Efficient SNARKs for Boolean Circuits via Sumcheck over Tower Fields

Tianyi Liu and Yupeng Zhang

University of Illinois Urbana-Champaign, Champaign IL 61820, USA
{tianyi28, zhangyp}@illinois.edu

Abstract. In this paper, we present efficient SNARKs for Boolean circuits, achieving significant improvements in the prover efficiency. The core of our technique is a novel tower sumcheck protocol and a tower zero-check protocol tailored for tower fields, which enable this efficiency boost. When instantiated with Wiedemann's binary tower fields with the base field of $GF(2)$ and the top-level field $GF(2^{2^\ell})$, assuming the quadratic complexity of multiplications $O(2^{2\ell})$ in the top-level field with 2^ℓ bits, the prover time of our sumcheck protocol is $O(2^{1.5\ell}N)$. It is faster than the standard sumcheck protocol over the large field with the complexity of $O(2^{2\ell}N)$. To achieve a reasonable security level, 2^ℓ is usually set to 128.

Leveraging this advancement, we improve the efficiency of IOP protocols over the binary or small characteristic fields for Plonkish, CCS, and GKR-based constraint systems. Moreover, to further improve the prover efficiency of the SNARKs, we introduce a basis-switching mechanism that efficiently transforms polynomial evaluations on the base-field polynomial to evaluations on the tower-field polynomial. With the basis-switching, we are able to compile the binary-field IOPs to SNARKs using large-field polynomial commitment schemes (PCS) that batch the witness over the base field. The size of the large-field PCS is only $\frac{1}{2^\ell}$ of the size of the witness over the base field. Combining the IOP and the PCS, the overall prover time of our SNARKs for Boolean circuits significantly faster than the naive approach of encoding Boolean values in a large field.

1 Introduction

Succinct Non-interactive Arguments of Knowledge (SNARKs) enable a prover to generate a short proof to convince a verifier that a statement on the prover's witness is true. Theoretical constructions for all NP statements were proposed by Kilian [29] and Micali [32] utilizing probabilistically checkable proofs. In recent years, there have been numerous constructions of modern SNARKs with good efficiency in practice.

A widely-used approach is compiling Interactive Oracle Proofs (IOPs) with Polynomial Commitment Schemes (PCS), as formalized by Bünz et al. [13]. Early constructions often combine IOPs with PCS based on Elliptic Curves (EC) due

to their small proof size, as proposed in papers such as [22, 25, 33]. However, secure elliptic curves are typically constructed on large prime fields, which in turn serve as the scalar field in the associated IOP protocols. For example, the order of the BN254 curve is a prime of 254 bits, making scalar-field operations computationally expensive. This overhead makes EC-based SNARKs impractical for large-scale applications.

Recent advancements in polynomial commitment protocols, such as FRI [8] and Brakedown [24], have significantly improved the efficiency by leveraging *error-correcting codes* over small-characteristic fields. Extending this approach further, some schemes construct PCS directly over characteristic-2. Notably, Binius [19] and FRI-Binius [20] demonstrate how to pack multiple bits into a large field, and only commit to a polynomial over the large field with the same number of bits as the witness in the binary field, which improves the efficiency of the PCS. Their protocols have shown exceptional performance in proving Boolean circuits, including applications such as the Keccak hashing circuit.

Despite these advancements, a key remaining challenge is to develop an efficient IOP protocol over small-characteristic fields. In particular, most circuits are defined over a base field, while the challenges are sampled from a significantly larger extension field to ensure the soundness property. This discrepancy introduces substantial computational overhead. For example, in Binius, although the circuit is defined over $GF(2)$, the IOP protocol has to be executed over $GF(2^{128})$ to sample the random challenges. The witness over $GF(2)$ is packed only in the PCS, but not the IOP protocol. This limitation significantly undermines the performance benefits of the packing technique used in the PCS, ultimately limiting the efficiency of the entire SNARK.

1.1 Our Contributions

In this paper, we propose several techniques to address the issue above:

Efficient Sumcheck and Zerocheck Protocols over Tower Fields. We introduce an efficient sumcheck protocol, referred to as the *tower sumcheck protocol*, which is defined over arbitrary tower fields where the extension degree is a prime power. This protocol is specifically designed to embed prover messages within the tower field and to utilize its hierarchical structure to reduce the prover's cost. Furthermore, we propose an optimized zerocheck protocol, named the tower zerocheck protocol, which builds upon the sumcheck framework while incorporating zerocheck-specific optimizations to further enhance efficiency. For the tower field with $GF(2)$ as the base field and extension degree 2, assuming the cost of a multiplication in the i-th level of the tower field is $\mathcal{M}_i = O(2^{2i})$, our protocol has a complexity of $O(2^{1.5\ell}N)$, whereas the previous protocol incurs a complexity of $O(2^{2\ell}N)$.

Efficient IOPs over Small-Characteristic Fields for Plonkish, CCS, GKR Arithmetizations. Using our tower sumcheck and zerocheck protocols, we design efficient IOP protocols over small-characteristic fields for several

widely used arithmetizations, including Plonkish [22], customizable constraint systems (CCS) [39], and GKR [23]. We achieve similar improvements on the prover time compared the baseline approach of embedding the elements of small-characteristic fields naively in large fields in these schemes.

Efficient SNARKs for Boolean Circuits. Finally, to construct SNARKs over small-characteristic fields, e.g., Boolean circuits, we propose an efficient basis-switching technique, which enables us to encode the witnesses into a polynomial defined over the large field with the same number of bits as the witness and run a PCS over the large field. Suppose the base-field witness vector has size 2^n and the polynomial commitment scheme uses a field $\mathcal{T}_{\mathsf{pcs}}$ with bit width $2^{\ell_{\mathsf{pcs}}}$. Compared with the naive solution that commits a polynomial embedding each bit into an extension field $\mathcal{T}_{\mathsf{pcs}}$, our approach only needs to commit a polynomial of size $\frac{2^n}{2^{\ell_{\mathsf{pcs}}}}$. Although FRI-Binius proposes a ring-switching technique that achieves a similar size reduction, our method offers greater efficiency for the prover. Specifically, our method only incurs a constant overhead on the prover, while the FRI-Binius reduction needs $O\left(2^{n-\ell_{\mathsf{pcs}}} \cdot \mathsf{poly}(\lambda, n)\right)$ prover time. With our basis-switching technique and an efficient PCS over the large field, we can compile these IOP protocols into highly efficient SNARKs.

1.2 Related Work

SNARK over Large Fields. There are many SNARK constructions over fields of large prime characteristics based on different cryptographic primitives, such as [12, 22, 25, 30, 33, 44]. We refer the readers to [43] for a survey on modern SNARK constructions.

Sumcheck and Zerocheck. The sumcheck protocol, introduced by Lund et al. [31], is a fundamental tool in interactive proofs and the building block of many SNARKs. Algorithms with linear prover time for sumcheck over the product of multilinear extensions were proposed in [42, 46]. Setty [38] presented a SNARK protocol based on sumcheck to prove R1CS instances. It also proposed zerocheck, which was formalized by Chen et al. [14] as the zerocheck IOP. As shown by Chen et al. [14], zerocheck IOP is important when constructing a SNARK scheme based on Plonkish arguments.

SNARKs over Small-Characteristic Fields. Small-characteristic fields, particularly binary fields, have recently gained attention for their compact representation and high performance, which align well with practical circuit implementations. Thanks to the polynomial commitment scheme based on error-correcting code [2–4, 8, 11, 24, 47–49], there has been a list of SNARK protocols based on small-characteristic field, such as [9,27] and practical implementations [34,35,41].

Ron-Zewi et al. [36] and Holmgren et al. [28] used tensor code and code-switching to construct the binary-field multi-sumcheck protocol, and then a suc-

cinct argument of circuit satisfiability problem. Although they achieved linear-time prover and sublinear-time verifier with a private linear preprocessing step, the soundness error is an arbitrarily small constant, not negligible.

Diamond et al. [19, 20] introduced constructions of polynomials defined on the tower fields with characteristic 2. By packing multiple bits into extension field elements, these works achieved significant improvements in prover efficiency. Considering the circuit is still written over $GF(2)$, however, they need to convert a base-field polynomial evaluation to an evaluation of the large-field polynomial. Soukhanov et al. [40] proposed an approach that exploits Frobenius mapping to construct the conversion. All of the approaches above leverage large-field PCS to commit base-field witness represented by the same number of bits, however, all these protocols propose seldom techniques to accelerate the IOP prover cost. When instantiated with the sumcheck protocol, even though the witnesses are defined over the base field, the prover's computational cost involves approximately $O(N)$ multiplications over a large extension field denoted by hundreds of bits, where N is the witness size.

Recent Optimization of Small-Characteristic Field Sumcheck. Gruen et al. [26] introduced several techniques to accelerate the zerocheck IOP protocol, which are not specific to small fields. Dao et al. [7, 16, 17] proposed further optimizations for the small-characteristic and binary-field sumcheck protocol. These techniques are also compatible with our algorithms and to further improve the concrete efficiency.

Applications. Theoretically, a Boolean circuit composed of AND and XOR gates is sufficient to describe any deterministic algorithm. With the development of practical SNARK protocols, there has been increasing effort to integrate binary fields into real-world applications. One such application is the computation of hash functions. Bertoni et al. [10] introduced the Keccak hash function, while more ZK-friendly hash functions over binary tower fields have been proposed by [6]. Additionally, Jolt [1, 5] has begun leveraging binary tower fields to construct *Zero-Knowledge Virtual Machine (ZKVM)*.

2 Technical Overview

As previously mentioned, a SNARK is constructed by first designing an IOP protocol and then compiling it using a PCS. Especially, when proving statements about a Boolean circuit, an option is to construct a multivariate polynomial IOP over a binary field (or $GF(2)$) and compile it using a PCS for polynomials in $GF(2)[\mathbf{X}]$.

Recently, multivariate IOP protocols have become increasingly popular due to their linear proving complexity, in contrast to univariate protocols, which exhibit quasilinear proving complexity. Sumcheck protocol serves as a fundamental building block for many multivariate IOP protocols. However, when applied to

a small-characteristic field, the traditional sumcheck protocol remains unoptimized. The main reason is that to ensure soundness, the challenges introduced in the sumcheck protocol must be sampled from a large extension field. For example, when running the sumcheck protocol on $\mathbb{F}_p$ with p as the 31-bit Babybear field, all the computations transition to the extension field $\mathbb{E} = \mathbb{F}_{p^4}$ after generating a challenge in the first round. In contrast, the univariate IOP for the constraints over the same field retains the most expensive computations, such as FFT and hashing, within the base field. In the multivariate case, this gap significantly increases the proving time after the first round, especially when $p = 2$ and the extension field is $\mathbb{E} = \mathbb{F}_{2^{128}}$. As a result, this overhead reduces the performance advantage of multivariate systems in practical implementations.

One intuitive approach to avoid this overhead is to pack multiple base-field witnesses into an extension field. For example, given three vectors denoted as $(a_0, \ldots, a_3), (b_0, \ldots, b_3), (c_0, \ldots, c_3) \in \mathbb{F}_p^4$, we have

$$(c_0, \ldots, c_3) = (a_0, \ldots, a_3) \oplus (b_0, \ldots, b_3) \ ,$$

where "$\oplus$" denotes the element-wise addition. This can be equivalently represented in the extension field as

$$(c_0, \ldots, c_3)_{\mathbb{E}} = (a_0, \ldots, a_3)_{\mathbb{E}} \oplus (b_0, \ldots, b_3)_{\mathbb{E}} \ ,$$

where $\mathbb{E} = \mathbb{F}_{p^4} = \mathbb{F}(\alpha)$, and the notation $(c_0, \ldots, c_3)_{\mathbb{E}}$ denotes an element $c \in \mathbb{E}$ as explained in Section 3.1. However, this simple packing technique fails when checking

$$(c_0, \ldots, c_3) = (a_0, \ldots, a_3) \otimes (b_0, \ldots, b_3) \ ,$$

where "$\otimes$" denotes the element-wise multiplication.

In our paper, instead of packing multiple witnesses into an extension field, we retain the witnesses in the base field and utilize the extension structure of tower fields within the protocols.

2.1 Tower Sumcheck Protocol

In this section, we introduce tower sumcheck protocol. For simplicity, we focus on a specific version designed for a simple case where the expression is given by the product of multilinear extensions:

$$\sigma = \sum_{\mathbf{b} \in \{0,1\}^n} \widetilde{f}_0(\mathbf{b}) \widetilde{f}_1(\mathbf{b}) \cdots \widetilde{f}_{d-1}(\mathbf{b}) \ , \tag{1}$$

where $\widetilde{f}_0(\mathbf{X}), \ldots, \widetilde{f}_{d-1}(\mathbf{X}) \in \mathcal{T}_0[\mathbf{X}]$ denote the multilinear extensions (see Definition 2) defined over the base field $\mathcal{T}_0$. This type of sumcheck equation appears commonly in many IOPs and SNARKs based on multivariate polynomials, such as the GKR protocol. To efficiently generate a proof for this equation, we design a protocol called *tower sumcheck protocol*. The entire proving process is conducted over the following tower fields, which is defined by fixing a base field and sequentially deriving an extension field of the previous field:

$$\mathfrak{T}_{p,t,\ell} = \left\{ \mathcal{T}_0 = GF\left(p^{t^0}\right), \mathcal{T}_1 = GF\left(p^{t^1}\right), \ldots, \mathcal{T}_\ell = GF\left(p^{t^\ell}\right) \right\} \ ,$$

where the witness polynomials are in $\mathcal{T}_0[\mathbf{X}]$ and the challenges are sampled from $\mathcal{T}_\ell$. For example, when $p = 2$, $t = 2$ and $\ell = 7$, $\mathcal{T}_0$ is the binary field $GF(2)$, and $\mathcal{T}_\ell$ is $GF(2^{2^7}) = GF(2^{128})$. For a general formulation of our protocol, please refer to Section 4.1.

The main idea behind the sumcheck protocol is to recursively reduce the summation identity into a new identity with half of the size in each step, until it becomes trivial to verify. For instance, in the first round, the prover sends a univariate polynomial $q(X)$ with degree d, and then the problem becomes proving the correctness of $q(X)$. This verification is further reduced to checking a single evaluation of $q(x_0)$ for a random challenge $x_0 \in \mathcal{T}_\ell$, which incurs a soundness error $\frac{d}{|\mathcal{T}_\ell|}$ as established by the Schwartz-Zippel lemma (Lemma 1). Unfortunately, due to the random challenge from $\mathcal{T}_\ell$, the computations are all defined over $\mathcal{T}_\ell$ after the first round, which are significantly slower than computations over the base field.

To solve this problem, the key observation behind our approach is that, when the polynomial q is evaluated at the basis element α_0 defining the first extension field in the tower fields i.e., $\mathcal{T}_1 = \mathcal{T}_0(\alpha_0)$, and the polynomial's degree d is smaller than the field extension degree t, the coordinates of $q(\alpha_0)$ retain the original polynomial structure. Utilizing this property, we replace the random challenge $x_0 \in \mathcal{T}_\ell$ by the basis element α_0 and reduce the verification of Equation 1 to verifying the evaluation $q(\alpha_0)$. This results in zero soundness error. Subsequently, we replace the random challenges in the first ℓ rounds with the fixed basis element $(\alpha_0, \ldots, \alpha_{\ell-1})$, where each extension satisfies $\mathcal{T}_{i+1} = \mathcal{T}_i(\alpha_i)$. After ℓ rounds, we switch back to random challenges from $\mathcal{T}_\ell$ as in the standard sumcheck. Based on the intuition above, we propose the *tower sumcheck protocol* in Protocol 1. Note that as the basis elements of the tower fields are known by both the prover and the verifier, the first ℓ rounds are deterministic and non-interactive without any challenge from the verifier.

We are able to show the soundness of the new tower sumcheck protocol by the following theorem:

Theorem 1 (Tower Sumcheck (Simplified for MLE Products)). *The protocol described in Protocol 1 is a sumcheck protocol with perfect completeness and a soundness error of* $\frac{d(n-\ell)}{|\mathcal{T}_\ell|}$.

Improvements on the Prover Time. The main advantange of the tower sumcheck protocol is that in the step $i < \ell$, the computation is done over $\mathcal{T}_i$ instead of $\mathcal{T}_\ell$, which is significantly faster than the original sumcheck protocol. To evaluate the computational cost of the prover, we decompose its computation into the following two distinct procedures. Suppose the challenges are $\mathbf{x} = (x_0, \ldots, x_{n-1})$ (in the tradition $\mathbf{x}$ are sampled in $\mathcal{T}_\ell$, while in our protocol $\mathbf{x} = (\boldsymbol{\alpha}, x_\ell, \ldots, x_{n-1})$, and only $x_\ell, \ldots, x_{n-1}$ are random challenges), then:

- **Computing the message $q^{(i)}(X)$ for $0 \leq i < n$:** In this procedure, the prover computes the following steps:

Protocol 1 (Tower Sumcheck Protocol (Simplified for MLE Products))
Tower sumcheck protocol is a specialized sumcheck protocol designed over the tower fields $\mathfrak{T}_{p,t,\ell}$ to prove Equation 1. The protocol proceeds as follows:

- $\mathsf{TSC.Prove}_{n,d,\ell}\left(\widetilde{f}_0(X),\ldots,\widetilde{f}_{d-1}(X)\right)$*: With the input $\widetilde{f}_i \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}]$ for $0 \leq i < d$, $\mathcal{P}$ goes through the following steps:*
  1. ***Phase 1.*** *For $i = 0,\ldots,\ell-1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X) \in \mathcal{T}_i^{(\leq d)}[X]$ to the verifier, where:*

$$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} \widetilde{f}_0(\alpha_0,\ldots,\alpha_{i-1},X,\mathbf{b})\cdots\widetilde{f}_{d-1}(\alpha_0,\ldots,\alpha_{i-1},X,\mathbf{b}) \ .$$

  2. ***Phase 2.*** *For $i = \ell,\ldots,n-1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X) \in \mathcal{T}_\ell^{(\leq d)}[X]$ to the verifier, where:*

$$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} \widetilde{f}_0(\boldsymbol{\alpha},\mathbf{x}_{\ell..i},X,\mathbf{b})\cdots\widetilde{f}_{d-1}(\boldsymbol{\alpha},\mathbf{x}_{\ell..i},X,\mathbf{b}) \ .$$

     (b) *Receive a challenge x_i from the verifier.*
- $\mathsf{TSC.Verify}_{n,d,\ell}^{\left(\widetilde{f}_i(\cdot)\right)_{0\leq i<d}}(\sigma)$*: With the input $\sigma \in \mathcal{T}_0$, and the oracles $\widetilde{f}_0(\cdot),\ldots,\widetilde{f}_{d-1}(\cdot)$, $\mathcal{V}$ goes through the following steps:*
  1. *Set $\sigma^{(0)} = \sigma$.*
  2. ***Phase 1.*** *For $i = 0,\ldots,\ell-1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ from the prover and check $\sigma^{(i)} = q^{(i)}(0) + q^{(i)}(1)$*
     (b) *Compute $\sigma^{(i+1)} = q^{(i)}(\alpha_i)$.*
  3. ***Phase 2.*** *For $i = \ell,\ldots,n-1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ from the prover and check $\sigma^{(i)} = q^{(i)}(0) + q^{(i)}(1)$.*
     (b) *Randomly generate $x_i \leftarrow \mathcal{T}_\ell$ and send to the prover.*
     (c) *Compute $\sigma^{(i+1)} = q^{(i)}(x_i)$.*
  4. *Query the oracle $F_{\mathsf{eval}} = \widetilde{f}_0(\boldsymbol{\alpha},\mathbf{x}_{\ell..n})\cdots\widetilde{f}_{d-1}(\boldsymbol{\alpha},\mathbf{x})$. Output $F_{\mathsf{eval}} \stackrel{?}{=} \sigma^{(n)}$.*

Figure 1: Tower Sumcheck Protocol (Simplified)

1. Define $\left(a_j^{(i)}X + b_j^{(i)}\right) := \widetilde{f}_j(\mathbf{x}_{0..i},X,\mathbf{b})$, $\mathbf{b}\in\{0,1\}^{n-i-1}$, and compute

$$q_{\mathbf{b}}^{(i)}(X) = \prod_{j=0}^{d-1}\left(a_j^{(i)}X + b_j^{(i)}\right), \text{ for each } \mathbf{b}\in\{0,1\}^{n-i-1}.$$

2. Compute $q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} q_{\mathbf{b}}^{(i)}(X)$.

Since the additions are much cheaper than multiplications, we only count the cost of Step 1. Then our protocol is much cheaper in the first ℓ rounds. Assuming $c(d)$ is the complexity to compute the product of d binomials, Step 1 only costs $O\left(c(d)\frac{N}{2^{i+1}}\right)$ multiplications. However, in the previous protocol, it costs $O\left(c(d)\frac{N}{2}\right)$ base-field multiplications in Round 0 and $O\left(c(d)\frac{N}{2^{i+1}}\right)$ top-level field multiplications in the remaining rounds.

– **Fixing variables in $\widetilde{f}_0(\mathbf{x}_{0..i+1}, \mathbf{X}_{i+1..n}), \ldots, \widetilde{f}_{d-1}(\mathbf{x}_{0..i+1}, \mathbf{X}_{i+1..n})$:** For each $0 \le j < d$ and $\mathbf{b} \in \{0,1\}^{n-i-1}$, the prover computes

$$\widetilde{f}_j(\mathbf{x}_{0..i+1}, \mathbf{b}) = \widetilde{f}_j(\mathbf{x}_{0..i}, 0, \mathbf{b}) + x_i \cdot (\widetilde{f}_j(\mathbf{x}_{0..i}, 1, \mathbf{b}) - \widetilde{f}_j(\mathbf{x}_{0..i}, 0, \mathbf{b})) \ .$$

We also improve this step a lot in the first ℓ rounds. For each polynomial, our cost is negligible because multiplying a field element by a basis in its extension resembles concatenation in the implementation. While the previous protocol costs $O\left(\frac{N}{2}\right)$ multiplications between $\mathcal{T}_0$ and $\mathcal{T}_\ell$ in Round 0, and $O\left(\frac{N}{2^{i+1}}\right)$ top-level field multiplications in the remaining rounds.

The complexity comparison is shown in Table 1. when instantiated with Wiedemann's binary tower fields, assuming that d and $c(d)$ are constants, and the complexity of multiplications is quadratic in the bitlength (i.e., $\mathcal{M}_i = O((2^{2^i})^2)$), the total prover time of our sumcheck protocol is $O(2^{1.5\ell}N)$, while the standard sumcheck protocol runs in $O(2^{2\ell}N)$.

Computing Messages	**Our Protocol**	**Previous Protocol**
Phase 1, Round 0	$O\left(c(d)\frac{N}{2}\right)\mathcal{M}_0$	$O\left(c(d)\frac{N}{2}\right)\mathcal{M}_0$
Phase 1, Round $i \ge 1$	$O\left(c(d)\frac{N}{2^{i+1}}\right)\mathcal{M}_i$	$O\left(c(d)\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$
Phase 2, Round $i \ge \ell$	$O\left(c(d)\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$	$O\left(c(d)\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$
Fixing Variables		
Phase 1, Round 0	Negligible	$O\left(d\frac{N}{2}\right)\overline{\mathcal{M}}_0$
Phase 1, Round $i \ge 1$	Negligible	$O\left(d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$
Phase 2, Round $i \ge \ell$	$O\left(d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$	$O\left(d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$

Table 1: Comparison of previous and our protocols for sumcheck proving Equation 1, where ℓ is the total level of the field tower, d is the degree of the expression, $c(d)$ is the cost to multiply d binomials of the form $(aX + b)$, $\mathcal{M}_i$ is the cost of multiplication in $\mathcal{T}_i$, $\overline{\mathcal{M}}_i$ is the multiplication between $\mathcal{T}_0$ and $\mathcal{T}_i$.

2.2 Tower Zerocheck Protocol

Verifying a product of base-field MLEs is insufficient to verify a complete arithmetic constraint system. To construct a complete IOP over small-characteristic fields, such as the Plonkish-style protocol, we also need to verify the following identity:

$$\sigma = \mathsf{out}(\mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^n} \widetilde{eq}(\mathbf{Y}, \mathbf{b})\widetilde{f}_0(\mathbf{b})\widetilde{f}_1(\mathbf{b}) \cdots \widetilde{f}_{d-1}(\mathbf{b}) \tag{2}$$

where $\widetilde{f}_0(\mathbf{X}), \ldots, \widetilde{f}_{d-1}(\mathbf{X}) \in \mathcal{T}_0[X]$ are the multilinear extensions. This is named zerocheck in the literature. From the prior subsection, the tower sumcheck protocol assumes that all polynomials are defined over the base field. However, in

Protocol 2 (Tower Zerocheck Protocol (Simplified for MLE Products)) *Tower zerocheck protocol is a specialized zerocheck protocol designed over the tower fields $\mathfrak{T}_{p,t,\ell}$ to prove Equation 2. The protocol proceeds as follows:*

- $\mathsf{TZC.Prove}_{n,d,\ell}\left(\widetilde{f}_0(X),\ldots,\widetilde{f}_{d-1}(X);\mathbf{y}\right)$*: With the input $\widetilde{f}_i \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}]$ for $0 \leq i < d$ and $\mathbf{y} = (\boldsymbol{\alpha}, \mathbf{y}_{\ell..n})$, $\mathcal{P}$ goes through the following steps:*
  1. ***Phase 1.*** *For $i = 0,\ldots,\ell-1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X) \in \mathcal{T}_\ell^{(\leq d)}[X]$ to the verifier, where:*
     $$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} \widetilde{eq}(\boldsymbol{\alpha}_{i+1..\ell}, \mathbf{y}_{\ell..n}; \mathbf{b})\widetilde{f}_0(\boldsymbol{\alpha}_{0..i}, X, \mathbf{b})\cdots\widetilde{f}_{d-1}(\boldsymbol{\alpha}_{0..i}, X, \mathbf{b}).$$
  2. ***Phase 2.*** *For $i = \ell,\ldots,n-1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X) \in \mathcal{T}_\ell^{(\leq d)}[X]$ to the verifier, where:*
     $$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} \widetilde{eq}(\mathbf{y}_{i..n}; \mathbf{b})\widetilde{f}_0(\boldsymbol{\alpha}, \mathbf{x}_{\ell..i}, X, \mathbf{b})\cdots\widetilde{f}_{d-1}(\boldsymbol{\alpha}, \mathbf{x}_{\ell..i}, X, \mathbf{b}).$$
     (b) *Receive a challenge x_i from the verifier.*
- $\mathsf{TZC.Verify}^{\left(\widetilde{f}_i(\cdot)\right)_{0\leq i<d}}_{n,d,\ell}(\sigma;\mathbf{y})$*: With the input $\sigma \in \mathcal{T}_\ell$ and the output evaluation point $\mathbf{y} = (\boldsymbol{\alpha}, \mathbf{y}_{\ell..n})$, $\mathcal{V}$ goes through the following steps:*
  1. *Set $\sigma^{(0)} = \sigma$.*
  2. ***Phase 1.*** *For $i = 0,\ldots,\ell-1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ from the prover and check $\sigma^{(i)} = (1-\alpha_i)q^{(i)}(0) + \alpha_i q^{(i)}(1)$.*
     (b) *Compute $\sigma^{(i+1)} = q^{(i)}(\alpha_i)$.*
  3. ***Phase 2.*** *For $i = \ell,\ldots,n-1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ and check $\sigma^{(i)} = (1-y_{i-\ell})q^{(i)}(0) + y_{i-\ell}q^{(i)}(1)$.*
     (b) *Randomly generate $x_i \leftarrow \mathcal{T}_\ell$ and send to the prover.*
     (c) *Compute $\sigma^{(i+1)} = q^{(i)}(x_i)$.*
  4. *Query the oracle $F_{\mathsf{eval}} = \widetilde{f}_0(\boldsymbol{\alpha}, \mathbf{x})\cdots\widetilde{f}_{d-1}(\boldsymbol{\alpha}, \mathbf{x})$. Output $F_{\mathsf{eval}} \stackrel{?}{=} \sigma^{(n)}$.*

Figure 2: Tower Zerocheck Protocol (Simplified)

zerocheck, the statements involves $\mathsf{eq}(\mathbf{y};\mathbf{X})$ with $\mathbf{y} \in \mathcal{T}_\ell^n$, necessitating polynomial operations over the top-level field from the outset. A natural question arises: Can we still achieve further optimization in this scenario? The answer is affirmative.

We propose the *tower zerocheck protocol*, which includes a special optimization introduced by Gruen [26, Section 3.2]. We present the protocol in Protocol 2 and the following theorem summarizes its properties:

Theorem 2 (Tower Zerocheck (Simplified for MLE Products)). *The protocol described in Protocol 2 is a zerocheck protocol with perfect completeness and a soundness error of $\frac{(\ell+d+2)(n-\ell)}{|\mathcal{T}_\ell|}$.*

For the complete version, please refer to Section 4.2.

Improvements on the Prover Time. Similarly, we compare the performance of our protocol with the existing zerocheck protocol [14]. We also decompose its computation into the following two distinct procedures. Suppose the initial evaluation point $\mathbf{y} = (y_0, \ldots, y_{n-1})$ (in the tradition $\mathbf{y} \in \mathcal{T}_\ell^n$, while in our protocol $\mathbf{y} = (\boldsymbol{\alpha}, y_\ell, \ldots, y_{n-1})$, and only $y_\ell, \ldots, y_{n-1} \in \mathcal{T}_\ell^{n-\ell}$ are random challenges). Suppose the elements assigned in the MLEs are denoted as $\mathbf{x} = (x_0, \ldots, x_{n-1})$ (the form of $\mathbf{x}$ is similar to $\mathbf{y}$), then:

- **Computing the message $q^{(i)}(X)$ for $0 \leq i < n$:** In this procedure, the prover computes the following steps:
  1. Define $\left(a_j^{(i)} X + b_j^{(i)}\right) := \widetilde{f}_j(\mathbf{x}_{0..i}, X, \mathbf{b})$, $\mathbf{b} \in \{0,1\}^{n-i-1}$, and compute:

$$\hat{q}_{\mathbf{b}}^{(i)}(X) = \prod_{j=0}^{d-1} \left(a_j^{(i)} X + b_j^{(i)}\right) \ , \tag{3}$$

$$q_{\mathbf{b}}^{(i)}(X) = \widetilde{eq}(\mathbf{y}_{i+1..n}, \mathbf{b}) \cdot \hat{q}_{\mathbf{b}}^{(i)}(X) \ , \tag{4}$$

     for each $\mathbf{b} \in \{0,1\}^{n-i-1}$.
  2. Compute $q^{(i)}(X) = \sum_{\mathbf{b} \in \{0,1\}^{n-i-1}} q_{\mathbf{b}}^{(i)}(X)$.

  Similarly to the previous subsection, we only account for Step 1 in the evaluation. Our protocol improves efficiency in the first ℓ rounds. Assuming $c(d)$ is the cost to compute the product of d binomials, computing Equation 3 only costs $O\left(c(d)\frac{N}{2^{i+1}}\right)$ multiplications in $\mathcal{T}_i$, and computing Equation 4 always requires $O\left(d\frac{N}{2^\ell}\right)$ top-level field multiplications. However, in the old protocol, it costs $O\left(c(d)\frac{N}{2}\right)$ base-field multiplications and $O\left(d\frac{N}{2}\right)$ multiplications between the base field and the top-level field in Round 0. In the remaining rounds, since all polynomials are defined over the top-level field, it costs $O\left(c(d)\frac{N}{2^{i+1}} + d\frac{N}{2^{i+1}}\right)$ large-field multiplications.
- **Fixing variables in $\widetilde{f}_0(\mathbf{x}_{0..i+1}, \mathbf{X}_{i+1..n}), \ldots, \widetilde{f}_{d-1}(\mathbf{x}_{0..i+1}, \mathbf{X}_{i+1..n})$:** For each $0 \leq j < d$ and $\mathbf{b} \in \{0,1\}^{n-i-1}$, the prover behaves identically as Section 2.1.

The complexity comparison is shown in Table 2

2.3 Efficient SNARKs over Binary Tower Fields

Using the proposed tower sumcheck and tower zerocheck protocols, we now have the building blocks to efficiently construct IOP protocols over binary field. We instantiate our protocols using Wiedemann's binary tower fields, supplemented by an additional small technique that effectively handles the case where the extension degree t is less than or equal to the polynomial degree d (refer to Section 4.1). In Section 5, we instantiate the optimized binary-field IOP protocols specially tailored for various arithmetic constraint systems, including Plonkish, CCS, and GKR. Finally in Section 6, we introduce a novel basis-switching technique, enabling us to compile our binary-field IOP protocols into SNARKs by leveraging existing large-field polynomial commitment schemes.

Computing Messages	**Our Protocol**	**Previous Protocol**
Phase 1, Round 0	$O\left(c(d)\frac{N}{2}\right)\mathcal{M}_0 + O\left(d\frac{N}{2^\ell}\right)\mathcal{M}_\ell$	$O\left(c(d)\frac{N}{2}\right)\mathcal{M}_0 + O\left(d\frac{N}{2}\right)\overline{\mathcal{M}}_0$
Phase 1, Round $i \geq 1$	$O\left(c(d)\frac{N}{2^{i+1}}\right)\mathcal{M}_i + O\left(d\frac{N}{2^\ell}\right)\mathcal{M}_\ell$	$O\left(c(d)\frac{N}{2^{i+1}} + d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$
Phase 2, Round $i \geq \ell$	$O\left(c(d)\frac{N}{2^{i+1}} + d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$	$O\left(c(d)\frac{N}{2^{i+1}} + d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$
Fixing Variables		
Phase 1, Round 0	Negligible	$O\left(d\frac{N}{2}\right)\overline{\mathcal{M}}_0$
Phase 1, Round $i \geq 1$	Negligible	$O\left(d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$
Phase 2, Round $i \geq \ell$	$O\left(d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$	$O\left(d\frac{N}{2^{i+1}}\right)\mathcal{M}_\ell$

Table 2: Comparison of previous and our protocols for zerocheck proving Equation 1, where ℓ is the total level of the field tower, d is the degree of the expression, $c(d)$ is the cost to multiply d binomials of the form $(aX + b)$, $\mathcal{M}_i$ is the cost of multiplication in $\mathcal{T}_i$, $\overline{\mathcal{M}}_i$ is the multiplication between $\mathcal{T}_0$ and $\mathcal{T}_i$.

3 Preliminaries

3.1 Notation

Indices, Variables, and Points. We use $\mathbf{b}$ to represent binary vectors, such as $\mathbf{b} = (b_0, \ldots, b_{n-1}) \in \{0,1\}^n$ and $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{F}^n$, where $\mathbb{F}$ is a finite field. The notation $s..t$ denotes the index sequence of $(s, \ldots, t-1)$. We define the following shorthand:

$$\begin{aligned}\mathbf{X}_{s..t} &= (X_s, X_{s+1}, \ldots, X_{t-1})\\ \mathbf{x}_{s..t} &= (x_s, x_{s+1}, \ldots, x_{t-1})\\ \mathbf{b}_{s..t} &= (b_s, b_{s+1}, \ldots, b_{t-1})\end{aligned}$$

We use $\langle i \rangle_{0..n}^{(k)}$ to denote the n-bit base-k representation of an index i, or

$$\langle i \rangle_{0..n}^{(k)} := (i_0, i_1, \ldots, i_{n-1}) \ , \text{ where } i = \sum_{j=0}^{n-1} i_j \cdot k^j$$

On the other direction, we use the following notation to map the binary representation to the index, or

$$(i_0, \ldots, i_{n-1})_k := i \ , \text{ where } i = \sum_{j=0}^{n-1} i_j \cdot k^j$$

Similarly, we use this notation to map a base-field array to an extension-field element, or

$$(a_0, \ldots, a_{t-1})_{\mathbb{E}} := a \in \mathbb{E} \ ,$$

$$\text{where } a_i \in \mathbb{F} \text{ for } 0 \leq i < t,\ \mathbb{E} = \mathbb{F}(\alpha) \text{ and } [\mathbb{E} : \mathbb{F}] = t, \text{ and } a = \sum_{i=0}^{t-1} a_i \alpha^i.$$

Polynomials, Oracles, and Evaluations. We define a vector of field elements as $a(\mathbf{b}) : \{0,1\}^n \to \mathbb{F}$, indexed by a binary string. Its multilinear extension (MLE) is defined by $\tilde{a}(\mathbf{X}) : \mathbb{F}^n \to \mathbb{F}$ through Definition 2, where $\mathbf{X} = (X_0, \ldots, X_{n-1})$ is a list of variables. Sometimes, we don't distinguish the notation of a vector from its MLE. For notation, we follow these conventions:

- $a(\mathbf{b})$ indexes a value in the vector;
- $a(\mathbf{x})$ represents the evaluation of $a(\mathbf{X})$ at some random point $\mathbf{x}$ (typically generated by a verifier in an interactive proof protocol);
- a_{eval} denotes the evaluation result;
- $a(\cdot)$ refers to the oracle of $a(\mathbf{X})$.

Additionally, we use $(\mathbf{b}_x, \mathbf{b}_s)$, $(\mathbf{x}, \mathbf{s})$ and $(\mathbf{X}, \mathbf{S})$ to denote the concatenation of bit strings, random points, and variables. We also use ";" to distinguish different groups of variables, such as $\widetilde{eq}(\mathbf{Y}; \mathbf{X})$, $F(\mathbf{X}; \mathbf{Y})$, etc.

We frequently use the following functions: For $\mathbf{X}$ and $\mathbf{Y}$, define:

$$\widetilde{eq}(\mathbf{Y}; \mathbf{X}) = \prod_{i=0}^{n-1} ((1 - Y_i)(1 - X_i) + Y_i X_i) \ .$$

We use $\mathbb{F}^{(\leq d)}[X]$ to denote the univariate polynomial with the degree less than or equal to d, or each $f(X) \in \mathbb{F}^{(\leq d)}[X]$ can be represented in the following form:

$$f(X) = \sum_{i=0}^{d} f_i X^i \ .$$

Similarly, we use $\mathbb{F}^{(\leq d)}[\mathbf{X}]$ to denote the set of multivariate polynomials where the degree of each variable does not exceed d. Any polynomial $f(\mathbf{X}) \in \mathbb{F}^{(\leq d)}[\mathbf{X}_{0..n}]$ can be written as:

$$f(\mathbf{X}) = \sum_{i=0}^{(d+1)^n - 1} f_i \prod_{j=0}^{n-1} X^{i_j} \ ,$$

where $(i_0, \ldots, i_{n-1}) = \langle i \rangle_{0..n}^{(d+1)}$ is the base-$(d+1)$ representation of i, padded to n bits. Moreover, we define $\mathbb{F}^{(\leq d_x; \leq d_y)}[\mathbf{X}; \mathbf{Y}]$ as the set of polynomials where $\deg(X_i) \leq d_x$ and $\deg(Y_j) \leq d_y$.

Tower Fields. We denote the tower field set as

$$\begin{gathered} \mathfrak{T}_{p,t,\ell} = \{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_\ell\} \ , \\ \text{where } \mathcal{T}_0 = GF(p), \mathcal{T}_1 = GF\left(p^{t^1}\right), \ldots, \mathcal{T}_\ell = GF\left(p^{t^\ell}\right) \\ \text{and } \mathcal{T}_i = \mathcal{T}_{i-1}(\alpha_{i-1}) \cong \mathcal{T}_{i-1}[X]/(h_{i-1}(X)) \ , \end{gathered} \tag{5}$$

for some irreducible polynomial $h_{i-1}(X) \in \mathcal{T}_{i-1}[X]$ of degree t, with a root $\alpha_{i-1} \in \mathcal{T}_i$. We use the following notation to denote a list of basis elements:

$$\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_{\ell-1})$$

3.2 Finite Fields and Tower Fields

Finite field, or *Galois field*, is a field with a finite number of elements, supporting addition, subtraction, multiplication, and division operations. The *characteristic* of a field $\mathbb{F}$ is defined as the least positive integer n such that $nr = 0$ for all $r \in \mathbb{F}$. It is well known that all finite fields have prime characteristics. Specifically, if $\mathbb{F}$ is a finite field of characteristic p, then its order must be p^t for some $t \in \mathbb{N}^*$. The Galois field of order p^t is denoted by $GF(p^t)$. The existence and uniqueness of finite fields are established by the following theorems:

Theorem 3. *For every prime p and every positive integer n, there exists a finite field $\mathbb{F}$ with p^n elements.*

Theorem 4. *Every subfield of the Galois field $GF(p^b)$ has p^a elements, where a divides b. Conversely , if $a \mid b$ for $a > 0$, then there exists a unique subfield of $GF(p^b)$ isomorphic to $GF(p^a)$.*

For the structure of a finite field, we present the following theorem with an important corollary:

Theorem 5. *If G is a finite subgroup of $\mathbb{F}^*$, the multiplicative group of nonzero elements of a field $\mathbb{F}$, then G is cyclic.*

Corollary 1. *Every finite extension $\mathbb{E}$ of a finite field $\mathbb{F}$ is a simple extension of $\mathbb{F}$, which means there exists $\alpha \in \mathbb{E}$ such that $\mathbb{E} = \mathbb{F}(\alpha)$.*

In this paper, we construct protocols over the tower fields which we formally defined as Equation 5. From Theorem 4, we know that for each $\mathcal{T}_i \in \mathfrak{T}_{p,t,\ell}$ exists uniquely.

Terminology. Throughout this paper, we adopt the following terminology: the term *tower field* refers to any field within $\mathfrak{T}_{p,t,\ell}$, and *field tower* refers to the entire set $\mathfrak{T}_{p,t,\ell}$. Specifically, we use the *base field* to refer to the field $\mathcal{T}_0$, and the *intermediate field* to refer to any other field. We also call the highest-level field $\mathcal{T}_\ell$ the *challenge field*. Notabily, the challenge field satisfies $|\mathcal{T}_\ell| = 2^{O(\lambda)}$ where λ is the security parameter. When referring to binary tower fields, we also use the term *binary field* to denote the base field.

The *Schwartz–Zippel Lemma* [18,37,50] (also known as the *DeMillo–Lipton–Schwartz–Zippel Lemma*) is a fundamental result in probabilistic polynomial identity testing. It provides a probabilistic method for determining whether a given multivariate polynomial is identically zero.

Lemma 1 (Schwartz, Zippel). *Let $P \in R[X_0, \ldots, X_{n_x-1}]$, be a non-zero polynomial of total degree $d \geq 0$ over an integral domain R. Let S be a finite subset of R and let $x_0, \ldots, x_{n_x-1}$ be selected at random independently and uniformly from S. Then*

$$\Pr[P(x_0, \ldots, x_{n_x-1}) = 0] \leq \frac{d}{|S|}$$

In this paper, we use this result in the case where R is a finite field and $S = R$.

3.3 Interactive Argument

Definition 1 (Interactive Argument). *We say that* $\mathsf{ARG} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is an interactive argument of knowledge for a relation* $\mathcal{R}$ *if it satisfies the following completeness and knowledge properties.*

- ***Completeness****: For every adversary* $\mathcal{A}$

$$\Pr\left[\begin{matrix}(\mathbb{x},\mathbb{w}) \notin \mathcal{R} \text{ or} \\ \langle \mathcal{P}(\mathsf{pp},\mathbb{x},\mathbb{w}), \mathcal{V}(\mathsf{pp},\mathbb{x})\rangle = 1\end{matrix} : \begin{matrix}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{x},\mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp})\end{matrix}\right] = 1$$

- ***Witness-extended emulation****:* ARG *has witness-extended emulation with knowledge error* κ *if there exists an expected polynomial-time algorithm* $\mathcal{E}$ *such that for every polynomial-size adversary* $\mathcal{A}$ *it holds that*

$$\left|\Pr\left[\begin{matrix}\mathcal{A}(\mathsf{aux},\mathsf{tr}) = 1 \\ \end{matrix} : \begin{matrix}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{x},\mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{tr} \leftarrow \langle \mathcal{A}(\mathsf{aux}), \mathcal{V}(\mathsf{pp},\mathbb{x})\rangle\end{matrix}\right]\right.$$
$$\left. - \Pr\left[\begin{matrix}\mathcal{A}(\mathsf{aux},\mathsf{tr}) = 1 \\ \textit{and if } \mathsf{tr} \textit{ is accepting} \\ \textit{then } (\mathbb{x},\mathbb{w}) \in \mathcal{R}\end{matrix} : \begin{matrix}\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda) \\ (\mathbb{x},\mathsf{aux}) \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{tr},\mathbb{w}) \leftarrow \mathcal{E}^{\mathcal{A}(\mathsf{aux})}(\mathsf{pp},\mathbb{x})\end{matrix}\right]\right| \leq \kappa(\lambda)$$

  Above $\mathcal{E}$ *has oracle access to (the next-message functions of)* $\mathcal{A}(\mathsf{aux})$*.*

If the interactive argument of knowledge protocol ARG is public-coin, it has been shown that by the Fiat-Shamir transformation [21], we can derive a non-interactive argument of knowledge from ARG. If the scheme further satisfies the following property:

- **Succinctness**. The proof size is $|\pi| = \mathsf{poly}(\lambda, \log|C|)$ and the verification time is $\mathsf{poly}(\lambda, |\mathbb{x}|, \log|C|)$,

then it is a *Succinct Non-interactive Argument of Knowledge (SNARK)* protocol.

3.4 Multilinear Extension

Multilinear extension is a type of multivariate polynomial, often represented with an array. The definition is as follows:

Definition 2 (Multilinear Extension [15]). *Let* $\mathbf{a} = (a_0, \ldots, a_{N-1})$ *be a vector. The* multilinear extension *of* a *is the unique polynomial* $\tilde{a} : \mathbb{F}^n \to \mathbb{F}$ *such that* $N = 2^n$*, and* $\tilde{a}(\langle i \rangle_{0..n}^{(2)}) = a_i$ *for all* $0 \leq i < N$*.* $\tilde{a}$ *can be expressed as:*

$$\begin{aligned}\tilde{a}(\mathbf{X}) &= \sum_{i=0}^{2^n-1} \widetilde{eq}(\mathbf{X}; \langle i \rangle_{0..n}^{(2)}) \cdot a_i \\ &= \sum_{i=0}^{2^n-1} \left(\prod_{j=0}^{n-1} (1 - X_i) \cdot (1 - i_j) + X_i \cdot i_j) \right) \cdot a_i,\end{aligned}$$

Protocol 3 (Sumcheck Protocol) *The sumcheck protocol is an interactive proof protocol between $\mathcal{P}$ and $\mathcal{V}$, described as follows:*

- $\mathsf{SC.Prove}_{n,d}(F(\mathbf{X}))$*: With the input $\sigma \in \mathbb{F}$, $F : \mathbb{F}^n \to \mathbb{F}$ with degree at most d for each variable, $\mathcal{P}$ goes through the following steps:*
  1. *For $i = 0, \ldots, n-1$, run the following steps:*
     (a) *Set*
     $$q^{(i)}(X) = \sum_{\mathbf{b} \in \{0,1\}^{n-i-1}} F(x_0, \ldots, x_{i-1}, X, \mathbf{b}).$$
     (b) *Compute $q^{(i)}(1), \ldots, q^{(i)}(d)$ and send to the verifier.*
     (c) *Receive a challenge x_i from the verifier.*
- $\mathsf{SC.Verify}^{F(\cdot)}_{n,d}(\sigma)$*: With the input $\sigma \in \mathbb{F}$, $\mathcal{V}$ goes through the following steps:*
  1. *Set $\sigma_0 = \sigma$.*
  2. *For $i = 0, \ldots, n-1$, run the following steps:*
     (a) *Receive $q^{(i)}(1), \ldots, q^{(i)}(d)$, and compute $q^{(i)}(0) = \sigma_i - q^{(i)}(1)$.*
     (b) *Randomly generate $x_i \leftarrow \mathbb{F}$ and send to the prover.*
     (c) *Recover $q^{(i)}(X)$ from $q^{(i)}(0), \ldots, q^{(i)}(d)$ and compute $\sigma_{i+1} = q^{(i)}(x_i)$.*
  3. *Query the oracle $F_{\mathsf{eval}} = F(x_0, \ldots, x_{n-1})$. If $F_{\mathsf{eval}} = \sigma_n$, output $\mathtt{1}$. Otherwise, output $\mathtt{0}$.*

Figure 3: Sumcheck Protocol

3.5 Sumcheck Protocol

Sumcheck protocol is one of the most important interactive proofs in the literature. The sumcheck problem is to prove that the sum of a multivariate polynomial $F : \mathbb{F}^n \to \mathbb{F}$ on all binary inputs is a certain value σ, i.e., $\sigma = \sum_{b_0,\ldots,b_{n-1} \in \{0,1\}} F(b_0, \ldots, b_{n-1})$. Calculating the sum directly requires exponential time in n, as there are 2^n combinations of $b_0, \ldots, b_{n-1}$. Lund et al. [31] proposed a *sumcheck* protocol that allows a verifier $\mathcal{V}$ to delegate the computation to a computationally unbounded prover $\mathcal{P}$, who can convince $\mathcal{V}$ that σ is the correct summation.

The sumcheck protocol for $F(\mathbf{X}) = \prod_{i=0}^{d-1} f_i(\mathbf{X})$ is presented in Protocol 3. Xie et al. [46] introduced a state-of-the-art algorithm for MLE products. In Round j, the algorithm leverages a book-keeping table to compute the evaluations of $f_i(\mathbf{x}_{0..j}, \mathbf{X}_{j..n})$ on the boolean hypercube $\{0,1\}^{n-j}$ for each $0 \le i < d$. The performance of this approach is summarized in the following lemma:

Lemma 2. *Assuming d is a constant, the sumcheck protocol for a product of MLEs with n variables runs in $O(2^n)$ time.*

3.6 Polynomial Interactive Oracle Proof (PIOP)

Definition 3 (Public-coin Polynomial Interactive Oracle Proof [13]). *Let $\mathcal{R}$ be a binary relation and $\mathbb{F}$ be a finite field. Let $X = (X_0, \ldots, X_{n-1})$ be a*

vector of n indeterminates. A (n, d) Polynomial IOP for $\mathcal{R}$ over $\mathbb{F}$ with soundness error ϵ and knowledge error δ consists of two stateful PPT algorithms, the prover $\mathcal{P}$, and the verifier $\mathcal{V}$, that satisfy the following requirements:

- ***Protocol syntax**. For each i-th round there is a prover state $\mathsf{st}_i^{\mathcal{P}}$ and a verifier state $\mathsf{st}_i^{\mathcal{V}}$. For any common input x and $\mathcal{R}$ witness w, at round 0 the states are $\mathsf{st}_0^{\mathcal{P}} = (x, w)$ and $\mathsf{st}_0^{\mathcal{V}} = x$. In the i-th round (starting at $i = 1$) the prover outputs a single proof oracle $\mathcal{P}(\mathsf{st}_{i-1}^{\mathcal{P}}) \to \pi_i$, which is a polynomial $\pi_i(X) \in \mathbb{F}[X]$. The verifier deterministically computes the query matrix $i \in \mathbb{F}^{n \times \ell}$ from its state and a string of public random bits $\mathsf{coins}_i \leftarrow \{0,1\}^*$, i.e, $\mathcal{V}(\mathsf{st}_{i-1}^{\mathcal{V}}, \mathsf{coins}_i) \to \Sigma_i$. This query matrix is interpreted as a list of ℓ points in $\mathbb{F}^n$ denoted $(\sigma_{i,1}, \ldots, \sigma_{i,\ell})$. The oracle π_i is queried on all points in this list, producing the response vector $(\pi_i(\sigma_{i,1}), \ldots, \pi_\ell(\sigma_{i,\ell})) = a_i \in \mathbb{F}^{1 \times \ell}$. The updated prover state is $\mathsf{st}_i^{\mathcal{P}} \leftarrow (\mathsf{st}_{i-1}^{P}), \Sigma_i)$ and verifier state is $\mathsf{st}_i^{V} \leftarrow (\mathsf{st}_{i-1}^{\mathcal{V}}, \Sigma_i, a_i)$. Finally, $\mathcal{V}(\mathsf{st}_t^{\mathcal{V}})$ returns 1 or 0.*
  (Extensions: multiple and prior round oracles; various arity. The syntax can be naturally extended such that multiple oracles are sent in the i-th round; that the verifier may query oracles sent in the i-th round or earlier; or that some of the oracles are polynomials in fewer variables than n.)

Furthermore, a PIOP is stateless if for each $0 \leq i < t, \mathcal{V}(\mathsf{st}_{i-1}^{\mathcal{V}}, \mathsf{coins}_i) = \mathcal{V}(i, \mathsf{coins}_i)$.

3.7 Interactive Oracle Polynomial Commitment Scheme (IOPCS)

We refer to FRI-Binius for the definition of IOPCS as follows:

Definition 4 (IOPCS [20, Definition 2.8]). *An* interactive oracle polynomial commitment scheme (IOPCS) *is defined as $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ with the following syntax:*

- $\mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, n)$. *On input the security parameter $\lambda \in \mathbb{N}$ and a number-of-variables parameter $n \in \mathbb{N}$, outputs pp, which includes, among other things, a field $\mathbb{E}$.*
- $\mathsf{com}_f \leftarrow \Pi.\mathsf{Commit}(\mathsf{pp}, f)$. *On input pp and a multilinear polynomial $f(\mathbf{X}) \in \mathbb{E}[\mathbf{X}]$, outputs a handle com_f to a vector.*
- $b \leftarrow \langle \mathcal{P}(\mathsf{com}_f, y, \mathbf{x}; f), \mathcal{V}(\mathsf{com}_f, y, \mathbf{x}) \rangle$ *is an IOP. The parties have as common input a vector handle com_f, an evaluation point $\mathbf{x} \in \mathbb{E}^n$, and a claimed evaluation $y \in \mathbb{E}$. $\mathcal{P}$ has as further input a multilinear polynomial $f(\mathbf{X}) \in \mathbb{E}[\mathbf{X}]$. $\mathcal{V}$ outputs a success bit $b \in \{0,1\}$.*

The security of IOPCS is as follows:

Definition 5 (IOPCS Security [20, Definition 2.9]). *For each interactive oracle polynomial commitment scheme Π, security parameter $\lambda \in \mathbb{N}$, and number-of-variables parameter $n \in N$, PPT query sampler $\mathcal{Q}$, PPT adversary $\mathcal{A}$, and PPT emulator $\mathcal{E}$, we define the following experiment:*

- *The experimenter samples* $\mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, n)$*, and gives* pp*, including* $\mathbb{E}$*, to* $\mathcal{A}$ *and* $\mathcal{E}$*.*
- *Adversary, after interacting arbitrarily with the vector oracle, outputs a handle* $\mathsf{com}_f \leftarrow \mathcal{A}(\mathsf{pp})$*.*
- *On input* $\mathcal{A}$*'s record of interactions with the oracle.* $\mathcal{E}$ *outputs* $f(\mathbf{X}) \in \mathbb{E}[\mathbf{X}]$*.*
- *The query sampler outputs* $\mathbf{x} \leftarrow \mathcal{Q}(\mathsf{pp})$*;* $\mathcal{A}$ *responds with an evaluation claim* $y \leftarrow \mathcal{A}(\mathbf{x})$*.*
- *The experimenter defines the following two random bits:*
  - *Obtain the bit* $b \leftarrow \langle \mathcal{A}(y, \mathbf{x}), \mathcal{V}(\mathsf{com}_f, y, \mathbf{x}) \rangle$ *by running the evaluation IOP with* $\mathcal{A}$ *as* $\mathcal{V}$*.*
  - *Obtain the further bit* $b' := f(\mathbf{x}) \stackrel{?}{=} y$*.*

The IOPCS Π *is secure if, for each PPT adversary* $\mathcal{A}$*, there is a PPT emulator* $\mathcal{E}$ *and a negligible function* negl *such that, for each* $\lambda \in \mathbb{N}$*, each* $n \in \mathbb{N}$*, and each PPT query sampler* $\mathcal{Q}$*,* $\Pr[b = 1 \wedge b' = 0] \leq \mathsf{negl}(\lambda)$*.*

Definition 6 (Small-Field IOPCS [20, Definition 2.10]). *A* small-field interactive oracle polynomial commitment scheme (small-field IOPCS) *is a tuple of algorithms* $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ *with the following syntax:*

- $\mathsf{pp} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, n, \mathbb{F})$*. On input the security parameter* $\lambda \in \mathbb{N}$ *and a number-of-variables parameter* $n \in \mathbb{N}$*, a field* $\mathbb{F}$*, outputs* pp*, which includes, among other things, a field* $\mathbb{E}/\mathbb{F}$*.*
- $\mathsf{com}_f \leftarrow \Pi.\mathsf{Commit}(\mathsf{pp}, f)$*. On input* pp *and a multilinear polynomial* $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$*, outputs a handle* com_f *to a vector.*
- $b \leftarrow \langle \mathcal{P}(\mathsf{com}_f, y, \mathbf{x}; f), \mathcal{V}(\mathsf{com}_f, y, \mathbf{x}) \rangle$ *is an IOP. The parties have as common input a vector handle* com_f*, an evaluation point* $\mathbf{x} \in \mathbb{E}^n$*, and a claimed evaluation* $y \in \mathbb{E}$*.* $\mathcal{P}$ *has as further input a multilinear polynomial* $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$*.* $\mathcal{V}$ *outputs a success bit* $b \in \{0, 1\}$*.*

The security of small-field IOPCS, Π exactly as in Definition 5 expect that we require that $\mathcal{E}$ output a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$.

4 Sumcheck Protocols over Tower Fields

4.1 Sumcheck Protocol over Tower Field

In this subsection, we introduce an efficient sumcheck protocol over tower fields. We define the *sumcheck relation* as follows:

Definition 7 (Sumcheck Relation). *The relation* $\mathcal{R}_{\mathsf{SUM}}^{(n_x, d)}$ *is the set of all tuples* $(\mathbb{x}; \mathbb{w}) = (\sigma, \mathbf{y}, F(\cdot;\cdot); F(\mathbf{X}; \mathbf{Y}))$ *such that:*

$$\sigma = \mathsf{out}(\mathbf{y}) = \sum_{\mathbf{b} \in \{0,1\}^{n_x}} F(\mathbf{b}; \mathbf{y}) \tag{6}$$

where $F(\mathbf{X}; \mathbf{Y}) \in \mathcal{T}_0^{(\leq d; \leq 1)}[\mathbf{X}_{0..n_x}; \mathbf{Y}_{0..n_y}]$*.*

The traditional sumcheck protocol establishes the validity of a summation identity by leveraging Schwartz-Zippel Lemma (Lemma 1) and reduction. Specifically, the protocol begins with the equation:

$$\mathsf{out}(\mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^n} F(\mathbf{b};\mathbf{Y}) \ . \tag{7}$$

At the first step, the verifier provides a random challenge $\mathbf{y}$, reducing the problem to verifying:

$$\mathsf{out}(\mathbf{y}) = \sigma^{(0)} = \sum_{\mathbf{b}\in\{0,1\}^n} F(\mathbf{b};\mathbf{y}) \tag{8}$$

When running the sumcheck protocol, at the beginning of each step i, the goal is to prove

$$\sigma^{(i)} = \sum_{\mathbf{b}\in\{0,1\}^{n-i}} F(\mathbf{x}_{0..i-1},\mathbf{b};\mathbf{y}) \ . \tag{9}$$

To achieve it, the prover computes $q^{(i)} = \sum_{\mathbf{b}\in\{0,1\}^{n-i}} F(\mathbf{x}_{0..i-1}, X, \mathbf{b}_{1..n-i-1};\mathbf{y})$, and upon receiving a fresh challenge x_i from the verifier, reduces the problem to verifying

$$\sigma^{(i+1)} = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} F(\mathbf{x}_{0..i},\mathbf{b};\mathbf{y}) \tag{10}$$

This reduction process continues iteratively until the problem is fully reduced to verify

$$\sigma^{(n)} = F(\mathbf{x};\mathbf{y}) \tag{11}$$

at which point the correctness is verified by the oracle access to $F(\mathbf{X};\mathbf{Y})$.

However, when constructing the sumcheck protocol on the field tower $\mathfrak{T}_{p,t,\ell}$, it is essential to account for the case where the initial evaluation point is set as $\mathbf{y} = (\boldsymbol{\alpha}, \mathbf{y}_{\ell..n})$, where $\boldsymbol{\alpha}$ represents the basis elements of each intermediate fields, and $\mathbf{y}_{\ell..n}$ consists of randomly sampled elements from the challenge field. The reason for considering this specific form of $\mathbf{y}$ will become clear in the following sections. To analyze the soundness error of our proposed protocol, we introduce a new reduction that preserves the variables $\mathbf{Y}$ and $\mathbf{X}$ throughout the execution of the protocol, as formalized in the following lemma:

Lemma 3 (Sumcheck Reduction Lemma). *With the polynomial* $F(\mathbf{X};\mathbf{Y}) \in \mathcal{T}_0^{(\leq d;\leq 1)}[\mathbf{X}_{0..n_x};\mathbf{Y}_{0..n_y}]$ *and* $\sigma^{(i)}(\mathbf{X}_{0..i};\mathbf{Y}) \in \mathcal{T}_0^{(\leq d;\leq 1)}[\mathbf{X};\mathbf{Y}]$*, there is:*

$$\sigma^{(i)}(\mathbf{X}_{0..i};\mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^{n_x-i}} F(\mathbf{X}_{0..i},\mathbf{b};\mathbf{Y}) \tag{12}$$

for $0 \leq i < n_x$ *if and only if there exists* $\sigma^{(i+1)}(\mathbf{X}_{0..i+1};\mathbf{Y}) \in \mathcal{T}_0^{(\leq d;\leq 1)}[\mathbf{X};\mathbf{Y}]$*, such that*

1. $\sigma^{(i)}(\mathbf{X}_{0..i};\mathbf{Y}) = \sigma^{(i+1)}(\mathbf{X}_{0..i},0;\mathbf{Y}) + \sigma^{(i+1)}(\mathbf{X}_{0..i},1;\mathbf{Y})$*;*
2. $\sigma^{(i+1)}(\mathbf{X}_{0..i+1};\mathbf{Y}) = \sum_{\mathbf{b}'\in\{0,1\}^{n-i-1}} F(\mathbf{X}_{0..i+1},\mathbf{b}';\mathbf{Y})$*.*

Typically, the sumcheck protocol utilizes the Schwartz-Zippel Lemma to succinctly validate the sumcheck reduction. To enhance the efficiency of the protocol over tower fields, we exploit the extension structure within the field tower and embed the message $q^{(i)}(\mathbf{X}_{0..i+1}; \mathbf{Y})$ into intermediate field elements. Before demonstrating the protocol, we prove a generalized Schwartz-Zippel Lemma, which is stated as follows:

Lemma 4 (Embedding Lemma). *Let $\mathfrak{T}_{p,t,\ell} = (\mathcal{T}_0, \ldots, \mathcal{T}_\ell)$ be a field tower with the basis elements $(\alpha_0, \ldots, \alpha_{\ell-1})$,. Consider a nonzero polynomial $q(\mathbf{X})$ over the base field, or $q(\mathbf{X}) \in \mathcal{T}_0[\mathbf{X}_{0..n_0+n_1}]$, where $n_0 \leq \ell$, each variable X_i for $i < n_0$ has degree less than t, and each variable X_i for $n_0 \leq i < n_0 + n_1$ is at most d. If $S \subseteq \mathbb{E}$ is a finite set, then for a uniformly random distribution $\mathcal{U}_S$ over S*

$$\Pr_{x_{n_0},\ldots,x_{n_0+n_1-1} \leftarrow \mathcal{U}_S}[q(\boldsymbol{\alpha}_{0..n_0}, \mathbf{x}_{n_0..n_0+n_1}) = 0] \leq \frac{dn_1}{|S|} \ .$$

Proof. Because $\deg(X_i) < t$ for $i < n_0$, then $q(\mathbf{X}) \neq 0$ if and only if:

$$q(\boldsymbol{\alpha}_{0..n_0}, \mathbf{X}_{n_0..n_0+n_1}) \neq 0 \ .$$

By applying Lemma 1 on $q'(\mathbf{X}) = q(\boldsymbol{\alpha}_{0..n_0}, \mathbf{X})$, we draw the conclusion in the lemma.

The tower sumcheck protocol is presented in Protocol 4. Currently, we only focus on the case where $s = 1$, $\boldsymbol{\alpha}^{(s)} = \boldsymbol{\alpha}$ and $\ell^{(s)} = \ell$. The protocol consists of two phases. Phase 1 consists of several non-interactive rounds, where in each round i, the prover sends the polynomial $q^{(i)}(X)$ to the verifier, and assigns an additional variable X_i in $F(\boldsymbol{\alpha}_{0..i}, X_{i..n})$ to the basis α_i. After that, Phase 2 proceeds identically to the traditional sumcheck protocol, completing the reduction process. Finally, the verifier queries the oracle $F(\cdot;\cdot)$ at the point $(\boldsymbol{\alpha}, \mathbf{x}_{\ell..n}; \mathbf{y})$ to verify the correctness. The oracle access can be instantiated directly using a polynomial commitment protocol or further verification steps. We summarize our result in the following theorem:

Theorem 6 (Sumcheck over Tower Fields (When $s = 1$)). *Assuming $n_x \geq \ell$ and $n_x \geq \ell$. Let $\mathfrak{T}_{p,t^s,\ell^{(s)}} = \left\{\mathcal{T}_0^{(s)}, \ldots, \mathcal{T}_{\ell^{(s)}}^{(s)}\right\}$ be a field tower, where $s = 1$, $\boldsymbol{\alpha}^{(s)} = \boldsymbol{\alpha}$ and $\ell^{(s)} = \ell$. The protocol as described in Protocol 4 is a sumcheck protocol to ensure whether $(\sigma, \mathbf{y} = (\boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}), F(\cdot;\cdot); F(\mathbf{X}; \mathbf{Y})) \in \mathcal{R}_{\mathsf{SUM}}^{(n_x,d)}$ where $d + 1 < t$, with perfect completeness, and $\frac{(\ell+1)(n_y-\ell)+d(n_x-\ell)}{|\mathcal{T}_\ell|}$ soundness error.*

Proof. The perfect completeness is obvious.

To prove the soundness error, Suppose there are $\left\{\bar{\sigma}^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y})\right\}_{0 \leq i \leq \ell}$ such that

$$\begin{aligned}
&\bar{\sigma}^{(0)}(\boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}) = \sigma^{(0)} \ , \\
&\bar{\sigma}^{(i+1)}(\boldsymbol{\alpha}_{0..i}, X_i; \boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}) = q^{(i)}(X_i) \quad \text{for } 0 \leq i < \ell \ ,
\end{aligned}$$

Protocol 4 (Tower Sumcheck Protocol) *Tower sumcheck protocol is a specially designed sumcheck protocol over the tower fields $\mathfrak{T}_{p,t^s,\ell^{(s)}}s$. It is described as follows:*

- $\mathsf{TSC.Prove}_{n,d,s,\ell^{(s)}}(F(\mathbf{X};\mathbf{Y});\mathbf{y})$*: With the input $F(\mathbf{X};\mathbf{Y}) \in \mathcal{T}_0^{(\leq d;\leq 1)}[\mathbf{X};\mathbf{Y}]$ and the initial evaluation point $\mathbf{y}$, $\mathcal{P}$ goes through the following steps:*
  1. ***Phase 1.*** *For $i = 0, \ldots, \ell^{(s)} - 1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X)$ to the verifier, where*
     $$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} F\left(\alpha_0^{(s)}, \ldots, \alpha_{i-1}^{(s)}, X, \mathbf{b}; \mathbf{y})\right) .$$
  2. ***Phase 2.*** *For $i = \ell^{(s)}, \ldots, n-1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X)$ to the verifier, where*
     $$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} F\left(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..i}, X, \mathbf{b}; \mathbf{y}\right) .$$
     (b) *Receive a challenge x_i from the verifier.*
- $\mathsf{TSC.Verify}^{F(\cdot,\cdot)}_{n,d,s,\ell^{(s)}}(\sigma,\mathbf{y})$*: With the input σ, the initial evaluation point $\mathbf{y}$, and oracle access to $F(\mathbf{X};\mathbf{Y})$, $\mathcal{V}$ goes through the following steps:*
  1. *Set $\sigma^{(0)} = \sigma$.*
  2. ***Phase 1.*** *For $i = 0, \ldots, \ell^{(s)} - 1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ from the prover and check $\sigma^{(i)} = q^{(i)}(0) + q^{(i)}(1)$.*
     (b) *Compute $\sigma^{(i+1)} = q^{(i)}(\alpha_i^{(s)})$.*
  3. ***Phase 1.*** *For $i = \ell^{(s)}, \ldots, n-1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ from the prover and check $\sigma^{(i)} = q^{(i)}(0) + q^{(i)}(1)$.*
     (b) *Randomly generate $x_i \leftarrow \mathcal{T}_\ell$ and send to the prover.*
     (c) *Compute $\sigma^{(i+1)} = q^{(i)}(x_i)$.*
  4. *Query the oracle $F_{\mathsf{eval}} = F(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..n}; \mathbf{y})$. Output $F_{\mathsf{eval}} \stackrel{?}{=} \sigma^{(n)}$.*

Figure 4: Tower Sumcheck Protocol

In Phase 2, from the analysis of the traditional sumcheck,

$$\sigma^{(\ell)} = \sum_{\mathbf{b}\in\{0,1\}^{n-\ell}} F(\boldsymbol{\alpha}, \mathbf{b}; \boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}) .$$

with soundness error $\frac{d(n_x-\ell)}{|\mathcal{T}_\ell|}$. From Lemma 4, and $d+1 < t$, it proves that there exists $\bar{\sigma}^{(\ell)}(\mathbf{X},\mathbf{Y})$ such that

$$\bar{\sigma}^{(\ell)}(\mathbf{X}_{0..\ell}, \mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^{n-\ell}} F(\mathbf{X}_{0..\ell}, \mathbf{b}; \mathbf{Y}) . \tag{13}$$

with $\frac{(n_y-\ell)+d(n_x-\ell)}{|\mathcal{T}_\ell|}$ soundness error.

As for Phase 1, we prove the statement by induction from Round $\ell - 1$ to 0. Suppose

$$\bar{\sigma}^{(i+1)}(\mathbf{X}_{0..i+1}, \mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^n} F(\mathbf{X}_{0..i+1}, \mathbf{b}; \mathbf{Y}) \tag{14}$$

with $\frac{(\ell-i)(n_y-\ell)+d(n_x-\ell)}{|\mathcal{T}_\ell|}$ soundness error, which is true when $i = \ell - 1$. Then from the algorithm, we have

$$\begin{aligned} \sigma^{(i)} &= q^{(i)}(0) + q^{(i)}(1) \text{ , or} \\ \bar{\sigma}^{(i)}(\boldsymbol{\alpha}_{0..i}; \boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}) &= \bar{\sigma}^{(i+1)}(\boldsymbol{\alpha}_{0..i}, 0; \boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}) + \bar{\sigma}^{(i+1)}(\boldsymbol{\alpha}_{0..i}, 1; \boldsymbol{\alpha}, \mathbf{y}_{\ell..n_y}) \text{ ,} \end{aligned}$$

which implies

$$\bar{\sigma}^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}) = \bar{\sigma}^{(i+1)}(\mathbf{X}_{0..i}, 0; \mathbf{Y}) + \bar{\sigma}^{(i+1)}(\mathbf{X}_{0..i}, 1; \mathbf{Y}) \tag{15}$$

with soundness error $\frac{(n_y-\ell)}{|\mathcal{T}_\ell|}$. By applying Lemma 3 on Equation 15, it implies that

$$\bar{\sigma}^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^{n_x-i}} F(\mathbf{X}_{0..i}, \mathbf{b}; \mathbf{Y})$$

with soundness error $\frac{(\ell+1-i)(n_y-\ell)+d(n_x-\ell)}{|\mathcal{T}_\ell|}$ and at the final step, it implies

$$\mathsf{out}(\mathbf{Y}) = \bar{\sigma}^{(0)}(\mathbf{Y}) = \sum_{\mathbf{b}\in\{0,1\}^n} F(\mathbf{b}; \mathbf{Y})$$

with soundness error $\frac{(\ell+1)(n_y-\ell)+d(n_x-\ell)}{|\mathcal{T}_\ell|}$.

Extend Protocols to Tower Fields with Small Extension Degree t. In the previous subsections, we propose an efficient sumcheck protocol under the assumption that $d + 1 < t$, where t denotes the extension degree between adjacent fields in $\mathfrak{T}_{p,t,\ell}$, and d represents the maximum degree of the variables in the given expression. However, many commonly used tower fields only satisfy $t \leq d + 1$. For example, Wiedemann [45] proposed a binary field tower with quadratic extensions, which has been adopted to construct SNARK and PCS by Binius and FRI-Binius. In this section, we extend our solution to tower fields where $t \leq d + 1$, provided that t is a prime power. We propose the following Lemma:

Lemma 5. *Let $\mathfrak{T}_{p,t,\ell}$ be a field tower where t is a prime power. Suppose the tower fields $\mathcal{T}_i, \mathcal{T}_j, \mathcal{T}_k \in \mathfrak{T}_{p,t,\ell}$ satisfy the following extension relationships:*

- *$\mathcal{T}_j = \mathcal{T}_i(\alpha)$, with extension degree t^a,*
- *$\mathcal{T}_k = \mathcal{T}_j(\beta)$ with extension degree t^b.*

Then $\mathcal{T}_k$ can be expressed as a direct extension of $\mathcal{T}_i$ via β with an overall extension degree of t^{a+b}, i.e.,

$$\mathcal{T}_k = \mathcal{T}_i(\beta), \qquad [\mathcal{T}_k : \mathcal{T}_i] = t^{a+b}.$$

Proof. By the tower rule of field extensions, we have $[\mathcal{T}_k : \mathcal{T}_i(\beta)] \cdot [\mathcal{T}_i(\beta) : \mathcal{T}_i] = [\mathcal{T}_k : \mathcal{T}_j] \cdot [\mathcal{T}_j : \mathcal{T}_i] = t^{a+b}$. Since t is a prime power, $[\mathcal{T}_i(\beta) : \mathcal{T}_i]$ satisfies $[\mathcal{T}_i(\beta) : \mathcal{T}_i] \mid t^a$ or $t^a \mid [\mathcal{T}_i(\beta) : \mathcal{T}_i]$. From the uniqueness of the fields (Theorem 4), $\mathcal{T}_i(\beta) \subseteq \mathcal{T}_j$ or $\mathcal{T}_j \subseteq \mathcal{T}_i(\beta)$. Considering $\beta \notin \mathcal{T}_j$, there should be $\mathcal{T}_j \subset \mathcal{T}_i(\beta)$, which implies $\mathcal{T}_j(\beta) \subseteq \mathcal{T}_i(\beta)$. On the other hand, there is $\mathcal{T}_i \subset \mathcal{T}_j$, which implies $\mathcal{T}_i(\beta) \subseteq \mathcal{T}_j(\beta)$. Therefore, $\mathcal{T}_i(\beta) = \mathcal{T}_j(\beta) = \mathcal{T}_k$.

We have the following corollary:

Corollary 2. *Let $\mathfrak{T}_{p,t,\ell} = \{\mathcal{T}_0, \ldots, \mathcal{T}_\ell\}$ be a field tower, where each intermediate field is defined as $\mathcal{T}_i = \mathcal{T}_{i-1}(\alpha_{i-1})$ for basis elements $(\alpha_0, \ldots, \alpha_{\ell-1})$ and t is a prime power, we can construct $\mathfrak{T}_{p,t^s,\ell^{(s)}} = \left\{\mathcal{T}_{0\cdot s}, \ldots, \mathcal{T}_{\lfloor \ell/s \rfloor \cdot s}\right\}$ for $s > 1$, where $\mathcal{T}_{i\cdot s} = \mathcal{T}_{(i-1)\cdot s}(\alpha_{i\cdot s-1})$.*

Therefore, when applying our protocol to a given tower field $\mathfrak{T}_{p,t,\ell}$ with $t \leq d$, we can construct $\mathfrak{T}_{p,t^s,\ell^{(s)}}$ such that $t^s > d+1$. We denote the basis for $\mathfrak{T}_{p,t^s,\ell^{(s)}}$ as $\boldsymbol{\alpha}^{(s)} = \left(\alpha_0^{(s)}, \ldots, \alpha_{\ell^{(s)}-1}^{(s)}\right)$ and set $\ell^{(s)} = \lfloor \ell/s \rfloor$.

Then we restate and generalize Theorem 6 as follows:

Theorem 7 (Sumcheck over Tower Fields). *Let $\mathfrak{T}_{p,t^s,\ell^{(s)}} = \left\{\mathcal{T}_0^{(s)}, \ldots, \mathcal{T}_{\ell^{(s)}}^{(s)}\right\}$ be a field tower. The protocol as described in Protocol 4 is a sumcheck protocol to ensure whether*

$$\left(\sigma, (\boldsymbol{\alpha}^{(s)}, \mathbf{y}_{0..n_y-\ell^{(s)}}), F(\cdot;\cdot); F(\mathbf{X};\mathbf{Y})\right) \in \mathcal{R}_{\mathsf{SUM}}^{(n_x,d)} \ ,$$

with perfect completeness, and $\frac{(\ell^{(s)}+1)(n_y-\ell^{(s)})+d(n_x-\ell^{(s)})}{|\mathcal{T}_\ell|}$ soundness error.

4.2 Zerocheck Protocol over Tower Field

In Section 4.1, we introduced the tower sumcheck protocol, building upon Lemma 3 and Lemma 4, to efficiently prove summation identities with base-field witnesses. However, to construct efficient IOP protocols, we mainly require an optimized method to verify the polynomial identities. Although a polynomial identity can be converted to a certain summation identity and proved by the sumcheck protocol, designing a special zerocheck protocol to prove the polynomial identities enables us to include more optimizations such as those proposed by Gruen [26, Section 3.2] and to further reduce the degree of the prover messages $q^{(i)}(X)$.

In this subsection, we introduce the *zerocheck relation*, which expresses the equality between $\mathsf{out}(\mathbf{X})$ and $F(\mathbf{X})$ over evaluations on the points in $\{0,1\}^n$.

Definition 8 (Zerocheck Relation). *The relation $\mathcal{R}_{\mathsf{IDENT}}^{(n,d)}$ is set of all tuples $(\mathbb{x}, \mathbb{w}) = (\sigma, \mathbf{y}, F(\cdot); F(\mathbf{X}))$ such that:*

$$\sigma = \mathsf{out}(\mathbf{y}) = \sum_{\mathbf{b}\in\{0,1\}^n} \widetilde{eq}(\mathbf{y}, \mathbf{b}) F(\mathbf{b}) \tag{16}$$

where $\widetilde{eq}(\mathbf{Y}, \mathbf{X})$ is defined in Section 3.1 and $F(\mathbf{X}) \in \mathcal{T}_0^{(\leq d)}[\mathbf{X}_{0..n}]$.

We also formally establish the following *zerocheck reduction lemma*, which serves as the foundation for our zerocheck protocol:

Lemma 6 (Zerocheck Reduction Lemma). *With the polynomial* $F(\mathbf{X}) \in \mathcal{T}_0^{(\leq d)}[\mathbf{X}_{0..n}]$ *and* $\sigma^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}_{i..n}) \in \mathcal{T}_0^{(\leq d;\leq 1)}[\mathbf{X}; \mathbf{Y}]$*, there is*

$$\sigma^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}_{i..n}) = \sum_{\mathbf{b}\in\{0,1\}^{n-i}} \widetilde{eq}(\mathbf{Y}_{i..n}; \mathbf{b}) F(\mathbf{X}_{0..i}, \mathbf{b}) \tag{17}$$

for $0 \leq i < n$ *if and only if there exists* $\sigma^{(i+1)}(\mathbf{X}_{0..i+1}, \mathbf{Y}_{i+1..n}) \in \mathcal{T}_0^{(\leq d;\leq 1)}[\mathbf{X}; \mathbf{Y}]$*, such that*

1. $\sigma^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}_{i..n}) = (1-Y_i)\sigma^{(i+1)}(\mathbf{X}_{0..i}, 0, \mathbf{Y}_{i+1..n}) + Y_i\sigma^{(i+1)}(\mathbf{X}_{0..i}, 1, \mathbf{Y}_{i+1..n})$;
2. $\sigma^{(i+1)}(\mathbf{X}_{0..i+1}; \mathbf{Y}_{i+1..n}) = \sum_{\mathbf{b}'\in\{0,1\}^{n-i-1}} \widetilde{eq}(\mathbf{Y}_{i+1..n}; \mathbf{b}') F(\mathbf{X}_{0..i+1}, \mathbf{b}')$.

Proof. We rewrite Equation 17 as follows:

$$\begin{aligned}
\sigma^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}_{i..n}) &= \sum_{\mathbf{b}\in\{0,1\}^{n-i}} \widetilde{eq}(\mathbf{Y}_{i..n}; \mathbf{b}) F(\mathbf{X}_{0..i}, \mathbf{b}) \\
&= \sum_{b\in\{0,1\}} \sum_{\mathbf{b}'\in\{0,1\}^{n-i-1}} \widetilde{eq}(Y_i, \mathbf{Y}_{i+1..n}; b, \mathbf{b}') F(\mathbf{X}_{0..i}, b, \mathbf{b}') \\
&= \sum_{b\in\{0,1\}} \widetilde{eq}(Y_i, b) \sum_{\mathbf{b}'\in\{0,1\}^{n-i-1}} \widetilde{eq}(\mathbf{Y}_{i+1..n}; \mathbf{b}') F(\mathbf{X}_{0..i}, b, \mathbf{b}')
\end{aligned}$$

Let $\sigma^{(i+1)}(\mathbf{X}_{0..i+1}; \mathbf{Y}_{i+1..n}) = \sum_{\mathbf{b}'\in\{0,1\}^{n-i-1}} \widetilde{eq}(\mathbf{Y}_{i+1..n}; \mathbf{b}') F(\mathbf{X}_{0..i+1}, \mathbf{b}')$, we can derive the following equation:

$$\sigma^{(i)}(\mathbf{X}_{0..i}; \mathbf{Y}_{i..n}) = \sum_{b\in\{0,1\}} \widetilde{eq}(Y_i; b) \cdot \sigma^{(i+1)}(\mathbf{X}_{0..i}, b; \mathbf{Y}_{i+1..n})$$

Building upon this reduction, we propose the *tower zerocheck protocol*, a specialized approach designed to generate a proof for the relation $\mathcal{R}_{\mathsf{IDENT}}$. This protocol also leverages the hierarchical structure of tower fields to improve computational efficiency.

The *tower zerocheck protocol* is demonstrated in Protocol 5. We summarize the result by the following theorem:

Theorem 8 (Zerocheck over Tower Fields). *With the tower fields* $\mathfrak{T}_{p,t^s,\ell^{(s)}} = \left\{\mathcal{T}_0^{(s)}, \ldots, \mathcal{T}_{\ell^{(s)}}^{(s)}\right\}$*, the protocol as described in Protocol 5 is to ensure whether*

$$\left(\sigma, (\boldsymbol{\alpha}^{(s)}, \mathbf{y}_{\ell^{(s)}..n}), F(\cdot); F(\mathbf{X})\right) \in \mathcal{R}_{\mathsf{IDENT}}^{(n,d)},$$

where $d < t^s$*, with perfect completeness,* $\frac{(\ell+d+2)(n-\ell^{(s)})}{|\mathcal{T}_\ell|}$ *soundness error.*

Proof. The proof is similar to Theorem 7.

Protocol 5 (Tower Zerocheck Protocol) *Tower zerocheck protocol is a specially designed zerocheck protocol over the tower fields $\mathfrak{T}_{p,t^s,\ell^{(s)}}$. It is described as follows*

- $\mathsf{TZC.Prove}_{n,d,s,\ell^{(s)}}(F(\mathbf{X});\mathbf{y})$*: With the input $F(\mathbf{X}) \in \mathcal{T}_0^{(\leq d)}[\mathbf{X}]$ and the initial evaluation point $\mathbf{y}$, $\mathcal{P}$ goes through the following steps:*
  1. ***Phase 1.*** *For $i = 0, \ldots, \ell^{(s)} - 1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X)$ to the verifier, where*

$$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} \widetilde{eq}\left(\mathbf{y}_{i+1..n};\mathbf{b}\right) F\left(\boldsymbol{\alpha}^{(s)}_{0..i-1}, X, \mathbf{b}\right) .$$

  2. ***Phase 2.*** *For $i = \ell^{(s)}, \ldots, n-1$, run the following steps:*
     (a) *Send the univariate polynomial $q^{(i)}(X)$ to the verifier, where*

$$q^{(i)}(X) = \sum_{\mathbf{b}\in\{0,1\}^{n-i-1}} \widetilde{eq}(\mathbf{y}_{i+1..n};\mathbf{b}) F(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..i}, X, \mathbf{b}) .$$

     (b) *Receive a challenge x_i from the verifier.*
- $\mathsf{TZC.Verify}^{F(\cdot)}_{n,d,s,\ell^{(s)}}(\sigma;\mathbf{y})$*: With the input $\sigma \in \mathcal{T}_\ell$ and the initial evaluation point $\mathbf{y}$, $\mathcal{V}$ runs the following steps:*
  1. *Set $\sigma^{(0)} = \sigma$.*
  2. ***Phase 1.*** *For $i = 0, \ldots, \ell^{(s)} - 1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ and check $\sigma^{(i)} = (1 - y_i)q^{(i)}(0) + y_i q^{(i)}(1)$.*
     (b) *Compute $\sigma^{(i+1)} = q^{(i)}(\alpha_i^{(s)})$.*
  3. ***Phase 2.*** *For $i = \ell^{(s)}, \ldots, n-1$, run the following steps:*
     (a) *Receive $q^{(i)}(X)$ and check $\sigma^{(i)} = (1 - y_i)q^{(i)}(0) + y_i q^{(i)}(1)$.*
     (b) *Randomly generate $x_i \leftarrow \mathcal{T}_\ell$ and send to the prover.*
     (c) *Compute $\sigma^{(i+1)} = q^{(i)}(x_i)$.*
  4. *Query the oracle $F_{\mathsf{eval}} = F\left(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..n}\right)$. Output $F_{\mathsf{eval}} \stackrel{?}{=} \sigma^{(n)}$.*

Figure 5: Tower Zerocheck Protocol

5 IOP Protocols over Tower Fields

In this section, we instantiate *tower IOP protocols* for three widely adopted arithmetic constraint systems, Plonkish, CCS, and GKR, using the algorithms developed in the previous section as fundamental building blocks. Before our work, protocols introduced by Chen et al. [14], Setty et al. [39] and Xie et al. [46] provided constructions based on large fields, named HyperPlonk, SuperSpartan, and Libra, respectively. However, these prior works did not explore methods to efficiently instantiate such protocols over small-characteristic fields, especially the fields with characteristic 2, other than employing the trivial approach. In this section, we fill the gap by proposing optimized instantiations, leveraging tower sumcheck and zerocheck protocols.

In the remainder of this paper, we fix the usage of Wiedemann's binary field tower and its derived towers, following Corollary 2. We adopt the notation $\mathfrak{T}_{2,2^s,\ell^{(s)}} = (\mathcal{T}_{0\cdot s}, \mathcal{T}_{1\cdot s}, \ldots, \mathcal{T}_{\ell^{(s)}\cdot s})$, where $\ell^{(s)} = \lfloor \ell/s \rfloor$ for $s \in \mathbb{N}^*$, to represent all potentially used tower fields.

5.1 Tower Protocol for Plonkish Relation

In this subsection, we present an IOP protocol for the Plonkish constraint system. Before introducing the main theorem, we propose the following definitions and lemmas.

Definition 9. *The relation $\mathcal{R}_{\mathsf{ZERO}}$ is the set of all tuples $(\mathbb{x}; \mathbb{w}) = (\mathcal{I}(\cdot); \mathcal{I}(\mathbf{X}))$ where $\mathcal{I}(\mathbf{X}) \in \mathcal{T}_0^{(\leq d)}[\mathbf{X}]$ and $\mathcal{I}(\mathbf{b}) = 0$ for all $\mathbf{b} \in \{0,1\}^n$.*

Lemma 7. *There exists a tower protocol to prove $\mathcal{R}_{\mathsf{ZERO}}$ over the field tower $\mathfrak{T}_{2,2^s,\ell^{(s)}}$ for some $2^s > d$, with perfect completeness and negligible soundness error.*

Proof. Because $(\mathcal{I}(\cdot); \mathcal{I}(\mathbf{X})) \in \mathcal{R}_{\mathsf{ZERO}}$ is equivalent to $(0, \mathbf{y}, \mathcal{I}(\cdot); \mathcal{I}(\mathbf{X})) \in \mathcal{R}_{\mathsf{IDENT}}$ with a randomly generated $\mathbf{y}$, then from Theorem 8, we derive this lemma.

Definition 10. *The indexed relation $\mathcal{R}_{\mathsf{PERM}}$ is defined as the set of tuples*

$$(\mathbb{i}; \mathbb{x}; \mathbb{w}) = (\mathsf{perm}(\mathbf{X}); f(\cdot), g(\cdot); f(\mathbf{X}), g(\mathbf{X})) \ ,$$

where

- *when evaluated on the Boolean hypercube, $\mathsf{perm} = \left(\mathsf{perm}_0, \ldots, \mathsf{perm}_{n-1}\right) : \{0,1\}^n \to \{0,1\}^n$ is a permutation function;*
- *each $\mathsf{perm}_j : \{0,1\}^n \to \{0,1\}$ (for $0 \leq j < n$) defines a mapping from $\mathbf{b}$ to the j-th bit of the destination index;*
- *$f(\mathbf{X}), g(\mathbf{X}) \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}]$ and $g(\mathbf{b}) = f(\mathsf{perm}(\mathbf{b}))$ for all $\mathbf{b} \in \{0,1\}^n$*

Lemma 8. *There exists a tower protocol to prove $\mathcal{R}_{\mathsf{PERM}}$ over $\mathfrak{T}_{2,2^s,\ell^{(s)}}$ for some $s \geq 2$, with perfect completeness and negligible soundness error.*

Proof. To prove the permutation relation, $(\mathsf{perm}(\mathbf{X}); f(\cdot), g(\cdot); f(\mathbf{X}), g(\mathbf{X})) \in \mathcal{R}_{\mathsf{PERM}}$, with a challenge $r \in \mathcal{T}_\ell$, we define $f'(\mathbf{X}), g'(\mathbf{X}), v(\mathbf{X}) \in \mathcal{T}_\ell[\mathbf{X}]$ as fol-

lows:

$$
\begin{aligned}
f'(\mathbf{X}) &:= r + \widetilde{eq}\left(\alpha^{(s)}_{0..\lceil\log(n+1)\rceil}; \langle n\rangle^{(2)}_{0..\lceil\log(n+1)\rceil}\right) f(\mathbf{X}) \\
&\quad + \sum_{j=0}^{n-1} \widetilde{eq}\left(\alpha^{(s)}_{0..\lceil\log(n+1)\rceil}; \langle j\rangle^{(2)}_{0..\lceil\log(n+1)\rceil}\right) \mathsf{perm}_j(\mathbf{X}) \ , \\
g'(\mathbf{X}) &:= r + \widetilde{eq}\left(\alpha^{(s)}_{0..\lceil\log(n+1)\rceil}; \langle n\rangle^{(2)}_{0..\lceil\log(n+1)\rceil}\right) g(\mathbf{X}) \\
&\quad + \sum_{j=0}^{n-1} \widetilde{eq}\left(\alpha^{(s)}_{0..\lceil\log(n+1)\rceil}; \langle j\rangle^{(2)}_{0..\lceil\log(n+1)\rceil}\right) \mathbf{X}_j, \\
v(0,\mathbf{b}) &:= \begin{cases} f'(\mathbf{b})/g'(\mathbf{b}) & \text{, if } g'(\mathbf{b}) \neq 0 \\ 0 & \text{, otherwise} \end{cases} \text{ for } \mathbf{b} \in \{0,1\}^n, \\
v(1,\mathbf{b}) &:= v(\mathbf{b},0) \cdot v(\mathbf{b},1) \text{ for } \mathbf{b} \in \{0,1\}^n,
\end{aligned}
$$

and then run the traditional zerocheck protocol on $\mathcal{T}_\ell$ on the following identity:

$$
\begin{aligned}
0 &= \sum_{\mathbf{b}\in\{0,1\}^n} \widetilde{eq}(\mathbf{Y},\mathbf{b}) \cdot (v(1,\mathbf{b}) - v(\mathbf{b},0) \cdot v(\mathbf{b},1)) \\
0 &= \sum_{\mathbf{b}\in\{0,1\}^n} \widetilde{eq}(\mathbf{Y},\mathbf{b}) \cdot g'(\mathbf{b}) \cdot (g'(\mathbf{b}) \cdot v(0,\mathbf{b}) - f'(\mathbf{b}))
\end{aligned}
$$

Using the building blocks above, we define *HyperPlonk Indexed Relation* as follows:

Definition 11. *(HyperPlonk Indexed Relation (Rewrite) [14, Definition 4.1]) Fix public parameters* $\mathsf{pp} := (\mathfrak{T}_{2,2,\ell}, M_{\mathsf{io}}, N, M_{\mathsf{wit}}, M_{\mathsf{sel}}, \mathcal{I})$ *where* $\mathfrak{T}_{2,2,\ell}$ *is Wiedemann's field tower,* $M_{\mathsf{io}} = 2^{m_{\mathsf{io}}}$ *is the public input length,* $N = 2^n$ *is the number of constraints,* $M_{\mathsf{wit}} = 2^{m_{\mathsf{wit}}}$ *and* $M_{\mathsf{sel}} = 2^{m_{\mathsf{sel}}}$ *are the number of witnesses and selectors per constraint, and* $\mathcal{I} : \mathcal{T}_0^{M_{\mathsf{wit}}+M_{\mathsf{sel}}} \to \mathcal{T}_0$ *is an algebraic map with degree* $d < t^s$*. The indexed relation* $\mathcal{R}_{\mathsf{PLONK}}$ *is the set of all tuples* $(\mathtt{i}, \mathtt{x}, \mathtt{w})$ *where* $\mathsf{perm}; \{0,1\}^{n+m_{\mathsf{wit}}} \to \{0,1\}^{n+m_{\mathsf{wit}}}$ *is a permutation, and*

$$
\begin{aligned}
\mathtt{i} &:= \left(\mathsf{sel}(\mathbf{X}) \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}_{0..n+m_{\mathsf{sel}}}], \mathsf{perm}(\mathbf{X}) \in \left(\mathcal{T}_0^{(\leq 1)}[\mathbf{X}_{0..m_{\mathsf{io}}}]\right)^n\right) \ ; \\
\mathtt{x} &:= \left(\mathsf{io}(\mathbf{X}) \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}_{0..m_{\mathsf{io}}}], \mathsf{wit}(\cdot)\right) \ ; \\
\mathtt{w} &:= \left(\mathsf{wit}(\mathbf{X}) \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}_{0..n+m_{\mathsf{wit}}}]\right) \ ;
\end{aligned}
$$

such that

- *the wiring identity is satisfied, that is* $(\mathsf{perm}(\mathbf{X}); (\mathsf{wit}(\cdot), \mathsf{wit}(\cdot)); \mathsf{wit}(\mathbf{X})) \in \mathcal{R}_{\mathsf{PERM}}$ *(Definition 10);*
- *the gate identity is satisfied, that is,* $(\mathcal{I}(\cdot); \mathcal{I}(\mathbf{X})) \in \mathcal{R}_{\mathsf{ZERO}}$ *(Definition 9), where the virtual polynomial* $\mathcal{I}(\mathbf{X}) \in \mathcal{T}_0^{(\leq d)}[\mathbf{X}_{0..n}]$ *is defined as:*

$$
\begin{aligned}
\mathcal{I}(\mathbf{X}) := \mathcal{I}\Big(&\mathsf{sel}(\langle 0\rangle^{(2)}_{0..m_{\mathsf{sel}}}, \mathbf{X}), \ldots, \mathsf{sel}(\langle M_{\mathsf{sel}} - 1\rangle^{(2)}_{0..m_{\mathsf{sel}}}, \mathbf{X}), \\
&\mathsf{wit}(\langle 0\rangle^{(2)}_{0..m_{\mathsf{wit}}}, \mathbf{X}), \ldots, \mathsf{wit}(\langle M_{\mathsf{wit}} - 1\rangle^{(2)}_{0..m_{\mathsf{wit}}}, \mathbf{X})\Big)
\end{aligned} \tag{18}
$$

– *the public input is consistent with the witness, that is, the public input polynomial* $\mathsf{io}(\mathbf{X}) \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}_{0..m_{\mathsf{io}}}]$ *is identical to* $\mathsf{wit}(\mathbf{0}^{n+m_{\mathsf{wit}}-m_{\mathsf{io}}}, \mathbf{X}) \in \mathcal{T}_0^{(\leq 1)}[\mathbf{X}_{0..m_{\mathsf{io}}}]$, *where* $\mathbf{0}^{n+m_{\mathsf{wit}}-m_{\mathsf{io}}}$ *denotes the array consisting of* $n + m_{\mathsf{wit}} - m_{\mathsf{io}}$ *zeros.*

With the Plonkish constraint system defined above, we have the following theorem:

Theorem 9. *Let* $\mathsf{pp} := (\mathfrak{T}_{2,2,\ell}, M_{\mathsf{io}}, N, M_{\mathsf{wit}}, M_{\mathsf{sel}}, \mathcal{I})$ *be the public parameters where* $M_{\mathsf{wit}}, M_{\mathsf{sel}} = O(1)$ *are some constants and* $d := \deg(\mathcal{I})$. *There exists a multivariate PIOP over the field tower* $\mathfrak{T}_{2,2^s,\ell^{(s)}}, s = \lfloor \log(d) \rfloor + 1$, *for relation* $\mathcal{R}_{\mathsf{PLONK}}$ *with perfect completeness and negligible soundness error.*

Proof. After instantiating Lemma 7 with the field tower $\mathfrak{T}_{2,2^{\lfloor \log(d) \rfloor + 1},\ell^{(\lfloor \log(d) \rfloor + 1)}}$ and Definition 10 with the field tower $\mathfrak{T}_{2,2,\ell}$, we can derive this result.

5.2 Tower Protocol for CCS Relation

In this subsection, we present a tower IOP protocol for the CCS constraints.

Definition 12. *(CCS (Rewrite) [39, Definition 2.2]) Fix public parameters* $\mathsf{pp} = (\mathfrak{T}_{2,2,\ell}, M, N, N_{\mathsf{nz}}, M_{\mathsf{io}}, m_M, m_S, d \in \mathbb{N})$ *where* $\mathfrak{T}_{2,2,\ell}$ *is Wiedemann's field tower,* M *is the number of constraints,* N *is the size of public and private witnesses,* N_{nz} *is the number of non-zero entries in the constraint matrices,* M_{io} *is the public input,* m_M, m_S *is the number of constraint matrices and sets defined below. We require that* $N > M_{\mathsf{io}}$.

The indexed relation $\mathcal{R}_{\mathsf{CCS}}$ *is the set of all tuples* $(\mathbb{i} := (\mathbb{i}_C, \mathbb{i}_S, \mathbb{i}_c), \mathbb{x} \in \mathcal{T}_0^{M_{\mathsf{io}}}, \mathbb{w} \in \mathcal{T}_0^{N-M_{\mathsf{io}}-1})$ *where*

– *a sequence of matrices* $\mathbb{i}_C = \left(C_0, \ldots, C_{m_M-1} \in \mathbb{F}^{M \times N}\right)$ *with at most* $N_{\mathsf{nz}} = \Omega(\max(M, N))$ *non-zero entries in total;*
– *a sequence of* m_S *multisets* $\mathbb{i}_S = (S_0, \ldots, S_{m_S-1})$, *where an element in each multiset is from the domain* $\{0, \ldots, m_M - 1\}$; *and the cardinality of each multiset is at most* d.
– *a sequence of* m_S *constants* $\mathbb{i}_c = (c_0, \ldots, c_{m_S-1})$, *where each constant is from* $\mathcal{T}_0$;

such that

$$\sum_{i=0}^{m_S-1} c_i \cdot \bigcirc_{j \in S_i} C_j \cdot W = \mathbf{0} \tag{19}$$

Here, $W = (\mathbb{w}, 1, \mathbb{x}) \in \mathcal{T}_0^N$, $C_j \cdot W$ *denotes matrix-vector multiplication,* $\bigcirc$ *denotes the Hadamard product between vectors, and* $\mathbf{0}$ *is an* M*-sized vector with entries equal to the additive identity in* $\mathcal{T}_0$.

With the CCS structure defined above, we have the following theorem.

Theorem 10. *There exists a tower IOP protocol over the field tower* $\mathfrak{T}_{2,2^s,\ell^{(s)}}, s = \lfloor \log(\max\{d, 3\}) \rfloor + 1$ *for* $\mathcal{R}_{CCS}$ *with perfect completeness and negligible soundness error.*

Proof. Firstly, we construct $\mathfrak{T}_{2,2^s,\ell^{(s)}}$ from $\mathfrak{T}_{2,2,\ell}$ with $s = \lfloor \log(\max\{d,3\}) \rfloor + 1$, following Corollary 2. Leveraging the field tower, the proving process can be split into two stages.

In the first stage, we define $f_i(\mathbf{Y}) := \sum_{\mathbf{b}\in\{0,1\}^n} C_i(\mathbf{Y},\mathbf{b})W(\mathbf{X})$, and then prove the following identity:

$$0(\mathbf{Z}) = \sum_{\mathbf{b}\in\{0,1\}^m} \widetilde{eq}(\mathbf{Z},\mathbf{b}) \sum_{i=0}^{m_S-1} c_i \cdot \prod_{j\in S_i} f_j(\mathbf{b})$$

which can be proved with Protocol 5.

In the second stage, we prove the following identity:

$$\begin{aligned}\mathsf{out}(\mathbf{Y}',\mathbf{Y}) &= \sum_{\mathbf{b}'\in\{0,1\}^{m_S}} \widetilde{eq}(\mathbf{b}',\mathbf{Y}')f_j(\mathbf{Y}) \\ &= \sum_{\mathbf{b}\in\{0,1\}^n} \left(\sum_{\mathbf{b}'\in\{0,1\}^{m_S}} \widetilde{eq}(\mathbf{b}',\mathbf{Y}')C_{\mathbf{b}'}(\mathbf{Y},\mathbf{b}) \right) W(\mathbf{b})\end{aligned}$$

which can be proved with Protocol 4.

5.3 Tower Protocol for GKR Relation

In this subsection, we present a tower IOP protocol for the GKR constraints.

Definition 13 (GKR). *Fix* $\mathsf{pp} = \left(\mathfrak{T}_{2,2,\ell}, D, (S_i)_{0\le i<D}, M_{\mathsf{io}} \in \mathbb{N}\right)$ *where* $\mathfrak{T}_{2,2,\ell}$ *is Wiedemann's field tower,* D *is the number of layers, where Layer* D *is the input and Layer* 0 *is the output.* $S_i = 2^{s_i}$ *is the size of each layer, and* M_{in} *are number of the public inputs. We require that* $S_D \ge M_{\mathsf{in}}$.

The indexed relation $\mathcal{R}_{\mathsf{GKR}}$ *is the set of all tuples* $(\mathbb{i} := \{(\mathsf{add}_i, \mathsf{mul}_i)\}_{0\le i<D}, \mathbb{x} \in \mathcal{T}_0^{M_{\mathsf{io}}}, \mathbb{w} \in \mathcal{T}_0^{S_D - M_{\mathsf{io}} - 1})$ *where* $\mathsf{mul}_i(z,x,y) = 1$ *if and only if there is a gate computing* $V_{i,z} = V_{i+1,x} \times V_{i+1,y}$*, otherwise,* $\mathsf{mul}_i(z,x,y) = 0$*, and* $\mathsf{add}_i(z,x,y) = 1$ *if and only if there is a gate computing* $V_{i,z} = V_{i+1,x} + V_{i+1,y}$*, otherwise,* $\mathsf{add}_i(z,x,y) = 0$.

If $(\mathbb{i} := \{(\mathsf{add}_i, \mathsf{mul}_i)\}_{0\le i<D}, \mathbb{x} \in \mathcal{T}_0^{M_{\mathsf{io}}}, \mathbb{w} \in \mathcal{T}_0^{S_D - M_{\mathsf{io}} - 1}) \in \mathcal{R}_{\mathsf{GKR}}$*, then there exists* $V_0, V_1, \ldots, V_D = (\mathbb{x}, \mathbb{w})$ *such that:*

$$V_{i,z} = \sum_{x=0}^{S_{i+1}-1} \sum_{y=0}^{S_{i+1}-1} (\mathsf{mul}_i(z,x,y)V_{i+1,x} \cdot V_{i+1,y} + \mathsf{add}_i(z,x,y)(V_{i+1,x} + V_{i+1,y}))$$

Theorem 11. *There exists a Tower IOP protocol over the field tower* $\mathfrak{T}_{2,2^s,\ell^{(s)}}, s = 3$ *for* $\mathcal{R}_{\mathsf{GKR}}$ *with perfect completeness and negligible soundness error.*

Proof. For the output layer, we prove the following identity:

$$\begin{aligned}\widetilde{V_i}(\mathbf{Z}) =& \sum\nolimits_{\mathbf{b}_x,\mathbf{b}_y\in\{0,1\}^{s_{i+1}}} f_i(\mathbf{Z},\mathbf{b}_x,\mathbf{b}_y) \\ =& \sum\nolimits_{\mathbf{b}_x,\mathbf{b}_y\in\{0,1\}^{s_{i+1}}} \widetilde{\mathsf{mul}}_{i+1}(\mathbf{Z},\mathbf{b}_x,\mathbf{b}_y)\widetilde{V}_{i+1}(\mathbf{b}_x)\widetilde{V}_{i+1}(\mathbf{b}_y) \\ &+ \sum\nolimits_{\mathbf{b}_x,\mathbf{b}_y\in\{0,1\}^{s_{i+1}}} \widetilde{\mathsf{add}}_{i+1}(\mathbf{Z},\mathbf{b}_x,\mathbf{b}_y)(\widetilde{V}_{i+1}(\mathbf{b}_x) + \widetilde{V}_{i+1}(\mathbf{b}_y)),\end{aligned}$$

where $\widetilde{V_i}$ is the MLE for V_i with s_i variables, $\widetilde{\mathsf{add}}$, $\widetilde{\mathsf{mul}}$ are the MLEs for add and mul. For the other layers, since there are two claims derived from the proved layers, the identity becomes:

$$\begin{aligned}
&\widetilde{V_i}(\mathbf{Z}) + W \cdot \widetilde{V_i}(\mathbf{Z}') = \sum\nolimits_{\mathbf{b}_x, \mathbf{b}_y \in \{0,1\}^{s_{i+1}}} f_i(\mathbf{Z}, \mathbf{Z}', W, \mathbf{b}_x, \mathbf{b}_y) \\
&= \sum_{\substack{\mathbf{b}_x, \mathbf{b}_y \in \\ \{0,1\}^{s_{i+1}}}} (\widetilde{\mathsf{mul}}_{i+1}(\mathbf{Z}, \mathbf{b}_x, \mathbf{b}_y) + W \cdot \widetilde{\mathsf{mul}}_{i+1}(\mathbf{Z}', \mathbf{b}_x, \mathbf{b}_y))\widetilde{V}_{i+1}(\mathbf{b}_x)\widetilde{V}_{i+1}(\mathbf{b}_y) \\
&+ \sum_{\substack{\mathbf{b}_x, \mathbf{b}_y \in \\ \{0,1\}^{s_{i+1}}}} (\widetilde{\mathsf{add}}_{i+1}(\mathbf{Z}, \mathbf{b}_x, \mathbf{b}_y) + W \cdot \widetilde{\mathsf{add}}_{i+1}(\mathbf{Z}', \mathbf{b}_x, \mathbf{b}_y))(\widetilde{V}_{i+1}(\mathbf{b}_x) + \widetilde{V}_{i+1}(\mathbf{b}_y)) \ .
\end{aligned}$$

Both identities can be proved by Protocol 4.

6 Polynomial Commitment Scheme

Building on the previous sections, we now introduce a *basis-switching technique*, which plays a crucial role in compiling our binary-field IOP protocol into a succinct argument. Similar to the *ring-switching technique* in FRI-Binius, this method enables us to transform a small-field polynomial evaluation problem into an equivalent large-field polynomial evaluation problem, As a result, our approach seamlessly integrates with the existing large-field PCS, allowing flexible and efficient instantiations.

6.1 Basis-Switching Technique

In this subsection, we introduce the *basis-switching technique*, which enables efficient polynomial evaluation transformation over Wiedemann's field tower $\mathfrak{T}_{2,2,\ell} = \{\mathcal{T}_0, \ldots, \mathcal{T}_\ell\}$. Given a base-field polynomial $f(\mathbf{X}) \in {\mathcal{T}_0}^{(\leq 1)}[\mathbf{X}_{0..n}]$, and its evaluation $f_{\mathsf{eval}} = f\left(\left(\alpha_i^{(s)}\right)_{i=0..\ell^{(s)}}, \mathbf{x}_{\ell^{(s)}..n}\right)$, where $\left(\alpha_i^{(s)}\right)_{i=0..\ell^{(s)}}$ are the basis elements of some derived field tower $\mathfrak{T}_{2,2^s,\ell^{(s)}}$ from $\mathfrak{T}_{2,2,\ell}$, we define the following identity

$$f\left(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..n}\right) = \sum_{\mathbf{b} \in \{0,1\}^{\ell}_{\mathsf{pcs}}} \widetilde{eq}\left(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..\ell_{\mathsf{pcs}}}; \mathbf{b}\right) f(\mathbf{b}, \mathbf{x}_{\ell_{\mathsf{pcs}}..n}) \tag{20}$$

where $\mathcal{T}_{\mathsf{pcs}} = \mathcal{T}_{\ell_{\mathsf{pcs}}} \in \mathfrak{T}_{2,2,\ell}$.

From Theorem 8, we have the following corollary:

Corollary 3 (Basis-Switching Protocol over Tower Fields). *There exists an tower zerocheck protocol over the field tower $\mathfrak{T}_{2,2,\ell}$ to reduce the evaluation from $f\left(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..n}\right)$ to $f(\boldsymbol{\alpha}, \mathbf{x}_{\ell_{\mathsf{pcs}}..n})$, with the perfect completeness and $O(\frac{(\ell_{\mathsf{pcs}}+1)(n-\ell^{(s)})}{|\mathcal{T}_\ell|})$ soundness error.*

Proof. Let $\mathbf{y} = \left(\boldsymbol{\alpha}^{(s)}, \mathbf{x}_{\ell^{(s)}..n}\right)$, $\sigma = f(\mathbf{y})$. Equation 20 is equivalent to

$$(\sigma, \mathbf{y}, f(\cdot); f(\mathbf{X})) \in \mathcal{R}_{\mathsf{IDENT}}$$

Considering $f(\mathbf{X})$ is an MLE, we run

$$\left\langle \mathsf{TZC.Prove}_{n,d,1,\ell_{\mathsf{pcs}}}(f(\mathbf{X}); \mathbf{y}), \mathsf{TZC.Verify}_{n,d,1,\ell_{\mathsf{pcs}}}(\sigma; \mathbf{y}) \right\rangle$$

in Protocol 5. In the end, the problem is reduced to the oracle query of $f(\boldsymbol{\alpha}, \mathbf{x}_{\ell_{\mathsf{pcs}}..n})$.

6.2 Small-field PCS to Large-field PCS Reduction

We define the small-field PCS $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$ for polynomials in $\mathcal{T}_0[\mathbf{X}]$ from the large-field PCS $\mathit{\Pi} = (\mathsf{Setup}', \mathsf{Commit}', \mathcal{P}', \mathcal{V}')$ for polynomials in $\mathcal{T}_\ell[\mathbf{X}]$ as follows:

- $\mathsf{Setup}(1^\lambda, \mathcal{T}_0[\mathbf{X}_{0..n}], \mathcal{T}_0) \to \mathsf{pp}$: compute $\mathsf{pp}' = \mathsf{Setup}'(1^\lambda, \mathcal{T}_{\mathsf{pcs}}[\mathbf{X}_{\ell_{\mathsf{pcs}}..n}])$, which includes a parameter $\mathcal{T}_{\mathsf{pcs}}$. Return pp'.
- $\mathsf{Commit}(f \in \mathcal{T}_0[\mathbf{X}_{0..n}], \mathsf{pp}) \to \mathsf{com}_f$: define the polynomial

$$f'(\mathbf{X}) \in \mathcal{T}_{\mathsf{pcs}}[\mathbf{X}_{0..n-\ell_{\mathsf{pcs}}}] = \sum_{\mathbf{b} \in \{0,1\}^{\ell_{\mathsf{pcs}}}} \widetilde{eq}(\boldsymbol{\alpha}_{0..\ell_{\mathsf{pcs}}}, \mathbf{b}) f(\mathbf{b}, \mathbf{X}_{0..n-\ell_{\mathsf{pcs}}}) \ . \tag{21}$$

  Run $\mathsf{com}_{f'} = \mathsf{Commit}'(f', \mathsf{pp})$ and return $\mathsf{com}_{f'}$.
- $b \leftarrow \langle \mathcal{P}(\mathsf{com}_f, y, \mathbf{x}; f), \mathcal{V}(\mathsf{com}_f, y, \mathbf{x}) \rangle$. Both parties run Protocol 5 to prove the identity Equation 20. After the final step, with the oracle query $y' = f'(\mathbf{x}_{\ell_{\mathsf{pcs}}..n})$, return

$$b' \leftarrow \left\langle \mathcal{P}'(\mathsf{com}_f, y', \mathbf{x}_{\ell_{\mathsf{pcs}}..n}; f'), \mathcal{V}'(\mathsf{com}_f, y', \mathbf{x}_{\ell_{\mathsf{pcs}}..n}) \right\rangle \ .$$

Theorem 12. *If $\mathit{\Pi}' = (\mathsf{Setup}', \mathsf{Commit}', \mathcal{P}', \mathcal{V}')$ is complete and sound, then so is $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$.*

Remark 1. Our basis-switching protocol only requires a constant-round sum-check protocol and $O\left(2^{\ell_{\mathsf{pcs}}}\right) \mathcal{M}_\ell$ prover time. This is much cheaper than the ring-switching technique in FRI-Binius, which is $O\left(2^{n-\ell_{\mathsf{pcs}}} \cdot \mathsf{poly}(\lambda, n)\right)$.

Remark 2. We can generalize the PCS construction to $\mathfrak{T}_{p,t,\ell}, t = 2^s$ with basis elements $(\alpha_0, \ldots, \alpha_{\ell-1})$, by using

$$\left(\alpha_0^{2^{s-1}}, \alpha_0^{2^{s-2}}, \ldots, \alpha_0^{2^0}, \ldots, \alpha_{\ell-1}^{2^{s-1}}, \alpha_{\ell-1}^{2^{s-2}}, \ldots, \alpha_{\ell-1}^{2^0}\right)$$

instead during the zerocheck protocol.

Acknowledgments. We express our gratitude to Ji Luo for suggesting the intuition behind the proof of Lemma 5. We also thank Tianren Liu and Binyi Chen for their valuable discussions on a more general case of Lemma 5. Additionally, we appreciate SungMing Wu for engaging in early discussions on some ideas of the paper and Yuncong Zhang for pointing out several typographical errors.

References

1. a16z: Binius - JoltBook — jolt.a16zcrypto.com. https://jolt.a16zcrypto.com/background/binius.html, [Accessed 11-02-2025]
2. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: lightweight sublinear arguments without a trusted setup. DCC **91**(11), 3379–3424 (2023). https://doi.org/10.1007/s10623-023-01222-8
3. Arnon, G., Chiesa, A., Fenzi, G., Yogev, E.: STIR: Reed-solomon proximity testing with fewer queries. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 380–413. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68403-6_12
4. Arnon, G., Chiesa, A., Fenzi, G., Yogev, E.: WHIR: Reed–solomon proximity testing with super-fast verification. Cryptology ePrint Archive, Paper 2024/1586 (2024), https://eprint.iacr.org/2024/1586
5. Arun, A., Setty, S.T.V., Thaler, J.: Jolt: SNARKs for virtual machines via lookups. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 3–33. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58751-1_1
6. Ashur, T., Mahzoun, M., Posen, J., Šijačić, D.: Vision mark-32: ZK-friendly hash function over binary tower fields. Cryptology ePrint Archive, Report 2024/633 (2024), https://eprint.iacr.org/2024/633
7. Bagad, S., Domb, Y., Thaler, J.: The sum-check protocol over fields of small characteristic. Cryptology ePrint Archive, Report 2024/1046 (2024), https://eprint.iacr.org/2024/1046
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Prague, Czech Republic (Jul 9–13, 2018). https://doi.org/10.4230/LIPIcs.ICALP.2018.14
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), https://eprint.iacr.org/2018/046
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer Berlin Heidelberg, Germany, Athens, Greece (May 26–30, 2013). https://doi.org/10.1007/978-3-642-38348-9_19
11. Block, A.R., Fang, Z., Katz, J., Thaler, J., Waldner, H., Zhang, Y.: Field-agnostic SNARKs from expand-accumulate codes. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 276–307. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68403-6_9
12. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press, San Francisco, CA, USA (May 21–23, 2018). https://doi.org/10.1109/SP.2018.00020

13. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 677–706. Springer, Cham, Switzerland, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45721-1_24
14. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 499–530. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30617-4_17
15. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Goldwasser, S. (ed.) ITCS 2012. pp. 90–112. ACM, Cambridge, MA, USA (Jan 8–10, 2012). https://doi.org/10.1145/2090236.2090245
16. Dao, Q., Thaler, J.: Constraint-packing and the sum-check protocol over binary tower fields. Cryptology ePrint Archive, Report 2024/1038 (2024), https://eprint.iacr.org/2024/1038
17. Dao, Q., Thaler, J.: More optimizations to sum-check proving. Cryptology ePrint Archive, Report 2024/1210 (2024), https://eprint.iacr.org/2024/1210
18. Demillo, R.A., Lipton, R.J.: A probabilistic remark on algebraic program testing. Information Processing Letters **7**(4), 193–195 (1978). https://doi.org/10.1016/0020-0190(78)90067-4
19. Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. Cryptology ePrint Archive, Report 2023/1784 (2023), https://eprint.iacr.org/2023/1784
20. Diamond, B.E., Posen, J.: Polylogarithmic proofs for multilinears over binary towers. Cryptology ePrint Archive, Report 2024/504 (2024), https://eprint.iacr.org/2024/504
21. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
22. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953
23. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 113–122. ACM Press, Victoria, BC, Canada (May 17–20, 2008). https://doi.org/10.1145/1374376.1374396
24. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic SNARKs for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 193–226. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38545-2_7
25. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer Berlin Heidelberg, Germany, Vienna, Austria (May 8–12, 2016). https://doi.org/10.1007/978-3-662-49896-5_11
26. Gruen, A.: Some improvements for the PIOP for ZeroCheck. Cryptology ePrint Archive, Report 2024/108 (2024), https://eprint.iacr.org/2024/108
27. Haböck, U., Levit, D., Papini, S.: Circle STARKs. Cryptology ePrint Archive, Report 2024/278 (2024), https://eprint.iacr.org/2024/278

28. Holmgren, J., Rothblum, R.D.: Faster sounder succinct arguments and IOPs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 474–503. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15802-5_17
29. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press, Victoria, BC, Canada (May 4–6, 1992). https://doi.org/10.1145/129712.129782
30. Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 359–388. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15985-5_13
31. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. In: 31st FOCS. pp. 2–10. IEEE Computer Society Press, St. Louis, MO, USA (Oct 22–24, 1990). https://doi.org/10.1109/FSCS.1990.89518
32. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453. IEEE Computer Society Press, Santa Fe, NM, USA (Nov 20–22, 1994). https://doi.org/10.1109/SFCS.1994.365746
33. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press, Berkeley, CA, USA (May 19–22, 2013). https://doi.org/10.1109/SP.2013.47
34. project, P.: Plonky2. https://github.com/mir-protocol/plonky2
35. project, P.: Plonky3. https://github.com/Plonky3/Plonky3
36. Ron-Zewi, N., Rothblum, R.D.: Proving as fast as computing: succinct arguments with constant prover overhead. In: Leonardi, S., Gupta, A. (eds.) 54th ACM STOC. pp. 1353–1363. ACM Press, Rome, Italy (Jun 20–24, 2022). https://doi.org/10.1145/3519935.3519956
37. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM **27**(4), 701–717 (Oct 1980). https://doi.org/10.1145/322217.322225
38. Setty, S.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 704–737. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56877-1_25
39. Setty, S., Thaler, J., Wahby, R.: Customizable constraint systems for succinct arguments. Cryptology ePrint Archive, Report 2023/552 (2023), https://eprint.iacr.org/2023/552
40. Soukhanov, L.: Hashcaster. https://github.com/morgana-proofs/hashcaster
41. StarkWare: ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582 (2021), https://eprint.iacr.org/2021/582
42. Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 71–89. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40084-1_5
43. Thaler, J., et al.: Proofs, arguments, and zero-knowledge. Foundations and Trends® in Privacy and Security **4**(2–4), 117–660 (2022)
44. Wahby, R.S., Tzialla, I., shelat, a., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: 2018 IEEE Symposium on Security and Privacy. pp. 926–943. IEEE Computer Society Press, San Francisco, CA, USA (May 21–23, 2018). https://doi.org/10.1109/SP.2018.00060
45. Wiedemann, D.: An iterated quadratic extension of gf (2). The Fibonacci Quarterly **26**(4), 290–295 (1988). https://doi.org/10.1080/00150517.1988.12429608

46. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 733–764. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_24
47. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 299–328. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15985-5_11
48. Zeilberger, H., Chen, B., Fisch, B.: BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part X. LNCS, vol. 14929, pp. 138–169. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68403-6_5
49. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: 2020 IEEE Symposium on Security and Privacy. pp. 859–876. IEEE Computer Society Press, San Francisco, CA, USA (May 18–21, 2020). https://doi.org/10.1109/SP40000.2020.00052
50. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, E.W. (ed.) Proceedings of the International Symposiumon on Symbolic and Algebraic Computation. p. 216–226. EUROSAM '79, Springer-Verlag, Berlin, Heidelberg (1979). https://doi.org/10.5555/646670.698972