mid \forall r \in \mathsf{S}_{\mathsf{rand}}, \mathsf{PredF}(\mathsf{pp}, x, \mathsf{TarF}(x,w,r), w, r) = 1\}$*. A triple of interactive algorithms* $(\mathsf{G}, \mathsf{P}, \mathsf{V})$ *is a* functional argument system (FAS) *for target function* TarF *and predicate function* PredF *if the following conditions hold:*

- ***Syntax****: The setup algorithm* G *receives the security parameter* 1^λ *and outputs a public parameter* $\mathsf{pp} \in \mathsf{S}_{\mathsf{pp}}$*. The verifier* V *is a deterministic algorithm that takes* pp*, an instance* $x \in \mathsf{S}_{\mathsf{ins}}$ *and a random tape* $r \in \mathsf{S}_{\mathsf{rand}}$ *as input. After interacting with the prover* P *(who takes* (pp, x, w) *as input), the verifier outputs a value* $y \in \mathsf{S}_{\mathsf{out}}$*. We denote this by* $y \leftarrow \langle \mathsf{P}(\mathsf{pp}, x, w), \mathsf{V}(\mathsf{pp}, x, r) \rangle$*.*
- ***Completeness****: For any* $(x,w) \in \mathcal{R}_{\mathsf{TarF},\mathsf{PredF}}$*, the verifier's output is the expected one except with probability* $\varepsilon_{\mathsf{cmplt}}$*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} [\langle \mathsf{P}(x,w), \mathsf{V}(x,r) \rangle = \mathsf{TarF}(x,w,r)] = 1 - \varepsilon_{\mathsf{cmplt}},$$

  where $\varepsilon_{\mathsf{cmplt}}$ *is the completeness error.*
- ***Soundness****: For any pair of (potentially malicious) polynomial-time deterministic provers* $(\mathsf{P}_1^*, \mathsf{P}_2^*)$ *and any assistant input* z*:*

$$\Pr_{r \in \mathsf{S}_{\mathsf{rand}}} \left[\mathsf{PredF}(\mathsf{pp}, x, y, w, r) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{G}, (x, w, \mathsf{st}) \leftarrow \mathsf{P}_1^*(\mathsf{pp}, z) \\ y \leftarrow \langle \mathsf{P}_2^*(\mathsf{pp}, x, \mathsf{st}, z), \mathsf{V}(\mathsf{pp}, x, r) \rangle \end{array} \right] \geq 1 - \varepsilon_{\mathsf{snd}},$$

  where $\varepsilon_{\mathsf{snd}}$ *is the soundness error.*

B Proof of Theorem 1

By definition, $(x_0, \cdots, x_{k-1}, w_0, \cdots, w_{k-1}) \in \mathcal{R}_{\mathsf{TarF},\mathsf{PredF}}$ if and only if $(x_i, w_i) \in \mathcal{R}_{\mathsf{TarF}_i,\mathsf{PredF}_i}$ for all i. Completeness follows immediately since the honest composed prover runs each component prover honestly, and the composed verifier outputs each component result.

For soundness, assume some malicious prover P^* makes the predicate fail with probability exceeding $\sum_{i \in [k]} \varepsilon_{\mathsf{snd}}^i$. Then by union bound, some component predicate PredF^{i_0} must fail with probability at least $\varepsilon_{\mathsf{snd}}^{i_0}$. We construct $\mathsf{P}_{i_0}^*$ that breaks soundness of the i_0-th FPS by simulating the full composed protocol: upon receiving challenge $c_j^{i_0}$ in round j, it computes the other challenges $\{c_j^i\}_{i \neq i_0} = \mathsf{ChallengeDerive}(i, j, c_j^{i_0})$ and executes P^* with the complete challenge vector. This perfectly simulates P^* interacting with V, causing PredF^{i_0} to fail with probability at least $\varepsilon_{\mathsf{snd}}^{i_0}$, contradicting the soundness of the i_0-th FPS. □

C Proof of Theorem 2

The analysis relies on the standard sum-check security analysis [LFKN90]. Without considering the efficiency, we assume in the standard sum-check the prover

directly sends the polynomial f to the verifier, they execute the sum-check protocol, and the verifier checks that $h_\mu(r_\mu) = f(\mathbf{r})$. By [LFKN90], this achieves perfect completeness and soundness error $\frac{\mu \cdot m}{|\mathbb{F}|}$ otherwise. We then argue completeness and soundness of the sum-check FPS as follows.

Completeness. For any $(c, f) \in \mathcal{R}_{\mathsf{TarF},\mathsf{PredF}}$, by definition $c = \sum_{\mathbf{b}\in\{0,1\}^\mu} f(\mathbf{b})$, standard sum-check completeness ensures $h_\mu(r_\mu) = f(\mathbf{r})$ with probability 1. Since the output is $y = h_\mu(r_\mu)$, we have $y = \mathsf{TarF}(c, f, \mathbf{r})$ with probability 1.

Soundness. For any (c, f) and any prover P^* interacting with the sum-check verifier, if $c = \sum_{\mathbf{b}\in\{0,1\}^\mu} f(\mathbf{b})$, by definition, $\mathsf{PredF}(c, y, f, \mathbf{r}) = 1$ regardless of $y, \mathbf{r}$, i.e., with probability one. Otherwise, let P^{**} be the prover interacting with the standard sum-check verifier with f sent to the verifier, and executing whatever P^* does. By soundness of the standard sum-check, the verifier rejects with probability at least $1 - \frac{\mu \cdot m}{|\mathbb{F}|}$, which means $y = h_\mu(r_\mu) \neq f(\mathbf{r})$ with probability at least $1 - \frac{\mu \cdot m}{|\mathbb{F}|}$. Soundness follows from the definition of PredF. □

D Proof of Theorem 3

For any sub-probability measure $w : [n] \to [0, 1]$, let

$$\mathsf{agree}_w(\mathbf{v}, \mathbf{c}) := \sum_{j\in[n],\mathbf{v}_{[j]}=\mathbf{c}_{[j]}} w(j)$$

and

$$\mathsf{agree}_w(\mathbf{v}, C) := \max_{\mathbf{c}\in C} \mathsf{agree}_w(\mathbf{v}, \mathbf{c}).$$

We recall the weighted correlated agreement from Theorem 7.2 of [BCI+20] as follows (with parameter $l = 1$):

Theorem 8 ([BCI+20], Theorem 7.2, Weighted correlated agreement). *Let $C = \mathsf{RS}[\mathbb{F}, L, \rho]$ and choose $\theta = 1 - \sqrt{\rho} \cdot (1 + 1/2m)$, for some integer $m \geq 3$. Assume a density function $\delta : [|L|] \to [0, 1] \cup \mathbb{Q}$ with common denominator $B \geq 1$, i.e., for all $j \in [|L|]$, $\delta(j) = \frac{m_x}{B}$, for an integer value $m_j \in [0, B]$, and let w be the sub-probability measure with density δ, defined by $w(j) = \delta(j)/|L|$. If for $\mathbf{v}_0, \mathbf{v}_1 \in \mathbb{F}^{|L|}$,*

$$\frac{1}{|\mathbb{F}|} \cdot |\{z \in \mathbb{F} : \mathsf{agree}_w((1 - z) \cdot \mathbf{v}_0 + z \cdot \mathbf{v}_1, C') \geq 1 - \theta\}| > \varepsilon(|L|, B, \theta),$$

where ε is the same as in Theorem 3, then there exist codewords $\mathbf{c}_0, \mathbf{c}_1 \in C$, and a set A with $w(A) \geq 1 - \theta$ on which $\mathbf{v}_0, \mathbf{v}_1$ coincide with $\mathbf{c}_0, \mathbf{c}_1$, respectively.

Completeness. For honest execution with witness $\mathbf{v}$, the definition of Fold ensures that the final message represents the correct evaluation $\hat{\mathbf{v}}(\mathbf{r})$.

Soundness. Let $\mathsf{Neighbor}(\mathbf{c}^{(0)})$ denote the set $\{\mathbf{c} \in C : \Delta(\mathbf{c}, \mathbf{c}^{(0)}) \leq \theta \cdot n^{(0)}\}$. For any sequence of $(k+1) \leq (\mu+1)$ vectors $(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k)})$ of the appropriate lengths, i.e., $\mathbf{c}^{(i)}$ has length $n^{(i)}$, and for $\alpha \in (0,1)$, we say this sequence is *α-good* w.r.t. the randomness $\mathbf{r} \in \mathbb{F}^k$ if there exists a valid codeword $\mathbf{h}$ of size $n^{(k)}$ such that

$$\mathbf{h} \notin \{\mathsf{Fold}(\mathbf{c}, r_1, \cdots, r_k) : \mathbf{c} \in \mathsf{Neighbor}(\mathbf{c}^{(0)})\}$$

and

$$\left|\left\{ j \in [n^{(0)}] : \begin{array}{c}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k)}) \text{ satisfies the query check at } j \\ \wedge \mathbf{c}^{(k)}_{[j \bmod n^{(k)}]} = \mathbf{h}_{[j \bmod n^{(k)}]}\end{array} \right\}\right| \geq \alpha \cdot n^{(0)}.$$

We prove the following lemma by induction.

Lemma 1. *Let $\theta = 1-(1+1/2m)\cdot\sqrt{\rho}$ and $\alpha = 1-\theta$. For any unbounded prover P^*, the probability that P^* produces an α-good sequence at the end of round $k \geq 0$ is bounded by*

$$\varepsilon_{\mathsf{cmplt}} = \varepsilon_1 + \cdots + \varepsilon_r,$$

where $\varepsilon_i = \varepsilon(n^{(i)}, 2^i, \theta) + 1/|\mathbb{F}|$.

Proof. We prove the lemma by induction on k. For the base case $k = 0$, note that for any $\mathbf{c}^{(0)}$ of size $n^{(0)}$, $\mathbf{c}^{(0)}$ alone cannot form an α-good sequence, because for any $\mathbf{h}$ of size $n^{(0)}$, the first condition of α-goodness requires that $\mathbf{c}^{(0)}$ agrees with $\mathbf{h}$ in at least $\alpha \cdot n^{(0)}$ positions, while the second condition requires the opposite.

Assume that the lemma holds for $k-1$ for $1 \leq k \leq n$. Assume for contradiction that a prover P^* produces an α-good sequence at the end of round k with probability larger than $(\varepsilon_1 + \cdots + \varepsilon_{k-1}) + \varepsilon_r$. For any $(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k-1)})$ and $r_1, \cdots, r_{k-1}$, let

$$\Omega_{r_1,\cdots,r_{k-1}}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k-1)}) := \Pr_{r_k}\left[\begin{array}{c}\exists \mathbf{c}^{(k)} \text{ such that } (\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k)}) \\ \text{is } \alpha\text{-good for } (r_1, \cdots, r_k)\end{array}\right].$$

Let A be the event that $\Omega_{r_1,\cdots,r_{k-1}}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k-1)}) \leq \varepsilon_k$ and $\bar{A}$ be the event that $\Omega_{r_1,\cdots,r_{k-1}}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k-1)}) > \varepsilon_k$. Then

$$\begin{aligned}(\varepsilon_1 + \cdots + \varepsilon_{k-1}) + \varepsilon_k &< \Pr\left[\begin{array}{c}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k)}) \\ \text{is } \alpha\text{-good for } (r_1, \cdots, r_k)\end{array}\right] \\ &= \Pr\left[\begin{array}{c}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k)}) \\ \text{is } \alpha\text{-good for } (r_1, \cdots, r_k)\end{array}\middle| A\right] \cdot \Pr[A] \\ &+ \Pr\left[\begin{array}{c}(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(k)}) \\ \text{is } \alpha\text{-good for } (r_1, \cdots, r_k)\end{array}\middle| \bar{A}\right] \cdot \Pr[\bar{A}] \\ &\leq \varepsilon_k + \Pr[\bar{A}].\end{aligned}$$

Therefore, $\Pr[\bar{A}] > \varepsilon_1 + \cdots + \varepsilon_{k-1}$, i.e., with larger than this probability, at the end of round $k-1$, the probability that the next verifier challenge r_k leads to a state where the prover is able to produce an α-good sequence is greater than ε_k.

By definition of α-goodness, for each of these r_k, there exists a valid codeword $\mathbf{h}$ such that $\mathbf{h} \notin \{\mathsf{Fold}(\mathbf{c}, r_1, \cdots, r_k) : \mathbf{c} \in \mathsf{Neighbor}(\mathbf{c}^{(0)})\}$ and the folding of $\mathbf{c}^{(k-1)}$ by r_k is close to $\mathbf{h}$ in the weighted sense, i.e.,

$$\mathsf{agree}_{w_k}((1-r_k)\cdot(\mathbf{c}_{[\mathsf{left}]}+\mathbf{c}_{[\mathsf{right}]})^{(k-1)}\circ\mathbf{t}_+^{(k)}+r_k\cdot(\mathbf{c}_{[\mathsf{left}]}^{(k-1)}-\mathbf{c}_{[\mathsf{right}]}^{(k-1)})\circ\mathbf{t}_-^{(k)},\mathbf{h})\geq\alpha$$

where w_k is the sub-probability measure over $[n^{(k)}]$ with density function:

$$w_k(j) := \frac{\left|\left\{\begin{array}{c}\ell\in[n^{(0)}]:\ell\equiv j \bmod n^{(k)}\wedge\\ (\mathbf{c}^{(0)},\cdots,\mathbf{c}^{(k-1)})\text{ satisfies the query check at }\ell\end{array}\right\}\right|}{2^k}.$$

By the weighted correlated agreement, there exists a subset B such that $w_k(B) \geq \alpha$ and $\mathbf{h}_{\mathsf{left}}, \mathbf{h}_{\mathsf{right}} \in C^{(k-1)}$ such that $(\mathbf{c}_{[\mathsf{left}]}^{(k-1)}+\mathbf{c}_{[\mathsf{right}]}^{(k-1)})\circ\mathbf{t}_+^{(k)}, (\mathbf{c}_{[\mathsf{left}]}^{(k-1)}-\mathbf{c}_{[\mathsf{right}]}^{(k-1)})\circ\mathbf{t}_-^{(k)}$ coincide with $\mathbf{h}_{\mathsf{left}}, \mathbf{h}_{\mathsf{right}}$ over B, respectively. Let $\mathbf{h}^{(k-1)} := \frac{1}{2}\cdot((\mathbf{h}_{\mathsf{left}}/\mathbf{t}_+^{(k)} + \mathbf{h}_{\mathsf{right}}/\mathbf{t}_-^{(k)})\|(-(\mathbf{h}_{\mathsf{left}}/\mathbf{t}_+^{(k)} - \mathbf{h}_{\mathsf{right}}/\mathbf{t}_-^{(k)})))$ and $B' = B \cup \{j + n^{(k)} : j \in B\}$. By the property of foldable code family introduced in Section 2.7, $\mathbf{c}^{(k-1)}$ coincides with $\mathbf{h}^{(k-1)}$ over B'. By definition of w_k and agree_{w_k}, we have

$$\left|\left\{\begin{array}{c}j \bmod n^{(k-1)}\in B'\wedge(\mathbf{c}^{(0)},\cdots,\mathbf{c}^{(k-1)})\\ \text{satisfies the query check at } j\end{array}\right\}\right| = 2^k\cdot\sum_{j\in B}w_k(j) = n^{(0)}\cdot w_{k-1}(B)$$

which is at least $\alpha \cdot n^{(0)}$. Note that $\mathbf{h}^{(k-1)} \notin \{\mathsf{Fold}(\mathbf{c}, r_1, \cdots, r_{k-1}) : \mathbf{c} \in \mathsf{Neighbor}(\mathbf{c}^{(0)})\}$, otherwise $\mathbf{h} = \mathsf{Fold}(\mathbf{h}^{(k-1)}, r_k)$ will contradict the assumption. By definition of α-goodness, we conclude that with probability greater than $\varepsilon_1 + \cdots + \varepsilon_{k-1}$, $(\mathbf{c}^{(0)}, \mathbf{c}^{(1)}, \cdots, \mathbf{c}^{(k-1)})$ is α-good for $r_1, \cdots, r_{k-1}$, contradicting the assumption for $k-1$. □

Using the above lemma, we argue soundness as follows.

Case (a). For any $\mathbf{c}^{(0)}$ that is within the distance $\delta \cdot n^{(0)}$, we show that y is either $\hat{\mathbf{v}}'(\mathbf{r})$ for some decoding $\mathbf{v}'$ of $\mathbf{c}^{(0)}$ or $\perp$. By the lemma, for any unbounded prover P^*, the final prover message sequence $(\mathbf{c}^{(0)}, \mathbf{c}^{(1)}, \cdots, \mathbf{c}^{(\mu)})$ is either α-good for $r_1, \cdots, r_\mu$ with probability at most $\varepsilon_{\mathsf{cmplt}} = \varepsilon_1 + \cdots + \varepsilon_\mu$, or not α-good, which means either:

- The final message $\mathbf{c}^{(\mu)}$ is not a codeword in $C^{(\mu)}$, in which case $y = \perp$.
- The final message $\mathbf{c}^{(\mu)}$ is a codeword of the form $\mathsf{Fold}(\mathbf{c}, r_1, \cdots, r_\mu)$ for some $\mathbf{c} \in \mathsf{Neighbor}(\mathbf{c}^{(0)})$ with $\mathbf{c} = \mathsf{Encode}(\mathbf{v}')$ for some $\mathbf{v}'$, in this case $y = \hat{\mathbf{v}}'(\mathbf{r})$.
- The density of j such that $(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(\mu)})$ satisfies the query check at j is smaller than $\alpha = 1 - \theta$. In this case, $y = \perp$ with probability at least $1 - (1-\theta)^\ell$.

By union bound, the probability that the predicate RPredF returns 1 with probability at least $1 - \varepsilon_{\mathsf{cmplt}} - (1-\theta)^\ell$.

Case (b). For any $\mathbf{c}^{(0)}$ that is not within the distance $\delta \cdot n^{(0)}$, we show that $y = \perp$. By the lemma, for any unbounded prover P^*, the final prover message sequence $(\mathbf{c}^{(0)}, \mathbf{c}^{(1)}, \cdots, \mathbf{c}^{(\mu)})$ is either α-good for $r_1, \cdots, r_\mu$ with probability at most $\varepsilon_1 + \cdots + \varepsilon_\mu$, or not α-good, which implies that either:

– The final message $\mathbf{c}^{(\mu)}$ is not a codeword, then $y = \perp$.
– The final message $\mathbf{c}^{(\mu)}$ is a codeword, but since $\mathsf{Neighbor}(\mathbf{c}^{(0)}) = \emptyset$, the density of j such that $(\mathbf{c}^{(0)}, \cdots, \mathbf{c}^{(\mu)})$ satisfies the query check at j must be smaller than $\alpha = 1 - \theta$. Then $y = \perp$ with probability at least $1 - (1-\theta)^{\ell}$.

By union bound, the probability that the predicate RPredF returns 1 with probability at least $1 - \varepsilon_{\mathsf{cmplt}} - (1-\theta)^{\ell}$. □

E Proof of Theorem 4

Perfect completeness follows from that of sum-check as FPS (Theorem 2), $\mathsf{FriEval}$ (Theorem 3), and Theorem 1, since both component protocols are public-coin and can share randomness, satisfying the composition conditions. Perfect binding follows from the injectivity of the encoding function $\mathsf{Encode}_{C^{(0)}}$.

For argument of knowledge, by soundness of sum-check as FPS, soundness of $\mathsf{FriEval}$, and Theorem 1, except with probability $\varepsilon_{\mathsf{snd}}^{fri} + \varepsilon_{\mathsf{snd}}^{sum}$, the outputs y_{sum} and y_{fri} satisfy:

$$\left(\forall \mathbf{v} \in \mathbb{F}^D, \sum_{\mathbf{b} \in \{0,1\}^{\mu}} \hat{\mathbf{v}}(\mathbf{b}) \cdot eq(\mathbf{z}, \mathbf{b}) = y \vee y_{sum} \neq \hat{\mathbf{v}}(\mathbf{r}) \cdot eq(\mathbf{z}, \mathbf{r})\right) \wedge$$
$$(y_{fri} = \perp \vee (\exists \mathbf{v} \in \mathbb{F}^D, \hat{\mathbf{v}}(\mathbf{r}) = y_{fri} \wedge \Delta(\mathsf{Encode}_{C^{(0)}}(\mathbf{v}), \mathbf{c}) < \delta \cdot D),$$

where $\delta = 1 - \sqrt{\rho} \cdot (1 + 1/2m)$ for some integer $m \geq 3$. By additionally asserting that $y_{sum} = y_{fri} \cdot eq(\mathbf{z}, \mathbf{r})$ and $y_{fri} \neq \perp$, the above conditions implies that

$$\exists \mathbf{v} \in \mathbb{F}^D, \hat{\mathbf{v}}(\mathbf{r}) = y_{fri} \wedge \Delta(\mathsf{Encode}_{C^{(0)}}(\mathbf{v}), \mathbf{c}) < \delta \cdot D) \wedge$$
$$\left(\sum_{\mathbf{b} \in \{0,1\}^{\mu}} \hat{\mathbf{v}}(\mathbf{b}) \cdot eq(\mathbf{z}, \mathbf{b}) = y \vee y_{fri} \cdot eq(\mathbf{z}, \mathbf{r}) \neq \hat{\mathbf{v}}(\mathbf{r}) \cdot eq(\mathbf{z}, \mathbf{r})\right),$$

which then implies

$$\exists \mathbf{v} \in \mathbb{F}^D, \Delta(\mathsf{Encode}_{C^{(0)}}(\mathbf{v}), \mathbf{c}) < \delta \cdot D \wedge \hat{\mathbf{v}}(\mathbf{z}) = y.$$

The binding property ensures that this $\mathbf{v}$ is exactly the evaluation of the committed polynomial f over $\{0,1\}^{\mu}$, so $f(\mathbf{z}) = \hat{\mathbf{v}}(\mathbf{z}) = y$. The extractor simply outputs the decoding of $\mathbf{c}$. □

F Proof of Theorem 5

Completeness holds due to the perfect completeness of the underlying sum-check protocol. For soundness, we first consider the case where the verifier rejects with high probability. If the output y is $\perp$ with probability at least $1 - \mathsf{negl}(\lambda)$, the

soundness condition holds trivially. We therefore assume the verifier accepts with non-negligible probability ε_0.

We construct an extractor that, given access to the cheating prover's internal state, can extract a valid witness. The extractor simulates the interaction between the prover and the verifier. By forking the execution at each round, the extractor builds a tree of accepting transcripts. The edges of this tree are labeled with the verifier's challenges. Since we assume the prover succeeds with non-negligible probability, this tree can be constructed in expected polynomial time via the standard rewind-and-fork technique.

The transcript tree is structured as follows. The root contains the initial prover message, a degree-2 polynomial, which we denote as $H_\emptyset(X)$, where the subscript $\emptyset$ stands for the path from root to this node (also the root). Each non-leaf node has four children, with the edges labeled by the four challenges r_1, r_2, r_3, r_4 sent by the verifier in this round. A path from the root to any node is labeled by the sequence of challenges $\mathbf{r}$ along this path. A non-leaf node reached via path $\mathbf{r}$ contains the polynomial $H_{\mathbf{r}}(X)$, while a leaf node contains the final value $y_{\mathbf{r}}$ output by the verifier. Each node indexed by $\mathbf{r}$ is also associated with $Y_{\mathbf{r}} \in \mathbb{G}$, computed from its parent's message and the corresponding challenge.

By definition of verifier acceptance in Protocol 6, a tree of accepting transcript satisfies two consistency conditions. First, for each leaf node indexed by $\mathbf{r}$, the final value $y_{\mathbf{r}}$ satisfies the check $\hat{\mathbf{G}}(\mathbf{r}) \cdot y_{\mathbf{r}} = Y_{\mathbf{r}}$. Second, for every non-leaf node, $H_{\mathbf{r}}(X)$ satisfies the sum-check verification $H_{\mathbf{r}}(0) + H_{\mathbf{r}}(1) = Y_{\mathbf{r}}$.

Consider a node at depth $\mu - 1$, corresponding to a challenge vector $\mathbf{r}$ of length $\mu - 1$. Let $f_{\mathbf{r}}(X)$ be the unique polynomial of degree at most 3 that interpolates the points $(r_j, y_{(\mathbf{r}, r_j)})$ for $j \in \{1, 2, 3, 4\}$. From the previous step, we have $H_{\mathbf{r}}(r_j) = \hat{\mathbf{G}}(\mathbf{r}, r_j) \cdot f_{\mathbf{r}}(r_j)$ for each j. By the definition of multilinear extension, $\hat{\mathbf{G}}(\mathbf{r}, X) = (1 - X)\hat{\mathbf{G}}(\mathbf{r}, 0) + X\hat{\mathbf{G}}(\mathbf{r}, 1)$. Therefore, we can write:

$$H_{\mathbf{r}}(r_j) = \Big((1 - r_j)\hat{\mathbf{G}}(\mathbf{r}, 0) + r_j\hat{\mathbf{G}}(\mathbf{r}, 1)\Big) \cdot f_{\mathbf{r}}(r_j) \quad \text{for } j \in \{1, 2, 3, 4\}. \tag{1}$$

Let us define two auxiliary polynomials. Let $g_{\mathbf{r}}(X)$ be the unique polynomial of degree at most 2 such that $g_{\mathbf{r}}(r_j) = (1-r_j)f_{\mathbf{r}}(r_j)$ for $j \in \{1, 2, 3\}$. Similarly, let $g'_{\mathbf{r}}(X)$ be the unique polynomial of degree at most 2 such that $g'_{\mathbf{r}}(r_j) = r_j f_{\mathbf{r}}(r_j)$ for $j \in \{1, 2, 3\}$. Recall that $H_{\mathbf{r}}(X)$ is a polynomial of degree at most 2. For $j \in \{1, 2, 3\}$, we have

$$H_{\mathbf{r}}(r_j) = g_{\mathbf{r}}(r_j)\hat{\mathbf{G}}(\mathbf{r}, 0) + g'_{\mathbf{r}}(r_j)\hat{\mathbf{G}}(\mathbf{r}, 1).$$

Since this equality holds for three distinct points and both sides are polynomials of degree at most 2, the following polynomial identity must hold:

$$H_{\mathbf{r}}(X) = g_{\mathbf{r}}(X)\hat{\mathbf{G}}(\mathbf{r}, 0) + g'_{\mathbf{r}}(X)\hat{\mathbf{G}}(\mathbf{r}, 1).$$

By evaluating this identity at r_4, we obtain the group value equality:

$$H_{\mathbf{r}}(r_4) = g_{\mathbf{r}}(r_4)\hat{\mathbf{G}}(\mathbf{r}, 0) + g'_{\mathbf{r}}(r_4)\hat{\mathbf{G}}(\mathbf{r}, 1).$$

We now have two expressions for $H_{\mathbf{r}}(r_4)$. Combining them, we get:

$$g_{\mathbf{r}}(r_4)\hat{\mathbf{G}}(\mathbf{r},0) + g'_{\mathbf{r}}(r_4)\hat{\mathbf{G}}(\mathbf{r},1) = (1-r_4)f_{\mathbf{r}}(r_4)\hat{\mathbf{G}}(\mathbf{r},0) + r_4 f_{\mathbf{r}}(r_4)\hat{\mathbf{G}}(\mathbf{r},1).$$

This can be rewritten as:

$$(g_{\mathbf{r}}(r_4) - (1-r_4)f_{\mathbf{r}}(r_4))\hat{\mathbf{G}}(\mathbf{r},0) + (g'_{\mathbf{r}}(r_4) - r_4 f_{\mathbf{r}}(r_4))\hat{\mathbf{G}}(\mathbf{r},1) = \mathbf{0}.$$

By the discrete logarithm assumption, if the coefficients are non-zero, we have found a collision. Therefore, with overwhelming probability, we must have:

$$g_{\mathbf{r}}(r_4) = (1-r_4)f_{\mathbf{r}}(r_4) \quad \text{and} \quad g'_{\mathbf{r}}(r_4) = r_4 f_{\mathbf{r}}(r_4).$$

This establishes $g_{\mathbf{r}}(r_j) = (1-r_j)f_{\mathbf{r}}(r_j)$ and $g'_{\mathbf{r}}(r_j) = r_j f_{\mathbf{r}}(r_j)$ for all $j \in \{1,2,3,4\}$. Summing these two equations gives $g_{\mathbf{r}}(r_j) + g'_{\mathbf{r}}(r_j) = f_{\mathbf{r}}(r_j)$ for all j. Since $g_{\mathbf{r}}(X)$ and $g'_{\mathbf{r}}(X)$ are polynomials of degree at most 2, their sum is also a polynomial of degree at most 2. As $f_{\mathbf{r}}(X)$ is a degree-3 polynomial that agrees with a degree-2 polynomial on four points, it must be identical to that polynomial, and thus its degree is at most 2.

Now consider the relation $g_{\mathbf{r}}(r_j) = (1-r_j)f_{\mathbf{r}}(r_j)$, which holds for four points. The polynomial $g_{\mathbf{r}}(X)$ has degree at most 2, and since we have just shown that $f_{\mathbf{r}}(X)$ has degree at most 2, the polynomial $(1-X)f_{\mathbf{r}}(X)$ has degree at most 3. As these two degree-3 polynomials agree on four points, they must be identical, so $g_{\mathbf{r}}(X) = (1-X)f_{\mathbf{r}}(X)$, implying that the degree of $f_{\mathbf{r}}(X)$ must be at most 1.

We now return to the equation (1). As this equality between two degree-2 group-valued polynomials holds for four points, it must be a polynomial identity:

$$H_{\mathbf{r}}(X) = (1-X)f_{\mathbf{r}}(X)\hat{\mathbf{G}}(\mathbf{r},0) + X f_{\mathbf{r}}(X)\hat{\mathbf{G}}(\mathbf{r},1).$$

From the sum-check condition, we have $Y_{\mathbf{r}} = H_{\mathbf{r}}(0) + H_{\mathbf{r}}(1)$. Evaluating the identity at $X=0$ and $X=1$ and summing the results, we get:

$$Y_{\mathbf{r}} = f_{\mathbf{r}}(0)\hat{\mathbf{G}}(\mathbf{r},0) + f_{\mathbf{r}}(1)\hat{\mathbf{G}}(\mathbf{r},1).$$

We now generalize this argument by induction. We claim that for every non-leaf node at depth k (with challenge vector $\mathbf{r}$ of length k), there exists a $(\mu-k)$-variate multilinear polynomial $f_{\mathbf{r}}$ such that:

$$Y_{\mathbf{r}} = \sum_{\mathbf{b}\in\{0,1\}^{\mu-k}} f_{\mathbf{r}}(\mathbf{b}) \cdot \hat{\mathbf{G}}(\mathbf{r},\mathbf{b}).$$

Base case: For $k = \mu-1$, we have shown that $Y_{\mathbf{r}} = f_{\mathbf{r}}(0)\hat{\mathbf{G}}(\mathbf{r},0) + f_{\mathbf{r}}(1)\hat{\mathbf{G}}(\mathbf{r},1)$, with $f_{\mathbf{r}}$ being a 1-variable multilinear polynomial.

Inductive step: Assume the claim holds for any node at depth $k+1$. Let $\mathbf{r}$ be a challenge vector of length k. The same analysis that we applied to the leaves of the tree can be applied here. Instead of a scalar value $y_{(\mathbf{r},r_j)}$, we now have a multilinear polynomial $f_{(\mathbf{r},r_j)}$. The final check equation is replaced by the claimed inner product relation. The argument proceeds in the same way, allowing us to

construct the desired $(\mu - k)$-variate polynomial $f_{\mathbf{r}}$ from the $(\mu - k - 1)$-variate polynomials $f_{(\mathbf{r},r_j)}$.

Applying the claim at the root of the tree (for $k = 0$ and $\mathbf{r} = \emptyset$), we obtain a μ-variate multilinear polynomial f such that $C = \sum_{\mathbf{b}\in\{0,1\}^\mu} f(\mathbf{b}) \cdot \hat{\mathbf{G}}(\mathbf{b})$. Let $\mathbf{v} = f(\{0,1\}^\mu)$. Then the relation becomes $C = \langle \mathbf{v}, \mathbf{G} \rangle$, which means we have extracted a valid witness vector $\mathbf{v}$. Furthermore, the construction of f ensures that for any path $\mathbf{r}$ to a leaf, the evaluation $f(\mathbf{r})$ coincides with the value $y_{\mathbf{r}}$.

In summary, the extraction procedure guarantees that any cheating prover must either provide an accepting transcript from which a valid witness can be extracted, or reveal a discrete logarithm collision. As the latter happens with negligible probability, the protocol is sound. □

G Proof of Theorem 6

Perfect completeness follows directly from the perfect completeness of the sum-check FPS (Theorem 2) and the BulletEval protocol (Theorem 5), and the composition theorem (Theorem 1).

For knowledge soundness, we apply the composition theorem (Theorem 1). For any polynomial-time prover P^*, there exists an extractor that, from an accepting transcript, outputs a witness $\mathbf{v}$ such that, except with probability $\varepsilon_{\mathsf{snd}}^{eval} + \varepsilon_{\mathsf{snd}}^{sum}$, the following conditions hold:

(1) From the soundness of the sum-check FPS (Theorem 2), we have:

$$\left(\sum_{\mathbf{b}\in\{0,1\}^\mu} \hat{\mathbf{v}}(\mathbf{b}) \cdot \hat{\mathbf{u}}(\mathbf{b}) = t \right) \vee (y_{sum} \neq \hat{\mathbf{v}}(\mathbf{r}) \cdot \hat{\mathbf{u}}(\mathbf{r}))$$

(2) From the knowledge soundness of BulletEval (Theorem 5), we have:

$$(y_{eval} = \bot) \vee (\hat{\mathbf{v}}(\mathbf{r}) = y_{eval} \wedge C = \langle \mathbf{G}, \mathbf{v} \rangle)$$

The verifier accepts only if $y_{eval} \neq \bot$ and $y_{sum} = y_{eval} \cdot \hat{\mathbf{u}}(\mathbf{r})$. When this is the case, the first part of condition (2) is false, so $\hat{\mathbf{v}}(\mathbf{r}) = y_{eval}$ and $C = \langle \mathbf{G}, \mathbf{v} \rangle$.

Substituting $y_{eval} = \hat{\mathbf{v}}(\mathbf{r})$ into the verifier's check gives $y_{sum} = \hat{\mathbf{v}}(\mathbf{r}) \cdot \hat{\mathbf{u}}(\mathbf{r})$. This makes the second part of condition (1) false. Therefore, the first part of condition (1) must be true: $\sum_{\mathbf{b}\in\{0,1\}^\mu} \hat{\mathbf{v}}(\mathbf{b}) \cdot \hat{\mathbf{u}}(\mathbf{b}) = t$, implying $\langle \mathbf{v}, \mathbf{u} \rangle = t$.

Thus, for any accepting transcript, the extractor outputs a vector $\mathbf{v}$ that satisfies both $C = \langle \mathbf{G}, \mathbf{v} \rangle$ and $\langle \mathbf{v}, \mathbf{u} \rangle = t$. □