Transparent SNARKs over Galois Rings

Yuanju Wei[1,2][0009−0008−6778−9794], Xinxuan Zhang[1,2][0000−0002−2739−7656], and Yi Deng[1,2][0000−0001−5948−0780]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{weiyuanju, zhangxinxuan, deng}@iie.ac.cn

Abstract. Recently, there is a growing need for SNARKs to operate over a broader range of algebraic structures, and one important structure is Galois ring. We present transparent SNARK schemes over arbitrary Galois rings. Compared with Rinocchio scheme in Ganesh et al. (J Cryptol 2023), our SNARK schemes do not require a trusted third party to establish a structured reference string (SRS).

In this paper, we present the expander code over arbitrary Galois rings, which can be encoded in $O(n)$ time. Using this expander code, we then extend the Brakedown commitment scheme in Golovnev et al. (CRYPTO 2023) to Galois rings. By combining the Libra framework in Xie et al. (CRYPTO 2019), we present a transparent SNARK for log-space uniform circuits over Galois rings, achieving $O(n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time. And by combining HyperPlonk in Chen et al. (EUROCRYPT 2023), we present a transparent SNARK for NP circuits over Galois rings, with $O(n \log^2 n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time.

Keywords: SNARKs · Galois rings · polynomial commitment.

1 Introduction

Succinct Non-interactive Arguments of Knowledge (SNARK) are cryptographic protocols that allow a verifier to efficiently check the validity of any NP statement without interacting with the prover [25,27,20,8]. One of the most important security properties of SNARKs is soundness. Soundness demands that for any incorrect NP statement, it is infeasible for any prover to generate a proof that will pass verification. To ensure the soundness property, SNARK protocols are typically designed for arithmetic circuits over a large prime field $\mathbb{F}_p$. Recently, there is a growing demand to deploy them over more general algebraic structures. While arithmetic circuits over large prime fields $\mathbb{F}_p$ can simulate various algebraic structures, such simulation often comes at an expensive efficiency cost. It is more practical to design SNARKs directly for arithmetic circuits tailored to specific algebraic structures. Arithmetic circuits over rings are both a natural extension of field-based arithmetic circuits and hold significant value in various applications. One important use case is in Fully Homomorphic Encryption

(FHE), where computations are performed over rings, and the need for SNARKs to prove correctness in these settings has gained attention. In particular, second-generation FHE schemes like BGV and (B)FV[11,12] are defined over large integer rings. In this paper, we focus on Galois rings. Galois rings are a type of finite commutative rings which generalize both the finite fields and the rings of integers modulo a prime power. According to the Chinese Remainder Theorem, any integer ring (extension) can be mapped one-to-one to several Galois rings.

SNARKs for arithmetic circuits over rings have garnered significant attention, but there are relatively few solutions specifically tailored for such settings. The Rinocchio protocol, proposed by Ganesh et al. [19], is the first complete SNARK designed for ring-based arithmetic circuits. Rinocchio is a ring version of the Pinocchio [28] and the Groth16 [23]. These SNARKs are based on Linear PCP constructions, and they require a trusted setup to generate a structured reference string (SRS). In recent years, newer SNARK schemes have moved away from Linear PCP-based constructions, instead adopting a combination of PIOP and polynomial commitment schemes. Examples include Libra [33], Plonk [18], Spartan [29], Marlin [16], STARK [4,3,5], Brakedown [22], and Orion [34]. Many of these do not require a trusted third party to set up an SRS. However, none of these can be directly applied to arithmetic circuits over rings because, to date, there are no known polynomial commitment schemes for ring arithmetic.

Current polynomial commitment schemes over finite fields can generally be classified into three categories. The first category is the KZG commitment scheme based on pairing structures, proposed by Kate et al. [24]. And it also relies on an SRS. The second category is based on the hardness of the discrete logarithm problem, such as Bulletproofs proposed by Bünz et al. [13] and Dory proposed by Lee [26]. The third category is based on encoding techniques, such as FRI (Fast Reed-Solomon Interactive Oracle Proof of Proximity) proposed by Ben-Sasson et al. [3,5], Brakedown proposed by Golovnev et al. [22], and Basefold proposed by Zeilberger et al. [35]. These polynomial commitment schemes rely on algebraic structures applicable to finite fields, such as pairing structures, which are difficult to satisfy over Galois rings. Thus, this naturally leads to the following question:

Is it possible to construct polynomial commitments over Galois rings, and furthermore, is there a transparent SNARK scheme over Galois rings?

We found that the Brakedown scheme only relies on linear codes with fixed relative distance. Since linear codes are typically designed for finite fields, and given the "similarity" between Galois rings and finite fields, the Brakedown scheme appears to be the most feasible scheme to implement over Galois rings, and it does not require a trusted setup for generating an SRS. Therefore, we present the Brakedown commitment scheme over Galois rings and proposes a SNARK scheme over arbitrary Galois rings without the need for an SRS.

1.1 Our Contribution

We present a polynomial commitment and transparent SNARK schemes over arbitrary Galois rings in this paper.

We extend expander codes to arbitrary Galois rings, enabling linear-time encoding with a fixed relative distance that is a constant depending on the Galois ring. Then, using this encoding, we present the Brakedown polynomial commitment scheme over Galois rings. Combining this construction with Libra, we present a transparent SNARK for log-space uniform circuits over Galois rings, achieving $O(n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time. And by combining HyperPlonk, we present a transparent SNARK for NP circuits over Galois rings with $O(n \log^2 n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time.

1.2 Technique Overview

We revisit the Brakedown polynomial commitment. A multilinear polynomial f with l variables can be written as

$$f(x_1, \cdots, x_l) = \sum_{\mathbf{b} \in \{0,1\}^l} \prod_{i \in [1,l]} \left((1 - x_i)(1 - b_i) + x_i b_i\right) f(\mathbf{b})$$

The values of f over the hypercube can be viewed as the basis of the polynomial, resulting in 2^l basis elements. These 2^l elements can be arranged into a $2^{l/2} \times 2^{l/2}$ matrix S:

$$\begin{bmatrix} f(0, \cdots, 0, 0, \cdots, 0) & f(0, \cdots, 0, 0, \cdots, 1) & \cdots & f(0, \cdots, 0, 1, \cdots, 1) \\ f(0, \cdots, 1, 0, \cdots, 0) & f(0, \cdots, 1, 0, \cdots, 1) & \cdots & f(0, \cdots, 1, 1, \cdots, 1) \\ \vdots & \vdots & \vdots & \vdots \\ f(1, \cdots, 1, 0, \cdots, 0) & f(1, \cdots, 1, 0, \cdots, 1) & \cdots & f(1, \cdots, 1, 1, \cdots, 1) \end{bmatrix}$$

To evaluate f at a point $\mathbf{r}$, one computes

$$f(r_1, \cdots, r_l) = \sum_{\mathbf{b} \in \{0,1\}^l} \prod_{i \in [1,l]} \left((1 - r_i)(1 - b_i) + r_i b_i\right) f(\mathbf{b})$$

Using matrix S, this calculation involves two vectors, $\mathbf{s}_1$ and $\mathbf{s}_2$, each of length $2^{l/2}$. Specifically,

$$\mathbf{s}_1 = \big((1 - r_1, r_1) \otimes \cdots \otimes (1 - r_{l/2}, r_{l/2})\big)$$

and

$$\mathbf{s}_2 = \big((1 - r_{l/2+1}, r_{l/2+1}) \otimes \cdots \otimes (1 - r_l, r_l)\big)$$

Therefore, $f(\mathbf{r}) = \mathbf{s}_1^\top S \mathbf{s}_2$.

The core of the Brakedown commitment is to ensure that the prover can correctly compute $\mathbf{s}_1^\top S$ using encoding techniques and then send the result to the verifier, who completes the remaining computation. By employing encoding

methods as described in Ligero (AHIV17)[1], proposed by Ames et al., the verifier can ensure that the prover correctly forms the linear combination of the encoded vectors.

Linear Codes over Galois Rings. During the encoding process of expander codes over large prime fields $\mathbb{F}_q$, a crucial step is performing linear combinations on k non-zero elements. The linear combinations are performed $b(k)$ times, where $b(k)$ is a function of k and $b(k) = \max(k+4, 1.28k)$. Let z is the probability that a single linear combination is zero. According to the union bound, the probability that all $b(k)$ linear combinations of any k non-zero elements are zero is bounded by:

$$\Pr[b(k) \text{ all zeros}] \leq q^k z^{b(k)}$$

To ensure codeword distance, this probability must be negligible. While this is easily achievable in large prime fields, however, in Galois rings $\mathrm{GR}(p^s, r)$, zero divisors significantly increases this probability, braking the code distance.

A key observation to solve this problem is that in a Galois ring $\mathrm{GR}(p^s, r)$ all zero divisors reside in the ideal (p), meaning the proportion of zero divisors is $\frac{1}{p^r}$. If p^r is exponentially large relative to the security parameter λ, then the proportion of zero divisors becomes negligible. However, the total number of elements in the Galois ring is p^{sr}, and as p^r increases, the total number of elements in the Galois ring grows accordingly. We need that any k non-zero elements satisfy the distance condition. At this time, $z = \frac{1}{p^r}$, so we have

$$\Pr[b(k) \text{ all zeros}] \leq p^{ksr} \left(\frac{1}{p^r}\right)^{b(k)} = \frac{p^{ksr}}{p^{b(k)r}}$$

In this case, as long as $s \geq 2$, the probability will not be negligible.

We solve this problem by performing a more refined parameter analysis. By analyzing the distribution of zero divisors in the Galois ring, we categorize the zero divisors into s classes based on their membership in the ideals $(p^{s-1}), \ldots, (p^2), (p)$. Each element is classified according to the smallest ideal it belongs to. For instance, if an element is in the ideal (p^i), it is also contained in (p^{i-1}), but it is classified as being in the i-th class rather than the $(i-1)$-th class. We then analyze the impact of the zero divisors from each of these s classes on the code distance. As i increases, the probability that a random linear combination of elements from the ideal (p^i) results in zero increases, thus having a greater impact on the code's distance. However, the proportion of such zero divisors in the entire Galois ring decreases. Through this more detailed parameter analysis, we prove that expander codes can maintain a constant relative code distance with $1 -$ negligible probability in Galois rings $\mathrm{GR}(p^s, r)$, provided that p^r is sufficiently large (exponential scale relative to the security parameter λ).

If the Galois ring $\mathrm{GR}(p^s, r)$ does not satisfy the large condition, we draw inspiration from the block-level encoding approach introduced in [17] and adapt it from binary fields to arbitrary Galois rings. Specifically, if a Galois ring $\mathrm{GR}(p^s, r)$ requires an expansion by a factor of k to meet the necessary conditions, we treat k elements from $\mathrm{GR}(p^s, r)$ as a single element in $\mathrm{GR}(p^s, kr)$ during the encoding

process. We further demonstrate that this encoding method maintains linearity over $\mathrm{GR}(p^s, r)$. This construction thus results in a linear code defined over any Galois ring $\mathrm{GR}(p^s, r)$.

Polynomial Commitments over Galois Rings. Using the linear codes defined over Galois rings, we apply the Brakedown construction framework to present a polynomial commitment scheme over Galois rings. Since Brakedown commitments rely on the linear code detection lemma from AHIV17 [1], we must first discuss this lemma for arbitrary Galois rings. If a Galois ring $\mathrm{GR}(p^s, r)$ does not satisfy the condition that p^r is sufficiently large, we can say this ring is a "small ring". We need to account for cases where the polynomial to be committed is defined over a small ring.

Reviewing the construction of the Brakedown polynomial commitment, the verifier must ensure that the prover correctly performs a linear combination on each row of the coefficient matrix. In the encoding step, we mentioned that adjacent k elements in the small ring are treated as a single element in the extension ring for encoding. If the verifier directly selects a challenge e from the extension ring, then e will multiply with the larger element formed by the k elements from the small ring. This results in the "mixing" of the original k coefficients of the polynomial, meaning the operation is no longer within the small ring. In such scenarios, it is preferable for the verifier's challenge to be selected from $\mathrm{GR}(p^s, r)$ to ensure that the coefficients used in the linear combination of the codewords are drawn from $\mathrm{GR}(p^s, r)$. However, if the verifier selects challenges only from the small ring, it cannot guarantee the soundness of the polynomial commitment.

We employed Interleaved codes and Block-wise relative distance [9] to discuss the "repetition" version of the lemma over Galois rings to solve this problem: the verifier can ensure soundness by repeatedly selecting challenges multiple times. This approach, to some extent, avoids the need for field expansion operations, leading to an improvement in computational efficiency.

In practical applications of polynomial commitment, it is often encountered that polynomial coefficients come from a smaller ring $\mathrm{GR}(p^s, r)$, but for security reasons, the challenges are drawn from $\mathrm{GR}(p^s, kr)$, such as sumcheck. In [17], this issue is solved by using a two-dimensional extension. In this work, we explain the problem from the perspective of "repetition", using a more intuitive expression. When the prover computes $\mathbf{s}_1^\top S$, only $\mathbf{s}_1$ comes from $\mathrm{GR}(p^s, kr)$. Therefore, each element of $\mathbf{s}_1$ can be broken down into k-dimensional vectors, with each dimension being computed separately. This allows for a more efficient polynomial commitment process by avoiding the need to pad each element in $\mathrm{GR}(p^s, r)$ to the larger ring $\mathrm{GR}(p^s, kr)$ during the commitment phase.

SNARKs over Galois Rings. We combine polynomial commitments over Galois rings with PIOP (Polynomial Interactive Oracle Proof) to obtain SNARKs schemes over Galois rings. While sumcheck protocols over rings have been widely discussed, such as the extension to infinite non-commutative rings in [30], some PIOP frameworks those involving set consistency checks, cannot be directly ex-

tended from fields to Galois rings. This issue arises from the presence of zero divisors in Galois rings, which complicates set consistency checks.

In a field, to determine whether two sets S_1 and S_2 are identical, we encode them as polynomials, $p_1(x) = \prod_{a \in S_1}(x - a)$ and $p_2(x) = \prod_{a \in S_2}(x - a)$, and compare the polynomials. However, in Galois rings, zero divisors can result in different sets being encoded as identical polynomials. For example, under modulo 8, we have $(x - 1)(x - 7) = (x - 5)(x - 3) \pmod 8$.

So we selected an IOP framework that avoids set consistency checks: Libra and HyperPlonk. Using Libra, we constructed a transparent SNARK scheme for log-space uniform circuits over Galois rings, with $O(n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time. Similarly, using HyperPlonk, we constructed a transparent SNARK for NP circuits over Galois rings, achieving $O(n \log^2 n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time.

1.3 Related Work

Proof Systems over Rings. Due to the widespread application of proof systems, some efforts have been made to extend these systems to broader algebraic structures. Several works aim to migrate proof systems to arithmetic circuits over rings. In [15], Chen et al. presented the sumcheck and GKR protocols to finite commutative rings, where the verifier's challenges are required to come from an exceptional set within the ring. In [10], Bootle et al. constructed a sumcheck protocol over rings and used it to solve the Rank-1 Constraint System (R1CS) problem over rings. Furthermore, Soria-Vazquez extended the sumcheck protocol to infinite non-commutative rings in [30]. These protocols are rooted in the information-theoretic framework. However, due to the absence of a polynomial commitment scheme, none of these protocols are complete SNARK schemes.

Ganesh et al. [19] proposed the Rinocchio protocol, the first SNARK scheme for arithmetic circuits over rings. This protocol is based on the Linear PCP framework and does not require a polynomial commitment, but it does rely on a trusted third party to generate a structured reference string (SRS), which can raise concerns in practical applications. In recent years, new SNARK schemes have been introduced that eliminate the need for an SRS but instead depend on polynomial commitments. However, there is no polynomial commitment schemes over rings. In this paper, we present the Brakedown polynomial commitment scheme over Galois rings and present a SNARK based on an PIOP + polynomial commitment framework, differing from Rinocchio. Unlike Rinocchio, the new SNARKs do not require an SRS.

Code based Polynomial Commitment. Some of the latest polynomial commitment schemes are code-based. Compared to previous approaches, these commitment schemes only rely on collision-resistant hash functions cryptographic primitive, and they are transparent.

The first representative scheme in this category is the FRI (Fast Reed-Solomon Interactive Oracle Proof of Proximity) scheme, proposed by Ben-Sasson et al. [3,5]. FRI-based polynomial commitments exploit the efficient detection of

Reed-Solomon codes to commit to polynomials. However, this scheme requires an "FFT-friendly" field and uses an iterative structure similar to FFT, resulting in a prover time complexity of $O(n \log n)$.

Another key scheme based on encoding is the Brakedown scheme, proposed by Golovnev et al. [22]. The construction of Brakedown is based on linear codes with fixed relative distances and the only cryptographic primitive used is a collision-resistant hash function. Brakedown provides linear-time prover efficiency but with larger proof sizes and slower verifier time.

Additionally, the recently proposed Basefold protocol by Zeilberger et al. [35] is also based on encoding techniques and serves as a trade-off between FRI and Brakedown. Although Basefold retains the same asymptotic complexity of $O(n \log n)$, it offers faster prover times compared to FRI and an improved verifier time of $O(\log n)$, whereas Brakedown has a verifier time of $O(\sqrt{n})$. The Basefold construction depends on foldable linear codes.

Given these comparisons, Brakedown has the minimum coding requirement and is the most "friendly" for Galois rings, which is why we choose the Brakedown framework.

2 Preliminaries

We denote a finite field by $\mathbb{F}$, a security parameter by λ, and a negligible function with respect to λ by $\mathrm{negl}(\lambda)$. We use PPT to denote probabilistic polynomial time. For any integer n, we define $\mathrm{Poly}(\mathbb{F}, n)$ as the set of polynomials with n variables and coefficients in the field $\mathbb{F}$.

2.1 Galois Rings

In this part, we will focus on discussing certain properties of Galois rings that will be used later. These properties are fundamental to establishing codes over Galois rings.

A Galois ring is constructed from the ring $\mathbb{Z}/p^s\mathbb{Z}$ similar to how a finite field $\mathbb{F}_{p^m}$ is constructed from $\mathbb{F}_p$ [32]. It is a Glaois extension of $\mathbb{Z}/p^s\mathbb{Z}$, when the concept of Galois extension is generalized beyond the context of fields.

Definition 1. *A Galois ring is a commutative ring of characteristic p^s which has p^{rs} elements, where p is a prime and s and r are positive integers. It is usually denoted $GR(p^s, r)$. It can be defined as a quotient ring*

$$GR(p^s, r) \cong \mathbb{Z}[x]/(p^s, f(x))$$

where $f(x) \in \mathbb{Z}[x]$ is monic polynomial of degree r which is irreducible modulo p. Up to isomorphism, the ring depends only on p, n, and r and not on the choice of f used in the construction.

Every Galois ring is a local ring. The unique maximal ideal is the principal ideal $(p) = p\mathrm{GR}(p^s, r)$, consisting of all elements which are multiples of p. Furthermore, $(0), (p^{s-1}), \cdots, (p), (1)$ are all the ideals.

If a Galois ring $\mathrm{GR}(p^s, r)$ does not satisfy the condition that p^r is exponential scale relative to the security parameter λ, we can say this ring is a "small ring".

Fact 1 *All zero divisors in the Galois ring $GR(p^s, r)$ are in the ideal (p).*

We define a ring homomorphism ϕ

$$\begin{aligned} \mathbb{Z}_{p^s} &\to \mathbb{F}_p \\ c_0 + c_1 p + \cdots + c_{p-1} p^{s-1} &\mapsto c_0 \end{aligned}$$

where $0 \le c_i \le p-1$ and its kernel is the ideal (p) of the ring $\mathbb{Z}_{p^s}$. And the ring homomorphism can be extended to ψ

$$\begin{aligned} \mathbb{Z}_{p^s}[x] &\to \mathbb{F}_p[x] \\ a_0 + a_1 x + \cdots + a_n x^n &\mapsto \bar{a}_0 + \bar{a}_1 x + \cdots + \bar{a}_n x^n \end{aligned}$$

where $\bar{a}_i = \phi(a_i)$ and the kernel of ψ is the ideal (p) of the ring $\mathbb{Z}_{p^s}[x]$. So the ideal $(h(x))$ is the ideal $(\bar{h}(x))$. So we induce a ring homomorphism Φ

$$\begin{aligned} \mathbb{Z}_{p^s}[x]/(h(x)) &\to \mathbb{F}_p[x]/(\bar{h}(x)) \\ a_0 + a_1 x + \cdots + a_{r-1} x^{r-1} + (h(x)) &\mapsto \bar{a}_0 + \bar{a}_1 x + \cdots + \bar{a}_{r-1} x^{r-1} + (\bar{h}(x)) \end{aligned}$$

The kernel of the ring homomorphism Φ is the ideal $(p + (h(x)))$ generated by $p + (h(x))$ in $(\mathbb{Z}_{p^s}[x]/h(x))$. By the fundamental theorem of homomoprphisms of rings

$$(\mathbb{Z}_{p^s}[x]/(h(x)))/(p + h(x)) \simeq \mathbb{F}_p/(\bar{h}(x))$$

From the above series of ring homomorphisms, we observe that the Galois ring is an algebraic structure closely related to a field. Consequently, many techniques applicable to fields can be smoothly transferred to rings.

Definition 2. *a is an element of ring $GR(p^s, r)$ and n is an integer. We define $gcd(a, n)$ as $gcd(a_0, \cdots, a_{r-1}, n)$. Where a is represented by $a_0 + a_1 x + \cdots + a_{r-1} x^{r-1}$.*

Fact 2 *Consider elements a and b in the Galois ring $GR(p^s, r)$. Let $d = gcd(a, p^s)$. The linear equation $ax = b$ has at most d^r solutions within $GR(p^s, r)$.*

Proof. If $\gcd(a, p^s) = 1$, then a is not in the ideal (p). According to the properties of the Galois ring, a is not a zero divisor in $\mathrm{GR}(p^s, r)$, implying that a has an inverse. Therefore, $x = b \cdot a^{-1}$ has a unique solution.

If $\gcd(a, p^s) = d$, then we have

$$ax = b \pmod{(p^s, f)}$$

The condition for this equation to have a solution is $d \mid b_i$ for all i, $0 \le i \le r-1$, where b_i is the coefficient in b. Thus, we have

$$\frac{a}{d} x = \frac{b}{d} \pmod{(p^s/d, f)}$$

where $\frac{a}{d}$ and $\frac{b}{d}$ mean each coefficient in a and b is divided by d. Let $\frac{a}{d} = a'$ and $\frac{b}{d} = b'$. Then a' and b' are elements of the Galois ring $\mathrm{GR}(p^s/d, r)$. Since $\gcd(a', p^s/d) = 1$, a' is not a zero divisor in the ring $\mathrm{GR}(p^s/d, r)$, so a' has an inverse. Therefore, x has a unique solution in the ring $\mathrm{GR}(p^s/d, r)$. This solution is obtained by determining each coefficient of x.

$$\begin{aligned} x_0 &\equiv c_0 \pmod{p^s/d} \\ x_1 &\equiv c_1 \pmod{p^s/d} \\ &\cdots \\ x_{r-1} &\equiv c_{r-1} \pmod{p^s/d} \end{aligned}$$

So

$$\begin{aligned} x_0 &\equiv c_0 + r_0\frac{p^s}{d} \pmod{p^s}, r_0 \in [0, d-1] \\ x_1 &\equiv c_1 + r_1\frac{p^s}{d} \pmod{p^s}, r_1 \in [0, d-1] \\ &\cdots \\ x_{r-1} &\equiv c_{r-1} + r_{r-1}\frac{p^s}{d} \pmod{p^s}, r_{d-1} \in [0, d-1] \end{aligned}$$

Obviously $r_i, i \in [0, d-1]$ has d values, so each coefficient x_i of x has at most d values, and x has at most d^r values. □

Fact 3 *In the Galois ring $GR(p^s, r)$, the probability that a degree-d polynomial f evaluates to zero at a randomly chosen point is at most $\frac{d}{p^r}$.*

Fact 4 *If a monic polynomial f is irreducible in the field $GF(p, r)$, then f is also irreducible in the Galois ring $GR(p^s, r)$.*

In Appendix A, we provide detailed proofs for 3 and 4.

Definition 3 (Exceptional Set [19]). *Let $A = \{a_1, \cdots, a_n\} \subset R$. We say that A is an exceptional set if $\forall i \neq j$, $a_i - a_j \in R^*$, where R^* is the set of all invertible elements in the ring R.*

Lemma 1 (Genralized Schwartz-Zippel Lemma [7,19]). *Let $f : R^n \to R$ be an n-variate nonzero polynomial. Let $A \subseteq R$ be an finite exceptional set. Let $\deg(f)$ denote the total degree of f. Then:*

$$\Pr_{\mathbf{a} \leftarrow A^n}[f(\mathbf{a}) = 0] \leq \frac{\deg(f)}{|A|}$$

2.2 SNARKs

We adapt the definition from Rinocchio [19]. Let $\mathcal{R}$ be an efficiently computable binary relation which consists of pairs of the form (x, w) where x is a statement

and w is a witness. Let $\mathbf{L}$ be the language associated with the relation $\mathcal{R}$, i.e.$\mathcal{L} = \{x|\exists w, \text{s.t.}\mathcal{R}(x,w) = 1\}$.

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

- $\text{Setup}(1^\lambda) \to (\sigma, \text{vk})$: take a security parameter λ and outputs a common (sturctured) reference string σ together with private verification information vk.
- $\text{Prove}(\sigma, x, w) \to \pi$: on input σ, a statement x and witness w, outputs an argument π.
- $\text{Verify}(\sigma, \text{vk}, x, \pi) \to 1/0$: on input σ, the private verification key vk, a statement x and a proof π, it outputs either 1 indicating accepting the argument or 0 for rejecting it.

Definition 4 (SNARK). *A triple of polynomial time algorithms* (**Setup, Prove**, **Verify**) *is a SNARK for an NP relation $\mathcal{R}$, if the following properties are satisfied:*

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

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

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

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

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

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

2.3 Polynomial Commitment Scheme

We adapt the definition from Brakedown [22].

Definition 5 (Polynomial Commitment Scheme). *A polynomial commitment scheme for multilinear polynomial over Galois Ring $GR(p^s, r)$ is a tuple of four protocols $PC = (Gen, Commit, Open, Eval)$:*

- *$pp \leftarrow Gen(1^\lambda, \mu)$ takes as input μ (the number of variables in a multilinear polynomial); produces public parameters pp.*
- *$\mathcal{C} \leftarrow Commit(pp, \mathcal{G}, \gamma)$: takes any input a $\mu-$variate multilinear polynomial over a Galois Ring $\mathcal{G} \in \text{Poly}(GR(p^s, r), \mu)$ and a random string γ produces a commitment $\mathcal{C}$.*

- $b \leftarrow \mathit{Open}(pp, \mathcal{C}, \mathcal{G}, \gamma)$*: verifies the opening of commitment* $\mathcal{C}$ *to the* $\mu-$*variate multilinear polynomial* $\mathcal{G} \in \mathrm{Poly}(GR(p^s, r), \mu)$*; outputs* $b \in \{0, 1\}$.
- $b \leftarrow \mathit{Eval}(pp, \mathcal{C}, z, v, \mu, \mathcal{G})$ *is a protocol between a PPT prover* $\mathcal{P}$ *and verifier* $\mathcal{V}$*. Both* $\mathcal{V}$ *and* $\mathcal{P}$ *hold a commitment* $\mathcal{C}$*, the number of variables* μ*, a scalar* $v \in GR(p^s, r)$ *and* $z \in GR(p^s, r)^\mu$*.* $\mathcal{P}$ *additionally knows a* $\mu-$*variate multilinear polynomial* $\mathcal{G} \in \mathrm{Poly}(GR(p^s, r), \mu)$*.* $\mathcal{P}$ *attempts to convince* $\mathcal{V}$ *that* $\mathcal{G}(z) = v$*. At the end of the protocol,* $\mathcal{V}$ *outputs* $b \in \{0, 1\}$.

A tuple of four protocol (Gen,Commit,Open,Eval) is an extractable polynomial commitment scheme for multilinear polynomials over Galois Ring $GR(p^s, r)$ *if the following conditions hold.*

- ***Completeness.*** *For any multilinear polynomial* $\mathcal{G} \in \mathrm{Poly}(GR(p^s, r), \mu)$,

$$\Pr\left[\begin{matrix} pp \rightarrow \mathit{Gen}(1^\lambda, \mu); \mathcal{C} \leftarrow \mathit{Commit}(pp, \mathcal{G}, \gamma) : \\ \mathit{Eval}(pp, \mathcal{C}, r, v, \mu, \mathcal{G}) = 1 \wedge v = \mathcal{G}(r) \end{matrix}\right] \geq 1 - \mathit{negl}(\lambda)$$

- ***Binding.*** *For any PPT adversary* $\mathcal{A}$*, size parameter* $\mu \geq 1$,

$$\Pr\left[\begin{matrix} pp \leftarrow \mathit{Gen}(1^\lambda, \mu); \mathcal{C}, \mathcal{G}_0, \mathcal{G}_1 \leftarrow \mathcal{A}(pp); \\ b_0 \leftarrow \mathit{Open}(pp, \mathcal{C}, \mathcal{G}_0, \gamma); b_1 \leftarrow \mathit{Open}(pp, \mathcal{C}, \mathcal{G}_1, \gamma); \\ b_0 = b_1 \neq 0 \wedge \mathcal{G}_0 \neq \mathcal{G}_1 \end{matrix}\right] \leq \mathit{negl}(\lambda)$$

- ***Knowledge soundness.*** *Eval is a succinct argument of knowledge for the following NP relation given* $pp \leftarrow \mathit{Gen}(1^\lambda, \mu)$.

$$\mathcal{R}_{\mathit{Eval}}(pp) = \{\langle (\mathcal{C}, z, v), (\mathcal{G}) \rangle : \mathcal{G} \in GR(p^s, r)[\mu] \wedge \mathcal{G}(z) = v \wedge \mathit{Open}(pp, \mathcal{C}, \mathcal{G}, \gamma) = 1\}$$

3 Linear Codes over Galois Rings

In SNARKs, the encoding is typically defined over a large prime field $\mathbb{F}_q$, as this ensures a large codeword distance and provides the verifier with a large challenge space. However, in the case of Galois rings, zero divisors can disrupt the code distance. In this section, we solve this problem and show expander codes over arbitrary Galois rings.

3.1 Linear Codes over Partial Galois Rings

We utilized the construction framework of expander codes from Brakedown [22] and use a similar approach when proving the code distance. When analyzing codeword distances, for lemmas (Lemma 2 and Lemma 3) that are applicable to large prime fields but not to Galois rings, we provide versions of these lemmas for Galois rings $\mathrm{GR}(p^s, r)$ and their proofs. We use the same encoding parameters as in Brakedown. While the failure probability in Brakedown does not exceed 2^{-100}, we proved that in Galois rings $\mathrm{GR}(p^s, r)$, the error probability does not exceed $s \cdot 2^{-100}$. In practical applications, s typically does not exceed 2^8, so our scheme remains secure.

Firstly, we need to define some parameters related to expander codes in Brakedown: $0 < \alpha < 1$, $0 < \beta < 1$, and $t > \frac{1+2\beta}{1-\alpha}$, with $c_n, d_n \geq 3$. Let $\mathcal{M}_{n,m,d} \subset \mathrm{GR}(p^s, r)^{n \times m}$ be a matrix distribution where each row has exactly d non-zero entries, with these d entries randomly selected from the non-zero elements of $\mathrm{GR}(p^s, r)$. For $x \in [0, 1]$, $H(x) = -x \log_2(x) - (1-x)\log_2(1-x)$.

$$
\begin{aligned}
c_n = \Bigg\lceil \min \Bigg(& \max\left(1.28\beta n, \beta n + 4\right), \\
& \frac{1}{\beta \log_2\left(\frac{\alpha}{1.28\beta}\right)} \left(\frac{100}{n} + H(\beta) + \alpha H\left(\frac{1.28\beta}{\alpha}\right)\right)\Bigg)\Bigg\rceil \\
d_n = \left\lceil \min\left(\left(2\beta + \frac{(t-1) + 100/n}{\log_2\left(p^r\right)}\right) n, D\right)\right\rceil & \\
D = \max\Bigg(& \frac{t\alpha H\left(\frac{\beta}{t}\right) + \mu H\left(\frac{\nu}{\mu}\right) + \frac{100}{n}}{\alpha\beta \log_2\left(\frac{\mu}{\nu}\right)}, \\
& \frac{t\alpha H\left(\frac{\beta}{t}\right) + \mu H\left(\frac{2\beta+0.03}{\mu}\right) + \frac{100}{n}}{\beta \log_2\left(\frac{\mu}{2\beta+0.03}\right)}, \\
& (2\beta + 0.03)\left(\frac{1}{\alpha t - \beta} + \frac{1}{\alpha\beta} + \frac{1}{\mu - 2\beta - 0.03}\right) + 1\Bigg)
\end{aligned}
$$

where $\mu = t - 1 - t\alpha, \nu = \beta + \alpha\beta + 0.03$.

Algorithm 1: Enc Algorithm: $\mathrm{GR}(p^s, r)^n \rightarrow \mathrm{GR}(p^s, r)^{tn}$

Input: $\mathbf{x} \in \mathrm{GR}(p^s, r)^n$
parameter: $\alpha, \beta, t, c_n, d_n$
Output: $\mathbf{w} \in \mathrm{GR}(p^s, r)^{tn}$

if $n < n_0$ **then**
  $\mathbf{w} = M\mathbf{x}$ and return $\mathbf{w}$;
Matrices $A^{(n)} \leftarrow \mathcal{M}_{n,\alpha n,c_n}$ and $B^{(n)} \leftarrow \mathcal{M}_{\alpha t n,(t-1-t\alpha)n,d_n}$ for are choosen in pre-processing;
$\mathbf{y}^\top = \mathbf{x}^\top \cdot A^{(n)} \in \mathrm{GR}(p^s, r)^{\alpha n}$;
$\mathbf{z} = \mathrm{Enc}(\mathbf{y}) \in \mathrm{GR}(p^s, r)^{t\alpha n}$;
$\mathbf{v}^\top = \mathbf{z}^\top \cdot B^{(n)} \in R^{(t-1-t\alpha)n}$;
$\mathbf{w} = \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \\ \mathbf{v} \end{pmatrix} \in \mathrm{GR}(p^s, r)^{tn}$;
return $\mathbf{w}$

Here, n_0 is a small constant and M can be any generator matrix that meets the code rate and distance parameters, such as selecting an Reed-Solomon code

and using padding zeros to meet the requirements. The Algorithm 1 is the encoding algorithm over Galois rings. We prove that by choosing α, β, t, c_n, and d_n with the same parameters as in the Brakedown scheme, it can still ensure the relative code distance over Galois rings with a probability of $1 - \mathrm{negl}(\lambda)$.

Since the proof framework for the relative code distance of expander codes over Galois rings is similar to that in Brakedown[22], we only highlight the differences specific to Galois rings here. The complete proof is provided in Appendix B.

Define the four events $E_{n,k}^{(1)}$, $E_{n,k}^{(2)}$, $E_{n,k}^{(3)}$, and $E_{n,k}^{(4)}$.

As explained in Brakedown [22], as long as the probability of these four events occurring does not exceed $s \cdot 2^{-100}$, Then, the relative code distance holds with a probability of $1 - s \cdot 2^{-100}$.

- $E_{n,k}^{(1)}$: There exists a set of k coordinates of $\mathbf{x} \in \mathrm{GR}(p^s, r)^n$ that doesn't "expend" into $b(k) = max(k + 4, 1.28k)$ coordinates of $\mathbf{x}^\top \cdot A$.
- $E_{n,k}^{(2)}$: Given the event that, given that every set of size k expands into a set of size at least $b(k)$ (that is, conditioned on the complement of $E_{n,k}^{(1)}$), there exists an $\mathbf{x}$ of Hamming weight $\|\mathbf{x}\|_0 = h$ such that $\mathbf{y}^\top = \mathbf{x}^\top \cdot A = \mathbf{0}$.
- $E_{n,k}^{(3)}$: There exists a set of k coordinates of $\mathbf{z} \in \mathrm{GR}(p^s, r)^{\alpha r n}$ that doesn't expand into $b'(k) = \left(\beta + k/n + \frac{(r-1)+110/n}{\log_2 q}\right) n$ coordinates of $\mathbf{v}^\top = \mathbf{z}^\top \cdot B$.
- $E_{n,k}^{(4)}$: Given that all sets of size k expand into at least $b'(k)$ coordinates, there exists a $\mathbf{z} \in \mathrm{GR}(p^s, r)^{\alpha r n}$ of Hamming weight $\|\mathbf{z}\|_0 = k$ which is mapped to $\mathbf{v}^\top = \mathbf{z}^\top \cdot B$ of Hamming weight $\|\mathbf{v}\|_0 < \beta n$.

Next, we demonstrate that by selecting an appropriate parameter r, the probability of the event $E_{n,k}^{(2)}$ occurring remains below $s \cdot 2^{-100}$.

Lemma 2. *For every $n \leq 2^{30}$ and every Galois Ring $GR(p^s, r)$ satisfying $p^r \geq 2^{127}$, for every n and $k \leq n$, $Pr[E_{n,k}^{(2)}] \leq s \cdot 2^{-100}$.*

Proof. Let $\mathbf{x}$ be an element in $\mathrm{GR}(p^s, r)^n$ with $\|\mathbf{x}\|_0 = k$, and let $K \subseteq [n]$ represent the indices of the nonzero elements in $\mathbf{x}$. Let T_K be a random variable denoting the number of columns of $A \leftarrow \mathcal{M}_{n,\alpha n,c_n}$ with at least one non-zero element in the rows with indices from K.

The occurrence of the event $E_{n,k}^{(2)}$ depends on the non-occurrence of the event $E_{n,k}^{(1)}$, we have that $T_k \geq b(k) = max(k + 4, 1.28k)$. We have at least T_k coordinates of $A\mathbf{x}$ are non-zero linear combinations of the non-zero coordinates of $\mathbf{x}$ with indices form $K_{\mathbf{x}}$.

Here, a more complicated analysis than that used in the Brakedown scheme is necessary. Drawing from the Fact 2, we know that for any two elements a and b in the Galois ring $\mathrm{GR}(p^s, r)$, if $d = \gcd(p^s, a)$, then the linear equation $ax = b$ has at most d^r solutions. If we treat the coefficients of the randomly selected linear combination as a variable x, and one of the k non-zero elements as a, then the condition that the random linear combination equals zero means that for some b, the equation $ax = b$ hold. According to Fact 2, the proportion

of x values that satisfy this condition is $\frac{d^r}{p^{sr}}$, which implies that the probability of the random linear combination being zero does not exceed $\frac{d^r}{p^{sr}}$. It is noted that the likelihood of a random linear combination being zero is influenced by the element with the smallest greatest common divisor with p^s. Therefore, we can categorize and discuss based on the smallest greatest common divisor of k elements. Assuming that the k nonzero elements in $\mathbf{x}$ have the smallest greatest common divisor with p^s as d, the probability that a random linear combination is zero is $\frac{d^r}{p^{sr}}$. There are $b(k)$ positions in $\mathbf{x}^\top \cdot A$ that are nonzero elements for the linear combination, thus the probability that all are zero is $\left(\frac{d^r}{p^{sr}}\right)^{b(k)}$. Given that $\mathbf{x}$ has k nonzero elements, and the greatest common divisor of these nonzero elements with p^s does not exceed d, the number of potential selections is $\binom{k}{n} \cdot \left(\left(\frac{p^s}{d}\right)^r\right)^k$. Consequently, when the k nonzero elements in x have the smallest common factor with p^s as d, the probability of event $E_{n,k}^{(2),d}$ occurring does not exceed:

$$\left(\frac{d^r}{p^{sr}}\right)^{b(k)} \cdot \left(\left(\frac{p^s}{d}\right)^r\right)^k \cdot \binom{k}{n} = \binom{k}{n}\left(\frac{d^{b(k)-k}}{p^{s(b(k)-k)}}\right)^r.$$

It is evident that the larger the value of d, the greater the probability of error. When d reaches its maximum at p^{s-1}, the error probability attains its highest value of:

$$\binom{k}{n}\left(\frac{1}{p^{b(k)-k}}\right)^r = \binom{k}{n}\left(\frac{1}{p^r}\right)^{b(k)-k}.$$

for $k \geq 15$, $n \leq 2^{30}$, and $p^r > 2^{127}$,

$$\begin{aligned}\Pr[E_{n,k}^{(2),d}] &\leq \binom{k}{n}\left(\frac{1}{p^r}\right)^{b(k)-k} \leq \binom{k}{n}(p^r)^{-0.28k} \leq \left(\frac{en}{k}\right)^k \cdot (p^r)^{-0.28k} \\ &= \left(\frac{en}{k(p^r)^{0.28}}\right)^k \leq 2^{-120}.\end{aligned}$$

for $k \leq 14$, $n \leq 2^{30}$, and $p^r > 2^{127}$,

$$\begin{aligned}\Pr[E_{n,k}^{(2),d}] &\leq \binom{k}{n}\left(\frac{1}{p^r}\right)^{b(k)-k} \leq \binom{k}{n}(p^r)^{-4} \leq \left(\frac{en}{k}\right)^k \cdot (p^r)^{-4} \\ &= \left(\frac{en}{14}\right)^{14} \leq 2^{-120}.\end{aligned}$$

Of course, this scenario pertains to when d attains its largest value. The maximum number of possible values for d is s, which includes 1, p, $\cdots$, p^{s-1}. According to the union bound, the soundness error can increase by at most s times. In this case, the error probability remains below $\frac{s}{2^{100}}$, a value which is still negligible. □

Lemma 3. *For every n and $\alpha\beta n \leq k \leq \beta n$, if $2\beta + \frac{(t-1)+110/n}{\log_2(p^r)} \leq t - 1 - t\alpha$, $Pr[E_{n,k}^{(4)}] \leq s \cdot 2^{-100}$.*

Proof. Assuming that every set of size k expands to at least

$$b'(k) = \left(\beta + \frac{k}{n} + \frac{(t-1) + \frac{110}{n}}{\log_2(p^r))}\right) n$$

we need to demonstrate that, with overwhelming probability, every element z in $\mathrm{GR}(p^s, r)^{\alpha t n}$ with a Hamming weight of $\|z\|_0 = k$ can be mapped to a vector $\mathbf{v} = B \cdot \mathbf{z}$ in $\mathrm{GR}(p^s, r)^{(t-1-t\alpha)n}$ with a Hamming weight $\|\mathbf{v}\|_0 \geq \beta n$.

Fix an element $\mathbf{z}$ in $\mathrm{GR}(p^s, r)$ with a Hamming weight $\|\mathbf{z}\|_0 = k$. Given that the non-zero coordinates of $\mathbf{z}$ will expand to at least $b'(k)$ coordinates, there are at least $b'(k)$ positions in the vector v comprising random linear combinations of non-zero vectors. As in the previous proof, if the k non-zero elements in x have the smallest greatest common divisor with p^s as d, then the probability of a random linear combination resulting in zero is $\frac{d^r}{p^{sr}}$. Consequently, with $\mathbf{z}$ fixed, the probability that $\|\mathbf{v}\|_0 \leq \beta n$ is bounded by:

$$\begin{aligned}\binom{b'(k)}{\geq b'(k) - \beta n} \left(\frac{d^r}{p^{sr}}\right)^{b'(k)-\beta n} &\leq \binom{(t-1-t\alpha)n}{\geq (t-1-t\alpha)n - \beta n} \left(\frac{d^r}{p^{sr}}\right)^{b'(k)-\beta n} \\ &\leq 2^{(t-1-t\alpha)n} \left(\frac{d^r}{p^{sr}}\right)^{b'(k)-\beta n}.\end{aligned}$$

assuming $b'(k) \leq \left(2\beta + \frac{(t-1)+\frac{110}{n}}{\log_2(p^r))}\right) n$. For all $\mathbf{z}$ in $\mathrm{GR}(p^s, r)^{\alpha t n}$ that satisfy $\|\mathbf{z}\|_0 = k$ and whose greatest common divisor with p^s is d, the upper bound is given by:

$$\begin{aligned}Pr[E_{n,k}^{(4),d}] &\leq \binom{t\alpha n}{k} \left(\left(\frac{p^s}{d}\right)^r\right)^k \cdot 2^{(t-1-t\alpha)n} \cdot \left(\frac{d^r}{p^{sr}}\right)^{b'(k)-\beta n} \\ &= \binom{t\alpha n}{k} \cdot 2^{(t-1-t\alpha)n} \cdot \left(\frac{d^r}{p^{sr}}\right)^{b'(k)-\beta n-k}.\end{aligned}$$

It is evident that the larger the value of d, the greater the probability of error. When d reaches its maximum at p^{s-1}, the error probability attains its highest value of:

$$\binom{t\alpha n}{k} \cdot 2^{(t-1-t\alpha)n} \cdot \left(\frac{1}{p^r}\right)^{b'(k)-\beta n-k}$$

$$\begin{aligned}Pr[E_{n,k}^{(4),d}] \leq \binom{t\alpha n}{k} \cdot 2^{(t-1-t\alpha)n} \cdot \left(\frac{1}{p^r}\right)^{b'(k)-\beta n-k} &\leq \frac{2^{t\alpha n+(t-1-t\alpha)n}}{(p^r)^{b'(k)-\beta n-k}} \\ \leq \frac{2^{(r-1)n}}{(p^r)^{\left(\frac{(t-1+100/n)}{\log_2(p^r)}\right)n}} << 2^{-100}.&\end{aligned}$$

Of course, this scenario pertains to when d attains its largest value. The maximum number of possible values for d is s, which includes 1, p, $\cdots$, p^{s-1}.

According to the union bound, the soundness error can increase by at most s times. In this case, the error probability remains below $\frac{s}{2^{100}}$, a value which is still negligible. □

3.2 Linear Codes over Extensions of Small Galois Rings

In the first part of this section, the encoding was designed for rings $\mathrm{GR}(p^s, r)$ where p^r is sufficiently large. However, if p^r is not exponentially large relative to the security parameter λ, what can be done? If the parameter p^r in a Galois ring requires a k-fold extension such that p^{kr} becomes exponentially large with respect to λ, a trivial solution is to perform a k-fold expansion by padding each element of $\mathrm{GR}(p^s, r)$ into $\mathrm{GR}(p^s, kr)$. However, this padding method is inefficient.

Inspired by the block-level coding approach from the Binius scheme [17], we treat k consecutive elements from $\mathrm{GR}(p^s, r)$ as a single element in $\mathrm{GR}(p^s, kr)$ for encoding, resulting in higher encoding efficiency.

In [32], Theorem 14.23 proves that the extension of any Galois ring is still a Galois ring.

Claim 1 ([32] Theorem 14.23) *For any Galois ring $R = GR(p^s, r)$, $h(x)$ is a monic irreducible polynomial of degree k over R. Then the residue class ring $R[x]/(h(x))$ is a Galois ring of characteristic p^s and it has p^{skr} elements and contains R as a subring. Thus*

$$R[x]/(h(x)) = GR(p^s, kr)$$

Through this claim, we can conclude that the extension of a Galois ring remains a Galois ring.

In this paper, if the Galois ring $G_1 = \mathrm{GR}(p^s, r)$ does not satisfy the requirement that p^r is large enough, and an extension $G_2 = \mathrm{GR}(p^s, kr)$ is needed. At this time, G_2 is obtained as an extension of G_1 by a degree-k irreducible polynomial. Elements of G_2 can be viewed as polynomials of degree $k-1$ with coefficients in G_1. Therefore, an element of G_2 can be represented as a vector consisting of k elements from G_1. The multiplication of an element a from G_1 with an element b from G_2 can be interpreted as performing the multiplication of a with each entry of the length-k vector corresponding to b. In this paper, the relationship holds for the corresponding Galois rings $\mathrm{GR}(p^s, r)$ and $\mathrm{GR}(p^s, kr)$.

From Fact 4, we conclude that if a monic polynomial is irreducible over $\mathrm{GF}(p, r)$, then it remains irreducible over $\mathrm{GF}(p^s, r)$. Therefore, we can search for irreducible polynomials over the Galois field corresponding to the Galois ring. The method for finding irreducible polynomials of degree d over a general Galois field has already been provided in [6].

For small Galois rings, the encoding involves two Galois rings. Below is the definition of the $[l, n, d]$-k-$\mathrm{GR}(p^s, r)$ code.

Definition 6. *($[l, n, d]$-k-$GR(p^s, r)$ Code) For a vector of length kn over Galois Ring $GR(p^s, r)$, treat each k consecutive elements as a single element in*

$GR(p^s, kr)$(we treat the adjacent k elements of $GR(p^s, r)$ as a k-dimensional vector.), where the parameters p, k, and r satisfy the condition that $\frac{1}{p^{kr}}$ is negligible relative to the security parameter λ. The generator matrix G is an $n \times l$ matrix defined over $GR(p^s, kr)$. After encoding, the result is a vector of length l over $GR(p^s, kr)$. Furthermore, for any two distinct vectors of length kn over $GR(p^s, r)$, at least d elements in $GR(p^s, kr)$ are different after encoding.

Algorithm 2: Enc' Algorithm: $\mathrm{GR}(p^s, r)^{kn} \to \mathrm{GR}(p^s, kr)^{tn}$

Input: $\mathbf{x} \in \mathrm{GR}(p^s, r)^{kn}$
parameter: $\alpha, \beta, t, c_n, d_n$
Output: $\mathbf{w} \in \mathrm{GR}(p^s, kr)^{tn}$($\mathrm{GR}(p^s, kr)$ is obtained as an extension of $\mathrm{GR}(p^s, r)$ by a degree-k irreducible polynomial.)

1 Treat the adjacent k elements of $\mathrm{GR}(p^s, r)$ as a k-dimensional vector. According to Claim 1, a k-length $\mathrm{GR}(p^s, r)$ vector is considered an element of $\mathrm{GR}(p^s, kr)$. This leads to treating the vector $\mathbf{x}$ of length kn into a vector $\mathbf{x}'$ of length n over $\mathrm{GR}(p^s, kr)$;
2 Call $\mathrm{Enc}(\mathbf{x}')$ to obtain $\mathbf{w}$ and output $\mathbf{w}$;

It can be observed that the encoding over $\mathrm{GR}(p^s, r)$ is essentially an encoding over $\mathrm{GR}(p^s, kr)$. However, we must demonstrate that Enc' supports linear combinations over $\mathrm{GR}(p^s, r)$; otherwise, it cannot be considered a linear code over $\mathrm{GR}(p^s, r)$.

Lemma 4. *Enc' supports linear combinations over $GR(p^s, r)$.*

Proof. One important point to note is that an element of $\mathrm{GR}(p^s, kr)$ can be viewed as a vector of k elements over $\mathrm{GR}(p^s, r)$. Let $a \in \mathrm{GR}(p^s, r)$ and $b \in \mathrm{GR}(p^s, kr)$. At the same time, a can be considered as an element in $\mathrm{GR}(p^s, kr)$.

The multiplication $a \cdot b$ in $\mathrm{GR}(p^s, kr)$ can be understood as the component-wise multiplication of a with each element of b. Therefore, the dot product of an element $a \in \mathrm{GR}(p^s, r)$ with a vector of length l over $\mathrm{GR}(p^s, kr)$ can be viewed as the dot product of a with a vector of length lk over $\mathrm{GR}(p^s, r)$.

Let $a_1, a_2 \in \mathrm{GR}(p^s, r)$, and let $\mathbf{b}_1, \mathbf{b}_2 \in \mathrm{GR}(p^s, r)^{kn}$, with $\mathbf{b'}_1$ and $\mathbf{b'}_2$ being the vectors padded to length n over $\mathrm{GR}(p^s, kr)$. Then we have:

$$\begin{aligned} a_1 \cdot \mathrm{Enc}'(\mathbf{b_2}) + a_2 \cdot \mathrm{Enc}'(\mathbf{b_2}) &= a_1 \cdot \mathrm{Enc}(\mathbf{b'}_1) + a_2 \cdot \mathrm{Enc}(\mathbf{b'}_2) \\ &= \mathrm{Enc}(a_1 \cdot \mathbf{b'}_1 + a_2 \cdot \mathbf{b'}_2) \\ &= \mathrm{Enc}(a_1 \cdot \mathbf{b_1} + a_2 \cdot \mathbf{b_2}) \end{aligned}$$

The second equality holds because Enc is a linear code defined over $\mathrm{GR}(p^s, kr)$, and $a \in \mathrm{GR}(p^s, kr)$. Therefore, it can be concluded that Enc' supports linear combinations over $\mathrm{GR}(p^s, r)$. □

3.3 Costs

Table 1 is the performance comparison for different parameters. While Brakedown applies its operations over a large 127-bit prime, our approach utilizes a Galois ring $\mathrm{GR}(p^s, r)$. Additionally, in Brakedown, the probability of encoding failure does not exceed 2^{-100}, while in the Galois ring $\mathrm{GR}(p^s, r)$, the failure probability does not exceed $s \cdot 2^{-100}$. In practical applications, s is almost never greater than 2^8, so the scheme remains secure over Galois rings.

Beyond those, the parameters α, β, t, c_n, and d_n can be selected in a manner consistent with those in the Brakedown scheme. If the encoding length is n, data from Brakedown suggest that, depending on the selected parameters, encoding a vector of length n will require between $13.2n$ to $25.5n$ multiplications on $\mathrm{GR}(p^s, r)$. We denote this constant as c_0, with c_0 approximately ranging from 13.2 to 25.5. The cost of the expander code in this work is consistent with that in Brakedown, except for the difference in the basic computational unit.

Table 1: Linear Code over Galois rings $\mathrm{GR}(p^s, r)$ performance according to Brakedown [22].

n	p^r	**Pr[failure]**	**Run-time**	**Distance**	**Rate**	α	β	t	c_n	d_n
$\leq 2^{30}$	$\geq 2^{127}$	$< s \cdot 2^{-100}$	$13.2n$	0.02	0.704	0.1195	0.0284	1.42	6	33
$\leq 2^{30}$	$\geq 2^{127}$	$< s \cdot 2^{-100}$	$14.3n$	0.03	0.68	0.138	0.0444	1.47	7	26
$\leq 2^{30}$	$\geq 2^{127}$	$< s \cdot 2^{-100}$	$15.8n$	0.04	0.65	0.178	0.061	1.521	7	22
$\leq 2^{30}$	$\geq 2^{127}$	$< s \cdot 2^{-100}$	$17.8n$	0.05	0.60	0.2	0.082	1.64	8	19
$\leq 2^{30}$	$\geq 2^{127}$	$< s \cdot 2^{-100}$	$20.5n$	0.06	0.61	0.211	0.097	1.616	9	21
$\leq 2^{30}$	$\geq 2^{127}$	$< s \cdot 2^{-100}$	$25.5n$	0.07	0.58	0.238	0.1205	1.72	10	23

4 Linear Time Polynomial Commitment

The Brakedown polynomial commitment relies on the lemma from AHIV17[1], so our first step is to prove that this lemma holds over arbitrary Galois rings. So we must consider the case where polynomial coefficients lie in a small ring. Recall that during encoding, we treat k elements from the small ring as a single element from the extension ring. If the verifier directly selects a challenge e from the extension ring, then e will multiply with the larger element formed by the k elements from the small ring. So the adjacent k coefficients will be mixed together. However, if the verifier selects challenges only from the small ring, the soundness cannot be guaranteed.

To solve this issue, we prove a "repetition" version of the AHIV17[1] lemma over arbitrary Galois rings. In the "repetition" version lemma, the verifier ensures soundness by selecting multiple challenges from the small ring, while also preventing the mixing of the k coefficients. To prove the "repetition" version lemma, it is necessary to utilize the definitions of Interleaved Codes and Blockwise Relative Distance.

Definition 7 (Interleaved code and Block-wise relative distance [9]). *Let $C \subset \mathbb{F}^n$ be an $[n, k, d]$ linear code over $\mathbb{F}$. We let C^m denote the $[mn, mk, d]$ (interleaved) code over $\mathbb{F}^m$ whose codewords are all $m \times n$ matrices A such that every row A_i of A satisfies $A_i \in C$. For $A \in C^m$ and $j \in [n]$, we denote by $A[j]$ the jth symbol of A.*

Moreover, we introduce the denifiniton of block-wise distance of (A, C^m):

$$d(A, L^m) := \frac{|\{j \in [n] | \exists i \ s.t. \ A_i[j] \neq c_i[j]\}|}{n}$$

where c_i denote the closest codeword with A_i in C.

Definition 8 (Interleaved code and Block-wise relative distance over Galois Rings). *Let $C \subset GR(p^s, kr)^n$ be an $[n, k, d]$ linear code over $GR(p^s, kr)$. We let C^m denote the $[mn, mk, d]$ (interleaved) code over $GR(p^s, kr)^m$ whose codewords are all $m \times n$ matrices A such that every row A_i of A satisfies $A_i \in C$. For $A \in C^m$ and $j \in [n]$, we denote by $A[j]$ the jth symbol of A.*

Moreover, we give the denifiniton of the block-wise distance of (A, C^m):

$$d(A, L^m) := \frac{|\{j \in [n] | \exists i \ s.t. \ A_i[j] \neq c_i[j]\}|}{n}$$

where c_i denote the closest codeword with U_i in C.

Claim 2 (Ames, Hazay, Ishai, and Venkitasubramaniam [1], Roth and Zémor) *Fix an arbitrary $[n, k, d]$-code $L \subset \mathbb{F}_q^n$, and a proximity parameter $e \in \{0, \cdots, \lfloor \frac{d-1}{3} \rfloor\}$. Supposed $d(U, C^m) > e$. Then for a random w^* in the row-span of U, we have*

$$\Pr[d(w^*, L) \leq e] \leq (e + 1)/|\mathbb{F}|$$

The presence of zero divisors in the Galois ring $\mathrm{GR}(p^s, r)$ increases the possibility $\frac{e+1}{q}$. To maintain this probability at a negligible level, we repeatedly select challenges to reduce it.

Claim 3 (AHIV17 Repetition Version) *Fix any k-$[l, n, d]$ code $C \subset GR(p^s, kr)^l$ over the Galois ring $GR(p^s, r)$, and a proximity parameter $e \in \{0, \cdots, \lfloor \frac{d-1}{3} \rfloor\}$. For a matrix $U \in GR(p^s, kr)^{m \times l}$ with $d(U, C^m) > e$, and a matrix $R \in GR(p^s, r)^{k \times m}$ where each element of R is randomly chosen from $GR(p^s, r)$, let $W = RU$. Then we have:*

$$\Pr[d(W, C^k) \leq e] \leq \frac{e + 1}{p^{rk}}.$$

We provide the proof of this claim in the Appendix C.1.

With the verification lemma established above, we can now give a polynomial commitment over the Galois ring $\mathrm{GR}(p^s, r)$.

This polynomial commitment is designed to compute a multilinear polynomial $f(x_1, \cdots, x_l)$ defined over $\mathrm{GR}(p^s, r)$, where polynomial coefficient $x_i \in$

$\mathrm{GR}(p^s, r)$. Assume that the encoding over $\mathrm{GR}(p^s, r)$ uses a k-$[l, n, d]$ code, where $k \geq 1$.

Polynomial Commitment for $\mathbf{GR}(p^s, r)$:

- **Commit Phase**
  - The prover splits the multilinear polynomial with l variables into a matrix U of size $2^{l/2} \times 2^{l/2}$. Let $m = 2^{l/2}$. Each row of the matrix is encoded using a k-$[tm, m, d]$ code to obtain a matrix $\hat{U}$ with m rows, $\hat{U}_1, \cdots, \hat{U}_m$, where $\hat{U}_i \in \mathrm{GR}(p^s, kr)^{tm/k}$, and $\hat{U}_i[j]$ represents the element in the i-th row and j-th column of the matrix $\hat{U}$. The prover constructs a Merkle Tree to commit to $\hat{U}$.
- **Testing Phase**
  - **Verifier** $\rightarrow$ **Prover**: A random matrix $R \in (\mathrm{GR}(p^s, r))^{k \times m}$, where each element of R is randomly chosen from $\mathrm{GR}(p^s, r)$.
  - **Prover** $\rightarrow$ **Verifier**: Treat every k adjacent elements in each row of matrix U as an element in $\mathrm{GR}(p^s, r)$, and compute $V = RU \in \mathrm{GR}(p^s, kr)^{k \times (m/k)}$ and calims that $V = RU$. Then, send V to the verifier.
  - **Verifier**: Choose a random set Q of size $l_Q = \Theta(\lambda)$, with $Q \subseteq [tm/k]$. For each $j \in Q$:
    - Verifier queries all m entries of the corresponding "column" of $\hat{U}$: $\hat{U}_1[j], \cdots, \hat{U}_m[j]$. The verifier receives these values and the associated Merkle Tree paths from the prover. If one of the paths is invalid, reject.
    - For each row V_i of matrix V, the verifier confirms that $\mathrm{Enc}(V_i)[j] = \sum_{s=1}^{m} R_i[s] \cdot \hat{U}_s[j]$, rejecting if the condition is not satisfied.
- **Evaluation Phase**
  - To compute $f(r_1, \cdots, r_l)$, calculate $\mathbf{q_1} = (1-r_1, r_1) \otimes \cdots \otimes (1-r_{l/2}, r_{l/2}) \in \mathrm{GR}(p^s, r)^m$ and $\mathbf{q_2} = (1-r_{l/2+1}, r_{l/2+1}) \otimes \cdots \otimes (1-r_l, r_l) \in \mathrm{GR}(p^s, r)^m$.
  - The evaluation phase is similar to the testing phase, but with R replaced by $\mathbf{q_1}$ to obtain $\mathbf{v}$.
  - If all previous checks pass, the verifier treats $\mathbf{v}$ as a vector of length m over $\mathrm{GR}(p^s, r)$, and computes $f(r_1, \cdots, r_l) = \langle \mathbf{v}, \mathbf{q_2} \rangle$.

The Proof of Completeness

Proof. To demonstrate completeness, we first express the matrix U as:

$$U = \begin{bmatrix} f(0, \cdots, 0, 0, \cdots, 0) & f(0, \cdots, 0, 0, \cdots, 1) & \cdots & f(0, \cdots, 0, 1, \cdots, 1) \\ f(0, \cdots, 1, 0, \cdots, 0) & f(0, \cdots, 1, 0, \cdots, 1) & \cdots & f(0, \cdots, 1, 1, \cdots, 1) \\ \vdots & \vdots & \vdots & \vdots \\ f(1, \cdots, 1, 0, \cdots, 0) & f(1, \cdots, 1, 0, \cdots, 1) & \cdots & f(1, \cdots, 1, 1, \cdots, 1) \end{bmatrix}$$

To compute $f(r_1, \cdots, r_l)$, we use the formula:

$$f(r_1, \cdots, r_l) = \sum_{\mathbf{b} \in \{0,1\}} \prod_{i \in [1,l]} \left((1 - r_i)(1 - b_i) + r_i b_i\right) f(\mathbf{b}),$$

which can be rewritten as:

$$f(r_1, \cdots, r_l) = \mathbf{q}_1^\top U \mathbf{q}_2.$$

An important observation is that if we treat the i-th row of the matrix U, denoted U_i, as m/k elements $w_1, \cdots, w_{m/k}$ in $\mathrm{GR}(p^s, kr)$, then for $a \in \mathrm{GR}(p^s, r)$, we have:

$$\begin{aligned}
&a \cdot (w_1, \cdots, w_{m/k}) \\
=&(a \cdot w_1, \cdots, a \cdot w_{m/k}) \\
=&(a(w_{1,1}, \cdots, w_{1,k}), \cdots, a(w_{(m/k),1}, \cdots, w_{(m/k),k})) \\
=&(aw_{1,1}, \cdots, aw_{1,k}, \cdots, aw_{(m/k),1}, \cdots, aw_{(m/k),k}) \\
=&a \cdot (U_i[1], \cdots, U_i[m]).
\end{aligned}$$

Since each w_i can be viewed as k elements from $\mathrm{GR}(p^s, r)$, the computation is the same whether we treat U as a matrix in $\mathrm{GR}(p^s, r)^{m \times m}$ or in $\mathrm{GR}(p^s, kr)^{m \times (m/k)}$. Thus, the result of $\mathbf{q}_1^\top U$ remains the same in "form" regardless of the interpretation. Therefore, the linear combination of the matrix can be seen as a linear combination over $\mathrm{GR}(p^s, r)$, ensuring the completeness of the polynomial commitment. □

The Proof of Soundness

Proof. The encoded coefficient matrix $\hat{U} \in \mathrm{GR}(p^s, kr)^{m \times (m/k)}$, where $N = m/k$, has a code distance of d and a relative distance of $\gamma = d/N$. First, we show that if the prover can pass the testing phase with a probability greater than $\frac{1}{p^{kr}} + \left(1 - \frac{\gamma}{3}\right)^{l_Q}$, then there must exist a codeword $[c_1, \cdots, c_m] \in C^m$ such that:

$$E := |\{j \in [N] : \exists i \in [m], \text{ such that } c_{i,j} \neq \hat{U}_{i,j}\}| \leq \left(\frac{\gamma}{3}\right) N.$$

Let $e = \left\lfloor \frac{\gamma N}{3} \right\rfloor$. If no such codeword $[c_1, \cdots, c_m]$ exists, then we have $d(\hat{U}, C^m) > e$. According to claim 3, since $V = R \cdot \hat{U}$, the probability that $d(V, C^k)$ is greater than e is at least $1 - \frac{e+1}{p^{kr}}$.

Thus, during the testing phase, the probability that the verifier randomly selects l columns without hitting any columns in $\Delta(V, C^k)$ is at most $(1 - \frac{e}{3})^{l_Q} = \left(1 - \frac{\gamma}{3}\right)^{l_Q}$. Define the event E_1 as $d(V, C^k) > e$, and the event E_2 as the verifier selecting l columns without hitting any in $\Delta(V, C^k)$. Then, we have:

$$\begin{aligned}
\Pr[P^* \text{wins}] &\leq \Pr[P^* \text{wins} | \bar{E_1}] \Pr[\bar{E_1}] + \Pr[P^* \text{wins} | E_1] \Pr[E_1] \\
&\leq \Pr[\bar{E_1}] + \Pr[E_2 | E_1] \Pr[E_1] \\
&< \frac{1}{p^{kr}} + \Pr[E_2 | E_1] = \frac{1}{p^{kr}} + (1 - \frac{\gamma}{3})^{l_Q}
\end{aligned}$$

This contradicts the fact that the prover could pass the testing phase, so there must exist such a codeword $c_1, \cdots, c_m$.

We can observe that each c_i is the closest codeword to each row U_i of $\hat{U}$; otherwise, the distance between two codewords would be less than d. Define $w := \sum_{i=1}^{m} q_{1,i} \cdot c_i$. Next, we show that if the prover can pass the testing phase with a probability greater than $\frac{e+1}{p^k} + \left(1 - \frac{d}{3}\right)^{l_Q}$, and pass the evaluation phase with a probability greater than $\left(1 - \frac{2\gamma}{3}\right)^{l_Q}$, then let $u := \sum_{i=1}^{m} q_{1,i} \cdot U_i$ and $w = \mathrm{Enc}(u)$.

If $w \neq \mathrm{Enc}(u)$, then w and $\mathrm{Enc}(v)$ are two distinct codewords in C, meaning that the number of positions where they are identical will not exceed $(1-\gamma)N$. Let the set of positions where the two codewords are identical be denoted as A. If the verifier selects a column $j \notin A \cup E$, then the test will fail because w and $\mathrm{Enc}(u)$ can only match within $A \cup E$. We have $|A \cup E| \leq |A| + |E| \leq (1-\gamma)N + \left(\frac{\gamma}{3}\right) N = \left(1 - \frac{2\gamma}{3}\right) N$. Since the verifier selects the set of columns Q randomly, if the prover is cheating, the probability of passing the evaluation phase will not exceed $\left(1 - \frac{2\gamma}{3}\right)^{l_Q}$.

Thus, if the prover can pass the testing phase with a probability greater than $\frac{e+1}{p^k} + \left(1 - \frac{d}{3}\right)^{l_Q}$, and the evaluation phase with a probability greater than $\left(1 - \frac{2\gamma}{3}\right)^{l_Q}$, the binding property of the scheme is ensured. □

4.1 Extractability

If the committed polynomial has l variables, let $m = 2^{l/2}$, so the matrix U being committed is an $m \times (m/k)$ matrix over $\mathrm{GR}(p^s, kr)$. Let $N = m/k$. The key to extraction lies in finding an invertible $N \times N$ matrix over the Galois ring. In a finite field, a matrix formed by selecting N linearly independent non-zero vectors of length N is invertible. Randomly selecting N vectors of length N results in a non-negligible probability of the matrix being invertible. However, in a Galois ring, due to the presence of zero divisors, a set of linearly independent vectors does not necessarily form an invertible matrix.

We prove that a randomly chosen $N \times N$ matrix over the Galois ring $\mathrm{GR}(p^s, r)$ is invertible with high probability. First, we consider the case where $p^r > N$ and $1 - \frac{N}{p^r}$ is non-negligible.

Lemma 5. *Assuming that $1 - \frac{N}{p^r} > 0$ and is non-negligible, then by randomly selecting an $N \times N$ matrix R from $GR(p^s, r)$, there is a probability of $1 - \frac{N}{p^r}$ that R is invertible.*

Proof. Using the adjugate matrix method for matrix inversion,

$$R^{-1} = \frac{A^*}{\det(A)} = \frac{1}{\det(A)} \begin{bmatrix} R^*_{11} & \cdots & R^*_{1N} \\ \vdots & \ddots & \vdots \\ R^*_{N1} & \cdots & R^*_{NN} \end{bmatrix}^\top$$

Let R^*_{ij} represent the algebraic cofactor of matrix R with respect to the (i, j) entry. We observe that if $\det(R)$ is non-zero and not a zero divisor, then R^{-1}

exists. Now, we calculate the probability that $\det(R)$ is non-zero and not a zero divisor.

$$\det(R) = \sum_{\sigma \in S_N} \operatorname{sgn}(\sigma) \prod_{i=1}^{N} A_{i,\sigma(i)}$$

where S_N is the set of all permutations on $\{1, 2, \cdots, N\}$, and $\operatorname{sgn}(\sigma)$ denotes the sign of permutation σ. If σ has an even number of inversions, then $\operatorname{sgn}(\sigma) = 1$; otherwise, $\operatorname{sgn}(\sigma) = -1$. In fact, $\det(R)$ can be viewed as a multilinear polynomial in the N^2 variables R_{ij}, and the degree of the polynomial is at most N. Let us define this polynomial as f_R.

By a known fact about Galois rings, all zero divisors in $\mathrm{GR}(p^s, r)$ lie in the ideal (p). Therefore, we only need to show that f_R is non-zero modulo p with a non-negligible probability. Consider the isomorphism map ψ

$$\begin{aligned} \mathbb{Z}_{p^s} &\to \mathbb{F}_p \\ c_0 + c_1 p + \cdots + c_{p-1} p^{s-1} &\mapsto c_0 \end{aligned}$$

For a polynomial f of degree at most d with coefficients in $\mathrm{GR}(p^s, r)$, we define a mapping Ψ, where $\Psi(f)$ is derived by applying ψ to each coefficient of f, resulting in a polynomial f'. Clearly, Ψ is a homomorphism. If at some point $\mathbf{x}$, we have $f(\mathbf{x}) = 0 \pmod{p}$, then it follows that $\Psi(f)(\psi(\mathbf{x})) = 0$.

Let $\Psi(f_R) = f'_R$. If f'_R evaluates to a non-zero value, then f_R will not be a zero divisor in $\mathrm{GR}(p^s, r)$. Since the variables R_{ij} are randomly chosen from $\mathrm{GR}(p^s, r)$, so $\psi(R_{ij})$ are randomly chosen from $\mathrm{GF}(p, r)$. By the Schwartz-Zippel lemma for fields, the degree of f'_R is at most N, so the probability that a randomly chosen point evaluates to 0 is at most $\frac{N}{p^r}$. Therefore, the probability that f'_R is non-zero is at least $1 - \frac{N}{p^r}$, which is non-negligible. This completes the proof. □

If the ring $\mathrm{GR}(p^s, r)$ does not satisfy $p^r > N$, then the $\mathrm{GR}(p^s, r)$ code used in encoding must be a $[l, n, d] - k - \mathrm{GR}(p^s, r)$ code, where $p^{kr} \gg N$. Choose a k' such that k' is a divisor of k and $p^{k'r} > N$. At this point, the extractor can send k' challenges from $\mathrm{GR}(p^s, r)$ to simulate a challenge from $\mathrm{GR}(p^s, k'r)$. Then, using Lemma 8, we can obtain an invertible $N \times N$ matrix with non-negligible probability.

If the probability that the matrix is invertible is p_i, by repeating the process $\mathrm{poly}(\frac{1}{p_i})$ times, we can ensure that with probability $1 - \mathrm{negl}(\lambda)$, the matrix U can be successfully extracted.

4.2 More Efficient

When using polynomial commitment schemes in SNARKs, it is typically required that the challenge space for the verifier is exponential in size relative to the security parameter λ, while the polynomial coefficients may come from a smaller Galois ring. Suppose the coefficients come from $\mathrm{GR}(p^s, r)$, while the challenges

come from $\mathrm{GR}(p^s, kr)$. In this case, when committing to the coefficients, each coefficient element needs to be treated as an element in $\mathrm{GR}(p^s, kr)$. This increases the cost of encoding and hashing by a factor of k.

In Binius [17], a two-dimensional expansion was used to address this issue. In this work, we explain the problem from the perspective of "repetition": using the observations that even though elements are treated as being in $\mathrm{GR}(p^s, kr)$ during the encoding phase, the commitment still supports linear combinations over $\mathrm{GR}(p^s, r)$. This allows the algebraic structures used during the commitment and evaluation phases to differ. Below, we present a polynomial commitment scheme where the coefficients come from $\mathrm{GR}(p^s, r)$, but the challenges come from $\mathrm{GR}(p^s, kr)$.

The polynomial commitment scheme is designed to compute a multilinear polynomial $f(x_1, \cdots, x_l)$ over $\mathrm{GR}(p^s, r)$, where the coefficients come from $\mathrm{GR}(p^s, r)$ and each $x_i \in \mathrm{GR}(p^s, kr)$. Assume that the encoding over $\mathrm{GR}(p^s, r)$ uses a k-$[l, n, d]$ code, where $k \geq 1$.

More Efficient Polynomial Commitment for Small Rings

- **Commit Phase**
  - The prover splits the multilinear polynomial with l variables into a matrix U of size $2^{l/2} \times 2^{l/2}$. Let $m = 2^{l/2}$. Each row of the matrix is encoded using a k-$[tm, m, d]$ code to obtain a matrix $\hat{U}$ with m rows, $\hat{U}_1, \cdots, \hat{U}_m$, where $\hat{U}_i \in \mathrm{GR}(p^s, kr)^{tm/k}$, and $\hat{U}_i[j]$ represents the element in the i-th row and j-th column of the matrix $\hat{U}$. The prover constructs a Merkle Tree to commit to $\hat{U}$.
- **Testing Phase**
  - **Verifier** $\rightarrow$ **Prover**: A random matrix $R \in (\mathrm{GR}(p^s, r))^{k \times m}$, where each element of R is randomly chosen from $\mathrm{GR}(p^s, r)$.
  - **Prover** $\rightarrow$ **Verifier**: Treat every k adjacent elements in each row of matrix U as an element in $\mathrm{GR}(p^s, r)$, and compute $V = RU \in \mathrm{GR}(p^s, kr)^{k \times (m/k)}$ and calims that $V = RU$. Then, send V to the verifier.
  - **Verifier**: Choose a random set Q of size $l_Q = \Theta(\lambda)$, with $Q \subseteq [tm/k]$. For each $j \in Q$:
    - Verifier queries all m entries of the corresponding "column" of $\hat{U}$: $\hat{U}_1[j], \cdots, \hat{U}_m[j]$. The verifier receives these values and the associated Merkle Tree paths from the prover. If one of the paths is invalid, reject.
    - For each row V_i of matrix V, the verifier confirms that $\mathrm{Enc}(V_i) = \sum_{s=1}^{m} R_i[s] \cdot \hat{U}_s[j]$, rejecting if the condition is not satisfied.
- **Evaluation Phase**
  - To compute $f(r_1, \cdots, r_l)$, calculate $\mathbf{q_1} = (1-r_1, r_1) \otimes \cdots \otimes (1-r_{l/2}, r_{l/2}) \in \mathrm{GR}(p^s, kr)^m$ and $\mathbf{q_2} = (1-r_{l/2+1}, r_{l/2+1}) \otimes \cdots \otimes (1-r_l, r_l) \in \mathrm{GR}(p^s, kr)^m$.
  - Split each element of $\mathbf{q_1}$ into a k-length vector over $\mathrm{GR}(p^s, r)$. Let $\mathbf{q}_{1,i}$ represent the i-th position of each element. Then, for each $\mathbf{q}_{1,i}$, perform the evaluation phase to get k vectors $\mathbf{v}_1, \cdots, \mathbf{v}_k$.

- Combine the k vectors into one m-length element $\mathbf{v}$ over $\mathrm{GR}(p^s, kr)$, where $\mathbf{v}_i[j]$ represents the i-th coefficient at the j-th position. The verifier then computes $\mathbf{v}^\top \mathbf{q_2}$ to obtain $f(r_1, \cdots, r_l)$.

To demonstrate the completeness of this scheme, we express $f(r_1, \cdots, r_l)$ as a matrix multiplication: $\mathbf{q}_1^\top U \mathbf{q}_2$. When computing $\mathbf{q}_1^\top$, note that only the elements of $\mathbf{q_1}$ come from $\mathrm{GR}(p^s, kr)$. Thus, when multiplying the elements of $\mathbf{q}_1$ with the elements of U, it can be viewed as multiplying each component of the elements in U with each component of the elements in $\mathbf{q}_1$.

$$\mathbf{q}_1^\top U = \begin{bmatrix} \mathbf{q}_{11}^\top \\ \vdots \\ \mathbf{q}_{1k}^\top \end{bmatrix} U$$

Thus, the verifier can correctly compute $f(r_1, \cdots, r_l)$. The binding and extractability properties of this protocol remain consistent with the previous proof. We now focus on analyzing the prover's computational cost. The prover needs to construct a Merkle Tree over $\frac{2^l}{k}$ elements from $\mathrm{GR}(p^s, r)$, which takes time $t_1 = O\left(\frac{2^l}{k}\right)$. Additionally, the prover needs to perform linear encoding on $\frac{2^{l/2}}{k}$ elements from $\mathrm{GR}(p^s, r)$, requiring $\frac{c_0 2^l}{k}$ multiplications in $\mathrm{GR}(p^s, kr)$. In the testing phase and evaluation phase, $t2^l k$ multiplications in $\mathrm{GR}(p^s, kr) \cdot \mathrm{GR}(p^s, r)$ are needed, where $\frac{1}{t}$ is the code rate. Moreover, the prover must open $|Q|2^{l/2}$ Merkle Tree nodes, taking time t_2.

Let ee represent the time required for one multiplication in $\mathrm{GR}(p^s, kr)$, and be represent the time for one multiplication in $\mathrm{GR}(p^s, kr) \cdot \mathrm{GR}(p^s, r)$. The prover's total cost is $t_1 + t_2 + \frac{c_0 2^l}{k} ee + t2^{l+1} kbe$.

Without this commitment scheme, if the prover commits to elements from $\mathrm{GR}(p^s, r)$ by directly calculating over $\mathrm{GR}(p^s, kr)$ in the commitment phase, they would need to construct a Merkle Tree over 2^l elements from $\mathrm{GR}(p^s, r)$, taking time $kt_1 = O\left(\frac{2^l}{k}\right)$. Then, the prover would perform linear encoding on $2^{l/2}$ elements from $\mathrm{GR}(p^s, r)$, requiring $c_0 2^l$ multiplications in $\mathrm{GR}(p^s, kr)$. In the testing phase and evaluation phase, $t2^l$ multiplications in $\mathrm{GR}(p^s, kr)$ are needed. Additionally, $|Q|2^{l/2}$ Merkle Tree nodes must be opened, taking time t_2. Thus, the total prover cost is $kt_1 + t_2 + (c_0 + 2t)2^l ee$.

It is clear that using this approach results in approximately k times more ee computations. And the ee operations are significantly more time-consuming than be and bb operations.

5 Proof Toolboxes over Galois Rings

The sumcheck protocol over rings has been widely discussed, and in [30], the sumcheck protocol was extended to infinite non-commutative rings. However, [30] did not propose a corresponding polynomial commitment scheme. In this section, we explore the sumcheck protocol over Galois rings.

5.1 Sumcheck Over Galois Rings

To ensure the soundness of the sumcheck protocol, the verifier's challenges can be selected from the exceptional set of the ring. Therefore, to ensure a sufficiently large challenge space, the exceptional set must be large enough. An extension method is typically used to expand the size of the exceptional set. In fact, if a ring has a sufficiently large exceptional set, it indicates that the proportion of zero divisors is very small, and in this case, according to Fact 3, the verifier's challenge can be selected from the entire Galois ring.

Below, we introduce the sumcheck protocol where the challenges are selected from the entire Galois ring $\mathrm{GR}(p^s, r)$, assuming that p^r is sufficiently large and the degree of the polynomial p does not exceed d. As long as the verifier's challenges are selected from the ring $\mathrm{GR}(p^s, r)$, where p^r is exponentially large in relation to the security parameter λ, security can be ensured. Meanwhile, the coefficients of the polynomial being proven can still come from a "smaller ring" G_1, as long as $\mathrm{GR}(p^s, r)$ is an extension of G_1. This allows arithmetic circuits defined over a "small ring" to be encoded into polynomials with coefficients that remain in the smaller ring. This aligns with the scenario mentioned at the end of the previous section and it can reduce the prover's time complexity.

Our sumcheck protocol is adapted from the protocol presented in Chapter 4 of the [31]. We use $\deg_i(g)$ to denote the highest degree of the i-th variable of the polynomial g.

Sumcheck Over Galois Rings Construction

- The prover sends a value H to the verifier, which is equal to

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_l \in \{0,1\}} g(x_1, ..., x_l)$$

- In the first round, the prover sends a univariate polynomial $g_1(X_1)$ claim to equal

$$\sum_{(x_2,...,x_l) \in \{0,1\}^{n-1}} g(X_1, x_2, ..., x_l)$$

  Then verifier checks

$$H = g_1(0) + g_1(1)$$

  and g_1 is a univarite polynomial of degree at most $\deg_1(g)$. If not, the verifier terminates and returns reject.
- The verifier randomly selects a challenge r_1 from the $\mathrm{GR}(p^s, r)$ and sends r_1 to the prover.
- In the ith round, $1 < i < n$, the prover sends a univariate polynomial $p_i(X_i)$ and claim it is equal to

$$\sum_{(x_{i+1},...,x_l) \in \{0,1\}^{l-i}} g(r_1, ..., r_{i-1}, X_i, x_{i+1}, ..., x_l)$$

  verifier checks whether g_i is a univariate polynomial with degree at most $\deg_i(g)$, and checks $g_{i-1}(r_{i-1}) = g_i(0) + g_i(1)$. If the check does not hold, then terminates and rejects.

- The verifier randomly selects a challenge r_i from $\mathrm{GR}(p^s, r)$ and sends r_i to the prover.
- In the last round, the prover sends a univariate polynomial $g_n(X_n)$ to the verifier and claims equal $g(r_1, ..., r_{l-1}, X_l)$ verifier checks whether g_l is a univariate polynomial whose degree does not exceed $deg_l(g)$ and checks $g_{l-1}(r_{l-1}) = g_l(0) + g_l(1)$. If not, terminate and rejects.
- Verifier chooses a random challenge r_l from $\mathrm{GR}(p^s, r)$ and evaluates $g(r_1, ..., r_l)$ with a single oracle query to g. Verifier checks $g_l(r_l) = g(r_1, ..., r_l)$, if not, terminate and rejects.
- If all the above checks pass, then the verifier accepts.

Lemma 6. *The soundness error of the sumcheck protocol over Galois rings will not exceed* $\frac{ld}{p^r}$.

In Appendix D.1, we provide the proof for this lemma.

In addition to the sumcheck protocol, constructing SNARKs often requires a ZeroCheck protocol to prove that a polynomial p evaluates to zero on $\{0,1\}^l$. We now present a scheme over the Galois ring $\mathrm{GR}(p^s, r)$, where p^r is exponentially large with respect to the security parameter λ. Define

$$eq(\mathbf{x}, \mathbf{y}) = \prod_{i \in [l]} \left((1 - x_i)(1 - y_i) + x_i y_i\right).$$

ZeroCheck Over Galois Rings Construction

- Statement: For a polynomial f of degree d, we want to assert that $f(\mathbf{x}) = 0$ for all $\mathbf{x} \in \{0,1\}^l$.
- Input: The prover has the polynomial p, and the verifier has an oracle for evaluating f at any point.
- Procedure:
  - The verifier randomly selects an l-length vector $\mathbf{a}$ from $\mathrm{GR}(p^s, r)$ and sends it to the prover.
  - Upon receiving $\mathbf{a}$, the prover and verifier execute the sumcheck protocol to prove that

$$\sum_{\mathbf{x} \in \{0,1\}^l} eq(\mathbf{a}, \mathbf{x}) f(\mathbf{x}) = 0,$$

Lemma 7. *The soundness error of the zero-check protocol over the Galois ring does not exceed* $\frac{l(d+2)}{p^r}$.

In Appendix D.2, we provide the proof for this lemma.

6 SNARKs over Galois Rings

With the polynomial commitment scheme, sumcheck protocol, and ZeroCheck protocol over Galois rings, we can construct transparent SNARK schemes over Galois rings. In this section, we primarily discuss two SNARK schemes: Libra and HyperPlonk. Libra is designed for log-space uniform circuits but achieves a prover runtime of $O(n)$. While HyperPlonk is applicable to arbitrary proof circuits but has a prover runtime of $O(n \log^2 n)$.

6.1 Libra Over Galois Rings

In Galois rings, the presence of zero divisors can affect set consistency checks. In fields, to determine whether two sets S_1 and S_2 are identical, we encode them as polynomials: $p_1(x) = \prod_{a \in S_1}(x - a)$ and $p_2(x) = \prod_{a \in S_2}(x - a)$. We then check if the polynomials are equal to verify the sets' equality. However, in Galois rings, due to the influence of zero divisors, it is possible for different sets S_1 and S_2 to produce the same polynomial encoding. For example, under modulo 8, $(x-1)(x-7) = (x-5)(x-3) \pmod 8$, making it impossible to distinguish between the sets $(1, 7)$ and $(5, 3)$. Libra targets log-space uniform circuits, avoiding the need for set consistency checks.

Libra [33] provides a linear IOP construction specifically for log-space uniform circuits. According to the discussion in section 5, the challenge must come from the Galois ring $\mathrm{GR}(p^s, r)$ that p^r is exponential with respect to the security parameter λ, and the performance over Galois rings is similar to that over the Galois field $\mathrm{GF}(p, r)$. By using Libra's IOP framework combining with the Brakedown polynomial commitment scheme over Galois rings, we can construct a transparent SNARK with linear prover time for log-space uniform circuits. This SNARK has $O(n)$ prover time, $O(\sqrt{n})$ proof size, and $O(\sqrt{n})$ verifier time.

If the circuit to be proven is defined over the ring $G_1 = \mathrm{GR}(p^s, r)$, the challenge needs to be selected from $G_2 = \mathrm{GR}(p^s, kr)$. Let ee denote the cost of performing a multiplication over the ring G_2, be represent the cost of multiplying an element from G_1 with an element from G_2, and bb signify the cost of multiplication within G_1. Additionally, t_1 is the time required for constructing a Merkle tree, t_2 is the time needed to open $|Q|$ positions of the Merkle tree. Taking parameters that maximize the encoding distance, $c_0 = 25.5$, $t = 1.72$, the cost for polynomial commitment is $t_1 + t_2 + \frac{25.5n}{k}ee + 3.44nkbe$.

Then the total prover time required for the entire protocol is:

$$t_1 + t_2 + \frac{25.5 + 19k}{k}nee + (3.44k + 11)nbe$$

The details of the protocol and analysis of prover's time is provided in Appendix E.

6.2 HyperPlonk over Galois Rings

HyperPlonk [14] is capable of handling all NP arithmetic circuits. It has a constraint system that is divided into two parts: gate constraints and permutation constraints. For each gate in the circuit, there are three parts: left input, right input, and output. Each output wires can become the input wires of another gates. We encode the wires according to each circuit gate, using a multilinear polynomial to represent the circuit to be proved. If the circuit has q public input gates, one output gate, and s remaining circuit gates (including secret input gates and intermediate calculation gates), let L_i represent the left input of the i-th gate, R_i represent the right input of the i-th gate, and O_i represent the output of the i-th gate. Define $\langle i \rangle$ as the binary representation of i. Let the multilinear

polynomial M be defined such that $M(0,0,\langle i\rangle) = L_i$, $M(0,1,\langle i\rangle) = R_i$, and $M(1,0,\langle i\rangle) = O_i$. Assume $q+s+1 = 2^v$, where M is a multilinear polynomial with $v+2$ variables. The multilinear polynomial M serves as the witness for the SNARK.

Gate Constraints. Let $S_1(x)$, $S_2(x)$ be multilinear polynomials with v variables. To prove the gate constraint, we need to show that the following polynomial holds for all $\mathbf{x} \in \{0,1\}^v$:

$$0 = S_1(\mathbf{x}) \cdot (M(0,0,\mathbf{x}) + M(0,1,\mathbf{x})) + S_2(\mathbf{x}) \cdot M(0,0,\mathbf{x}) \cdot M(0,1,\mathbf{x}) - M(1,0,\mathbf{x}) + I(\mathbf{x}) \tag{1}$$

For different multiplication, addition, and input-output gates, the multilinear polynomials S_1, S_2 take on different values.

- for an addition gate: $S_1(\langle i\rangle) = 1$, $S_2(\langle i\rangle) = 0$, so $L_i + R_i = O_i$.
- for an multiplication gate: $S_1(\langle i\rangle) = 0$,$S_2(\langle i\rangle) = 1$, so $L_i \cdot R_i = O_i$.
- when $i < q$ or $i = q+s$, $S_1(\langle i\rangle) = 0$,$S_2(\langle i\rangle) = 0$, this applies to input-output gates, so $O_i = I(\langle i\rangle)$.

By invoking the ZeroCheck protocol over the Galois ring, we can prove that Equation 1 holds.

Permutation Constraints. We adopted the second permutation constraint from HyperPlonk. Due to the presence of zero divisors in Galois rings, set consistency check is affected. HyperPlonk's first approach to prove permutation constraint needs set consistency check[14]. Similarly, when using the Spartan IOP framework [29], sparse polynomial processing requires the Sparse technique, which also involves set consistency checks. Therefore, we ultimately used the approach in HyperPlonk to express permutation constraints directly in the form of a sumcheck, thereby avoiding this problem.

Given a permutation $\sigma : \{0,1\}^l \to \{0,1\}^l$, the permutation is to demonstrate that for any $\mathbf{x} \in \{0,1\}^l$, the equation $f(\mathbf{x}) = f(\sigma(\mathbf{x}))$ holds. The permutation σ can be decomposed into l bits, resulting in $\tilde{\sigma} = (\sigma_1(\mathbf{x}), \ldots, \sigma_l(\mathbf{x})) : \mathrm{GR}(p^s, r)^l \to \mathrm{GR}(p^s, r)^l$, where σ_i signifies the position of the i-th bit following the permutation. Notably, for all $\mathbf{x} \in \{0,1\}^l$, each $\sigma_i(\mathbf{x})$ falls within $\{0,1\}$. Consequently, the expression to be verified is:

$$f(\tilde{\sigma}(\mathbf{x})) - g(\mathbf{x}) = 0, \text{ for all } \mathbf{x} \in \{0,1\}^l.$$

Since $f(\tilde{\sigma}(\mathbf{x}))$ cannot be easily described using a multilinear polynomial, it is represented in multilinear form through an equivalence polynomial, denoted as:

$$\sum_{\mathbf{y} \in \{0,1\}^l} (f(\mathbf{y}) \cdot eq(\tilde{\sigma}(\mathbf{x}), \mathbf{y}) - g(\mathbf{y}) \cdot eq(\mathbf{x}, \mathbf{y})) = 0, \text{ for all } \mathbf{x} \in \{0,1\}^l.$$

Then use ZeroCheck to introduce eq and random numbers to turn it into a sumcheck protocol.

$$\sum_{\mathbf{x}\in\{0,1\}^l} eq(\mathbf{a},\mathbf{x}) \cdot \sum_{\mathbf{y}\in\{0,1\}^l} (f(\mathbf{y})eq(\tilde{\sigma}(\mathbf{x}),\mathbf{y}) - g(\mathbf{y})eq(\mathbf{x},\mathbf{y})) = 0$$

To facilitate better computation, the above expression can be rewritten as

$$\sum_{\mathbf{x}\in\{0,1\}^l} eq(\mathbf{r},\mathbf{x}) \cdot \sum_{\mathbf{y}\in\{0,1\}^l} (f(\mathbf{y})eq(\mathbf{x},\tilde{\sigma}^{-1}(\mathbf{y})) - g(\mathbf{y})eq(\mathbf{x},\mathbf{y})) = 0$$

$\tilde{\sigma}^{-1}$ is the inverse of $\tilde{\sigma}$, and we can split every bit just like $\tilde{\sigma}$.

If the circuit to be proven is defined over the ring $G_1 = \mathrm{GR}(p^s, r)$, the challenge needs to be selected from $G_2 = \mathrm{GR}(p^s, kr)$. Let ee denote the cost of performing a multiplication over the ring G_2, be represent the cost of multiplying an element from G_1 with an element from G_2, and bb signify the cost of multiplication within G_1. Additionally, t_1 is the time required for constructing a Merkle tree, t_2 is the time needed to open $|Q|$ positions of the Merkle tree. If the circuit to be proven has a size of n, in HyperPlonk, the permutation constraints apply to the polynomial M, where the size of the coefficients of M is $4n$, with $2^l = 4n$.

The total prover time cost of the protocol is:

$$\frac{39k + 2(\log n + 3)^2 k + 204}{k} nee + (13.76k + 2\log n + 25)nbe$$
$$+ (2(\log n + 4)(\log n + 2) + 7)nbb + 4t_1 + 4t_2$$

The details of the protocol and analysis of prover's time is provided in Appendix F.

7 Comparison with Rinocchio

For a circuit with ℓ wires, t addition gates, m multiplication gates and public input and output x, we provide the following comparison.

Table 2: Comparison with Rinocchio

	Rinocchio	**This work (Libra)**	**This work (HyperPlonk)**
circuit type	NP circuit	log-space uniform circuit	NP circuit
prover time	$O(\ell + m\log m)$	$O(t+m)$	$O((t+m)\log^2(t+m)))$
proof size	$O(1)$	$O(\sqrt{t+m})$	$O(\sqrt{t+m})$
verifier time	$O(\|x\|)$	$O(\sqrt{t+m})$	$O(\sqrt{t+m})$
SRS length	$O(\ell+m)$	–	–

The Rinocchio and our system primarily provide proofs for two-input circuits. Beyond the initial input gates, each circuit gate corresponds to both a left and right input wire, hence ℓ is approximately twice $(t+m)$. More details are provided in G.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable suggestions, and we also thank Sihuang Hu, Chong Shangguan, Kaijie Jiang, Hexiang Huang, Yuanting Shen and others for their valuable discussions on this work. We are supported by the National Key Research and Development Project of China (Grant No. 2023YFB4503203), the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDB0690200) and the National Natural Science Foundation of China (Grant No. 62372447 and No. 61932019).

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham, B., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 2087–2104. ACM (2017), https://doi.org/10.1145/3133956.3134104
2. Bagad, S., Domb, Y., Thaler, J.: The sum-check protocol over fields of small characteristic. IACR Cryptol. ePrint Arch. p. 1046 (2024), https://eprint.iacr.org/2024/1046
3. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: Chatzigiannakis, I., Kaklamanis, C., Marx, D., Sannella, D. (eds.) 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018), https://doi.org/10.4230/LIPIcs.ICALP.2018.14
4. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptol. ePrint Arch. p. 46 (2018), http://eprint.iacr.org/2018/046
5. Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S., Saraf, S.: Proximity gaps for reed-solomon codes. In: Irani, S. (ed.) 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020. pp. 900–909. IEEE (2020), https://doi.org/10.1109/FOCS46700.2020.00088
6. Benger, N., Scott, M.: Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In: Hasan, M.A., Helleseth, T. (eds.) Arithmetic of Finite Fields, Third International Workshop, WAIFI 2010, Istanbul, Turkey, June 27-30, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6087, pp. 180–195. Springer (2010), https://doi.org/10.1007/978-3-642-13797-6_13
7. Bishnoi, A., Clark, P.L., Potukuchi, A., Schmitt, J.R.: On zeros of a polynomial in a finite grid. Comb. Probab. Comput. **27**(3), 310–333 (2018), https://doi.org/10.1017/S0963548317000566
8. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Goldwasser, S. (ed.) Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 326–349. ACM (2012), https://doi.org/10.1145/2090236.2090263
9. Bootle, J., Chiesa, A., Groth, J.: Linear-time arguments with sublinear verification from tensor codes. In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography - 18th

International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12551, pp. 19–46. Springer (2020), https://doi.org/10.1007/978-3-030-64378-2_2
10. Bootle, J., Chiesa, A., Sotiraki, K.: Sumcheck arguments and their applications. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12825, pp. 742–773. Springer (2021), https://doi.org/10.1007/978-3-030-84242-0_26
11. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical gapsvp. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7417, pp. 868–886. Springer (2012), https://doi.org/10.1007/978-3-642-32009-5_50
12. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014), https://doi.org/10.1145/2633600
13. 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, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 315–334. IEEE Computer Society (2018), https://doi.org/10.1109/SP.2018.00020
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.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14005, pp. 499–530. Springer (2023), https://doi.org/10.1007/978-3-031-30617-4_17
15. Chen, S., Cheon, J.H., Kim, D., Park, D.: Verifiable computing for approximate computation. IACR Cryptol. ePrint Arch. p. 762 (2019), https://eprint.iacr.org/2019/762
16. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zksnarks with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12105, pp. 738–768. Springer (2020), https://doi.org/10.1007/978-3-030-45721-1_26
17. Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. IACR Cryptol. ePrint Arch. p. 1784 (2023), https://eprint.iacr.org/2023/1784
18. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. p. 953 (2019), https://eprint.iacr.org/2019/953
19. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: Snarks for ring arithmetic. J. Cryptol. **36**(4), 41 (2023), https://doi.org/10.1007/s00145-023-09481-3
20. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011. pp. 99–108. ACM (2011), https://doi.org/10.1145/1993636.1993651
21. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. J. ACM **62**(4), 27:1–27:64 (2015), https://doi.org/10.1145/2699436

22. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for R1CS. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II. Lecture Notes in Computer Science, vol. 14082, pp. 193–226. Springer (2023), https://doi.org/10.1007/978-3-031-38545-2_7
23. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer (2016), https://doi.org/10.1007/978-3-662-49896-5_11
24. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6477, pp. 177–194. Springer (2010), https://doi.org/10.1007/978-3-642-17373-8_11
25. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Kosaraju, S.R., Fellows, M., Wigderson, A., Ellis, J.A. (eds.) Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada. pp. 723–732. ACM (1992), https://doi.org/10.1145/129712.129782
26. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Nissim, K., Waters, B. (eds.) Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13043, pp. 1–34. Springer (2021), https://doi.org/10.1007/978-3-030-90453-1_1
27. Micali, S.: CS proofs (extended abstracts). In: 35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994. pp. 436–453. IEEE Computer Society (1994), https://doi.org/10.1109/SFCS.1994.365746
28. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013. pp. 238–252. IEEE Computer Society (2013), https://doi.org/10.1109/SP.2013.47
29. Setty, S.T.V.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 704–737. Springer (2020), https://doi.org/10.1007/978-3-030-56877-1_25
30. Soria-Vazquez, E.: Doubly efficient interactive proofs over infinite and non-commutative rings. In: Kiltz, E., Vaikuntanathan, V. (eds.) Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13747, pp. 497–525. Springer (2022), https://doi.org/10.1007/978-3-031-22318-1_18
31. Thaler, J.: Proofs, arguments, and zero-knowledge. Found. Trends Priv. Secur. **4**(2-4), 117–660 (2022), https://doi.org/10.1561/3300000030

32. Wan, Z.X.: Finite fields and Galois rings. World Scientific Publishing Company (2011)
33. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 733–764. Springer (2019), https://doi.org/10.1007/978-3-030-26954-8_24
34. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13510, pp. 299–328. Springer (2022), https://doi.org/10.1007/978-3-031-15985-5_11
35. Zeilberger, H., Chen, B., Fisch, B.: Basefold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part X. Lecture Notes in Computer Science, vol. 14929, pp. 138–169. Springer (2024), https://doi.org/10.1007/978-3-031-68403-6_5

Appendix

A Galois Rings

Proof. Considering the homomorphism $\psi : \mathrm{GR}(p^s, r) \to \mathrm{GF}(p, r)$, we define the mapping as follows:

$$\begin{array}{ccc} \mathbb{Z}_{p^s}[x] & \to & \mathbb{F}_p[x] \\ a_0 + a_1 x + \cdots + a_n x^n & \mapsto & \bar{a}_0 + \bar{a}_1 x + \cdots + \bar{a}_n x^n \end{array}$$

where each coefficient a_i in the Galois ring $\mathrm{GR}(p^s, r)$ is mapped to its corresponding residue $\bar{a}_i$ modulo p.

For a polynomial f of degree at most d with coefficients in $\mathrm{GR}(p^s, r)$, we define a mapping Ψ, where $\Psi(f)$ is derived by applying ψ to each coefficient of f, resulting in a polynomial f'. Clearly, Ψ is a homomorphism. If at some point $\mathbf{x}$, we have $f(\mathbf{x}) = 0$, then it follows that $\Psi(f)(\psi(\mathbf{x})) = 0$, because the computation of $f(\mathbf{x})$ modulo p^s is equivalent to computing $\Psi(f)(\psi(\mathbf{x}))$ modulo p.

Under modulo p calculations within the field $\mathbb{F}_p$, the degree of $\Psi(f)$ remains the same as that of f, which does not exceed d. Thus, $\Psi(f)$ has at most d roots in $\mathbb{F}_p$. For each root in $\mathbb{F}_p$, there are at most $p^{(s-1)r}$ preimages under ψ in $\mathrm{GR}(p^s, r)$. Therefore, the number of roots of f in $\mathrm{GR}(p^s, r)$ does not exceed $dp^{(s-1)r}$.

Consequently, the probability of randomly selecting a root of f is at most $\frac{dp^{(s-1)r}}{p^{sr}} = \frac{d}{p^r}$. □

Proof. Proof by contradiction: Suppose f is reducible in the ring $\mathrm{GR}(p^s, r)$. Then it can be expressed as

$$f = a \times b \pmod{p^s}$$

where (a) and (b) both belong to $\mathbb{Z}_{p^s}[x]$. Since the above equation holds modulo (p^s), it must also hold modulo p. Thus, we have

$$f = \bar{a} \times \bar{b} \pmod{p}$$

where $\bar{a}$ and $\bar{b}$ are polynomials obtained by reducing each coefficient in a and b modulo p. This contradicts the fact that f is irreducible in $\mathrm{GF}(p^r)$. Therefore, the proof is complete. □

B The Proofs of Expander Code over Galois Rings

The following holds with all but negligible probability over the choices of random matrices A, B:

(1) for every $0 < \|\mathbf{x}\|_0 < \beta n, \mathbf{y}^\top = \mathbf{x}^\top \cdot A \neq \mathbf{0}$;
(2) for every $\alpha\beta n \leq \|\mathbf{z}\|_0 < \beta n, \mathbf{v}^\top = \mathbf{z}^\top \cdot B$ has $\|\mathbf{v}\|_0 \geq \beta n$.

Assuming that these two properties hold, one can show that Enc_n has distance β , that is , for every $\mathbf{x} \neq \mathbf{0}, \mathbf{w} = Enc_n(\mathbf{x})$ satisfies $\|\mathbf{w}\|_0 \geq \beta n$. To this end, we consider the following three cases.

1 $\|\mathbf{x}\|_0 \geq \beta n$. In this case, $\mathbf{w} = \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \\ \mathbf{v} \end{pmatrix}$ trivially satisfies $\|\mathbf{w}\|_0 \geq \|\mathbf{x}\|_0 \geq \beta n$.
2 $\mathbf{z}^\top = \mathrm{Enc}_{\alpha n}(\mathbf{x}^\top \cdot A)$ satisfies $\|\mathbf{z}\|_0 \geq \beta n$. Again, since $\mathbf{w}$ contains $\mathbf{z}$, we have $\|\mathbf{w}\|_0 \geq \|\mathbf{z}\|_0 \geq \beta n$.
3 $0 < \|\mathbf{x}\|_0 < \beta n$ and $\|\mathbf{z}\|_0 < \beta n$. In this case, by Property (1) above, we have that $\mathbf{y}^\top = \mathbf{x}^\top \cdot A \neq \mathbf{0}$. By the code property of $\mathrm{Enc}_{\alpha n}$, we have that every non-zero vector $\mathbf{y}$ is mapped to $\mathbf{z} = \mathrm{Enc}_{\alpha n}(\mathbf{y})$ of Hamming weight $\|\mathbf{z}\| \geq \alpha\beta n$. Now we have that $\alpha\beta n \leq \|\mathbf{z}\| < \beta n$, and by the Property (2) above, $\|\mathbf{v}\|_0 \geq \beta n$, which finishes the proof.

It remains to choose the values of the parameters $\alpha, \beta, t, c_n, d_n$ such that the Property (1) and (2) are satisfied with probability at least $1 - s \cdot 2^{-100}$ over the choice of matrices A and B.

The probability of the occurrence of events $E^{(1)}_{n,k}$ and $E^{(3)}_{n,k}$ is less than 2^{-100}. Because both probabilities are independent of whether they are related to a ring or a field, so the proof aligns with the proof of the Brakedown scheme [22] Claim 1 and Claim 3 in section 5, and is not discussed further here. However, proving events $E^{(2)}_{n,k}$ and $E^{(4)}_{n,k}$ in the context of the ring is more difficult, due to the impact of zero divisors. The zero divisors affect the probability that elements of the ring $\mathrm{GR}(p^s, r)$ will be zero after a random linear combination, which is not simply $\frac{1}{p^{rs}}$ as might be expected. In the worst-case scenario, this probability may reach $\frac{1}{p^r}$.

Since the encoded elements are from the Galois ring $\mathrm{GR}(p^s, r)$, the total number of possible of k non-zero elements is p^{ksr}. The probability of events $E^{(2)}_{n,k}$ and $E^{(4)}_{n,k}$ occurring will significantly increase based on the union bound. The solution follows a similar approach to the Basefold scheme when analyzing codeword distance: not all elements in a Galois ring have the same probability of a random linear combination equaling zero, which is not always $\frac{1}{p^r}$. The higher the probability that a random linear combination equals zero, the fewer such elements there are in the Galois ring. We need to leverage the properties of Galois rings and classify the elements into s categories based on the probability of a random linear combination equaling zero, then calculate the probabilities for each category separately.

C The Proofs of Polynomial Commitment over Galois Rings

C.1 The Proof of AHIV17 Repetition Version

Claim 4 (AHIV17 Repetition Version) *Fix any k-$[l, n, d]$ code $C \subset GR(p^s, kr)^l$ over the Galois ring $GR(p^s, r)$, and a proximity parameter $e \in \left\{0, \cdots, \lfloor \frac{d-1}{3} \rfloor\right\}$. For a matrix $U \in GR(p^s, kr)^{m \times l}$ with $d(U, C^m) > e$, and a matrix $R \in GR(p^s,$*

$r)^{k\times m}$ where each element of R is randomly chosen from the $GR(p^s, r)$, let $W = RU$. Then we have:

$$\Pr[d(W, C^k) \le e] \le \frac{e+1}{p^{rk}}.$$

We say that $\Delta(v, C)$ represents the set of positions where v differs from the closest codeword in C, and $\Delta(V, C^m) = \bigcup_{i\in[m]} \Delta(V_i, C^m)$, where V_i is the i-th row of matrix V.

Proof. Let $\mathcal{W}$ be the distribution of all possible values of W. We first prove that not all W satisfy $d(W, C^k) \le e$. Using proof by contradiction, assume that for all $W \in \mathcal{W}$, we have $d(W, C^k) \le e$. Then we can select a specific $W^* \in \mathcal{W}$ such that for all $W \in \mathcal{W}$, $d(W^*, C^k) \ge d(W, C^k)$. And we still have $d(W^*, C^k) \le e$. From the conditions of the theorem, we know that $d(U, C^m) > e$, which implies that there exists a submatrix U^* consisting of some rows of U. We assume U^* consists of m' rows such that $\Delta(U^*, C^{m'}) \setminus \Delta(W^*, C^k) \neq \varnothing$ and $\Delta(U^*, C^{m'}) \cup \Delta(W^*, C^k) \ge e + 1$.

Let each row of the matrix W^* be W_i^*, and each row of U^* be U_i^*. The closest codeword to W_i^* is denoted as L_i, and we have $W_i^* = L_i + A_i$. Similarly, the closest codeword to U_i^* is denoted as M_i, and we have $U_i^* = M_i + B_i$. The matrices formed by L_i, M_i, A_i, and B_i are denoted as L, M, A, and B, respectively. Thus, we can write $W^* = L + A$ and $U^* = M + B$. Additionally, we know that $\Delta(A, \mathbf{0}^k) \le e$ and $\Delta(B_i, \mathbf{0}) \le e$, otherwise, this would contradict our previous assumption.

Randomly select k vectors $\mathbf{r_1}, \cdots, \mathbf{r_k}$, each of length m', with elements chosen from $\mathrm{GR}(p^s, r)$. Define the vector $W_i' = W_i^* + \mathbf{r_i}[1]U_1^* + \cdots + \mathbf{r_i}[m']U_{m'}^*$. Then,

$$\begin{aligned} W_i' &= L_i + A_i + \mathbf{r_i}[1]U_1^* + \cdots + \mathbf{r_i}[m']U_{m'}^* \\ &= L_i + A_i + \mathbf{r_i}[1](M_1 + B_1) + \cdots + \mathbf{r_i}[m'](M_m + B_m) \\ &= (L_i + \mathbf{r_i}[1]M_1 + \cdots + \mathbf{r_i}[m']M_{m'}) + (A_i + \mathbf{r_i}[1]B_1 + \cdots + \mathbf{r_i}[m']B_{m'}) \end{aligned}$$

The k rows W_i' form the matrix W'. We now consider each column of the matrix W'. Let $t \in (\Delta(U^*, C^m)$. In the t-th column, there must exist a row where $B_{i^*}[t] \neq 0$. Therefore, in the j-th row and t-th column of W',

$$\begin{aligned} W_j'[t] = (L_j + \mathbf{r_j}[1]M_1[t] + \cdots + \mathbf{r_j}[m']M_{m'}[t]) \\ +((A_j + \mathbf{r_j}[1]B_1[t] + \cdots + \mathbf{r_j}[m']B_{m'}[t])) \end{aligned}$$

To make the expression $((A_j + \mathbf{r_j}[1]B_1[t] + \cdots + \mathbf{r_j}[m']B_{m'}[t]))$ equal to zero, there is a term $r_j[i^*]B_{i^*}[t]$, where we treat $\mathbf{r_j}[i^*]$ as a variable, and it becomes a univariate polynomial in terms of this variable. In order for this term to equal a specific value. According to Fact 2, the solutions constitute at most $\frac{1}{p^r}$ of the total possible values. For a particular row j, if every row evaluates to zero, each row has a solution set of size $\frac{1}{p^r}$. This applies to a specific row j. If all rows are zero, the proportion of solutions for each row is $\frac{1}{p^r}$. Given that all k rows need to satisfy the condition, the combined proportion of solutions over all possible values is $\left(\frac{1}{p^r}\right)^k = \frac{1}{p^{kr}}$.

Since $\Delta(U^*, C^m) < l$, as long as $\frac{l}{p^{kr}} < l$, there must exist a set of random values such that not every column of W' in $\Delta(U^*, C^m)$ is zero. Let $\hat{L}$ be the matrix where each row is defined as $L_i + \mathbf{r_i}[1]M_1 + \cdots + \mathbf{r_i}[m']M_{m'}$. Because $\Delta(U^*, C^{m'}) \setminus \Delta(W^*, C^k) \neq \varnothing$, so $\Delta(W', \hat{L}) \geq d(W^*, C^k) + 1$. At this point, we need to demonstrate that there is no other codeword C_0 such that $d(C_0, W') \leq d(W^*, C^k)$. If such a codeword C_0 exists, then we have

$$\begin{aligned} d &\leq d(C_0, \hat{L}) \\ &\leq d(C_0, W') + d(W', \hat{L}) \\ &\leq d(W^*, C^k) + e \leq e + e < d \end{aligned}$$

We assume $d(W', \hat{L}) < e$ based on the initial assumption that for all $W \in \mathcal{W}$, we have $d(W, C^k) \leq e$. At this point, we obtain $d(W', C^k) > d(W^*, C^k)$, which contradicts the definition of W^*. Therefore, we conclude that not all W satisfy $d(W, C^k) \leq e$.

Claim 5 *Consider a fixed $[l, n, d]$ code $C \subset GR(p^s, kr)^l$ over the Galois ring $GR(p^s, r)$, and a proximity parameter $e \in \left\{0, \cdots, \lfloor \frac{d-1}{3} \rfloor\right\}$. For any matrices $W \in GR(p^s, kr)^{k \times l}$ and $V \in GR(p^s, kr)^{k \times l}$, we define an "affine transformation" as follows. Let $S \in GR(p^s, r)^{k \times k}$ be a $k \times k$ matrix where each element $S_i[j]$ (representing the element in the i-th row and j-th column) is randomly chosen from $GR(p^s, r)$. The elements $W_i[j]$ and $V_i[j]$ denote the (i, j) elements of matrices W and V, respectively. Define a matrix L such that each row of L is given by $L_i = W_i + \sum_{j \in [k]}(S_i[j]V_j)$. In matrix notation, we have $L = W + SV$. The distribution of all possible values of L is denoted by $\mathcal{L}_{W,V}$. Then, either (1) for all $L \in \mathcal{L}_{W,V}$, we have $d(L, C^k) \leq e$, or (2) at most a proportion of $\frac{e+1}{p^{kr}}$ of the possible values of L satisfy $d(L, C^k) \leq e$.*

Proof. As demonstrated in AHIV17, we observe that for any distribution $\mathcal{L}_{W,V}$, there exist N points within a C^k-distance of at most e if and only if the distribution $\mathcal{L}_{W+C_0, V+C_1}$ also contains N points within a C^k-distance of at most e, where C_0 and C_1 are arbitrary codewords in C^k.

Now, consider the case when S takes the value S_0, where a point $P = W + S_0V$ is within a distance of less than d from a point $C' \in C^k$. It follows that the distributions of $P + SV$ and $W + SV$ are identical. Additionally, the number of points in $\mathcal{L}_{P,V}$ and $\mathcal{L}_{P-C',V}$ that are within a C^k-distance of at most e is the same, and we have $d(P - C', \mathbf{0}^k) \leq e$. Therefore, it suffices to consider only the case where $d(W, \mathbf{0}) \leq e$.

At this point, if $d(V, \mathbf{0}^k) \geq e + 1$ and the closest point in C^k to V is $\hat{C}$, let $V = V' + \hat{C}$, then we have $|V'| \geq e + 1$. Additionally, $\Delta(V, C_0) \cap \Delta(W, C^k) \leq \Delta(W, C^k) = e$. Let $L = W + SV = W + SV' + S\hat{C}$. For the t-th column in $\Delta(V, C_0) \cap \Delta(W, \mathbf{0}^k)$, there must exist a row q^* such that $V'_{q^*}[t] \neq 0$. In the i-th

row and t-th column of L, we have

$$\begin{aligned} L_i[t] &= W_i[t] + S_i[1]V_1[t] + \cdots + S_i[k]V_k[t] \\ &= W_i[t] + (S_i[1]V_1'[t] + \cdots + S_i[k]V_k'[t]) \\ &+ (S_i[1]\hat{C}_1[t] + \cdots + S_i[k]\hat{C}_k[t]) \end{aligned}$$

If the t-th column of L is not in $\Delta(L, C^k)$, then for all $i \in [k]$, we have $W_i[t] = S_i[1]V_1'[t] + \cdots + S_i[k]V_k'[t]$. This equation for the i-th row can be viewed as a linear equation in the variable $S_i[q^*]$. According to Fact 2, since $V_{q^*}'[t] \neq 0$, the number of solutions to this equation is at most $\frac{1}{p^r}$. Therefore, in the i-th row of matrix S, at most $\frac{1}{p^r}$ values satisfy the condition. Since this must hold for all i, there are at most $\frac{1}{p^r}$ satisfying values per row, and for k rows, only $\left(\frac{1}{p^r}\right)^k = \frac{1}{p^{kr}}$ values satisfy the condition.

We also need to prove that only $S\hat{C}$ can satisfy $d(L, S\hat{C}) \leq e$ as the closest codeword. There cannot be another codeword $\hat{C}' \in C^k$ such that $d(\hat{C}', L) \leq e$. Otherwise, we would have

$$\begin{aligned} d(S\hat{C}, \hat{C}') &\leq d(S\hat{C}, L) + d(L, \hat{C}') \\ &\leq e + e = 2e < d \end{aligned}$$

which leads to a contradiction.

For a given column, the probability is $\frac{1}{p^{kr}}$, and the total number of possible columns is bounded by $|\Delta(V, C_0) \cap \Delta(W, C^k)| \leq |\Delta(W, C^k)| = e$. According to the union bound, we have at most $\frac{e}{p^{kr}}$ points that satisfy $d(L, C^k) \leq e$. Additionally, if all elements in S are set to zero, this also satisfies $d(L, C^k) \leq e$. Therefore, the total probability does not exceed

$$\frac{e}{p^{kr}} + \frac{1}{p^{skr}} \leq \frac{e+1}{p^{kr}}.$$

At this point, we only need to consider the cases where $d(W, \mathbf{0}^k) \leq e$ and $d(V, \mathbf{0}) \leq e$:

1 $|Support(W) \cup Support(V)| \leq e$. This means that all points are contained within a "ball" centered at $\mathbf{0}^k$ with radius e. Naturally, this ensures that for all $L \in \mathcal{L}_{W,V}$, we have $d(L, C^k) \leq e$.
2 $|Support(W) \cup Support(V)| \geq e + 1$. Since $d(W, \mathbf{0}^k) \leq e$ and $d(V, \mathbf{0}) \leq e$, it follows that $\Delta(W, \mathbf{0}^k) \cap \Delta(V, \mathbf{0}) \leq e - 1$. For each column $t \in \Delta(W, \mathbf{0}^k) \cap \Delta(V, \mathbf{0})$, there must exist a row q^* such that $V_{q^*}[t] \neq 0$. In the i-th row and t-th column of L, we have:

$$L_i[t] = W_i[t] + S_i[1]V_1[t] + \cdots + S_i[k]V_k[t]$$

If the t-th column of L is not in $\Delta(L, C^k)$, then for all $i \in [k]$, we have $W_i[t] = S_i[1]V_1[t] + \cdots + S_i[k]V_k[t]$. For the i-th row, this equation can be

viewed as a linear equation in the variable $S_i[q^*]$. According to Fact 2, since $V_{q^*}[t] \neq 0$, the proportion of solutions to this equation will not exceed $\frac{1}{p^r}$. In the i-th row of matrix S, at most $\frac{1}{p^r}$ values satisfy this condition. Since this condition must hold for all i, there are at most $\frac{1}{p^r}$ satisfying values per row, and for k rows, only $\left(\frac{1}{p^r}\right)^k = \frac{1}{p^{kr}}$ values will satisfy this condition.
Because $\Delta(W, \mathbf{0}^k) \cap \Delta(V, \mathbf{0}) \leq e-1$, using the union bound, the total number of points that satisfy $d(L, C^k) \leq e$ is at most $\frac{e-1}{p^{kr}}$.
Additionally, if all elements of S are set to zero, this also satisfies $d(L, C^k) \leq e$. Therefore, the total probability does not exceed

$$\frac{e-1}{p^{kr}} + \frac{1}{p^{skr}} \leq \frac{e}{p^{kr}}.$$

Therefore, $\mathcal{L}_{W,V}$ contains at most a proportion of $\frac{e}{p^{kr}}$ points within a "ball" centered at $\mathbf{0}^k$ with radius e.
Next, we show that no other points in C^k are within a distance of e from any point in $\mathcal{L}_{W,V}$. If there exists some $C' \in C^k$ such that there is a point $L \in \mathcal{L}_{W,V}$ with $d(C', L) \leq e$, then we have:

$$\begin{aligned} d(C', \mathbf{0}^k) &\leq d(C', L) + d(L, \mathbf{0}) \\ &\leq e + |\text{Support}(W)| + |\text{Support}(V)| \leq e + 2e = 3e \end{aligned}$$

This contradicts the fact that $d(C', \mathbf{0}^k) \geq d$. Therefore, such a C' cannot exist, and at most e points satisfy $d(L, C^k) \leq e$. This holds.

□

In conclusion, we have:

$$\Pr[d(W, C^k) \leq e] \leq \frac{e+1}{p^{rk}}.$$

□

C.2 The Proof of Completeness

Proof. To demonstrate completeness, we first express the matrix U as:

$$U = \begin{bmatrix} f(0,\cdots,0,0,\cdots,0) & f(0,\cdots,0,0,\cdots,1) & \cdots & f(0,\cdots,0,1,\cdots,1) \\ f(0,\cdots,1,0,\cdots,0) & f(0,\cdots,1,0,\cdots,1) & \cdots & f(0,\cdots,1,1,\cdots,1) \\ \vdots & \vdots & \vdots & \vdots \\ f(1,\cdots,1,0,\cdots,0) & f(1,\cdots,1,0,\cdots,1) & \cdots & f(1,\cdots,1,1,\cdots,1) \end{bmatrix}$$

To compute $f(r_1, \cdots, r_l)$, we use the formula:

$$f(r_1, \cdots, r_l) = \sum_{\mathbf{b} \in \{0,1\}} \prod_{i \in [1,l]} \left((1-r_i)(1-b_i) + r_i b_i\right) f(\mathbf{b}),$$

which can be rewritten as:

$$f(r_1, \cdots, r_l) = \mathbf{q}_1^\top U \mathbf{q}_2.$$

An important observation is that if we treat the i-th row of the matrix U, denoted U_i, as m/k elements $w_1, \cdots, w_{m/k}$ in $\mathrm{GR}(p^s, kr)$, then for $a \in \mathrm{GR}(p^s, r)$, we have:

$$\begin{aligned}
&a \cdot (w_1, \cdots, w_{m/k}) \\
=&(a \cdot w_1, \cdots, a \cdot w_{m/k}) \\
=&(a(w_{1,1}, \cdots, w_{1,k}), \cdots, a(w_{(m/k),1}, \cdots, w_{(m/k),k})) \\
=&(aw_{1,1}, \cdots, aw_{1,k}, \cdots, aw_{(m/k),1}, \cdots, aw_{(m/k),k}) \\
=&a \cdot (U_i[1], \cdots, U_i[m]).
\end{aligned}$$

Since each w_i can be viewed as k elements from $\mathrm{GR}(p^s, r)$, the computation is the same whether we treat U as a matrix in $\mathrm{GR}(p^s, r)^{m \times m}$ or in $\mathrm{GR}(p^s, kr)^{m \times (m/k)}$. Thus, the result of $\mathbf{q}_1^\top U$ remains the same in "form" regardless of the interpretation. Therefore, the linear combination of the matrix can be seen as a linear combination over $\mathrm{GR}(p^s, r)$, ensuring the completeness of the polynomial commitment. □

C.3 The Proof of Soundness

Proof. The encoded coefficient matrix $\hat{U} \in \mathrm{GR}(p^s, kr)^{m \times (m/k)}$, where $N = m/k$, has a code distance of d and a relative distance of $\gamma = d/N$. First, we show that if the prover can pass the testing phase with a probability greater than $\frac{1}{p^{kr}} + \left(1 - \frac{\gamma}{3}\right)^{l_Q}$, then there must exist a codeword $[c_1, \cdots, c_m] \in C^m$ such that:

$$E := |\{j \in [N] : \exists i \in [m], \text{ such that } c_{i,j} \neq \hat{U}_{i,j}\}| \leq \left(\frac{\gamma}{3}\right) N.$$

Let $e = \left\lfloor \frac{\gamma N}{3} \right\rfloor$. If no such codeword $[c_1, \cdots, c_m]$ exists, then we have $d(\hat{U}, C^m) > e$. According to claim 3, since $V = R \cdot \hat{U}$, the probability that $d(V, C^k)$ is greater than e is at least $1 - \frac{e+1}{p^{kr}}$.

Thus, during the testing phase, the probability that the verifier randomly selects l columns without hitting any columns in $\Delta(V, C^k)$ is at most $(1 - \frac{e}{3})^{l_Q} = \left(1 - \frac{\gamma}{3}\right)^{l_Q}$. Define the event E_1 as $d(V, C^k) > e$, and the event E_2 as the verifier selecting l columns without hitting any in $\Delta(V, C^k)$. Then, we have:

$$\begin{aligned}
\Pr[P^* \text{wins}] &\leq \Pr[P^* \text{wins} | \bar{E_1}] \Pr[\bar{E_1}] + \Pr[P^* \text{wins} | E_1] \Pr[E_1] \\
&\leq \Pr[\bar{E_1}] + \Pr[E_2 | E_1] \Pr[E_1] \\
&< \frac{1}{p^{kr}} + \Pr[E_2 | E_1] = \frac{1}{p^{kr}} + (1 - \frac{\gamma}{3})^{l_Q}
\end{aligned}$$

This contradicts the fact that the prover could pass the testing phase, so there must exist such a codeword $c_1, \cdots, c_m$.

We can observe that each c_i is the closest codeword to each row U_i of $\hat{U}$; otherwise, the distance between two codewords would be less than d. Define $w := \sum_{i=1}^{m} q_{1,i} \cdot c_i$. Next, we show that if the prover can pass the testing phase with a probability greater than $\frac{e+1}{p^k} + \left(1 - \frac{d}{3}\right)^{l_Q}$, and pass the evaluation phase with a probability greater than $\left(1 - \frac{2\gamma}{3}\right)^{l_Q}$, then let $u := \sum_{i=1}^{m} q_{1,i} \cdot U_i$ and $w = \mathrm{Enc}(u)$.

If $w \neq \mathrm{Enc}(u)$, then w and $\mathrm{Enc}(v)$ are two distinct codewords in C, meaning that the number of positions where they are identical will not exceed $(1-\gamma)N$. Let the set of positions where the two codewords are identical be denoted as A. If the verifier selects a column $j \notin A \cup E$, then the test will fail because w and $\mathrm{Enc}(u)$ can only match within $A \cup E$. We have $|A \cup E| \leq |A| + |E| \leq (1-\gamma)N + \left(\frac{\gamma}{3}\right) N = \left(1 - \frac{2\gamma}{3}\right) N$. Since the verifier selects the set of columns Q randomly, if the prover is cheating, the probability of passing the evaluation phase will not exceed $\left(1 - \frac{2\gamma}{3}\right)^{l_Q}$.

Thus, if the prover can pass the testing phase with a probability greater than $\frac{e+1}{p^k} + \left(1 - \frac{d}{3}\right)^{l_Q}$, and the evaluation phase with a probability greater than $\left(1 - \frac{2\gamma}{3}\right)^{l_Q}$, the binding property of the scheme is ensured. □

C.4 The Proof of Extractability

We prove that a randomly chosen $N \times N$ matrix over the Galois ring $\mathrm{GR}(p^s, r)$ is invertible with high probability. This allows us to solve for the coefficient matrix U.

First, we consider the case where $p^r > N$ and $1 - \frac{N}{p^r}$ is non-negligible.

Lemma 8. *Assuming that $1 - \frac{N}{p^r} > 0$ and is non-negligible, then by randomly selecting an $N \times N$ matrix R from $GR(p^s, r)$, there is a probability of $1 - \frac{N}{p^r}$ that R is invertible.*

Proof. Using the adjugate matrix method for matrix inversion,

$$R^{-1} = \frac{A^*}{\det(A)} = \frac{1}{\det(A)} \begin{bmatrix} R^*_{11} & \cdots & R^*_{1N} \\ \vdots & \ddots & \vdots \\ R^*_{N1} & \cdots & R^*_{NN} \end{bmatrix}^\top$$

Let R^*_{ij} represent the algebraic cofactor of matrix R with respect to the (i, j) entry. We observe that if $\det(R)$ is non-zero and not a zero divisor, then R^{-1} exists. Now, we calculate the probability that $\det(R)$ is non-zero and not a zero divisor.

$$\det(R) = \sum_{\sigma \in S_N} \mathrm{sgn}(\sigma) \prod_{i=1}^{N} A_{i,\sigma(i)}$$

where S_N is the set of all permutations on $\{1, 2, \cdots, N\}$, and $\text{sgn}(\sigma)$ denotes the sign of permutation σ. If σ has an even number of inversions, then $\text{sgn}(\sigma) = 1$; otherwise, $\text{sgn}(\sigma) = -1$. In fact, $\det(R)$ can be viewed as a multilinear polynomial in the N^2 variables R_{ij}, and the degree of the polynomial is at most N. Let us define this polynomial as f_R.

By a known fact about Galois rings, all zero divisors in $\text{GR}(p^s, r)$ lie in the ideal (p). Therefore, we only need to show that f_R is non-zero modulo p with a non-negligible probability. Consider the isomorphism map ψ

$$\begin{aligned} \mathbb{Z}_{p^s} &\to \mathbb{F}_p \\ c_0 + c_1 p + \cdots + c_{p-1} p^{s-1} &\mapsto c_0 \end{aligned}$$

For a polynomial f of degree at most d with coefficients in $\text{GR}(p^s, r)$, we define a mapping Ψ, where $\Psi(f)$ is derived by applying ψ to each coefficient of f, resulting in a polynomial f'. Clearly, Ψ is a homomorphism. If at some point $\mathbf{x}$, we have $f(\mathbf{x}) = 0 \pmod p$, then it follows that $\Psi(f)(\psi(\mathbf{x})) = 0$.

Let $\Psi(f_R) = f'_R$. If f'_R evaluates to a non-zero value, then f_R will not be a zero divisor in $\text{GR}(p^s, r)$. Since the variables R_{ij} are randomly chosen from $\text{GR}(p^s, r)$, so $\psi(R_{ij})$ are randomly chosen from $\text{GF}(p, r)$. By the Schwartz-Zippel lemma for fields, the degree of f'_R is at most N, so the probability that a randomly chosen point evaluates to 0 is at most $\frac{N}{p^r}$. Therefore, the probability that f'_R is non-zero is at least $1 - \frac{N}{p^r}$, which is non-negligible. This completes the proof. □

If the ring $\text{GR}(p^s, r)$ does not satisfy $p^r > N$, then the $\text{GR}(p^s, r)$ code used in encoding must be a $[l, n, d] - k - \text{GR}(p^s, r)$ code, where $p^{kr} \gg N$. Choose a k' such that k' is a divisor of k and $p^{k'r} > N$. At this point, the extractor can send k' challenges from $\text{GR}(p^s, r)$ to simulate a challenge from $\text{GR}(p^s, k'r)$. Then, using Lemma 8, we can obtain an invertible $N \times N$ matrix with non-negligible probability.

If the probability that the matrix is invertible is p_i, by repeating the process $\text{poly}(\frac{1}{p_i})$ times, we can ensure that with probability $1 - \text{negl}(\lambda)$, the matrix U can be successfully extracted.

D The Proofs of Toolboxes over Galois Rings

D.1 The Proof of Sumcheck over Galois Rings

Proof. If the prover cheats, then there must exist at least one round i where the prover's univariate polynomial $p_i(X_i)$ does not equal the following polynomial:

$$s_i(X_i) = \sum_{(x_{i+1}, \cdots, x_l) \in \{0,1\}^{l-i}} g(r_1, \cdots, r_{i-1}, X_i, x_{i+1}, \cdots, x_l).$$

However, $s_i(r_i) = g_i(r_i)$ is still satisfied. For each round, both s_i and g_i have a degree at most d. According to the fact 3, the probability of randomly choosing r_i from $\text{GR}(p^s, r)$ such that $s_i(r_i) = g_i(r_i)$ is at most $\frac{d}{p^r}$. Since the protocol involves l rounds, the soundness error, by the union bound, does not exceed $\frac{ld}{p^r}$. □

D.2 The Proof of ZeroCheck over Galois Rings

Proof. Note that $g(\mathbf{a}) = \sum_{\mathbf{x}\in\{0,1\}^l} eq(\mathbf{a},\mathbf{x})f(\mathbf{x})$ is a multilinear polynomial in the variables $\mathbf{a}$, and its value on $\{0,1\}^l$ is identical to that of f. A multilinear polynomial is uniquely determined by its values on $\{0,1\}^l$. If f is zero on all points in $\{0,1\}^l$, then g is the zero polynomial and evaluates to zero at any point. If f is not zero everywhere on $\{0,1\}^l$, then g is not the zero polynomial. Moreover, the degree of g does not exceed l, according to fact 3, the probability that g evaluates to zero at a random point is at most $\frac{l}{p^r}$. During the sumcheck protocol, since the eq function involves each variable only once and each round's univariate polynomial has a degree at most $d+1$, the soundness error of the sumcheck protocol is at most $\frac{l(d+1)}{p^r}$. Therefore, the overall soundness error of the protocol does not exceed $\frac{l(d+1)+l}{p^r} = \frac{l(d+2)}{p^r}$. □

E Prover Cost of Libra over Galois Ring

Firstly we review the GKR protocol [21].

For a log-space uniform circuit defined over a Galois ring $\mathrm{GR}(p^s, r)$, the circuit is divided into l layers from top to bottom, with the input layer as the l-th layer. The i-th layer contains n_i circuit gates, where $n_i = 2^{s_i}$. The values in the arithmetic circuit of the i-th layer are encoded as a polynomial V_i, where $V_i(\mathbf{x})$ represents the value of the gate at index x, and $\mathbf{x} \in \{0,1\}^{s_i}$. The multilinear extension of V_i is denoted as $\tilde{V}_i(\mathbf{x})$.

We define two polynomials, add_i and $\mathrm{mult}_i : \{0,1\}^{s_{i-1}+2s_i} \to \{0,1\}$. The first input to these polynomials is the index of a gate in the $(i-1)$-th layer, denoted by $\mathbf{z} \in \{0,1\}^{s_{i-1}}$, and the other two inputs are the indices of gates in the i-th layer, $\mathbf{x}, \mathbf{y} \in \{0,1\}^{s_i}$. The polynomial add_i outputs 1 if the gate z is an addition gate and x, y are its left and right inputs, respectively; otherwise, it outputs 0. Similarly, mult_i constrains the multiplication relationship between the i-th layer and the $(i-1)$-th layer.

Thus, we have:

$$V_i(z) = \sum_{x,y\in\{0,1\}^{s_{i+1}}} (\mathrm{add}_{i+1}(z,x,y))(V_{i+1}(x) + V_{i+1}(y)) + \mathrm{mult}_{i+1}(z,x,y)(V_{i+1}(x)V_{i+1}(y))$$

Thus, V_i can be computed from V_{i+1}, which implies that the multilinear extension $\tilde{V}_i$ can also be computed from $\tilde{V}_{i+1}$. If we wish to compute $\tilde{V}_i(g)$, we have:

$$\tilde{V}_i(g) = \sum_{x,y\in\{0,1\}^{s_{i+1}}} (\tilde{\mathrm{add}}_{i+1}(z,x,y))(\tilde{V}_{i+1}(x) + \tilde{V}_{i+1}(y)) + \tilde{\mathrm{mult}}_{i+1}(z,x,y)(\tilde{V}_{i+1}(x)\tilde{V}_{i+1}(y))$$

Let the Galois ring $\mathrm{GR}(p^s, r)$ be G_1, and $\mathrm{GR}(p^s, kr)$ be G_2, where p^{kr} is exponential in the security parameter λ. Let the circuit $C : G_1^n \to G_1^s$ be a circuit of depth l. The i-th layer contains n_i gates, where $n_i = 2^{s_i}$. The prover aims to prove that $\mathrm{out} = C(\mathrm{in})$.

- Encode out as a multilinear polynomial $\tilde{V}_0$. The verifier randomly selects $g \in G_2^{s_0}$ and sends it to the prover; both parties then compute $\tilde{V}_0(g)$.
- The prover and verifier execute the sumcheck protocol, with the randomness drawn from $G_2^{s_1}$, to prove that

$$\tilde{V}_0(g) = \sum_{x,y \in \{0,1\}^{s_1}} \tilde{\mathrm{mult}}_1(g,x,y)(\tilde{V}_1(x)\tilde{V}_1(y)) + \tilde{\mathrm{add}}_1(g,x,y)(\tilde{V}_1(x) + \tilde{V}_1(y)).$$

  In the final round of the sumcheck protocol, the prover sends $\tilde{V}_1(u^{(1)})$ and $\tilde{V}_1(v^{(1)})$ to the verifier. The verifier computes $\tilde{\mathrm{mult}}_1(g, u^{(1)}, v^{(1)})$ and $\tilde{\mathrm{add}}_1(g, u^{(1)}, v^{(1)})$v and checks that the last round of the sumcheck is satisfied.
- For each $i \in [l-1]$:
  - The verifier randomly selects $\alpha^{(i)}$ and $\beta^{(i)}$ and sends them to the prover.
  - The prover and verifier execute the sumcheck protocol, with randomness drawn from $G_2^{s_{i+1}}$, to check that

$$\begin{aligned}\alpha^{(i)}\tilde{V}_i(u^{(i)}) + \beta^{(i)}\tilde{V}_i(v^{(i)}) = \\ \sum_{x,y \in \{0,1\}^{s_{i+1}}} \left(\alpha^{(i)}\tilde{\mathrm{mult}}_{i+1}(u^{(i)},x,y) + \beta^{(i)}\tilde{\mathrm{mult}}_{i+1}(v^{(i)},x,y)\right)(\tilde{V}_{i+1}(x)\tilde{V}_{i+1}(y)) \\ + \left(\alpha^{(i)}\tilde{\mathrm{add}}_{i+1}(u^{(i)},x,y) + \beta^{(i)}\tilde{\mathrm{add}}_{i+1}(v^{(i)},x,y)\right)(\tilde{V}_{i+1}(x) + \tilde{V}_{i+1}(y))\end{aligned}$$

  - In the final round of the sumcheck, the prover sends $\tilde{V}_{i+1}(u^{(i+1)})$ and $\tilde{V}_{i+1}(v^{(i+1)})$ to the verifier, who then computes $\tilde{\mathrm{mult}}_{i+1}(u^{(i)}, u^{(i+1)}, v^{(i+1)})$, $\mathrm{mult}_{i+1}(v^{(i)}, u^{(i+1)}, v^{(i+1)})$, $\tilde{\mathrm{add}}_{i+1}(u^{(i)}, u^{(i+1)}, v^{(i+1)})$, and $\tilde{\mathrm{add}}_{i+1}(v^{(i)}, u^{(i+1)}, v^{(i+1)})$, verifying the last round of the sumcheck. In the next round, the sumcheck protocol is used to verify that $\tilde{V}_{i+1}(u^{(i+1)})$ and $\tilde{V}_{i+1}(v^{(i+1)})$ are correct.
- In the final layer l, the prover uses a polynomial commitment scheme to open $\tilde{V}_l(u^{(l)})$ and $\tilde{V}_l(v^{(l)})$.

Using the linear techniques from Libra [33], it is possible to perform a sumcheck protocol in linear time for expressions of the form

$$\sum_{x,y \in \{0,1\}^l} f_1(g,x,y) f_2(x) f_3(y)$$

provided that $f_1(g,x,y)$ satisfies a sparsity property, where it is non-zero in approximately $O(2^l)$ positions. Clearly, the functions $\tilde{\mathrm{add}}_i$ and $\tilde{\mathrm{mult}}_i$ fulfill this sparsity condition.

In Libra, the expression $\sum_{x,y\in\{0,1\}^l} f_1(g,x,y)f_2(x)f_3(y)$ is first rewritten in the following way:

$$\begin{aligned}\sum_{x,y\in\{0,1\}^l} f_1(g,x,y)f_2(x)f_3(y) &= \sum_{x\in\{0,1\}^l} f_2(x) \sum_{y\in\{0,1\}^l} f_1(g,x,y)f_3(y) \\ &= \sum_{x\in\{0,1\}^l} f_2(x)h_g(x),\end{aligned}$$

where $h_g(x) = \sum_{y\in\{0,1\}^l} f_1(g,x,y)f_3(y)$.

The sumcheck protocol is then conducted in two phases. In the first phase, the protocol runs for l rounds, reducing $\sum_{x\in\{0,1\}^l} f_2(x)h_g(x)$ to evaluating $f_2(u)h_g(u)$, where $u \in G_2^l$. In the second phase, the protocol verifies $\sum_{y\in\{0,1\}^l} f_1(g,u,y)f_3(y)$. In the first phase, both f_2 and h_g are multilinear polynomials in x. In the second phase, $f_2(u)$ is a constant, and $f_1(g,u,y)$ and $f_3(y)$ are multilinear polynomials in y.

For multilinear polynomials, it suffices to determine their values over $\{0,1\}^l$, allowing the sumcheck protocol to be completed in $O(n)$ time. According to Libra, the evaluation of $h_g(x)$ and $f_1(g,u,y)$ on the hypercube $\{0,1\}^l$ is known as the initialization process. Let A_f denote the array of values that the multilinear polynomial f takes on $\{0,1\}^l$. The operation Precompute(g) evaluates $G[z] = eq(g,z) = \prod_{i=1}^{l}((1-g_i)(1-z_i)+g_iz_i)$ for all $z \in \{0,1\}^l$.

Algorithm 3: PhaseOne Initialization

Input: Multilinear f_1 and f_3, initial bookkeeping tabels A_{f_3},random $g = g_1, \cdots, g_l$

Output: Bookkeeping table A_{h_g}

```
G ← Precompute(g);
Set G[0] = 1;
for i = 1 to l − 1 do
    for b ∈ {0,1}^i do
        G[b,0] = G[b] · (1 − g_{i+1});
        G[b,1] = G[b] · g_{i+1};
∀x ∈ {0,1}^l, set A_{h_g}[x] = 0;
for every (z,x,y) such that f_1(z,x,y) is no-zero do
    A_{h_g}[x] = A_{h_g}[x] + G[z] · f_1(z,x,y) · A_{f_3}[y];
return A_{h_g};
```

Algorithm 4: PhaseTwo Initialization

Input: Multilinear f_1 random $g = g_1, \cdots, g_l$ and $u = u_1, \cdots, u_l$
Output: Bookkeeping table A_{f_1}

$G \leftarrow \text{Precompute}(g)$;
$U \leftarrow \text{Precompute}(u)$;
$\forall y \in \{0,1\}^l$, set $A_{f_1}[y] = 0$;
for *every* (z, x, y) *such that* $f_1(z, x, y)$ *is non-zero* **do**
  $A_{f_1}[y] = A_{f_1}[y] + G[z] \cdot U[x] \cdot f_1(z, x, y)$;
return A_{f_1};

According to the Libra protocol, the sumcheck computation is divided into four phases: Initialization1, Sumcheck1, Initialization2, and Sumcheck2. The prover's computational costs are analyzed separately for each of these phases. Let the cost of a multiplication within the ring G_1 be denoted as bb, the cost of multiplying an element from G_1 with an element from G_2 as be, and the cost of a multiplication within G_2 as ee.

In practical application, the prover aims to prove that

$$f_1(g,x,y)f_2(x)f_3(y) = (\alpha^{(i)}\tilde{\text{mult}}_{i+1}(u^{(i)})+\beta^{(i)}\tilde{\text{mult}}_{i+1}(v^{(i)},x,y))(\hat{V}_{i+1}(x)\hat{V}_{i+1}(y)),$$

where $f_1(g,x,y) = \alpha^{(i)}\tilde{\text{mult}}_{i+1}(u^{(i)}) + \beta^{(i)}\tilde{\text{mult}}_{i+1}(v^{(i)},x,y)$, $f_2(x) = \hat{V}_{i+1}(x)$, and $f_3(y) = \hat{V}_{i+1}(y)$.

Initialization1 To initialize $h_g(x)$, while computing the array G, it requires n_{i+1} multiplications within G_2 (denoted as ee). Then, when calculating the array A_{h_g}, since there are at most n_{i+1} non-zero positions, A_{h_g} needs to be updated at most $2^{s_{i+1}}$ times. Each update involves first calculating $f_1(z,x,y) \cdot A_{f_3}[y]$, which requires one multiplication between elements from G_1 and G_2 (denoted as be), and then multiplying the result by $G[z]$, which requires one ee operation. Therefore, the total cost to initialize A_{h_g} is $2 \cdot n_{i+1}ee + n_{i+1}be$.

Sumcheck1 In the first phase of the sumcheck protocol, the process can be viewed as performing sumcheck on the product of two multilinear polynomials. One polynomial has coefficients from the ring G_1, while the other has coefficients from G_2, and the challenges are chosen from G_2. Let the univariate polynomial sent by the prover in the i-th round be denoted as s_i. According to the method described in [2], in the first round of sumcheck, the prover requires n_{i+1} multiplications between elements from G_1 and G_2 (denoted as be) to compute $s_1(0)$ and $s_1(1)$. Then, $s_1(2)$ can be computed using $\frac{n_{i+1}}{2}$ additional be operations. Moreover, $\frac{n_{i+1}}{2}$ ee operations are required to update the array A_{h_g}, and $\frac{n_{i+1}}{2}$ be operations are needed to update the array A_{f_2}.

During the j-th round, where $j \geq 2$, $\frac{n_{i+1}}{2^j}$ ee operations are needed to compute $s_i(0)$, and $s_i(1)$ can be derived by subtraction. Additionally, $\frac{n_{i+1}}{2^j}$ ee operations are required to compute $s_i(2)$. Furthermore, updating the arrays A_{f_2} and A_{h_g} each requires $\frac{n_{i+1}}{2^j}$ ee operations. Hence, the j-th round requires a total of $\frac{2n_{i+1}}{2^j}$ ee operations.

The total cost of performing the sumcheck protocol across all rounds is $\frac{5n_{i+1}}{2}ee + 2n_{i+1}be$.

Initialization2 To initialize the array for $f_1(g, u, y)$, since the array G has already been computed in the previous step, there is no need to recompute it. Only the array U needs to be calculated, which requires n_{i+1} multiplications within G_2 (denoted as ee). As $f_1(z, x, y)$ has only n_{i+1} non-zero positions, and each position requires 2 multiplications, the computation of $f_1(g, u, y)$ involves a total of $2n_{i+1}$ ee operations.

Sumcheck2 In the second phase of the sumcheck protocol, the process is similar to the first phase: performing sumcheck on the product of two multilinear polynomials. One polynomial has coefficients from the ring G_1, and the other from G_2, with challenges chosen from G_2. Therefore, the second round of sumcheck also requires $\frac{5n_{i+1}}{2}ee + 2n_{i+1}be$ operations.

The total cost of performing this sumcheck is $9n_{i+1}ee + 5n_{i+1}be$.

The prover also needs to demonstrate that:

$$f_1(g, x, y)f_2(x)f_3(y) = (\alpha^{(i)}\tilde{add}_{i+1}(u^{(i)}) + \beta^{(i)}\tilde{add}_{i+1}(v^{(i)}, x, y))\hat{V}_{i+1}(x)$$

The handling of this equation is similar, except here $f_3(y) = 1$, which is a constant polynomial.

Initialization1 When initializing the array A_{h_g}, there is no need to recompute the array G. Therefore, this process only requires n_{i+1} operations of ee and be.

Sumcheck1 During the first round of the sumcheck protocol, it similarly involves performing a sumcheck over the product of two multilinear polynomials. One polynomial has coefficients in the ring G_1, while the other has coefficients in G_2, with challenges chosen from G_2. The total computational cost for this step is $\frac{5n_{i+1}}{2}ee + 2n_{i+1}be$.

Initialization2 For the second round of computation, the initialization of the array A_{f_1} for $f_1(g, u, y)$ is required. Since the arrays G and U have already been computed previously, no recomputation is needed. The array A_{f_1} requires only n_{i+1} updates, with a computational cost of $2n_{i+1}$ operations of ee.

Sumcheck2 During the second round of the sumcheck protocol, since both f_2 and f_3 are constants, the sumcheck is only performed over $f_1(g, u, y)$. Each round requires updating only the array A_{f_1}, which has a total cost of n_{i+1} operations of ee.

Therefore, the total cost of performing this sumcheck is $\frac{13n_{i+1}}{2}ee + 3n_{i+1}be$.

Finally, the prover also needs to prove

$$f_1(g, x, y)f_2(x)f_3(y) = (\alpha^{(i)}\tilde{add}_{i+1}(u^{(i)}) + \beta^{(i)}\tilde{add}_{i+1}(v^{(i)}, x, y))(\hat{V}_{i+1}(y)),$$

where $f_2(x) = 1$ is a constant polynomial.

Initialization1 During the initialization of the array A_{h_g}, there is no need to recompute the G array. Additionally, while performing the previous sumcheck,

the non-zero values of $f_1(z,x,y)$ multiplied by $G[z]$ can be precomputed and stored. The only remaining step is to multiply these values by $A_{f_3}(y)$, which requires a total of n_{i+1} operations of be.

Sumcheck1 During the first round of the sumcheck protocol, since $f_2(x) = 1$ is a constant, the sumcheck is effectively over a single multilinear polynomial. Each round only requires updating the array A_{h_g}, which involves n_{i+1} operations of ee.

Initialization2 For the second round, it is necessary to initialize the array A_{f_1} for $f_1(g,u,y)$. Since the same random numbers are used as in the previous sumcheck, the array A_{f_1} from the previous sumcheck can be backed up and reused.

Sumcheck2 The second round of the sumcheck protocol involves the product of two multilinear polynomials. One polynomial has coefficients from the ring G_1, while the other has coefficients from G_2, and the challenges are selected from G_2. This sumcheck requires $\frac{5n_{i+1}}{2}$ operations of ee and $2n_{i+1}$ operations of be.

Therefore, the total computational cost for this sumcheck is $\frac{7}{2}n_{i+1}ee+3n_{i+1}be$.

To execute the i-th round of the Libra protocol, the computational cost is at least:

$$9n_{i+1}ee+5n_{i+1}be+\frac{13n_{i+1}}{2}ee+3n_{i+1}be+\frac{7}{2}n_{i+1}ee+3n_{i+1}be = 19n_{i+1}ee+11n_{i+1}be$$

Since $n_1+\cdots+n_l < n_0+n_1+\cdots+n_l = n$, the prover needs to perform at most $19n_{i+1}ee + 11n_{i+1}be$ computations to execute the sumcheck protocol, excluding the cost of the polynomial commitment.

Table 3: Prover Cost of Libra over Galois Ring without Polynomial Commitment

Phase	$\tilde{mult}(z,x,y)\tilde{V}(x)\tilde{V}(y)$	$\tilde{add}(z,x,y)\tilde{V}(x)$	$\tilde{add}(z,x,y)\tilde{V}(y)$	**all**
Initialization1	$2nee + nbe$	$nee + nbe$	nbe	$3nee + 3nbe$
Sumcheck1	$\frac{5n}{2}ee + 2nbe$	$\frac{5n}{2}ee + 2nbe$	nee	$6nee + 4nbe$
Initialization2	$2nee$	$2nee$		$4nee$
Sumcheck2	$\frac{5n}{2}ee + 2nbe$	nee	$\frac{5n}{2}ee + 2nbe$	$6nee + 4nbe$
all	$9nee + 5nbe$	$\frac{13n}{2}ee + 3nbe$	$\frac{7}{2}nee + 3nbe$	$19nee + 11nbe$

The polynomial commitment only needs to be made for the content of the last layer of circuit gates, with $n_i \leq n$, considering the polynomial's size as n. According to the analysis in Section 4, the cost to commit and open a polynomial of size n, where coefficients are in G_1 and challenges are in G_2, is $t_1+t_2+\frac{c_0 n}{k}ee+t2nkbe$. Here, t_1 is the time required for constructing a Merkle tree, t_2 is the time to open $|Q|$ positions of the Merkle tree, c_0 is the time for encoding, and t^{-1} is the code rate.

Taking parameters that maximize the encoding distance, $c_0 = 25.5$, $t = 1.72$, the cost for polynomial commitment is $t_1 + t_2 + \frac{25.5n}{k}ee + 3.44nkbe$.

The total prover time required for the entire protocol is:

$$t_1 + t_2 + \frac{25.5 + 19k}{k} nee + (3.44k + 11)nbe$$

F Prover Cost of HyperPlonk over Galois Rings

F.1 Gate Constraints.

$$0 = S_1(\mathbf{x}) \cdot (M(0,0,\mathbf{x}) + M(0,1,\mathbf{x})) + S_2(\mathbf{x}) \cdot M(0,0,\mathbf{x}) \cdot M(0,1,\mathbf{x}) - M(1,0,\mathbf{x}) + I(\mathbf{x})$$

To prove the sumcheck protocol, the prover can divide it into four parts: $S_1(\mathbf{x})M(0,0,\mathbf{x})$, $S_1(\mathbf{x})M(0,1,\mathbf{x})$, $S_2(\mathbf{x})M(0,0,\mathbf{x})M(0,1,\mathbf{x})$, and $M(1,0,\mathbf{x}) + I(\mathbf{x})$. Treating the left input, right input, and output as three polynomials, the sumcheck involves six polynomials in total. Let $2^v = n$. The coefficients of these polynomials are from the ring G_1, while the challenges are from the ring G_2. Each polynomial requires $\frac{n}{2}be + \frac{n}{2}ee$ for updating arrays during sumcheck.

In the first round, with d multilinear polynomials being multiplied in the sumcheck protocol, the univariate polynomial is of degree d, so $d+1$ points need to be sent. This requires $\frac{(d+1)(d-1)n}{2}$ bb operations. Substituting $d = 2, 2, 3, 1$ yields $\frac{3\cdot n}{2}, \frac{3\cdot n}{2}, \frac{8\cdot n}{2}, 0$ bb operations respectively. Additionally, updating arrays for the six multilinear polynomials requires $\frac{6\cdot n}{2}$ be operations.

In the j-th round, where $j \geq 2$, the sent univariate polynomial is of degree d, thus requiring $d + 1$ points. Each point calculation requires $\frac{(d-1)n}{2^j}$ ee operations, and computing all d points needs $\frac{d(d-1)2^l}{2^j}$ ee operations. Substituting $d = 2, 2, 3, 1$, this results in $\frac{n}{2^j}, \frac{n}{2^j}, \frac{3\cdot n}{2^j}, 0$ ee operations respectively. Furthermore, updating arrays for the six multilinear polynomials each round requires $\frac{6\cdot n}{2^j}$ ee operations. Therefore, the total cost for round j is $\frac{11\cdot n}{2^j}$ ee operations.

Overall, the sumcheck for gate constraints requires $11 \cdot nee + 3 \cdot nbe + 7 \cdot nbb$.

F.2 Permutation Constraints.

We can use a similar technique to Libra. Let $h(\mathbf{x}) = \sum_{\mathbf{y}\in\{0,1\}^l} (f(\mathbf{y})\text{eq}(\mathbf{x}, \tilde{\sigma}^{-1}(\mathbf{y})) - g(\mathbf{y})\text{eq}(\mathbf{x}, \mathbf{y}))$. In the first sumcheck phase, reduce $\sum_{\mathbf{x}\in\{0,1\}^l} \text{eq}(\mathbf{r}, \mathbf{x})h(\mathbf{x})$ to $\sum_{\mathbf{x}\in\{0,1\}^l} \text{eq}(\mathbf{r}, \mathbf{u})h(\mathbf{u})$, and then in the second phase, prove

$$\sum_{\mathbf{y}\in\{0,1\}^l} \text{eq}(\mathbf{r}, \mathbf{y})(f(\mathbf{y})\text{eq}(\mathbf{u}, \tilde{\sigma}^{-1}(\mathbf{y})) - g(\mathbf{y})\text{eq}(\mathbf{u}, \mathbf{y}))$$

Initialization1 Initialize the values of $h(\mathbf{x})$ over $\{0,1\}^l$. For each $\mathbf{x} \in \{0,1\}^l$, there is only one corresponding $\mathbf{y}$ such that $\text{eq}(\mathbf{x}, \mathbf{y}) = 1$, with all other values

being zero. Similarly, there is only one $\mathbf{y}$ for which $\mathrm{eq}(\tilde{\sigma}^{-1}(\mathbf{y}), \mathbf{x}) = 1$, with all other values being zero. While initializing the array A_h for the function h, since the values of eq are in $\{0, 1\}$, no multiplication is required; only 2^l additions are necessary. Initializing the array A_{eq} for $\mathrm{eq}(\mathbf{r}, \mathbf{x})$ requires 2^l multiplications in ee.

Sumcheck1 During the first stage of the sumcheck protocol, two multilinear polynomials are multiplied, where the coefficients of $\mathrm{eq}(\mathbf{g}, x)$ come from G_2, and the coefficients of h come from G_1. Therefore, the computational cost is $\frac{5 \cdot 2^l}{2}$ operations in ee and $2 \cdot 2^l$ operations in be.

Initialization2 In the second stage, we prove $\sum_{\mathbf{y} \in \{0,1\}^l} \mathrm{eq}(\mathbf{g}, \mathbf{u})(f(\mathbf{y})\mathrm{eq}(\mathbf{u}, \tilde{\sigma}^{-1}(\mathbf{y})) - g(\mathbf{y})\mathrm{eq}(\mathbf{u}, \mathbf{y}))$, where $\mathrm{eq}(\mathbf{g}, \mathbf{u})$ is a constant. Initializing $\mathrm{eq}(\mathbf{u}, \mathbf{g})$ requires 2^l operations in ee.

Sumcheck2 The expression $g(\mathbf{y}) \cdot \mathrm{eq}(\mathbf{u}, \mathbf{y})$ can be seen as the product of two multilinear polynomials, with one polynomial's coefficients in G_1 and the other in G_2. Since the challenge is in G_2, performing sumcheck requires $\frac{5 \cdot 2^l}{2} ee + 2 \cdot 2^l be$.

The expression $f(\mathbf{y}) \cdot \mathrm{eq}(\mathbf{u}, \tilde{\sigma}^{-1}(\mathbf{y}))$ can be seen as the product of $l+1$ multilinear polynomials during sumcheck. All the polynomials' coefficients come from G_1. In the first round, computing $s_1(0), \cdots, s_1(l+1)$ requires $\frac{(l+2)l \cdot 2^l}{2}$ operations in bb. Additionally, updating the arrays $A_f, A_{\tilde{\sigma}_1}, \cdots, A_{\tilde{\sigma}_l}$ requires $\frac{(l+1) \cdot 2^l}{2}$ operations in be.

In the j-th round ($j \geq 2$), the calculations involve computing $s_j(1), \cdots, s_j(l)$, where $s_j(0)$ is derived by subtracting $s_j(1)$ from the sum $s_j(0) + s_j(1)$. Calculating each $s_j(u), u \in [1, l+1]$ requires $\frac{l \cdot 2^l}{2^j}$ operations in ee, and calculating all $l+1$ values requires $\frac{l(l+1) \cdot 2^l}{2^j}$. Additionally, updating the arrays $A_f, A_{\tilde{\sigma}_1}, \cdots, A_{\tilde{\sigma}_l}$ requires $\frac{(l+1) \cdot 2^l}{2^j}$ operations in ee, thus the j-th round requires $\frac{(l+1)^2 \cdot 2^l}{2^j}$ operations in total. Therefore, performing sumcheck for the product of $l+1$ multilinear polynomials requires $\frac{(l+1)^2 \cdot 2^l}{2} ee + \frac{(l+1) \cdot 2^l}{2} be + \frac{l(l+1) \cdot 2^l}{2} bb$.

The total time required for the second stage is $\frac{((l+1)^2+7) \cdot 2^l}{2} ee + \frac{(l+5) \cdot 2^l}{2} be + \frac{l(l+1) \cdot 2^l}{2} bb$.

The overall time cost for handling permutation constraints is:

$$\frac{((l+1)^2 + 14) \cdot 2^l}{2} ee + \frac{(l+9) \cdot 2^l}{2} be + \frac{l(l+2) \cdot 2^l}{2} bb.$$

Now, consider the computational cost required for polynomial commitment. According to prior calculations, committing to a polynomial incurs the following cost:

$$t_1 + t_2 + \frac{25.5}{k} \cdot nee + 3.44k \cdot nbe.$$

In the constraints of HyperPlonk, the polynomials $S_1, S_2, \sigma_1, \cdots, \sigma_l$ need to be committed only once during the preprocessing phase. For each instance, the polynomials $M(0, 0, \mathbf{x}), M(0, 1, \mathbf{x}), M(1, 0, \mathbf{x}), I(\mathbf{x})$ require a commitment as

well. If the circuit to be proven has a size of n, in HyperPlonk, the permutation constraints apply to the polynomial M, where the size of the coefficients of M is $4n$, with $2^l = 4n$. Thus, committing to these four polynomials will require the following total cost:

$$4t_1 + 4t_2 + \frac{102}{k} \cdot nee + 13.76k \cdot nbe.$$

Table 4: HyperPlonk over Galois Ring Prover Cost

Phase	**Cost**
Gate Constraints	$11 \cdot nee + 3nbe + 7 \cdot nbb$
Permutation Constraints	$((\log n + 3)^2 + 14) \cdot 2nee + (\log n + 11) \cdot 2nbe + (\log n + 4)(\log n + 2)2nbb$
Polynomial Commitment	$4t_1 + 4t_2 + \frac{102 \cdot n}{k} ee + 13.76nkbe$

The total prover time cost of the protocol is:

$$\frac{39k + 2(\log n + 3)^2 k + 204}{k} nee + (13.76k + 2\log n + 25)nbe$$
$$+ (2(\log n + 4)(\log n + 2) + 7)nbb + 4t_1 + 4t_2$$

G Comparison with Rinocchio

The Rinocchio protocol [19] is based on a Linear PCP construction, adapted from the Pinocchio [28] and Groth16 [23] for ring settings. The Groth16 protocol relies on bilinear groups and the generic group model. However, such models and assumptions do not exist in the ring setting, so Rinocchio uses linear-only encoding to replace the bilinear groups and the generic group model in the field setting. In Rinocchio, the linear-only encoding is instantiated as (R)LWE ciphertexts. For an arithmetic circuit with wire size ℓ and m multiplication gates, according to Rinocchio's description in Section 6, Figure 3 of the paper [19], during the SRS generation phase, a trusted third party needs to encode $\ell + 2m + 2$ elements over the ring. In the proof generation phase, the prover needs to perform linear combinations on these $\ell + 2m + 2$ encodings. The prover must perform linear operations on the encoded elements for the computation of A, B, and C, requiring $m+1$, $m+1$, and about $\ell+m$ operations, respectively. In total, there are $\ell+3m+2$ linear operations (since computing C does not involve public inputs or outputs, the number of linear operations does not exceed $\ell+m$). In addition, when computing the polynomial $h(x)$, the prover needs to multiply two degree-m polynomials, which requires $O(m \log m)$ operations.

In Rinocchio, if the exceptional set of the ring is not large enough, an extension is required to ensure a sufficiently large exceptional set. For a Galois ring $\mathrm{GR}(p^s, r)$, the size of the exceptional set is p^r, so the required extension size is

the same as in this paper. However, during the SRS generation phase, Rinocchio encodes challenges as (R)LWE ciphertexts. RLWE can encrypt a polynomial, and whether the ring is original or extended, it will be encrypted as an RLWE ciphertext.

For a circuit with ℓ wires, t addition gates, m multiplication gates and public input and output x, we provide the following comparison.

Table 5: Comparison with Rinocchio

	Rinocchio	**This work (Libra)**	**This work (HyperPlonk)**
circuit type	NP circuit	log-space uniform circuit	NP circuit
prover time	$O(\ell + m \log m)$	$O(t + m)$	$O((t + m) \log^2(t + m)))$
proof size	$O(1)$	$O(\sqrt{t + m})$	$O(\sqrt{t + m})$
verifier time	$O(\|x\|)$	$O(\sqrt{t + m})$	$O(\sqrt{t + m})$
SRS length	$O(\ell + m)$	–	–

The Rinocchio and our system primarily provide proofs for two-input circuits. Beyond the initial input gates, each circuit gate corresponds to both a left and right input wire, hence ℓ is approximately twice $(t + m)$.