

Subring VOLE over Galois Rings with Applications to ZK over $\mathbb{Z}_{p^k}$

Ignacio Cascudo¹  and Xiang Liu^{1,2} 

¹ IMDEA Software Institute, Madrid, Spain
`name.surname@imdea.org`

² Universidad Politécnica de Madrid, Madrid, Spain

Abstract. Vector oblivious linear evaluation (VOLE) is a type of correlation that is widely used in multiparty computation (MPC) and zero-knowledge (ZK) proofs. Recently, the generation of VOLE correlation has become very efficient due to the pseudorandom correlation generator (PCGs) paradigm (Boyle et al. CCS 2018) and SoftSpokenOT (Roy Crypto 2022). This has driven a line of research on VOLE-based ZK, which enjoys linear prover time, low memory cost and post-quantum security. However, most existing works build VOLE and VOLE-based ZK over finite fields, whereas the constructions over integer rings are less satisfactory, especially in terms of communication and public verifiability.

In this work, we address some of these problems using a newly introduced primitive called subring VOLE (srVOLE), which is a generalization of subfield VOLE to Galois rings. Specifically,

- (1) We propose two maliciously secure srVOLE protocols. One is a PCG-like protocol that achieves extremely low amortized communication. The other is a SoftSpoken-like protocol, which is compatible with the VOLE-in-the-head (VOLEith) technique and thus can be used to construct publicly verifiable VOLE-based ZK.
- (2) We find that the VOLE correlation over $\mathbb{Z}_{2^k}$ used in Moz $\mathbb{Z}_{2^k}$ arella (Baum et al. Crypto 2022) is a special case of our srVOLE. Therefore, based on our construction, their designated-verifier ZK protocol can be made publicly verifiable.
- (3) We adapt the QuickSilver (Yang et al. CCS 2021) protocols to any Galois ring and compare with existing VOLE-based ZK protocols over rings. For circuit satisfiability, our protocol only communicates 1 subring element per multiplication gate, reducing the communication by more than half. For polynomial satisfiability, our protocol supports arbitrary low-degree relations, overcoming the restrictions of existing work on degree-2 relations.

Keywords: Vector oblivious linear evaluation · Zero-knowledge proof · Pseudorandom correlation generator · Public verifiability.

Table of Contents

Subring VOLE over Galois Rings with Applications to ZK over $\mathbb{Z}_{p^k}$	1
<i>Ignacio Cascudo[✉] and Xiang Liu[✉]</i>	
1 Introduction	3
1.1 Our Contributions	4
1.2 Technical Overview	5
2 Preliminaries	6
2.1 Notations	6
2.2 Galois Rings	7
2.3 Universal Hash Functions	7
2.4 Security Model and Ideal Functionalities	8
3 SoftSpoken-like srVOLE Protocols	9
3.1 Ideal Functionalities	9
3.2 Construction for Low-Degree Galois Rings	10
3.3 Construction for High-Degree Galois Rings	11
3.4 VOLE Protocols for Moz $\mathbb{Z}_{2^k}$ arella	13
4 PCG-like srVOLE Protocols	13
4.1 Ideal Functionalities	14
4.2 Single-Point srVOLE	14
4.3 srVOLE Extension	15
5 ZK Protocols over Galois Rings	17
5.1 VOLE Commitment	17
5.2 ZKP for Circuit Satisfiability over Galois Rings	18
5.3 ZKP for Polynomial Satisfiability over Galois Rings	20
A More Ideal Functionalities	26
B LPN over Galois Rings	26
B.1 LPN Noise Distribution	26
B.2 Reductions for LPN over $\text{GR}(p^k, d)$	26
B.3 LPN with Static Leakage	28
C GGM Algorithm	28
D Detailed VOLE Protocols for Moz $\mathbb{Z}_{2^k}$ arella	30
E Security Proofs for PCG-like Protocols	31
E.1 Proof of Theorem 5	31
E.2 Proof of Theorem 6	33
F Security Proofs for ZK protocols	33
F.1 Proof of Theorem 7	33
F.2 Proof of Theorem 8	34

1 Introduction

Vector oblivious linear evaluation (VOLE) is a basic primitive in multiparty computation (MPC). In a VOLE correlation over a finite ring $\mathcal{R}$, the sender holds two vectors $\mathbf{u}, \mathbf{v} \in \mathcal{R}^\ell$ while the receiver holds a vector $\mathbf{q} \in \mathcal{R}^\ell$ and a scalar $\Delta \in \mathcal{R}$, such that $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$. VOLE has found numerous applications, including 1-out-of- N oblivious transfer (OT) [BCG⁺19b, BCG⁺19a, Roy22], oblivious pseudorandom functions (OPRF) [BDFH25, YBH⁺25, AvBH⁺26], private set intersection (PSI) [RS21, RR22, vP25, BC25], and zero-knowledge (ZK) proofs [BCGI18, WYKW21, DIO21, BBMH⁺21, BBMHS22, DILO22].

The practicality of VOLE has been greatly improved by recent advances in pseudorandom correlation generators (PCGs) [BCGI18, BCG⁺19b]. Building on this line of work, there are a number of highly scalable VOLE protocols achieving extremely low amortized costs. On the other hand, the efficient generation of VOLE motivates the construction of efficient VOLE-based ZK proofs [WYKW21, DIO21, BMRS21, YSWW21, DILO22]. These protocols enjoy the advantages of linear prover time, low memory consumption and post-quantum security if the underlying VOLE protocol is post-quantum secure. Very recently, Baum et al. [BBD⁺23] proposed the VOLE-in-the-Head (VOLEitH) technique, which can make some kind of VOLE-based ZK publicly verifiable, and immediately leads to some post-quantum signatures based on this notion [BBB⁺26, ABB⁺26].

VOLE Protocols. The state-of-the-art VOLE protocols can be divided into two categories. One consists of PCG-like protocols, and was initiated by Boyle et al. [BCGI18, BCG⁺19b]. In their work, the ring $\mathcal{R}$ is actually a finite field $\mathbb{F}_{p^r}$. They start from generating sparse VOLE instances and then expand them to get random VOLE instances using the learning parity with noise (LPN) assumption. They also propose a variant called subfield VOLE (sVOLE), where the vector $\mathbf{u}$ is defined over the subfield $\mathbb{F}_p$. The subsequent works [BCG⁺19a, SGRR19, WYKW21, CRR21, BCG⁺22, RRT23, KPR⁺25, KPRR25, GLH⁺26] have continued to work on sVOLE and improve efficiency and security. For general commutative rings, [BCG⁺20] proposes pseudorandom correlation functions (PCFs) for VOLE, but they do not provide any concrete and efficient instantiation. [CD23, OSY21, BCM⁺25] continue to work on PCF-based VOLE, but these protocols are not as efficient as PCG-like protocols and they all work over finite fields.

The other category consists of SoftSpoken-like protocols. These protocols require black-box access to OT, but apart from this, they rely only on standard symmetric-key primitives. In the initial work SoftSpokenOT [Roy22], Roy defines a new variant of VOLE named subspace VOLE. In subspace VOLE, the correlation becomes $\mathbf{U} \cdot \mathbf{G}_c \cdot \text{diag}(\Delta) + \mathbf{V} = \mathbf{Q}$, where $\mathbf{U} \in \mathbb{F}_p^{\ell \times k_c}$, $\Delta \in \mathbb{F}_{p^r}^{n_c}$, $\mathbf{V}, \mathbf{Q} \in \mathbb{F}_{p^r}^{\ell \times n_c}$ and $\mathbf{G}_c \in \mathbb{F}_p^{k_c \times n_c}$ is a generator matrix of some error correcting code. Baum et al. [BBD⁺23] further generalize it to get subset VOLE, where $\Delta \in S_\Delta$ and $S_\Delta \subseteq \mathbb{F}_{p^r}^{n_c}$. Subset VOLE is useful especially when $\mathbb{F}_{p^r}$ is exponentially large.

VOLE-based ZK. The random VOLE correlation $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$, where $\mathbf{u}, \mathbf{v}, \mathbf{q}, \Delta$ are all random, can be interpreted as an information-theoretic (linearly) homomorphic message authentication code (MAC) for the random message $\mathbf{u}$. In this case, we write the correlation as $\mathbf{u} \cdot \Delta + M[\mathbf{u}] = K[\mathbf{u}]$. It can be converted to a commitment to an arbitrary message $\mathbf{x}$ by sending the derandomization $\mathbf{x} - \mathbf{u}$. This interpretation leads to ZK proofs for *circuit satisfiability*, where the prover and verifier hold some circuit $\mathcal{C}$, and the prover proves that it knows the witness $\mathbf{w}$ such that $\mathcal{C}(\mathbf{w}) = 1$. A key advantage is the linear homomorphism of this VOLE-MAC, which makes the computation of MACs of addition-gate outputs essentially cost-free in terms of communication, so the complexity mainly lies in proving correctness of computation of multiplication gates.

Boyle et al. [BCGI18] initiated the study of VOLE-based ZK. The following works have developed various efficient multiplication check techniques [WYKW21, DIO21, BMRS21, YSWW21, DILO22, LHLA25]. All of them have linear prover communication. We refer the reader to [BDSW23] for a survey.

Beyond linear proofs for circuit satisfiability, several works focus on other aspects. [WYY⁺22, LXY24a, LXY25] build ZK proofs for circuit satisfiability with sublinear communication. They are all designated-verifier, and some require more assumptions or incur high prover complexity and memory cost. [WYY⁺22, BCC⁺25] design protocols for single instruction multiple data (SIMD) circuits. [YHH⁺23, GHAK24, YHH⁺24, HHK⁺24, Yan25] study VOLE-based ZK for disjunctions. Another line of work focuses on VOLE-based ZK for mixed arithmetic [BBMH⁺21, WYX⁺21, ABBS25].

The main limitation of VOLE-based ZK is its designated-verifier nature because the underlying VOLE protocol is private-coin. To overcome this issue, Baum et al. [BBD⁺23] introduced the

VOLEitH technique, which can compile certain OT-based designated-verifier ZK protocols³ to publicly verifiable ones. VOLEitH was further improved in [BBM⁺24, BBB⁺25]. Moreover, combined with Fiat-Shamir transform and ideas from MPC-in-the-Head (MPCitH) signatures [KKW18], VOLE-based ZK and VOLEitH have led to several post-quantum signatures [BBB⁺26, ABB⁺26], which are both third round candidates in the NIST call for additional post-quantum digital signatures.

Very recently, [NRS26] builds a ZK protocol for circuit satisfiability with polylogarithmic communication from chosen-input VOLE (CI-VOLE) [ARS24, AMR25, NRS26]. In contrast to random VOLE, CI-VOLE allows the sender to choose the vector $\mathbf{u}$. This enables the sender to commit to the witness directly and eliminates the derandomization step, thereby saving $|\mathbf{w}|$ communication. However, the generation of CI-VOLE is much more computationally expensive. Moreover, these CI-VOLE protocols are not compatible with VOLEitH, leaving the public verifiability unclear.

ZK over Integer Rings. While most existing protocols work over finite fields, it is also valuable to study ZK over integer rings such as $\mathbb{Z}_{2^k}$. For example, when we are proving that a program is executed correctly, it is more natural to transform it into a circuit over $\mathbb{Z}_{2^k}$ because CPUs perform arithmetic over $\mathbb{Z}_{2^k}$.

However, to the best of our knowledge, there are only a few works that build ZK over $\mathbb{Z}_{2^k}$. Based on SPD $\mathbb{Z}_{2^k}$ -like VOLE [CDE⁺18], [BBMH⁺21] adapts the multiplication checks in Wolverine [WYKW21] and Mac’n’Cheese [BMRS21] to work over $\mathbb{Z}_{2^k}$. While their constructions require a maliciously secure VOLE, they do not provide an efficient VOLE protocol. Moz $\mathbb{Z}_{2^k}$ arella [BBMHS22] addresses this issue by proposing a PCG protocol for SPD $\mathbb{Z}_{2^k}$ -like VOLE. Moreover, they adapt QuickSilver [YSWW21] to the $\mathbb{Z}_{2^k}$ setting, obtaining a new protocol called QuarkSilver. They prove the security of QuarkSilver by bounding the number of solutions of the quadratic equation in a special form over $\mathbb{Z}_{2^k}$. Nevertheless, QuarkSilver only supports degree-2 relations because the number of roots grows exponentially with increasing degree, which undermines its soundness. Besides, QuarkSilver is not publicly verifiable because the underlying PCG-like VOLE is not compatible with VOLEitH. Recently, Lin et al. [LXY24b] build ZK protocols over $\mathbb{Z}_{2^k}$ via Galois rings. Their idea is to pack a batch of elements over $\mathbb{Z}_{2^k}$ into a single element over a Galois ring using reverse multiplication friendly embeddings (RMFEs, [CCXY18, CRX21]). Then, they can prove the statements in batches. Although RMFEs can amortize the computation and communication cost, their protocol suffers from two points: (1) they need to prove that the RMFE packing is performed honestly; (2) they need to re-embed the elements after each multiplication since RMFEs can only preserve one multiplication. Both bring an extra cost. [LXY24b] also proposes a SoftSpoken-like VOLE protocol to achieve public verifiability. However, their construction does not adapt subspace VOLE to Galois rings. Instead, they simply repeat the ZK protocols over low-degree Galois rings to amplify the soundness. As a result, their protocol is not as efficient as that over finite fields.

There are also some other ZK protocols over $\mathbb{Z}_{2^k}$ not based on VOLE. Braun et al. [BDJ⁺23] propose an MPCitH protocol over $\mathbb{Z}_{2^k}$ using linear secret sharing over Galois rings. [GNS23, JLX⁺25] build zero-knowledge succinct non-interactive arguments of knowledge (zkSNARK) over $\mathbb{Z}_{2^k}$ via Galois rings. All these works incur high prover computation and memory cost.

1.1 Our Contributions

In this work, we focus on (random) VOLE protocols and VOLE-based ZK over Galois rings and obtain the following results.

1. We introduce a new VOLE variant named subring VOLE (srVOLE), which is a generalization of sVOLE to Galois rings. srVOLE is a more flexible correlation as it implies both subfield VOLE over finite fields and VOLE over integer rings.
2. We provide two maliciously secure instantiations for srVOLE. One is a PCG-like protocol in Sect. 4, which achieves extremely low amortized communication when generating a large amount of VOLE instances. The other is a SoftSpoken-like protocol in Sect. 3, which is compatible with VOLEitH and thus can be used to construct publicly verifiable ZK proofs. We also generalize the subset VOLE protocol over finite fields in [BBD⁺23] to Galois rings, which is useful when the Galois ring has a large characteristic. Our SoftSpoken-like srVOLE works for both low-degree and high-degree Galois rings by adapting the subspace VOLE with repetition codes to Galois rings.

³ The commonly used OT protocol for publicly verifiable VOLE-based ZK is SoftSpokenOT.

3. We note that the VOLE correlation over $\mathbb{Z}_{2^k}$ used in Moz $\mathbb{Z}_{2^k}$ arella [BBMHS22] is a special case of our subset VOLE over Galois rings. Therefore, we explicitly provide a SoftSpoken-like VOLE protocol for Moz $\mathbb{Z}_{2^k}$ arella. Based on our construction, the designated-verifier ZK protocol in Moz $\mathbb{Z}_{2^k}$ arella can be made publicly verifiable. See Sect. 3.4 for details.
4. In Sect. 5, we generalize QuickSilver [YSWW21] protocols for circuit satisfiability and polynomial satisfiability to Galois rings using srVOLE. Our ZK protocol for circuits achieves optimal amortized communication – 1 subring element per multiplication gate, and better soundness. Our ZK protocol for polynomials supports any low-degree relations, which addresses the limitation of existing work. Table 1 gives a concrete comparison of the communication cost between different protocols for circuit satisfiability.

Table 1. Concrete comparison of online communication cost per multiplication gate (in bits) for ZK proofs of circuit satisfiability, where the witness is a vector over $\mathbb{Z}_{2^k}$. λ is the security parameter. $\mathcal{P}$ denotes the prover cost and $\mathcal{V}$ denotes the verifier cost. $\mathcal{R}$ is the ring over which the computations are carried out. As in [LXY24b], we use (16, 45)-RMFEs for $\lambda = 40$ and (27, 85)-RMFEs for $\lambda = 80$. We note that the messages from $\mathcal{V}$ to $\mathcal{P}$ are uniformly random challenges, which can be compressed by using a PRG. Furthermore, our protocol can be made publicly verifiable, so the verifier communication totally disappears when we derive the challenges from random oracle using Fiat-Shamir transform.

k	λ	[BBMHS22]			[LXY24b]			$\Pi_{\text{circuitZK}}^{n,t,d}$		
		$\mathcal{P}$	$\mathcal{V}$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{R}$
32	40	130	49	$\mathbb{Z}_{2^{130}}$	90	3	$\text{GR}(2^{32}, 45)$	32	42	$\text{GR}(2^{32}, 42)$
	80	212	90	$\mathbb{Z}_{2^{212}}$	101	3	$\text{GR}(2^{32}, 85)$	32	82	$\text{GR}(2^{32}, 82)$
64	40	162	49	$\mathbb{Z}_{2^{162}}$	180	3	$\text{GR}(2^{64}, 45)$	64	42	$\text{GR}(2^{64}, 42)$
	80	244	90	$\mathbb{Z}_{2^{244}}$	202	3	$\text{GR}(2^{64}, 85)$	64	82	$\text{GR}(2^{64}, 82)$

1.2 Technical Overview

srVOLE is a Galois ring analogue of sVOLE. In srVOLE $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$, $\mathbf{u} \in \text{GR}(p^k, d_0)^\ell$, $\mathbf{v}, \mathbf{q} \in \text{GR}(p^k, d_1)^\ell$, $\Delta \in \text{GR}(p^k, d_1)$ and $\text{GR}(p^k, d_0)$ is a subring of $\text{GR}(p^k, d_1)$. In this section, we briefly introduce the techniques we use in our srVOLE constructions and VOLE-based ZK protocols over Galois rings.

SoftSpoken-like srVOLE. SoftSpokenOT starts from a $(N - 1)$ -out-of- N random OT, where $N = p^r$. The sender gets random vectors $\mathbf{t}_i \in \mathbb{F}_p^\ell, i \in [N]$ and the receiver obtains $\Delta \in \mathbb{F}_{p^r}$ and all $\mathbf{t}_i$ except for $\mathbf{t}_\Delta$ (we implicitly identify $[N]$ and $\mathbb{F}_{p^r}$ via some bijection, since $N = p^r$).

Then, the sender parts of the sVOLE correlation are constructed as follows

$$\mathbf{u} := \sum_{i \in \mathbb{F}_{p^r}} \mathbf{t}_i, \mathbf{v} := - \sum_{i \in \mathbb{F}_{p^r}} \mathbf{t}_i \cdot i$$

The uniform distribution of $\mathbf{u}, \mathbf{v}$ comes from the fact that $\{1\} \times \mathbb{F}_{p^r}$ contains a basis of the $\mathbb{F}_p$ -linear space $\mathbb{F}_p \times \mathbb{F}_{p^r}$. We examine this structure and find that this construction implicitly defines a group homomorphism $\varphi : (\mathbb{F}_p)^{p^r} \rightarrow \mathbb{F}_p \times \mathbb{F}_{p^r}$. Since all $\mathbf{t}_i$ are uniformly random, it suffices to prove that φ is surjective. The only problem is that, when we adapt it to the Galois ring $\text{GR}(p^k, d_1)$, the computation cost is exponentially large in k . We borrow the idea from [LXY24b] to restrict $\Delta \in \mathbb{F}_{p^{d_1}}$, which does not affect correctness and security. Indeed, sampling Δ from the larger ring $\text{GR}(p^k, d_1)$ does not improve the soundness. This follows from the observation that although the size of the challenge space increases by a factor of $p^{(k-1)d_1}$, the number of solutions increases by the same factor. We notice that this restriction is actually a special case of subset VOLE, where $\Delta \in S_\Delta \subseteq \text{GR}(p^k, d_1)$. Therefore, we further generalize it to get subset VOLE over Galois rings using the above idea, which works as long as the surjectivity of φ holds.

We also investigate how to generalize subspace VOLE to Galois rings. An obstacle here is that general ring-linear codes (even free codes) do not have as good properties as codes over finite fields.

However, the repetition code over Galois rings works perfectly, and this is good enough in many applications due to its large minimum distance.

PCG-like srVOLE. Our PCG-like srVOLE protocol generalizes the sVOLE protocol in Wolverine [WYKW21]. This generalization is natural due to the similarity between Galois rings and Galois fields. Specifically, we first construct single-point subring VOLE (sp-srVOLE), where $\mathbf{u}$ has a single non-zero coordinate. Then we stack t sp-srVOLE instances to get a srVOLE correlation $\mathbf{e} \cdot \Delta + \mathbf{b} = \mathbf{c}$, where $\mathbf{e} \in \text{GR}(p^k, d_0)^{n_c}$ is a regular noise with invertible non-zero entries and $\mathbf{b}, \mathbf{c} \in \text{GR}(p^k, d_1)^{n_c}$ are random vectors. Then, we compute $\mathbf{x} := \mathbf{u} \cdot \mathbf{A} + \mathbf{e}$, $\mathbf{y} := \mathbf{v} \cdot \mathbf{A} + \mathbf{b}$, $\mathbf{z} := \mathbf{q} \cdot \mathbf{A} + \mathbf{c}$, where $\mathbf{A} \in \text{GR}(p^k, d_0)^{k_c \times n_c}$ is a generator matrix in LPN and $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$ is a length- k_c random srVOLE instance. By LPN assumption over Galois rings, $\mathbf{x} \cdot \Delta + \mathbf{y} = \mathbf{z}$ is a random srVOLE instance with length n_c .

Moreover, following the reductions between LPN over integer rings and LPN over finite fields in [LWYY24], we provide the reductions between LPN over $\text{GR}(p^k, d)$ and LPN over $\mathbb{F}_{p^d}$ to estimate the security of the underlying LPN assumption.

ZK Protocols from srVOLE. VOLE-based ZK follows the commit-and-prove paradigm. When working over a field, the VOLE-MAC is a statistically binding commitment scheme. If a malicious prover wants to open a different message u' , we have $(u - u') \cdot \Delta = M[u'] - M[u]$. This means that the prover successfully guesses the value of Δ , which happens with probability at most $1/|\mathbb{F}|$. However, this is not the case for $\mathbb{Z}_{p^k}$. As pointed out by [CDE⁺18, BBMHS22], the adversary can choose u' such that $u - u' = p^{k-1}$. Then it only needs to guess $\Delta \bmod p$ with probability $1/p$.

The above analysis leads to two observations. First, instead of sampling Δ from the whole ring $\mathbb{Z}_{p^k}$, it is sufficient to have $\Delta \in \mathbb{Z}_p$ without sacrificing security, which has been done in [LXY24b]. Second, if we are working over $\text{GR}(p^k, d_1)$, intuitively, the adversary needs to successfully guess all d_1 (independent) coordinates of Δ to break the binding property, which happens with probability at most $1/p^{d_1}$.

Based on VOLE-MAC over Galois rings, we follow QuickSilver [YSWW21] to construct ZK proofs for circuit satisfiability and polynomial satisfiability. For the i -th multiplication gate $w_\alpha^i \cdot w_\beta^i = w_\gamma^i$, observe that

$$\begin{aligned} B_i &= K[w_\alpha^i] \cdot K[w_\beta^i] - K[w_\gamma^i] \cdot \Delta \\ &= (w_\alpha^i w_\beta^i - w_\gamma^i) \cdot \Delta^2 + (w_\alpha^i \cdot M[w_\beta^i] + w_\beta^i \cdot M[w_\alpha^i] - M[w_\gamma^i]) \cdot \Delta \\ &\quad + M[w_\alpha^i] \cdot M[w_\beta^i] \\ &= \underbrace{(w_\alpha^i \cdot M[w_\beta^i] + w_\beta^i \cdot M[w_\alpha^i] - M[w_\gamma^i]) \cdot \Delta}_{A_{1,i}} + \underbrace{M[w_\alpha^i] \cdot M[w_\beta^i]}_{A_{0,i}} \end{aligned} \tag{1}$$

where the verifier can locally compute B_i and the prover can locally compute $A_{0,i}, A_{1,i}$. It remains to check $B_i = A_{1,i} \cdot \Delta + A_{0,i}$, which is an OLE correlation. To check multiple multiplications at once, we use a linear combination to aggregate a batch of them into a single equation.

Furthermore, unlike RMFEs that can only preserve one multiplication, the subring isomorphism has infinite multiplication capacity. Therefore, we generalize the above idea as in QuickSilver to prove any low-degree polynomials over Galois rings.

2 Preliminaries

2.1 Notations

We denote the set $\{1, 2, \dots, n\}$ by $[n]$. We use bold lower-case letters to denote column vectors and bold capital letters to denote matrices. The i -th coordinate of a vector $\mathbf{a}$ is denoted by a_i . The i -th row of a matrix $\mathbf{H}$ is denoted by $\mathbf{H}_i$ and the i -th column is $\mathbf{H}^i$. For a set G , $\mathbf{a}_G$ is the subvector indexed by G and $\mathbf{H}_G$ is the submatrix whose columns are indexed by G . Moreover, we write $\mathbf{H}_{-G}$ to denote the submatrix that contains all columns except for those indexed by G . We use $\mathbf{1}_n$ to denote an all-one row vector of length n . Given a vector $\mathbf{x}$, we denote by $\text{diag}(\mathbf{x})$ the diagonal matrix that has $\mathbf{x}$ on the diagonal. For a set S , we use $x \leftarrow S$ to denote sampling a uniformly random element x from S . We denote the prover by $\mathcal{P}$ and the verifier by $\mathcal{V}$. We use the abbreviation PPT to denote probabilistic polynomial time. We focus only on finite commutative rings with identity.

2.2 Galois Rings

A Galois ring $\text{GR}(p^k, d)$ is defined as $\mathbb{Z}_{p^k}[x]/(f(x))$, where p is a prime and $f(x)$ is a monic basic irreducible polynomial of degree d in $\mathbb{Z}_{p^k}[x]$. We define a ring homomorphism,

$$\begin{aligned} - : \mathbb{Z}_{p^k} &\rightarrow \mathbb{F}_p \\ a &\mapsto \bar{a} := a \bmod p \end{aligned}$$

and extend it to Galois rings,

$$\begin{aligned} - : \text{GR}(p^k, d) &\rightarrow \mathbb{F}_{p^d} \\ a_0 + a_1x + \dots + a_{d-1}x^{d-1} &\mapsto \bar{a}_0 + \bar{a}_1x + \dots + \bar{a}_{d-1}x^{d-1} \end{aligned}$$

It is easy to see $\text{Ker}(-) = (p)$. Most of the following are taken from [Wan03].

Teichmüller set. For any $\text{GR}(p^k, d)$, there exists an element $\xi \in \text{GR}(p^k, d)$ such that $\text{ord}(\xi) = p^d - 1$. The set $\mathcal{T}_{k,d} := \{0, 1, \xi, \xi^2, \dots, \xi^{p^d-2}\}$ is called the Teichmüller set of $\text{GR}(p^k, d)$. When the context is clear, we simply write $\mathcal{T}$ for $\mathcal{T}_{k,d}$. $\mathcal{T}$ is isomorphic to $\mathbb{F}_{p^d}$. We denote the Teichmüller lift function by $\iota : \mathbb{F}_{p^d} \rightarrow \mathcal{T}$, which is bijective and multiplicative. We write $\iota(\mathbf{x})$ to denote the vector obtained by applying ι to each component of $\mathbf{x}$. For brevity, we omit $\iota(\cdot)$ and simply write $a \cdot b$ to denote $a \cdot \iota(b)$ whenever $a \in \text{GR}(p^k, d), b \in \mathbb{F}_{p^d}$.

Subring. $\text{GR}(p^k, d')$ is a subring of $\text{GR}(p^k, d)$ if and only if $d' \mid d$. In fact, $\text{GR}(p^k, d)$ is a free $\text{GR}(p^k, d')$ -module with basis $\{1, \xi, \dots, \xi^{m-1}\} \subset \mathcal{T}_{k,d}$, where $m = \frac{d}{d'}$.

p-adic representation. Every element $\alpha \in \text{GR}(p^k, d)$ can be written uniquely as $\alpha = a_0 + a_1p + \dots + a_{k-1}p^{k-1}$, where $a_0, a_1, \dots, a_{k-1} \in \mathcal{T}$. This is called the p -adic representation. We define a decomposition function $\text{Decomp} : \text{GR}(p^k, d) \rightarrow \mathcal{T}^k$ such that it decomposes any element over $\text{GR}(p^k, d)$ to its p -adic representation, i.e., $\text{Decomp}(\alpha) = (a_0, a_1, \dots, a_{k-1})$. α is a unit if and only if $a_0 \neq 0$. We denote by $\text{GR}(p^k, d)^*$ the set of units in $\text{GR}(p^k, d)$.

Similar to the Schwartz-Zippel lemma in finite fields, we have the same result in Galois rings.

Lemma 1 ([EXY22]). *A non-zero degree- D polynomial over $\text{GR}(p^k, d)$ has at most $Dp^{(k-1)d}$ roots.*

Corollary 1. *Let $f(x)$ be a non-zero polynomial of degree D over $\text{GR}(p^k, d)$, and $r \leftarrow \text{GR}(p^k, d)$ be a uniform element, then $\Pr[f(r) = 0] \leq Dp^{-d}$.*

Lemma 2. *Let $f(x_1, \dots, x_n)$ be a non-zero polynomial of total degree D over $\text{GR}(p^k, d)$. Let $r_1, \dots, r_n$ be sampled uniformly and independently from $\text{GR}(p^k, d)$. Then $\Pr[f(r_1, \dots, r_n) = 0] \leq Dp^{-d}$.*

Remark 1. Lemma 2 also holds when $r_1, \dots, r_n \leftarrow \mathbb{F}_{p^d}$. Let $f(x_1, \dots, x_n) = p^i \cdot f'(x_1, \dots, x_n)$ for some $0 \leq i < k$ and $p \nmid f'$. Then we have $\bar{f}' := f' \bmod p$ is a non-zero polynomial of total degree at most D over $\mathbb{F}_{p^d}$. If $f(r_1, \dots, r_n) = 0$, $f'(r_1, \dots, r_n) \in (p)$, which means $\bar{f}'(r_1, \dots, r_n) = 0$. However, this happens with probability at most Dp^{-d} by the Schwartz-Zippel lemma over fields.

2.3 Universal Hash Functions

We borrow the definitions of linear universal hash functions from [BBD⁺23] and adapt them to work over rings.

Definition 1. *Let $\mathcal{R}$ be a ring. A linear ε -almost universal family of hashes is a family of matrices $\mathcal{H} \subseteq \mathcal{R}^{r \times \ell}$ such that for any nonzero $\mathbf{x} \in \mathcal{R}^\ell$,*

$$\Pr_{\mathbf{H} \leftarrow \mathcal{H}}[\mathbf{H}\mathbf{x} = 0] \leq \varepsilon$$

Definition 2. *Let $\mathcal{R}$ be a ring. A linear ε -almost uniform family of hashes is a family of matrices $\mathcal{H} \subseteq \mathcal{R}^{r \times \ell}$ such that for any nonzero $\mathbf{x} \in \mathcal{R}^\ell$ and all $\mathbf{y} \in \mathcal{R}^r$,*

$$\Pr_{\mathbf{H} \leftarrow \mathcal{H}}[\mathbf{H}\mathbf{x} = \mathbf{y}] \leq \varepsilon$$

Definition 3. Let $\mathcal{R}$ be a ring and $\mathcal{R}_0$ be its subring. A matrix $\mathbf{H} \in \mathcal{R}^{r \times (\ell+h)}$ is $\mathcal{R}_0^\ell$ -hiding if the distribution of $\mathbf{H}\mathbf{x}$ is independent of $\mathbf{x}_{[\ell]}$ when $\mathbf{x}_{[\ell+1, \ell+h]} \leftarrow \mathcal{R}_0^h$. A hash family $\mathcal{H} \subseteq \mathcal{R}^{r \times (\ell+h)}$ is $\mathcal{R}_0^\ell$ -hiding if every $\mathbf{H} \in \mathcal{H}$ is $\mathcal{R}_0^\ell$ -hiding.

To provide efficient constructions of universal hash functions over Galois rings, we propose the following theorem.

Theorem 1. If $\mathcal{H} \subseteq \mathbb{F}_{p^d}^{r \times \ell}$ is a linear ε -almost universal(uniform) family of hashes in $\mathbb{F}_{p^d}$, for any $k \geq 1$, $\mathcal{H}' := \{\mathbf{H}' \in \text{GR}(p^k, d)^{r \times \ell} : \overline{\mathbf{H}'} \in \mathcal{H}\}$ (where $\overline{\mathbf{H}'}$ is obtained by applying $-$ entry-wise to $\mathbf{H}'$) is a linear ε -almost universal(uniform) family of hashes in $\text{GR}(p^k, d)$.

Proof. We only prove the linear ε -almost uniform case, and the other case can be proven in a similar way. Note that for any $\mathbf{H} \in \mathcal{H}$, $|\{\mathbf{H}' \in \text{GR}(p^k, d)^{r \times \ell} : \overline{\mathbf{H}'} = \mathbf{H}\}| = |(p)^{(r \times \ell)}|$. So for any $\mathbf{H} \in \mathcal{H}$, $\Pr_{\mathbf{H}' \leftarrow \mathcal{H}'}[\overline{\mathbf{H}'} = \mathbf{H}] = \frac{1}{|\mathcal{H}|}$.

Assume there exists $0 \neq \mathbf{x} \in \text{GR}(p^k, d)^\ell$ and $\mathbf{y} \in \text{GR}(p^k, d)^r$ such that $\Pr_{\mathbf{H}' \leftarrow \mathcal{H}'}[\mathbf{H}'\mathbf{x} = \mathbf{y}] > \varepsilon$. Let $\mathbf{x} = p^t \mathbf{a}$ for some $0 \leq t < k$ and there exists some $i \in [\ell]$ such that $p \nmid a_i$. Then we have $\bar{\mathbf{a}} \neq 0$ and $\mathbf{y} \in p^t \text{GR}(p^k, d)^r$. Let $\mathbf{y} = p^t \mathbf{b}$. Since $p^t(\mathbf{H}'\mathbf{a} - \mathbf{b}) = \mathbf{H}'\mathbf{x} - \mathbf{y} = 0$, we have $\mathbf{H}'\mathbf{a} - \mathbf{b} \in p^{k-t} \text{GR}(p^k, d)^r$, which means $\overline{\mathbf{H}'}\bar{\mathbf{a}} = \bar{\mathbf{b}}$.

Therefore, $\Pr_{\mathbf{H} \leftarrow \mathcal{H}}[\mathbf{H}\bar{\mathbf{a}} = \bar{\mathbf{b}}] \geq \Pr_{\mathbf{H}' \leftarrow \mathcal{H}'}[\mathbf{H}'\mathbf{x} = \mathbf{y}] > \varepsilon$, which leads to a contradiction. $\square$

Remark 2. We note that $\mathcal{H}'$ is still an ε -almost universal(uniform) hash family when we only choose one representative element from the preimage of every $\mathbf{H} \in \mathcal{H}$. For efficiency, we can choose $\mathbf{H}'$ such that for each entry $h_{i,j} = a_0 + a_1x + \dots + a_{d-1}x^{d-1} \in \text{GR}(p^k, d)$ of $\mathbf{H}'$, for all $0 \leq j < d$, $a_j \in [0, p-1]$.

The following theorem is adapted from FAEST [BBB⁺26].

Theorem 2. Let $\mathcal{H} \subseteq \text{GR}(p^k, d)^{r \times \ell}$ be a linear ε -almost uniform hash family. Let $\mathcal{H}' \subseteq \text{GR}(p^k, d)^{r \times (\ell+r)}$ be the family $\{[\mathbf{H} \ \mathbf{I}_r] : \mathbf{H} \in \mathcal{H}\}$, where $\mathbf{I}_r$ is the $r \times r$ identity matrix. Then, it holds that (1) $\mathcal{H}'$ is ε -almost universal, and (2) $\mathcal{H}'$ is $\text{GR}(p^k, d)^\ell$ -hiding.

Proof. Let $\mathbf{x} = [\mathbf{x}_0 \ \mathbf{x}_1]^\top$ be nonzero, where $\mathbf{x}_0 \in \text{GR}(p^k, d)^\ell$ and $\mathbf{x}_1 \in \text{GR}(p^k, d)^r$. If $\mathbf{H}' \leftarrow \mathcal{H}'$, $\mathbf{H}'\mathbf{x} = 0$ implies $\mathbf{H}\mathbf{x}_0 = -\mathbf{x}_1$. We have $\mathbf{x}_0 \neq 0$ otherwise $\mathbf{x} = 0$. So $\mathbf{H}'\mathbf{x} = 0$ holds with probability at most ε . For (2), the hiding property holds because $\mathbf{I}_r$ ensures that if $\mathbf{x}_1$ are uniform then it perfectly masks the rest. $\square$

Combining Theorem 1 and Theorem 2, we can easily construct linear ε -almost universal and $\text{GR}(p^k, d)^\ell$ -hiding hash family in $\text{GR}(p^k, d)$ from a linear ε -almost uniform hash family in $\mathbb{F}_{p^d}$.

2.4 Security Model and Ideal Functionalities

In this work, we analyze the security of our protocols in the simplified universally composable security (SUC) framework [CCL15] in the presence of a static, malicious adversary. We give the ideal functionalities of ZK proofs in Figures 1 and 2. In our PCG-like srVOLE protocols, we need to run OT and an equality test. We give their functionalities in Appendix A.

Functionality $\mathcal{F}_{\text{circuitZK}}$

Upon receiving (prove, $\mathcal{C}, \mathbf{w}$) from $\mathcal{P}$ and (verify, $\mathcal{C}$) from $\mathcal{V}$, send (true) to $\mathcal{V}$ if $\mathcal{C}(\mathbf{w}) = 1$ and (false) otherwise.

Fig. 1. Ideal Functionality of $\mathcal{F}_{\text{circuitZK}}$

Functionality $\mathcal{F}_{\text{polyZK}}$

Upon receiving (prove, $\{f_i\}_{i \in [t]}, \mathbf{w}$) from $\mathcal{P}$ and (verify, $\{f_i\}_{i \in [t]}$) from $\mathcal{V}$, send (true) to $\mathcal{V}$ if $f_i(\mathbf{w}) = 0$ for all $i \in [t]$ and (false) otherwise.

Fig. 2. Ideal Functionality of $\mathcal{F}_{\text{polyZK}}$

3 SoftSpoken-like srVOLE Protocols

SoftSpokenOT [Roy22, BBD⁺23] builds sVOLE from $(N-1)$ -out-of- N OT, where $N = |S_\Delta|$ and $\Delta \leftarrow S_\Delta$. SoftSpoken-like protocols have two properties: first, the sender of sVOLE plays the role of sender in OT; second, the choice of the OT receiver can be delayed to the very end of the protocol. [BBD⁺23] call the protocols that satisfy these properties *OT-admissible*. Using their VOLEitH technique, VOLE-based ZK protocols that are OT-admissible can be transformed to publicly verifiable ZK proofs. Therefore, SoftSpoken-like protocols are very attractive in scenarios that require public verifiability.

The bottleneck of SoftSpoken-like protocols is the computation cost, which is linear in the size of S_Δ . When requiring sVOLE correlation over a large extension field (e.g., building VOLE-based ZK with soundness error negligible in the security parameter λ), the computation complexity could be exponential in λ . To address this issue, [Roy22, BBD⁺23] propose to repeat sVOLE over small fields $\mathbb{F}_{p^r}$ to get τ instances with different Δ_i . $\mathcal{P}$ then corrects all u_i to the same u and gets $u \cdot \Delta_i + v_i = q_i, i \in [\tau]$. Finally, $\mathcal{P}$ lifts $(v_1, \dots, v_\tau) \hookrightarrow v \in \mathbb{F}_{p^{r\tau}}$ and $\mathcal{V}$ lifts $(\Delta_1, \dots, \Delta_\tau) \hookrightarrow \Delta \in \mathbb{F}_{p^{r\tau}}$ and $(q_1, \dots, q_\tau) \hookrightarrow q \in \mathbb{F}_{p^{r\tau}}$. This approach achieves a computation-communication trade-off by choosing different r and τ .

In this section, we generalize SoftSpoken sVOLE to Galois rings. We first give some ideal functionalities in Sect. 3.1. Then we construct a srVOLE protocol for low-degree Galois rings in Sect. 3.2. Based on this construction, we use the repetition approach to build srVOLE for high-degree Galois rings in Sect. 3.3 with a consistency check to realize malicious security. In Sect. 3.4, we describe how to generate the SPDZ_{2k}-like VOLE used in MozZ_{2k} arella [BBMHS22] as a special case of our subset srVOLE.

3.1 Ideal Functionalities

The ideal functionality of $(N-1)$ -out-of- N random OT is given in Fig. 3. The srVOLE functionality given in Fig. 4 is a generalization of the subspace VOLE presented in [Roy22, BBD⁺23] to Galois rings. The differences are as follows: while in [BBD⁺23] the entries of $\mathbf{V}, \mathbf{Q}$ were in the fields $\mathbb{F}_{p^{d_1}}$ and those of $\mathbf{U}$ were in the subfield $\mathbb{F}_{p^{d_0}}$, in our case their entries are respectively in the Galois ring $\text{GR}(p^k, d_1)$ and subring $\text{GR}(p^k, d_0)$. However, Δ is still in $\mathbb{F}_{p_1^d}$ as in [LXY24b] for efficiency. The other difference is that the version of [BBD⁺23] allows the rows of $\mathbf{U}$ to be encoded with an error-correcting code before being multiplied by Δ . This brings some problems in the Galois rings as we discussed in Sect. 1.2 and we leave it as an interesting problem. If we take $k=1$, we recover the functionality from [BBD⁺23] where the error correcting code used is the trivial $[n, 1, n]$ -linear code over $\mathbb{F}_{p^{d_0}}$.

Functionality $\mathcal{F}_{\text{ROT-I}}^N$

Init: Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$:

1. If $\mathcal{P}$ is honest, sample $s_i \leftarrow \{0, 1\}^\lambda$ for all $i \in \mathbb{F}_N$. Otherwise receive $\{s_i \in \{0, 1\}^\lambda\}_{i \in \mathbb{F}_N}$ from $\mathcal{A}$.
2. If $\mathcal{V}$ is honest, sample $\Delta \leftarrow \mathbb{F}_N$ and send Δ to $\mathcal{V}$. Otherwise receive $\Delta \in \mathbb{F}_N$ and $\{s_i^*\}_{i \in \mathbb{F}_N \setminus \{\Delta\}}$ from $\mathcal{A}$, then set $s_i = s_i^*$ for all $i \in \mathbb{F}_N \setminus \{\Delta\}$.
3. Send $\{s_i\}_{i \in \mathbb{F}_N}$ to $\mathcal{P}$ and (done) to $\mathcal{V}$. Ignore any subsequent (init) commands.

Get: Upon receiving (get) from $\mathcal{V}$, send $\{s_i\}_{i \in \mathbb{F}_N \setminus \{\Delta\}}$ to $\mathcal{V}$.

Fig. 3. Ideal Functionality of $\binom{N}{N-1}$ -OT

Functionality $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,n,\mathcal{L}}$

Parameters: p is a prime and $d_0 \mid d_1$.

Init: Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$:

1. If $\mathcal{P}$ is honest, sample $\mathbf{U} \leftarrow \text{GR}(p^k, d_0)^{\ell \times n}$, $\mathbf{V} \leftarrow \text{GR}(p^k, d_1)^{\ell \times n}$. Otherwise receive $\mathbf{U} \in \text{GR}(p^k, d_0)^{\ell \times n}$, $\mathbf{V} \in \text{GR}(p^k, d_1)^{\ell \times n}$ and $L \in \mathcal{L}$ from $\mathcal{A}$.
2. If $\mathcal{V}$ is honest, sample $\Delta \leftarrow \mathbb{F}_{p^{d_1}}^n$ and compute $\mathbf{Q} := \mathbf{V} + \mathbf{U} \cdot \text{diag}(\Delta)$. Otherwise receive $\Delta \in \mathbb{F}_{p^{d_1}}^n$, $\mathbf{Q} \in \text{GR}(p^k, d_1)^{\ell \times n}$ from $\mathcal{A}$ and recompute $\mathbf{V} := \mathbf{Q} - \mathbf{U} \cdot \text{diag}(\Delta)$.
3. Send $(\mathbf{U}, \mathbf{V})$ to $\mathcal{P}$ and ignore any subsequent (init) commands.

Get: Upon receiving (get) from $\mathcal{V}$, if $\mathcal{P}$ is corrupted and $\Delta \notin L$, send (check failed) to $\mathcal{V}$ and abort. Otherwise send $(\Delta, \mathbf{Q})$ to $\mathcal{V}$.

Fig. 4. Ideal Functionality of Subring VOLE

3.2 Construction for Low-Degree Galois Rings

We construct srVOLE protocol for low-degree Galois rings in the $\mathcal{F}_{\text{ROT-I}}^N$ -hybrid model. The details are given in Fig. 5.

Subring VOLE Protocol $\Pi_{\text{low-srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$ **for Low Degree GR**

Parameters:

- $\text{PRG} : \{0, 1\}^\lambda \rightarrow \text{GR}(p^k, d_0)^\ell$, $N = p^{d_1}$, $d_0 \mid d_1$.
- Ideal functionality $\mathcal{F}_{\text{ROT-I}}^N$ defined in Fig. 3.

Init:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\text{ROT-I}}^N$.
2. $\mathcal{P}$ receives $\{s_i\}_{i \in \mathbb{F}_N}$ from $\mathcal{F}_{\text{ROT-I}}^N$, where $s_i \in \{0, 1\}^\lambda$.
3. For all $i \in \mathbb{F}_N$, $\mathcal{P}$ computes $\mathbf{t}_i := \text{PRG}(s_i)$.
4. $\mathcal{P}$ computes and outputs $\mathbf{u} := \sum_{i \in \mathbb{F}_N} \mathbf{t}_i$, $\mathbf{v} := -\sum_{i \in \mathbb{F}_N} \mathbf{t}_i \cdot i$.

Get:

1. $\mathcal{V}$ sends (get) to $\mathcal{F}_{\text{ROT-I}}^N$ and receives $\Delta \in \mathbb{F}_N$, $\{s_i\}_{i \in \mathbb{F}_N \setminus \{\Delta\}}$.
2. For all $i \in \mathbb{F}_N \setminus \{\Delta\}$, $\mathcal{V}$ computes $\mathbf{t}_i := \text{PRG}(s_i)$.
3. $\mathcal{V}$ computes $\mathbf{q} := \sum_{i \in \mathbb{F}_N \setminus \{\Delta\}} \mathbf{t}_i \cdot (\Delta - i)$.
4. $\mathcal{V}$ outputs $(\Delta, \mathbf{q})$.

Fig. 5. Subring VOLE Protocol $\Pi_{\text{low-srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$ for Low Degree GR

Theorem 3. The protocol $\Pi_{\text{low-srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$ given in Fig. 5 SUC-realizes $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$ in the $\mathcal{F}_{\text{ROT-I}}^{p^{d_1}}$ -hybrid model, with malicious security.

Proof. We first prove the correctness. We have

$$\mathbf{u} \cdot \Delta + \mathbf{v} = \sum_{i \in \mathbb{F}_N} \mathbf{t}_i \cdot \Delta - \sum_{i \in \mathbb{F}_N} \mathbf{t}_i \cdot i = \sum_{i \in \mathbb{F}_N} \mathbf{t}_i \cdot (\Delta - i) = \sum_{i \in \mathbb{F}_N \setminus \{\Delta\}} \mathbf{t}_i \cdot (\Delta - i) = \mathbf{q}$$

To prove that each component (u, v) is uniform over $\text{GR}(p^k, d_0) \times \text{GR}(p^k, d_1)$, consider a group homomorphism φ^4 ,

$$\begin{aligned} \varphi : \text{GR}(p^k, d_0)^N &\rightarrow \text{GR}(p^k, d_0) \times \text{GR}(p^k, d_1) \\ (t_0, \dots, t_{N-1}) &\mapsto \left(\sum_{i \in \mathbb{F}_N} t_i, \sum_{i \in \mathbb{F}_N} t_i \cdot i \right) \end{aligned}$$

⁴ Throughout, we identify $[0, N)$ with $\mathbb{F}_N$ via a fixed bijection.

Then it is sufficient to prove that φ is surjective. Let $d_1 = m \cdot d_0$, for any $(u^*, v^*) \in \text{GR}(p^k, d_0) \times \text{GR}(p^k, d_1)$, we write v^* as $a_0 + a_1\xi + \dots + a_{m-1}\xi^{m-1}$, where $a_0, a_1, \dots, a_{m-1} \in \text{GR}(p^k, d_0)$ and $\xi \in \text{GR}(p^k, d_1)$ is an element of order $p^{d_1} - 1$. Let $c(\cdot)$ be the function such that $c(\xi^i) = a_i$ for all $i \in [m-1]$ and $c(1) = a_0$.

Define $\mathcal{I} := \{i \in \mathbb{F}_N : \iota(i) \in \{1, \xi, \dots, \xi^{m-1}\}\}$. For all $i \in \mathcal{I}$, let $t_i^* := c(\iota(i))$. Let $t_0^* := u^* - \sum_{i \in \mathcal{I}} t_i^*$ and $t_j^* := 0$ for all $j \in \mathbb{F}_N^* \setminus \mathcal{I}$. Then we have $\varphi(t_0^*, \dots, t_{N-1}^*) = (u^*, v^*)$. Therefore, φ is surjective.

Malicious $\mathcal{P}$. The simulator $\mathcal{S}$ emulates the role of $\mathcal{F}_{\text{ROT-}\bar{1}}^N$. After receiving $\{s_i\}_{i \in \mathbb{F}_N}$ and $L = \mathbb{F}_{p^{d_1}}$ from $\mathcal{A}$, $\mathcal{S}$ computes $\mathbf{u}, \mathbf{v}$ honestly and sends $\mathbf{u}, \mathbf{v}, L$ to $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$. In both real and ideal world, Δ is uniformly random over $\mathbb{F}_{p^{d_1}}$ and $\mathbf{q}$ is computed as $\mathbf{u} \cdot \Delta + \mathbf{v}$. Therefore, these two worlds are identical.

Malicious $\mathcal{V}$. The simulator $\mathcal{S}$ emulates the role of $\mathcal{F}_{\text{ROT-}\bar{1}}^N$. After receiving $\{s_i^*\}_{i \in \mathbb{F}_N \setminus \{\Delta\}}$, Δ from $\mathcal{A}$, $\mathcal{S}$ computes $\mathbf{q}$ honestly and sends $\Delta, \mathbf{q}$ to $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$. In the ideal world, $\mathbf{u}$ is uniformly sampled. In the real world, since s_Δ is uniform and independent from $\{s_i^*\}_{i \in \mathbb{F}_N \setminus \{\Delta\}}$, $\mathbf{u} = \text{PRG}(s_\Delta) + \sum_{i \in \mathbb{F}_N \setminus \{\Delta\}} \text{PRG}(s_i^*)$ is indistinguishable from a uniformly random element over $\text{GR}(p^k, d_0)^\ell$. In both worlds, $\mathbf{v}$ is computed as $\mathbf{q} - \mathbf{u} \cdot \Delta$. Therefore, these worlds are indistinguishable. $\square$

Remark 3. We note that the functionality $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,n,\{\mathbb{F}_{p^{d_1}}^n\}}$ can be SUC-realized by n parallel executions of $\mathcal{F}_{\text{low-srVOLE}}^{p,k,d_0,d_1,\ell,1,\{\mathbb{F}_{p^{d_1}}\}}$, where n ℓ -dimensional vectors $\mathbf{u}_i, \mathbf{v}_i, \mathbf{q}_i$ are stacked into matrices $\mathbf{U} \in \text{GR}(p^k, d_0)^{\ell \times n}$, $\mathbf{V} \in \text{GR}(p^k, d_1)^{\ell \times n}$, $\mathbf{Q} \in \text{GR}(p^k, d_1)^{\ell \times n}$.

Subset srVOLE. We note that we can actually sample Δ from any polynomial-size subset $S_\Delta \subseteq \text{GR}(p^k, d_1)$, which is especially useful in the following cases: (1) the characteristic p is very large such that the computation cost remains high even when $d_1 = 1$; (2) we are generating VOLE instances over integer rings (i.e., $d_1 = 1$) for VOLE-based ZK. In this case, as we discuss in Sect. 1.2, sampling Δ from the whole ring does not improve soundness.

To ensure correctness, we require that $\{1\} \times S_\Delta$ must contain a free basis $\{(1, b_0), \dots, (1, b_m)\}$ of the free $\text{GR}(p^k, d_0)$ -module $\text{GR}(p^k, d_0) \times \text{GR}(p^k, d_1)$. When we sample the coefficients $t_0, \dots, t_m$ from $\text{GR}(p^k, d_0)$, $\sum_{i=0}^m t_i \cdot (1, b_i)$ could cover every element in $\text{GR}(p^k, d_0) \times \text{GR}(p^k, d_1)$, which means φ is surjective. From the proof of Theorem 3, one can find that $S_\Delta^* = \{0\} \cup \mathcal{I}$ satisfies the requirement. In this case, $|S_\Delta^*| = \frac{d_1}{d_0} + 1$, which is independent of p and also minimal. Therefore, we can use any S_Δ that satisfies $S_\Delta^* \subseteq S_\Delta \subseteq \mathbb{F}_N$.

Optimization for Integer Rings. When $d_0 = 1$, i.e., $\mathbf{U} \in \mathbb{Z}_{p^k}^{\ell \times n}$, we do not need to apply $\iota(\cdot)$ when computing $\mathbf{V}, \mathbf{Q}$. Instead, we treat $\mathbb{F}_N$ as a subset of $\text{GR}(p^k, d_1)$: $S_\Delta = \{a_0 + a_1x + \dots + a_{d_1-1}x^{d_1-1} \in \text{GR}(p^k, d_1) : \forall i, a_i \in [0, p-1]\}$. This approach could improve computation efficiency since now we do not need to do the Teichmüller lift. To see why this works, we first observe that $\{1\} \times \{0, 1, x, \dots, x^{d_1-1}\}$ is a free basis of the free $\mathbb{Z}_{p^k}$ -module $\mathbb{Z}_{p^k} \times \text{GR}(p^k, d_1)$. Since $\{0, 1, x, \dots, x^{d_1-1}\} \subset S_\Delta$, the correctness holds.

3.3 Construction for High-Degree Galois Rings

For Galois rings with large extension degree d , we first generate submodule srVOLE with $[\tau, 1, \tau]$ -repetition code over the subring $\text{GR}(p^k, d_1)$, where $d = \tau \cdot d_1$. Then we lift the vectors over $\text{GR}(p^k, d_1)$ to the elements $\text{GR}(p^k, d)$ to obtain the result.

To construct submodule srVOLE over Galois rings, we adapt the techniques from [Roy22]. Given the functionality $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$, we obtain τ independent instances of srVOLE over $\text{GR}(p^k, d_1)$. Then $\mathcal{P}$ sends the corrections to $\mathcal{V}$ to derandomize all $\mathbf{U}^i$ to the same $\mathbf{u}$. To achieve malicious security, we adapt the consistency check based on linear universal hash functions to our setting. However, this consistency check would allow a malicious $\mathcal{P}$ to guess Δ . Specifically, $\mathcal{P}$ can try to guess if Δ belongs to some structured set $L \in \mathcal{L}$, where $\mathcal{L}$ is the collection of all valid guesses. As in [Roy22], we define $\mathcal{L}$ as the set of all affine subspaces of $\mathbb{F}_{p^{d_1}}^\tau$, where all elements can be lifted to $\mathbb{F}_{p^d}$.

Subring VOLE Protocol $\Pi_{\text{high-srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$ for High Degree GR

Parameters:

- $d = \tau \cdot d_1 = \tau \cdot m \cdot d_0$. $\mathcal{H} \subseteq \text{GR}(p^k, d_0)^{r \times (\ell+h)}$ is a family of $\text{GR}(p^k, d_0)^\ell$ -hiding, linear ε -almost universal hash functions.
- Ideal functionality $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$ defined in Fig. 4.

Init:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$.
2. $\mathcal{P}$ receives $\mathbf{U} \in \text{GR}(p^k, d_0)^{(\ell+h) \times \tau}$, $\mathbf{V} \in \text{GR}(p^k, d_1)^{(\ell+h) \times \tau}$ from $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$.
3. Let $\mathbf{u} := \mathbf{U}^\top$. $\mathcal{P}$ computes $[0 \ \mathbf{C}] := \mathbf{U} - \mathbf{u} \cdot \mathbf{1}_\tau$ and sends $\mathbf{C} \in \text{GR}(p^k, d_0)^{(\ell+h) \times (\tau-1)}$.
4. $\mathcal{V}$ samples and sends a uniformly random hash function $\mathbf{H} \leftarrow \mathcal{H}$.
5. $\mathcal{P}$ computes and sends $\tilde{\mathbf{u}} := \mathbf{H}\mathbf{u}$, $\tilde{\mathbf{V}} := \mathbf{H}\mathbf{V}$.
6. For all $i \in [\ell]$, $\mathcal{P}$ lifts $\mathbf{V}_i \in \text{GR}(p^k, d_1)^\tau \hookrightarrow v_i \in \text{GR}(p^k, d)$.
7. $\mathcal{P}$ outputs $\mathbf{u}_{[\ell]}$, $\mathbf{v} := (v_1, \dots, v_\ell)^\top$.

Get:

1. $\mathcal{V}$ sends (get) to $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$ and receives $\Delta' \in \mathbb{F}_{p^{d_1}}^\tau$, $\mathbf{Q}' \in \text{GR}(p^k, d_1)^{(\ell+h) \times \tau}$.
2. $\mathcal{V}$ computes $\mathbf{Q} := \mathbf{Q}' - [0 \ \mathbf{C}] \text{diag}(\Delta')$.
3. $\mathcal{V}$ aborts if $\mathbf{H}\mathbf{Q} \neq \tilde{\mathbf{u}} \cdot \mathbf{1}_\tau \cdot \text{diag}(\Delta') + \tilde{\mathbf{V}}$.
4. $\mathcal{V}$ lifts $\Delta' \in \mathbb{F}_{p^{d_1}}^\tau \hookrightarrow \Delta \in \mathbb{F}_{p^d}$ and $\mathbf{Q}_i \in \text{GR}(p^k, d_1)^\tau \hookrightarrow q_i \in \text{GR}(p^k, d)$ for all $i \in [\ell]$.
5. $\mathcal{V}$ outputs Δ , $\mathbf{q} := (q_1, \dots, q_\ell)^\top$.

Fig. 6. Subring VOLE Protocol $\Pi_{\text{high-srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$ for High-Degree GR

Furthermore, to make the protocol compatible with Fiat-Shamir transform, we analyze the security in a similar way to [BBD⁺23], which is independent of the choice of Δ and only relies on the property of universal hashing. We describe the details in Fig. 6.

Theorem 4. *The protocol $\Pi_{\text{high-srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$ given in Fig. 6 SUC-realizes $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$ in the $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$ -hybrid model, with malicious security. And the distinguisher in malicious model has advantage at most $\varepsilon \binom{\tau}{2}$.*

Proof. The correctness follows from the construction.

Malicious $\mathcal{V}$. The proof is exactly the same as SoftSpokenOT except that we are working on Galois rings. We omit the details here.

Malicious $\mathcal{P}$. We present the simulator in Fig. 7, which is adapted from [BBD⁺23]. The only problem is that the simulator can abort in step 7 while the real adversary passes the consistency check. We define a bad event as in [BBD⁺23] such that if the simulator aborts in step 7 then the bad event must be triggered.

Bad event. Every row of $\mathbf{H}\mathbf{U}'_{-G}$ is a repetition of a single element while at least one row of $\mathbf{U}'_{-G}$ is not.

If the simulator aborts in step 7, there exists a row $\mathbf{u}_{-G}$ of $\mathbf{U}'_{-G}$ that is not a repetition of a single element. However, $\mathbf{H}\mathbf{U}'_{-G} = \hat{\mathbf{U}}_{-G} + (\tilde{\mathbf{u}} \cdot \mathbf{1}_\tau)_{-G} = (\tilde{\mathbf{u}} \cdot \mathbf{1}_\tau)_{-G}$. So the bad event is triggered.

Then we bound the probability that the bad event occurs. There must be some $\mathbf{p} = \mathbf{e}_i - \mathbf{e}_j$, where $\mathbf{e}_i, \mathbf{e}_j$ are unit vectors in $\text{GR}(p^k, d_0)^{\tau-|G|}$ and $i < j$, such that $\mathbf{u}_{-G} \cdot \mathbf{p} \neq 0$. So $\mathbf{U}'_{-G} \mathbf{p} \neq 0$. Therefore, $\Pr[\mathbf{H}\mathbf{U}'_{-G} \mathbf{p} = 0] \leq \varepsilon$ since $\mathbf{H}$ is sampled from an ε -almost universal hash family. By a union bound for all possible $\mathbf{p}$, $\Pr[\text{Bad event}] \leq \varepsilon \binom{\tau}{2}$, which completes the proof. $\square$

Remark 4. The proof for general subspace over finite fields in [BBD⁺23] does not directly extend to general linear codes over Galois rings due to the existence of zero-divisors. However, we can obtain the same result as [BBD⁺23] for a restricted class of codes: free codes that are still free after arbitrary puncturing. Nevertheless, this property seems too strong. So we only give the construction for repetition code, which is good enough and sufficient for many applications. We note that Reed-Solomon codes also satisfy this property, which may be of independent interest.

Simulator for malicious $\mathcal{P}$ in $\Pi_{\text{high-srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$

On (init) sent to $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell+h,\tau,\{\mathbb{F}_{p^{d_1}}^\tau\}}$:

1. Receive $\mathbf{U}, \mathbf{V}$ from $\mathcal{A}$ and send them to $\mathcal{P}$.
2. Receive $\mathbf{C}$ from $\mathcal{P}$. Compute the derandomization $\mathbf{U}' := \mathbf{U} - [0 \ \mathbf{C}]$. We have:

$$\mathbf{Q} = \mathbf{Q}' - [0 \ \mathbf{C}] \text{diag}(\Delta') = \mathbf{U}' \text{diag}(\Delta') + \mathbf{V}$$

3. Sample and send a uniformly random hash $\mathbf{H} \leftarrow \mathcal{H}$ to $\mathcal{P}$.
4. Receive the response $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{V}}$ from $\mathcal{P}$.
5. Compute the response error $\hat{\mathbf{U}} := \mathbf{H}\mathbf{U}' - \tilde{\mathbf{u}} \cdot \mathbf{1}_\tau$ and $\hat{\mathbf{V}} := \mathbf{H}\mathbf{V} - \tilde{\mathbf{V}}$.
6. The consistency check equation is $-\hat{\mathbf{V}} = \hat{\mathbf{U}} \text{diag}(\Delta')$. Abort if no solutions for Δ' exist. Otherwise, there exists a set of guessed columns $G := \{i \in [\tau] : \hat{\mathbf{U}}^i \neq 0\}$ and values $\{\Delta_i^*\}_{i \in G}$ such that the solution set is $L' := \{\Delta' \in \mathbb{F}_{p^{d_1}}^\tau : \forall i \in G, \Delta'_i = \Delta_i^*\}$.
7. Abort if not all columns $\mathbf{U}'_{-G}$ are equal. Otherwise let $\mathbf{u}^*$ be the first column of $\mathbf{U}'_{-G}$.
8. Pick an arbitrary $\Delta^* \in L'$ and recover $\mathbf{V}^* := \mathbf{V} + (\mathbf{U}' - \mathbf{u}^* \cdot \mathbf{1}_\tau) \text{diag}(\Delta^*)$ from $\mathcal{A}$'s guesses. We have $\mathbf{V}^* = \mathbf{Q} - \mathbf{u}^* \cdot \mathbf{1}_\tau \cdot \text{diag}(\Delta')$ if the consistency check passes because $\mathbf{U}' - \mathbf{u}^* \cdot \mathbf{1}_\tau$ is zero except for the columns in G , and $\Delta'_i = \Delta_i^*$ for $i \in G$.
9. Lift every $\Delta' \in L'$ to $\mathbb{F}_{p^d}$ to get a set $L \in \mathcal{L}$. For all $i \in [\ell]$, lift $\mathbf{V}_i^* \in \text{GR}(p^k, d_1)^\tau \hookrightarrow v_i^* \in \text{GR}(p^k, d)$.
10. Send $\mathbf{u}_{[\ell]}^*, \mathbf{v}^* := (v_1^*, \dots, v_\ell^*)^\top, L$ to $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$.

Fig. 7. Simulator for malicious $\mathcal{P}$ in $\Pi_{\text{high-srVOLE}}^{p,k,d_0,d,\ell,1,\mathcal{L}}$

3.4 VOLE Protocols for Moz $\mathbb{Z}_{2^k}$ arella

Moz $\mathbb{Z}_{2^k}$ arella [BBMHS22] uses SPD $\mathbb{Z}_{2^k}$ -like VOLE correlations $\tilde{\mathbf{u}} \cdot \Delta + \mathbf{v} = \mathbf{q} \bmod 2^\sigma$, where $\tilde{\mathbf{u}}, \mathbf{v}, \mathbf{q} \in \mathbb{Z}_{2^\sigma}^\ell$, $\Delta \in \mathbb{Z}_{2^s}$ and $\sigma > s$. This functions as a commitment to $\mathbf{u} = \tilde{\mathbf{u}} \bmod 2^k$ for some $k < \sigma$. Since they work over integer rings, their protocol enjoys high computational efficiency. However, they only provide a PCG-like protocol to generate such VOLE instances, which is not OT-admissible and thus is not compatible with the VOLEitH technique. That is to say, we still do not know how to make their QuarkSilver protocol publicly verifiable.

Fortunately, we find that this SPD $\mathbb{Z}_{2^k}$ -like VOLE correlation is implied by our SoftSpoken-like subset srVOLE as a special case, where the Galois ring is $\text{GR}(2^\sigma, 1) = \mathbb{Z}_{2^\sigma}$ and the subset is $S_\Delta = \mathbb{Z}_{2^s} \subset \mathbb{Z}_{2^\sigma}$. However, when s is large, which is typically $\Omega(\lambda)$, the computation cost is exponential as in the case of high-degree Galois rings. We thus use a 2^s -ary decomposition to aggregate τ VOLE instances over small rings to get a VOLE instance over large rings, where $s = \tau \cdot s'$, reducing the computation from $O(2^s)$ to $O(\tau \cdot 2^{s'})$.

Therefore, combined with our construction and VOLEitH, the QuarkSilver protocol can be made publicly verifiable. And it can be further made non-interactive using Fiat-Shamir transform. We give more details in Appendix D.

4 PCG-like srVOLE Protocols

PCG-like protocols build sVOLE based on the LPN assumption. The key observation is that we can generate sVOLE correlations for sparse vectors with only sublinear communication (in the length of the vectors). [BCGI18] proposes two constructions based on primal LPN and dual LPN respectively. Informally, the primal sVOLE generator based on primal LPN only allows for a quadratic stretch in the size of the seed, while the dual sVOLE generator based on dual LPN can realize an arbitrary polynomial stretch. In terms of communication and storage, the dual sVOLE generator is much more efficient. However, it suffers from the costly matrix-vector multiplication, which makes it less practical. Although the following works [BCG⁺22, RRT23] propose some structured codes to accelerate the computation, they all work on finite fields and the security over rings needs to be further analyzed.

Therefore, we build our srVOLE protocol based on primal LPN⁵. We first give the functionalities of PCG-like srVOLE and sp-srVOLE in Sect. 4.1. Then we construct sp-srVOLE in Sect. 4.2, which

⁵ We note that one can also adapt the construction based on dual LPN to Galois rings in a similar way. We omit the details here.

is a Galois ring analogue of sVOLE in [WYKW21]. Based on our sp-srVOLE, we realize srVOLE extension in Sect. 4.3.

4.1 Ideal Functionalities

The ideal functionality $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ of PCG-like srVOLE is given in Fig. 8, which generalizes the sVOLE functionality in [WYKW21]. When we take $k = 1$, they are the same. Then we define the functionality of sp-srVOLE as an extension of $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ in Fig. 9, which allows us to generate srVOLE instances for vectors with a single non-zero entry. There is one difference compared to single-point subfield VOLE (sp-sVOLE) over finite fields: while sp-sVOLE only allows the adversary to guess the whole Δ , the adversary in sp-srVOLE can try to guess the lower digits of Δ . This difference comes from the fact that any linear equation with a single variable over fields has at most one solution while there could be many over rings. We note that although we can sample Δ from $\mathbb{F}_{p^{d_1}}$ as we do in SoftSpoken-like srVOLE, here we sample from the whole Galois ring to make things more natural.

Functionality $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$

Parameters: p is a prime and $d_0 \mid d_1$.

Init: Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$, sample $\Delta \leftarrow \text{GR}(p^k, d_1)$ if $\mathcal{V}$ is honest, otherwise receive $\Delta \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$. Store Δ and send it to $\mathcal{V}$. Ignore any subsequent (init) commands.

Extend: Upon receiving (extend, ℓ) from both parties:

1. If $\mathcal{P}$ is honest, sample $\mathbf{u} \leftarrow \text{GR}(p^k, d_0)^\ell$, $\mathbf{v} \leftarrow \text{GR}(p^k, d_1)^\ell$. Otherwise receive $\mathbf{u} \in \text{GR}(p^k, d_0)^\ell$, $\mathbf{v} \in \text{GR}(p^k, d_1)^\ell$ from $\mathcal{A}$.
2. If $\mathcal{V}$ is honest, compute $\mathbf{q} := \mathbf{u} \cdot \Delta + \mathbf{v}$. Otherwise receive $\mathbf{q} \in \text{GR}(p^k, d_1)^\ell$ from $\mathcal{A}$ and recompute $\mathbf{v} := \mathbf{q} - \mathbf{u} \cdot \Delta$.
3. Send $(\mathbf{u}, \mathbf{v})$ to $\mathcal{P}$ and $\mathbf{q}$ to $\mathcal{V}$.

Global-key Query: If $\mathcal{P}$ is corrupted, receive (Guess, Δ', k') from $\mathcal{A}$ with $\Delta' \in \text{GR}(p^{k'}, d_1)$ and $k' \in [k]$. If $\Delta' = \Delta \bmod p^{k'}$, send (success) to $\mathcal{P}$. Otherwise, send (abort) to both parties and abort.

Fig. 8. Ideal Functionality of PCG-like Subring VOLE

Functionality $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$

This functionality extends $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ in Fig. 8. **Init**, **Extend** and **Global-key Query** are identical.

SP-Extend: Upon receiving (SP-extend, ℓ) from $\mathcal{P}$ and $\mathcal{V}$:

1. If $\mathcal{P}$ is honest, sample $\mathbf{u} \leftarrow \text{GR}(p^k, d_0)^\ell$ with a single entry invertible and zeros everywhere else, $\mathbf{v} \leftarrow \text{GR}(p^k, d_1)^\ell$. Otherwise receive $\mathbf{u} \in \text{GR}(p^k, d_0)^\ell$ with at most one non-zero entry and $\mathbf{v} \in \text{GR}(p^k, d_1)^\ell$ from $\mathcal{A}$. Compute $\mathbf{q} := \mathbf{u} \cdot \Delta + \mathbf{v}$.
2. If $\mathcal{V}$ is corrupted:
 - (a) Receive $\mathbf{q} \in \text{GR}(p^k, d_1)^\ell$ from $\mathcal{A}$ and recompute $\mathbf{v} := \mathbf{q} - \mathbf{u} \cdot \Delta$.
 - (b) Receive a set $I \subseteq [\ell]$ from $\mathcal{A}$. Let $\alpha \in [\ell]$ be the index of the non-zero entry of $\mathbf{u}$. If $\alpha \in I$, send (success) to $\mathcal{V}$ and continue. Otherwise send (abort) to both parties and abort.
3. Send $(\mathbf{u}, \mathbf{v})$ to $\mathcal{P}$ and $\mathbf{q}$ to $\mathcal{V}$.

Fig. 9. Ideal Functionality of Single-Point Subring VOLE

4.2 Single-Point srVOLE

Our construction of sp-srVOLE is based on an interactive punctured pseudorandom function (PPRF). In an interactive PPRF $F : [\ell] \rightarrow \text{GR}(p^k, d_1)$, the sender $\mathcal{V}$ inputs a key k and the receiver $\mathcal{P}$ inputs

an index $\alpha \in [\ell]$. $\mathcal{P}$ outputs a punctured key k^* that is punctured at α . By evaluating F at all points except α , $\mathcal{P}$ and $\mathcal{V}$ get two length- ℓ vectors $\mathbf{v}, \mathbf{q}$ respectively, where $q_i = v_i$ for all $i \in [\ell] \setminus \{\alpha\}$ and v_α remains undefined. Therefore, it remains to set $v_\alpha := q_\alpha - \beta \cdot \Delta$ for some $\beta \in \text{GR}(p^k, d_0)$ sampled from some LPN noise distribution. Then we get a srVOLE correlation $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$ where $u_\alpha = \beta$ is the only non-zero entry of $\mathbf{u}$. To resist the reduction attack in [LWYY24], we further require that β is invertible as [BBMHS22] does.

To do this, both parties first generate a random srVOLE instance $a \cdot \Delta + b = c$. Then $\mathcal{P}$ acts as the srVOLE sender (note that it acts as the receiver in PPRF) and sends the correction $a' = \beta - a$ so both parties can update the instance to $\gamma = \beta \cdot \Delta + \delta$. Then $\mathcal{V}$ sends $g = \gamma - \sum_{j \in [\ell]} q_j$ and $\mathcal{P}$ defines $v_\alpha := \delta - g - \sum_{j \in [\ell] \setminus \{\alpha\}} v_j$. For correctness, we have

$$\begin{aligned} \beta \cdot \Delta + v_\alpha &= \beta \cdot \Delta + \delta - g - \sum_{j \in [\ell] \setminus \{\alpha\}} v_j \\ &= \beta \cdot \Delta + \delta - \gamma + \sum_{j \in [\ell]} q_j - \sum_{j \in [\ell] \setminus \{\alpha\}} v_j \\ &= q_\alpha \end{aligned}$$

We instantiate the interactive PPRF using the GGM-PPRF from [BCG⁺19a], which achieves $O(\log \ell \cdot \lambda)$ communication. We give more details about GGM-PPRF in Appendix C.

The above construction only achieves semi-honest security. While a malicious $\mathcal{P}$ can only cheat in oblivious transfer, which is equivalent to changing its input α , a malicious $\mathcal{V}$ could cheat in two ways: (1) send inconsistent messages in different layers in GGM-PPRF; (2) send an incorrect g . To address these issues, we adapt the consistency check from [WYKW21] to our subring case. The idea is to check the correctness of the final output using a linear combination

$$\chi_\alpha \cdot \beta \cdot \Delta + \sum_{i \in [\ell]} \chi_i \cdot v_i = \sum_{i \in [\ell]} \chi_i \cdot q_i$$

where $\beta \cdot \Delta$ can be shared by consuming a random OLE instance so the parties can compute each side respectively and test the equality.

However, the consistency check allows the adversary to attack in other ways. By sending a malformed summation in the equality test, a corrupted $\mathcal{P}$ can try to guess Δ and a corrupted $\mathcal{V}$ can try to guess the index of non-zero entry in $\mathbf{u}$. These attacks have been incorporated into the functionality.

We give details of our sp-srVOLE protocol in Fig. 10. The security is stated in Theorem 5. We defer the proof to Appendix E.1.

Theorem 5. *The protocol $\Pi_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ given in Fig. 10 SUC-realizes $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ in the $(\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}, \mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{EQ}})$ -hybrid model, with malicious security. And the distinguisher in malicious model has advantage at most $1/p^{d_1} + \text{negl}(\lambda)$.*

4.3 srVOLE Extension

Given sp-srVOLE, we construct srVOLE extension protocol based on the primal LPN assumption. The idea is to generate two srVOLE instances: one is for a uniformly random vector $\mathbf{u} \in \text{GR}(p^k, d_0)^{k_c}$, the other is for a regular noise $\mathbf{e} \in \text{GR}(p^k, d_0)^{n_c}$. Then we can extend the short srVOLE instance $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$ of length k_c to a longer one, where the message is defined as $\mathbf{x} := \mathbf{u} \cdot \mathbf{A} + \mathbf{e}$ and $\mathbf{A} \in \text{GR}(p^k, d_0)^{k_c \times n_c}$ is an LPN generator matrix. The uniform distribution of $\mathbf{x}$ comes from the primal LPN assumption over Galois rings, which is defined as follows.

Definition 4 ([LXY24b]). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{t,n_c}(\mathcal{R})\}_{t,n_c}$ be the family of distributions over the ring $\mathcal{R}$ where $t \leq n_c$, $\text{Im}(\mathcal{D}_{t,n_c}(\mathcal{R})) \subset \mathcal{R}^{n_c}$. Let $\mathcal{G}$ be a PPT algorithm such that $\mathcal{G}(k_c, n_c, \mathcal{R})$ outputs a generator matrix $\mathbf{A} \in \mathcal{R}^{k_c \times n_c}$. Let k_c, n_c, t be implicit functions of security parameter λ . We say that the decisional $(\mathcal{D}, \mathcal{G}, \mathcal{R})$ -LPN(k_c, n_c, t) problem is (T, ε) -hard if for every probabilistic distinguisher $\mathcal{B}$ running in time T , we have*

$$|\Pr[\mathcal{B}(\mathbf{A}, \mathbf{b} = \mathbf{s} \cdot \mathbf{A} + \mathbf{e}) = 1] - \Pr[\mathcal{B}(\mathbf{A}, \mathbf{u}) = 1]| \leq \varepsilon$$

where $\mathbf{A} \leftarrow \mathcal{G}(k_c, n_c, t)$, $\mathbf{s} \leftarrow \mathcal{R}^{k_c}$, $\mathbf{e} \leftarrow \mathcal{D}_{t,n_c}(\mathcal{R})$ and $\mathbf{u} \leftarrow \mathcal{R}^{n_c}$.

Single-Point Subring VOLE Protocol $\Pi_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$

Parameters: $d_1 = m \cdot d_0$. $\xi \in \text{GR}(p^k, d_1)$ is an element of order $p^{d_1} - 1$. The ideal functionalities $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ defined in Fig. 8, $\mathcal{F}_{\text{EQ}}$ defined in Fig. 15 and $\mathcal{F}_{\text{OT}}$ defined in Fig. 16.

For the (init) and (extend) operations, the parties simply query $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$.

SP-Extend: For input $\ell = 2^h$:

1. Both parties send (extend, 1) to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$. $\mathcal{P}$ receives $a \in \text{GR}(p^k, d_0)$, $b \in \text{GR}(p^k, d_1)$ and $\mathcal{V}$ receives $c \in \text{GR}(p^k, d_1)$ such that $c = a \cdot \Delta + b$.
2. $\mathcal{P}$ samples $\alpha \leftarrow [\ell]$, $\beta \leftarrow \text{GR}(p^k, d_0)^*$ and lets $\mathbf{u} \in \text{GR}(p^k, d_0)^\ell$ be the vector with $u_\alpha = \beta$ and $u_i = 0$ for all $i \neq \alpha$.
3. $\mathcal{P}$ sets $\delta := b$ and sends $a' := \beta - a$ to $\mathcal{V}$. $\mathcal{V}$ computes $\gamma := c + a' \cdot \Delta$. Now $\gamma = \beta \cdot \Delta + \delta$.
4. $\mathcal{V}$ computes $k^0 \leftarrow \text{GGM.KeyGen}(1^\lambda)$, and $(\{q_j\}_{j \in [\ell]}, \{(K_0^i, K_1^i)\}_{i \in [h]}) \leftarrow \text{GGM.Gen}(\ell, k^0)$. Write $\alpha - 1 = \sum_{i \in [h]} 2^{h-i} \alpha_i$, for $\alpha_i \in \{0, 1\}$. Let $\bar{\alpha}_i$ denote the complement of α_i . For $i \in [h]$, both parties call $\mathcal{F}_{\text{OT}}$, where $\mathcal{P}$ inputs $\bar{\alpha}_i$ and $\mathcal{V}$ inputs (K_0^i, K_1^i) , then $\mathcal{P}$ receives $K_{\bar{\alpha}_i}^i$.
5. $\mathcal{P}$ computes $\{q_j\}_{j \in [\ell] \setminus \{\alpha\}} \leftarrow \text{GGM.PuncEval}(\ell, \alpha, \{K_{\bar{\alpha}_i}^i\}_{i \in [h]})$.
6. $\mathcal{V}$ sends $g := \gamma - \sum_{j \in [\ell]} q_j$ to $\mathcal{P}$. $\mathcal{P}$ defines $\mathbf{v} \in \text{GR}(p^k, d_1)^\ell$ such that $v_j := q_j$ for all $j \neq \alpha$ and $v_\alpha := \delta - g - \sum_{j \in [\ell] \setminus \{\alpha\}} v_j$. Then $\mathbf{q} := \mathbf{u} \cdot \Delta + \mathbf{v}$.
7. *Consistency check:*
 - (a) Both parties send (extend, m) to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$. $\mathcal{P}$ receives $\mathbf{x} \in \text{GR}(p^k, d_0)^m$, $\mathbf{y} \in \text{GR}(p^k, d_1)^m$ and $\mathcal{V}$ receives $\mathbf{z}^* \in \text{GR}(p^k, d_1)^m$ such that $\mathbf{z}^* = \mathbf{x} \cdot \Delta + \mathbf{y}$.
 - (b) $\mathcal{P}$ samples $\chi \leftarrow \text{GR}(p^k, d_1)^\ell$, and writes $\chi_\alpha = \sum_{i \in [m]} \chi_{\alpha,i} \cdot \xi^{i-1}$. Let $\chi_\alpha := (\chi_{\alpha,1}, \dots, \chi_{\alpha,m}) \in \text{GR}(p^k, d_0)^m$. $\mathcal{P}$ computes $\mathbf{x}^* := \chi_\alpha \cdot \beta - \mathbf{x}$ and sends $(\chi, \mathbf{x}^*)$ to $\mathcal{V}$, who computes $\mathbf{z} := \mathbf{z}^* + \mathbf{x}^* \cdot \Delta$.
 - (c) $\mathcal{P}$ computes $Y := \sum_{i \in [m]} y_i \cdot \xi^{i-1}$ and $V_P := \sum_{i \in [\ell]} \chi_i \cdot v_i - Y$. $\mathcal{V}$ computes $Z := \sum_{i \in [m]} z_i \cdot \xi^{i-1}$ and $V_V := \sum_{i \in [\ell]} \chi_i \cdot q_i - Z$. Then both parties call $\mathcal{F}_{\text{EQ}}$, where $\mathcal{P}$ inputs V_P and $\mathcal{V}$ inputs V_V . If either party receives (false) or (abort) from $\mathcal{F}_{\text{EQ}}$, it aborts.
8. $\mathcal{P}$ outputs $(\mathbf{u}, \mathbf{v})$ and $\mathcal{V}$ outputs $\mathbf{q}$.

Fig. 10. Single-Point Subring VOLE Protocol $\Pi_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$

We note that both T and ε are functions of the security parameter λ . Following [LWYY24], we only consider random linear codes and write $(\mathcal{D}, \mathcal{R})\text{-LPN}(k_c, n_c, t)$ to simplify the notation. We refer to Appendix B.1 for more background on LPN noise distributions.

Similar to previous work, the communication of our srVOLE extension is sublinear in the output length n_c . That is because for a regular noise $\mathbf{e}$ with hamming weight $t = o(n_c)$, we can generate the srVOLE instance by running sp-srVOLE t times, which achieves sublinear communication. For the uniform vector $\mathbf{u}$, the communication is also sublinear since $k_c = o(n_c)$.

For better concrete efficiency, we also adapt the bootstrapping technique from [YWL+20, WYKW21]. That is to say, whenever we generate a srVOLE instance of length n_c in one iteration, we save $k_c + t(1 + m)$ of them for the next iteration and only output the rest. The detailed protocol is given in Fig. 11.

Security. The security of $\Pi_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ is based on the LPN assumption over Galois rings. As generalizations of the reductions that focus on integer rings and $\text{GR}(2^k, d)$ in [LWYY24, LXY24b], we provide the reductions between primal LPN over $\mathbb{F}_{p^d}$ and primal LPN over $\text{GR}(p^k, d)$ under different noise distributions. The details are deferred to Appendix B.2.

On the other hand, since our sp-srVOLE functionality allows a malicious $\mathcal{V}$ to guess the index of non-zero entry in the noise, the underlying LPN assumption is not the standard one defined in Definition 4. We actually rely on the assumption called LPN with static leakage that is widely used in previous PCG-like protocols [BCG+19a, WYKW21, BBMHS22, LXY24b], which captures the index guessing that leaks 1 bit information in general. The formal definition is given in Appendix B.3.

Theorem 6. *If the decisional $(\text{RHW}^*, \text{GR}(p^k, d_0))\text{-LPN}(k_c, n_c, t)$ with static leakage assumption holds, then the protocol $\Pi_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ SUC-realizes $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ in the $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ -hybrid model with malicious security.*

We defer the detailed proof to Appendix E.2.

Subring VOLE Extension Protocol $\Pi_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$

Parameters: $d_1 = m \cdot d_0$. $\mathbf{A} \in \text{GR}(p^k, d_0)^{k_c \times n_c}$ is a generator matrix of $\text{LPN}(k_c, n_c, t)$, where n_c is a multiple of t . The ideal functionality $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ defined in Fig. 9.

Init: This must be called by the parties first and is executed only once.

1. Both parties send (init) to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ and $\mathcal{V}$ receives $\Delta \in \text{GR}(p^k, d_1)$.
2. Both parties send (extend, k_c) to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. $\mathcal{P}$ receives $\mathbf{u} \in \text{GR}(p^k, d_0)^{k_c}$, $\mathbf{v} \in \text{GR}(p^k, d_1)^{k_c}$ and $\mathcal{V}$ receives $\mathbf{q} \in \text{GR}(p^k, d_1)^{k_c}$, such that $\mathbf{q} = \mathbf{u} \cdot \Delta + \mathbf{v}$.

Extend: This procedure can be executed multiple times.

1. For $i \in [t]$, both parties send (SP-extend, n_c/t) to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$, which returns $\mathbf{e}_i \in \text{GR}(p^k, d_0)^{n_c/t}$, $\mathbf{b}_i \in \text{GR}(p^k, d_1)^{n_c/t}$ to $\mathcal{P}$ and $\mathbf{c}_i \in \text{GR}(p^k, d_1)^{n_c/t}$ to $\mathcal{V}$ such that $\mathbf{c}_i = \mathbf{e}_i \cdot \Delta + \mathbf{b}_i$ and $\mathbf{e}_i$ has exactly one entry invertible and zeros everywhere else.
2. Define $\mathbf{e} := (\mathbf{e}_1, \dots, \mathbf{e}_t)$, $\mathbf{b} := (\mathbf{b}_1, \dots, \mathbf{b}_t)$ and $\mathbf{c} := (\mathbf{c}_1, \dots, \mathbf{c}_t)$. $\mathcal{P}$ computes $\mathbf{x} := \mathbf{u} \cdot \mathbf{A} + \mathbf{e} \in \text{GR}(p^k, d_0)^{n_c}$ and $\mathbf{y} := \mathbf{v} \cdot \mathbf{A} + \mathbf{b} \in \text{GR}(p^k, d_1)^{n_c}$. $\mathcal{V}$ computes $\mathbf{z} := \mathbf{q} \cdot \mathbf{A} + \mathbf{c} \in \text{GR}(p^k, d_1)^{n_c}$.
3. $\mathcal{P}$ updates $\mathbf{u} := \mathbf{x}_{[k_c]}$ and $\mathbf{v} := \mathbf{y}_{[k_c]}$, and outputs $(\mathbf{x}_{[k_c+1, n_c]}, \mathbf{y}_{[k_c+1, n_c]})$. $\mathcal{V}$ updates $\mathbf{q} := \mathbf{z}_{[k_c]}$, and outputs $\mathbf{z}_{[k_c+1, n_c]}$.

Fig. 11. Subring VOLE Extension Protocol $\Pi_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$

5 ZK Protocols over Galois Rings

In this section, we build VOLE-based ZK protocols over Galois rings. Note that SoftSpoken-like protocols are compatible with VOLEitH, which can be made publicly verifiable. We thus focus only on the constructions based on SoftSpoken-like srVOLE. One can also build ZK protocols over Galois rings from PCG-like srVOLE in a similar way, and the online phase will be the same if we sample Δ from $\mathbb{F}_{p^d}$. We omit the details here for brevity.

We first introduce the general VOLE commitment in Sect. 5.1. Then we give our ZK protocols for circuit satisfiability and polynomial satisfiability in Sect. 5.2 and Sect. 5.3.

5.1 VOLE Commitment

A VOLE correlation $\mathbf{u} \cdot \Delta + \mathbf{v} = \mathbf{q}$, where $\mathbf{u} \in \text{GR}(p^k, d_0)^\ell$, $\Delta \in \mathbb{F}_{p^d}$, $\mathbf{v}, \mathbf{q} \in \text{GR}(p^k, d)^\ell$ and $d_0 \mid d$, can be viewed as a commitment to $\mathbf{u}$. In this case, we write the VOLE correlation in the form $M[\mathbf{u}] + \mathbf{u} \cdot \Delta = K[\mathbf{u}]$, or use $[\mathbf{u}]$ to denote this authenticated value.

Statistically binding. If $\mathcal{P}$ opens the commitment to a different message $\mathbf{u}'$, we have $(\mathbf{u} - \mathbf{u}') \cdot \Delta = (\mathbf{v}' - \mathbf{v})$. Since $\mathbf{u} \neq \mathbf{u}'$, there exists some $j \in [\ell]$ such that $u_j \neq u'_j$. Since Δ is uniformly random over $\mathbb{F}_{p^d}$ and kept secret from $\mathcal{P}$, the probability that $\mathcal{P}$ breaks the binding property is at most $\frac{1}{p^d}$ by Lemma 2.

Perfectly hiding. This follows directly from that $M[\mathbf{u}]$ is uniform over $\text{GR}(p^k, d)$.

Degree- D VOLE Commitment. Yang et al. [YSWW21] generalized the degree-2 VOLE commitment to higher degree. Here we adapt it to Galois rings. In a degree- D VOLE commitment for message $x \in \text{GR}(p^k, d_0)$, $\mathcal{P}$ holds a polynomial $p_x(Y) = a_0 + a_1 \cdot Y + \dots + a_{D-1} \cdot Y^{D-1} + x \cdot Y^D \in \text{GR}(p^k, d)[Y]$ where d is a multiple of d_0 . $\mathcal{V}$ holds $\Delta \in \mathbb{F}_{p^d}$ and $q_x = p_x(\Delta) \in \text{GR}(p^k, d)$. This commitment is statistically binding except with probability D/p^d by Lemma 2.

To generate a degree- D VOLE correlation, we extend the functionality $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,n,\mathcal{L}}$ to get $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d_1,\ell,n,\mathcal{L}}$ in Fig. 12. To securely realize $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d_1,\ell,n,\mathcal{L}}$, one can easily adapt the approaches in [YSWW21, BBM⁺24] to Galois rings⁶ and we omit the details here.

⁶ The protocol in [YSWW21] is designed for interactive cases and [BBM⁺24] simplifies it for non-interactive cases.

Functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d_1,\ell,n,\mathcal{L}}$

This functionality extends $\mathcal{F}_{\text{srVOLE}}^{p,k,d_0,d_1,\ell,n,\mathcal{L}}$ in Fig. 4. **Init** and **Get** are identical.

Vector Oblivious Polynomial Evaluation: Upon receiving (VOPE, D) from $\mathcal{P}$ and $\mathcal{V}$:

1. If $\mathcal{P}$ is honest, sample $A_i \leftarrow \text{GR}(p^k, d_1)$ for all $0 \leq i \leq D$. Otherwise receive $\{A_i\}_{0 \leq i \leq D}$ with $A_i \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$.
2. If $\mathcal{V}$ is honest, compute $B := \sum_{i=0}^D A_i \cdot \Delta^i$. Otherwise receive $B \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$ and recompute $A_0 := B - \sum_{i=1}^D A_i \cdot \Delta^i$.
3. Send $\{A_i\}_{0 \leq i \leq D}$ to $\mathcal{P}$ and B to $\mathcal{V}$.

Fig. 12. Ideal Functionality of Extended Subring VOLE

VOLE Commitment based on Subset VOLE. While it is possible to use general subset srVOLE correlation to build a commitment scheme, we need to analyze the binding property carefully, because Lemma 2 does not hold for general subsets. It is also possible to commit to a small range of messages. For example, the $\text{SPD}\mathbb{Z}_{2^k}$ -like VOLE correlation is the commitment to $\mathbf{u} \bmod 2^k$, where $\Delta \in \mathbb{Z}_{2^s}$, $M[\mathbf{u}], K[\mathbf{u}] \in \mathbb{Z}_{2^\sigma}$ and $\sigma \geq k + s$. In $\text{Moz}\mathbb{Z}_{2^k}$ arella [BBMHS22], the authors generalized it to a degree-2 commitment, where they require $\sigma \geq k + 2s$ and $s = \lambda + \log \lambda + 3$ to ensure binding except with probability $2^{-\lambda}$.

5.2 ZKP for Circuit Satisfiability over Galois Rings

Zero-Knowledge Protocol $\Pi_{\text{circuitZK}}^{n,t,d}$ **for Circuit Satisfiability**

Parameters: Ideal functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$ defined in Fig. 12.

Input: Both parties hold a circuit $\mathcal{C}$ over any Galois ring $\text{GR}(p^k, d_0)$ with t multiplication gates. $\mathcal{P}$ also holds a witness $\mathbf{w}$ such that $\mathcal{C}(\mathbf{w}) = 1$ and $|\mathbf{w}| = n$.

Preprocessing phase:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$. $\mathcal{P}$ receives $\mathbf{u} \in \text{GR}(p^k, d_0)^{n+t}$, $M[\mathbf{u}] \in \text{GR}(p^k, d)^{n+t}$, while $\mathcal{V}$ gets (done).
2. $\mathcal{V}$ sends (get) to $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$ and receives $\Delta \in \mathbb{F}_{p^d}$, $K[\mathbf{u}] \in \text{GR}(p^k, d)^{n+t}$.
3. $\mathcal{P}$ and $\mathcal{V}$ send (VOPE, 1) to $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$, which returns A_0^*, A_1^* to $\mathcal{P}$ and B^* to $\mathcal{V}$, such that $A_0^* + A_1^* \cdot \Delta = B^*$.

Online phase:

1. $\mathcal{P}$ computes $\delta := \mathbf{w} - \mathbf{u}_{[n]}$ and sends it to $\mathcal{V}$. Both parties compute $[\mathbf{w}] := [\mathbf{u}_{[n]}] + \delta$.
2. For each gate $(\alpha, \beta, \gamma, T) \in \mathcal{C}$, in a topological order,
 - If $T = \text{Add}$, then both parties locally compute $[w_\gamma] := [w_\alpha] + [w_\beta]$.
 - If $T = \text{Mult}$ and this is the i -th multiplication gate, $\mathcal{P}$ sends $c_i := w_\alpha \cdot w_\beta - u_{n+i}$ to $\mathcal{V}$. Then both parties compute $[w_\gamma] := [u_{n+i}] + c_i$.
3. For the i -th multiplication gate, both parties hold an authenticated triple $([w_\alpha], [w_\beta], [w_\gamma])$ from the previous step and execute the following,
 - $\mathcal{P}$ computes $A_{0,i} := M[w_\alpha] \cdot M[w_\beta]$ and $A_{1,i} := w_\alpha \cdot M[w_\beta] + w_\beta \cdot M[w_\alpha] - M[w_\gamma]$.
 - $\mathcal{V}$ computes $B_i := K[w_\alpha] \cdot K[w_\beta] - K[w_\gamma] \cdot \Delta$.
4. Let h be the single output wire. Both parties perform the following check:
 - $\mathcal{V}$ samples a uniformly random $\chi \leftarrow \mathbb{F}_{p^d}^{t+1}$ and sends it to $\mathcal{P}$.
 - $\mathcal{P}$ computes $A_1 := \sum_{i \in [t]} \chi_i \cdot A_{1,i} + \chi_{t+1} \cdot M[w_h - 1] + A_1^*$, $A_0 := \sum_{i \in [t]} \chi_i \cdot A_{0,i} + A_0^*$ and sends (A_0, A_1) to $\mathcal{V}$.
 - $\mathcal{V}$ computes $B := \sum_{i \in [t]} \chi_i \cdot B_i + \chi_{t+1} \cdot K[w_h - 1] \cdot \Delta + B^*$ and checks if $B = A_0 + A_1 \cdot \Delta$. If the check fails, $\mathcal{V}$ outputs (false) and aborts.

Fig. 13. Zero-Knowledge Protocol $\Pi_{\text{circuitZK}}^{n,t,d}$ for Circuit Satisfiability over Any Galois Rings

We adapt the QuickSilver [YSWW21] protocol for circuit satisfiability to Galois rings. After committing to all input wires, since VOLE commitment is linearly homomorphic, the prover $\mathcal{P}$ only needs to prove that (1) every multiplication gate is honestly evaluated; (2) the output $C(\mathbf{w}) = 1$. To avoid degree expansion, for each multiplication gate, $\mathcal{P}$ additionally commits to the output wire w_γ^i and proves that $[w_\gamma^i] = [w_\alpha^i] \cdot [w_\beta^i]$, where w_α^i, w_β^i are input wires. Note that $[w_\gamma^i]$ is a degree-1 VOLE commitment and $[w_\alpha^i] \cdot [w_\beta^i]$ is a degree-2 VOLE commitment, so we need to raise $[w_\gamma^i]$ to degree-2 by multiplying Y or Δ and then check the equality of the leading coefficient like Equation (1). For (2), $\mathcal{P}$ only needs to open the output wire $[w_h]$ and let $\mathcal{V}$ check if $w_h = 1$.

To improve efficiency, we can batch all these t checks, where t is the number of multiplication gates, into a single equation using a linear combination as follows

$$\underbrace{\sum_{i \in [t]} \chi_i \cdot B_i}_B = \underbrace{\sum_{i \in [t]} \chi_i \cdot A_{0,i}}_{A_0} + \Delta \cdot \underbrace{\sum_{i \in [t]} \chi_i \cdot A_{1,i}}_{A_1}$$

where $\chi_i \leftarrow \mathbb{F}_{p^d}$. Since sending A_0, A_1 directly could leak additional information, we need to use an extra random OLE instance to mask both sides.

Furthermore, unlike previous work that opens $w_h = 1$ in an independent message, we also aggregate this opening into the batch check of multiplications. This is achieved by computing the degree-2 VOLE commitment $[w_h - 1]$

$$\begin{aligned} p_{w_h-1}(Y) &= (w_h - 1) \cdot Y^2 + M[w_h - 1] \cdot Y \\ q_{w_h-1} &= p_{w_h-1}(\Delta) = K[w_h - 1] \cdot \Delta \end{aligned}$$

where $w_h - 1$ should be 0, and then adding it to the aggregated degree-2 VOLE commitment. This not only reduces prover communication by the size of one opening, but also slightly improves the soundness. In this case, the adversary can only cheat in one message while an independent opening gives it another chance. This reduces the soundness error from $4/p^d$ to $3/p^d$. The details are given in Fig. 13. The security is stated in Theorem 7. We defer the detailed proof to Appendix F.1.

Theorem 7. *The protocol $\Pi_{\text{circuitZK}}^{n,t,d}$ given in Fig. 13 SUC-realizes $\mathcal{F}_{\text{circuitZK}}$ in the $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$ -hybrid model with soundness error $3/p^d$ and information-theoretic security.*

Efficiency. The prover sends $n + t$ elements in $\text{GR}(p^k, d_0)$ and 2 elements in $\text{GR}(p^k, d)$. The verifier sends $t + 1$ elements in $\mathbb{F}_{p^d}$. So we have

$$\begin{aligned} \text{Comm}_{\mathcal{P}} &= (2d + nd_0 + td_0) \cdot k \log p \\ \text{Comm}_{\mathcal{V}} &= (t + 1) \cdot d \log p \end{aligned}$$

Table 2 summarizes the communication cost of ZK protocols for circuit satisfiability over $\mathbb{Z}_{2^k}$. Here we only focus on $\mathcal{P}$'s communication because the uniform random challenge of $\mathcal{V}$ can be compressed to a short seed using PRG⁷. When t is much larger than n and λ , our protocol achieves nearly optimal amortized communication for each multiplication gate, as shown in Table 1. For general cases, compared with Moz $\mathbb{Z}_{2^k}$ arella [BBMHS22], our protocol has lower communication when $n + t \geq k$. [LXY24b] amortizes the communication using RMFEs while introducing a factor $\frac{\lambda}{w}$. Assuming $\lambda \approx 4m$, which is generous, our protocol is better in prover communication when $n + t \geq \frac{8}{3}m$. Furthermore, while [LXY24b] relies on SIMD circuits to amortize its communication cost, our protocol achieves optimal communication for a single circuit. Therefore, our approach is particularly advantageous when batching is unavailable or only a small number of instances can be batched.

In terms of computation efficiency, our protocol may be less efficient than Moz $\mathbb{Z}_{2^k}$ arella due to the use of high-degree ring extension. The comparison with [LXY24b] is less clear. While their protocol can amortize the computation over m instances, our protocol avoids modular reductions when computing $A_{1,i} = w_\alpha \cdot M[w_\beta] + w_\beta \cdot M[w_\alpha] - M[w_\gamma]$. We leave the implementation and evaluation of our protocol for future work.

⁷ If we aim at public verifiability, then this disappears when we make it non-interactive.

Complementarity with [LXY24b]. We note that our srVOLE based protocol is complementary with [LXY24b]. For example, when the number of batched instances m is not large enough to achieve the best ratio $\frac{\lambda}{m} \approx 4$, [LXY24b] needs to repeat their protocol several times to achieve a negligible soundness error, which incurs high communication cost. We notice that we can use (m, d_0) -RMFEs for some d_0 such that $d_0 \approx 4m$ and $d_0 \mid \lambda$ (or $\tau \cdot d_0$ is slightly larger than λ for some constant τ). Then we can combine our protocol and their RMFEs based protocol in this way: after embedding the witness $\mathbf{w} \in \mathbb{Z}_{2^k}^{n \times m}$ to $\text{GR}(2^k, d_0)^n$, we run their RMFEs based protocol over $\text{GR}(2^k, \lambda)$ using srVOLE. So we can enjoy both low amortized complexity and negligible soundness error.

Table 2. Asymptotic comparison of communication cost of ZK proofs for circuit satisfiability, where the witness is a vector over $\mathbb{Z}_{2^k}$. λ is the security parameter. $s = \lambda + \log \lambda + 3$. $\mathcal{P}$ denotes the prover cost and $\mathcal{V}$ denotes the verifier cost. $\mathcal{R}$ is the ring over which the computations are carried out. We use (m, λ) -RMFEs for [LXY24b].

Protocol	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{R}$	Soundness error
[BBMHS22]	$(n + t + 4)(k + 2s)$	ts	$\mathbb{Z}_{2^{k+2s}}$	$1/2^{\lambda-1}$
[LXY24b]	$(n + t + 3)k\lambda/m$	$t\lambda/m$	$\text{GR}(2^k, \lambda)$	$1/2^{\lambda-2}$
$\Pi_{\text{circuitZK}}^{n,t,\lambda}$	$(2\lambda + n + t)k$	$(t + 1)\lambda$	$\text{GR}(2^k, \lambda)$	$3/2^\lambda$

5.3 ZKP for Polynomial Satisfiability over Galois Rings

We adapt the ZK protocol for polynomial satisfiability from QuickSilver to Galois rings. The idea is to generalize the multiplication check in Equation (1), which is degree-2, to an arbitrary low degree D . For any degree- D polynomial $f \in \text{GR}(p^k, d_0)[X_1, \dots, X_n]$, we write it in a "degree-separated" form $f = \sum_{h=0}^D f_h$, where all terms in f_h are degree- h . For a witness $\mathbf{w} \in \text{GR}(p^k, d_0)^n$, we compute

$$\begin{aligned}
& \sum_{h=0}^D f_h(K[w_1], \dots, K[w_n]) \cdot \Delta^{D-h} \\
&= \sum_{h=0}^D f_h(M[w_1] + w_1 \cdot \Delta, \dots, M[w_n] + w_n \cdot \Delta) \cdot \Delta^{D-h} \\
&= \sum_{h=0}^D \left(f_h(w_1, \dots, w_n) \cdot \Delta^D + \sum_{j=0}^{h-1} A_h^j \cdot \Delta^{j+D-h} \right) \\
&= \sum_{h=0}^{D-1} A_h \cdot \Delta^h + f(w_1, \dots, w_n) \cdot \Delta^D \\
&= \sum_{h=0}^{D-1} A_h \cdot \Delta^h
\end{aligned} \tag{2}$$

To check this equation, we need a degree- D VOLE commitment that commits to 0 to mask the coefficients, which can be generated using $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$. We give the details of our ZK protocol for polynomial satisfiability in Fig. 14. The detailed proof of Theorem 8 is deferred to Appendix F.2.

Theorem 8. *The protocol $\Pi_{\text{polyZK}}^{n,t,d,D}$ given in Fig. 14 SUC-realizes $\mathcal{F}_{\text{polyZK}}$ in the $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$ -hybrid model with soundness error $(D + 1)/p^d$ and information-theoretic security.*

Efficiency. The prover sends n elements in $\text{GR}(p^k, d_0)$ and D elements in $\text{GR}(p^k, d)$. The verifier sends t elements in $\mathbb{F}_{p^d}$. We have

$$\begin{aligned}
\text{Comm}_{\mathcal{P}} &= (nd_0 + Dd) \cdot k \log p \\
\text{Comm}_{\mathcal{V}} &= td \log p
\end{aligned}$$

Zero-Knowledge Protocol $\Pi_{\text{polyZK}}^{n,t,d,D}$ for Polynomial Satisfiability

Parameters: Ideal functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$ defined in Fig. 12.

Input: Both parties hold t degree- D polynomials $f_1, \dots, f_t \in \text{GR}(p^k, d_0)[X_1, \dots, X_n]$. Each polynomial $f_i = \sum_{h=0}^D f_{i,h}$ where all terms in $f_{i,h}$ have degree h . $\mathcal{P}$ also holds a witness $\mathbf{w} \in \text{GR}(p^k, d_0)^n$ such that $f_i(\mathbf{w}) = 0$ for all $i \in [t]$.

Preprocessing phase:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$. $\mathcal{P}$ receives $\mathbf{u} \in \text{GR}(p^k, d_0)^n$, $M[\mathbf{u}] \in \text{GR}(p^k, d)^n$, while $\mathcal{V}$ gets (done).
2. $\mathcal{V}$ sends (get) to $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$ and receives $\Delta \in \mathbb{F}_{p^d}$, $K[\mathbf{u}] \in \text{GR}(p^k, d)^n$.
3. $\mathcal{P}$ and $\mathcal{V}$ send (VOPE, $D-1$) to $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$, which returns $\{A_h^*\}_{0 \leq h < D}$ to $\mathcal{P}$ and B^* to $\mathcal{V}$, such that $\sum_{h=0}^{D-1} A_h^* \cdot \Delta^h = B^*$.

Online phase:

1. $\mathcal{P}$ computes $\delta := \mathbf{w} - \mathbf{u}$ and sends it to $\mathcal{V}$. Both parties compute $[\mathbf{w}] := [\mathbf{u}] + \delta$.
2. From $i = 1$ to t , for the i -th polynomial f_i , $\mathcal{P}$ and $\mathcal{V}$ perform the following
 - $\mathcal{V}$ computes $B_i := \sum_{h=0}^D f_{i,h}(K[w_1], \dots, K[w_n]) \cdot \Delta^{D-h}$.
 - $\mathcal{P}$ defines a univariate degree- D polynomial over $\text{GR}(p^k, d)$ as $g_i(x) := \sum_{h=0}^D f_{i,h}(M[w_1] + w_1 \cdot x, \dots, M[w_n] + w_n \cdot x) \cdot x^{D-h}$, and computes the coefficients $\{A_{i,h}\}_{0 \leq h \leq D}$ such that $g_i(x) = \sum_{h=0}^D A_{i,h} \cdot x^h$. Since $A_{i,D} = f_i(w_1, \dots, w_n) = 0$, $g_i(x)$ can be written as $\sum_{h=0}^{D-1} A_{i,h} \cdot x^h$.
3. $\mathcal{P}$ and $\mathcal{V}$ perform the following check
 - $\mathcal{V}$ samples $\chi \leftarrow \mathbb{F}_{p^d}^t$ and sends it to $\mathcal{P}$.
 - For all $0 \leq h < D$, $\mathcal{P}$ computes $A_h := \sum_{i \in [t]} \chi_i \cdot A_{i,h} + A_h^*$ and sends it to $\mathcal{V}$.
 - $\mathcal{V}$ computes $B := \sum_{i \in [t]} \chi_i \cdot B_i + B^*$ and checks if $B = \sum_{h=0}^{D-1} A_h \cdot \Delta^h$. $\mathcal{V}$ outputs (false) if the check fails otherwise (true).

Fig. 14. Zero-Knowledge Protocol $\Pi_{\text{polyZK}}^{n,t,d,D}$ for Polynomial Satisfiability over Any Galois Rings

Table 3 gives a comparison between different schemes. Although [LXY24b] does not provide a ZK protocol for polynomial satisfiability, we find that it is easy to extend their technique to general degree-2 polynomials. We compute the communication cost and show it in Table 3.

Similar to the analysis in Sect. 5.2, we only care about prover communication. Our protocol is better than Moz $\mathbb{Z}_{2^k}$ arella when $n+2 \geq k$. Assuming $\lambda \approx 4m$, our protocol is better than [LXY24b] when $n \geq \frac{8}{3}(m-1)$. Therefore, our protocol is more efficient when the number of variables n is large. We also stress that our protocol supports arbitrary low-degree polynomials, while [BBMHS22, LXY24b] only support degree-2 polynomials.

Table 3. Asymptotic comparison of communication cost of ZK proofs for degree-2 polynomials, where the witness is a vector over $\mathbb{Z}_{2^k}$. λ is the security parameter. $s = \lambda + \log \lambda + 3$. $\mathcal{P}$ denotes the prover cost and $\mathcal{V}$ denotes the verifier cost. $\mathcal{R}$ is the ring over which the computations are carried out. We use (m, λ) -RMFEs for [LXY24b].

Protocol	$\mathcal{P}$	$\mathcal{V}$	$\mathcal{R}$	Soundness error
[BBMHS22]	$(n+2)(k+2s)$	ts	$\mathbb{Z}_{2^{k+2s}}$	$1/2^{\lambda-1}$
[LXY24b]	$(n+2)k\lambda/m$	$t\lambda/m$	$\text{GR}(2^k, \lambda)$	$3/2^\lambda$
$\Pi_{\text{polyZK}}^{n,t,\lambda,2}$	$(n+2\lambda)k$	$t\lambda$	$\text{GR}(2^k, \lambda)$	$3/2^\lambda$

Acknowledgements. The authors would like to thank the anonymous reviewers of ASIACRYPT2026 for their helpful comments. This work was supported by the Grant Excelencia María de Maeztu CEX2024-001471-M funded by MICIU/AEI/10.13039/501100011033.

References

- ABB⁺26. Carlos Aguilar Melchor, Slim Bettaleb, Loïc Bidoux, Thibault Feneuil, Philippe Gaborit, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovahery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH — Syndrome Decoding in the Head. Technical report, National Institute of Standards and Technology, 2026. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-3-additional-signatures>.
- ABBS25. Amit Agarwal, Carsten Baum, Lennart Braun, and Peter Scholl. Low-bandwidth mixed arithmetic in VOLE-based ZK from low-degree PRGs. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part IV*, volume 15604 of *LNCS*, pages 396–426. Springer, Cham, May 2025.
- AMR25. Damiano Abram, Giulio Malavolta, and Lawrence Roy. Succinct oblivious tensor evaluation and applications: Adaptively-secure laconic function evaluation and trapdoor hashing for all circuits. In Michal Koucký and Nikhil Bansal, editors, *57th ACM STOC*, pages 1875–1886. ACM Press, June 2025.
- ARS24. Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 301–330. Springer, Cham, May 2024.
- AvBH⁺26. Diego F. Aranha, Aron van Baarsen, Adam Blatchley Hansen, Kent Nielsen, and Peter Scholl. A maliciously-secure post-quantum OPRF from crypto dark matter. Cryptology ePrint Archive, Paper 2026/533, 2026.
- BBB⁺25. Carsten Baum, Ward Beullens, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Kloof, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. Shorter, tighter, FAESTer: Optimizations and improved (QROM) analysis for VOLE-in-the-head signatures. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 124–156. Springer, Cham, August 2025.
- BBB⁺26. Carsten Baum, Lennart Braun, Ward Beullens, Cyprien Delpech de Saint Guilhem, Michael Kloof, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. Technical report, National Institute of Standards and Technology, 2026. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-3-additional-signatures>.
- BBD⁺23. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Kloof, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Cham, August 2023.
- BBM⁺24. Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part I*, volume 15484 of *LNCS*, pages 463–493. Springer, Singapore, December 2024.
- BBMH⁺21. Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoît Razet, and Peter Scholl. Apzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and Z2k. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 192–211. ACM Press, November 2021.
- BBMHS22. Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz $\mathbb{Z}_{2^k}$ arella: Efficient vector-ole and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 329–358, Cham, 2022. Springer Nature Switzerland.
- BC25. Dung Bui and Kelong Cong. Efficient fuzzy labeled PSI from vector ring-OLE. In Yongdae Kim, Atsuko Miyaji, and Mehdi Tibouchi, editors, *CANS 25*, volume 16351 of *LNCS*, pages 173–198. Springer, Singapore, November 2025.
- BCC⁺25. Dung Bui, Haotian Chu, Geoffroy Couteau, Xiao Wang, Chenkai Weng, Kang Yang, and Yu Yu. An efficient ZK compiler from SIMD circuits to general circuits. *Journal of Cryptology*, 38(1):10, January 2025.
- BCG⁺19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In

- Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.
- BCG⁺19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO~2019, Part~III*, volume 11694 of *LNCS*, pages 489–518. Springer, Cham, August 2019.
- BCG⁺20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, pages 1069–1080. IEEE Computer Society Press, November 2020.
- BCG⁺22. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO~2022, Part~II*, volume 13508 of *LNCS*, pages 603–633. Springer, Cham, August 2022.
- BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.
- BCM⁺25. Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi. Fast pseudorandom correlation functions from sparse LPN. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT~2025, Part~VII*, volume 16251 of *LNCS*, pages 439–473. Springer, Singapore, December 2025.
- BDFH25. Ward Beullens, Lucas Dodgson, Sebastian H. Faller, and Julia Hesse. The 2Hash OPRF framework and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT~2025, Part~VIII*, volume 15608 of *LNCS*, pages 332–362. Springer, Cham, May 2025.
- BDJ⁺23. Lennart Braun, Cyprien Delpéch de Saint Guilhem, Robin Jadoul, Emmanuela Orsini, Nigel P. Smart, and Titouan Tanguy. ZK-for-Z2K: MPC-in-the-head zero-knowledge proofs for $\mathbb{Z}_2^k$. In Elizabeth A. Quaglia, editor, *19th IMA International Conference on Cryptography and Coding*, volume 14421 of *LNCS*, pages 137–157. Springer, Cham, December 2023.
- BDSW23. Carsten Baum, Samuel Dittmer, Peter Scholl, and Xiao Wang. Sok: vector OLE-based zero-knowledge protocols. *DCC*, 91(11):3527–3561, 2023.
- BMRS21. Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac’n’cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *CRYPTO~2021, Part~IV*, volume 12828 of *LNCS*, pages 92–122, Virtual Event, August 2021. Springer, Cham.
- CCL15. Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO~2015, Part~II*, volume 9216 of *LNCS*, pages 3–22. Springer, Berlin, Heidelberg, August 2015.
- CCXY18. Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. Amortized complexity of information-theoretically secure MPC revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO~2018, Part~III*, volume 10993 of *LNCS*, pages 395–426. Springer, Cham, August 2018.
- CD23. Geoffroy Couteau and Clément Ducros. Pseudorandom correlation functions from variable-density LPN, revisited. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC~2023, Part~II*, volume 13941 of *LNCS*, pages 221–250. Springer, Cham, May 2023.
- CDE⁺18. Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD $\mathbb{Z}_2^k$: Efficient MPC mod 2^k for dishonest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO~2018, Part~II*, volume 10992 of *LNCS*, pages 769–798. Springer, Cham, August 2018.
- CRR21. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO~2021, Part~III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Cham.
- CRX21. Ronald Cramer, Matthieu Rambaud, and Chaoping Xing. Asymptotically-good arithmetic secret sharing over $\mathbb{Z}/p^\ell\mathbb{Z}$ with strong multiplication and its applications to efficient MPC. In Tal Malkin and Chris Peikert, editors, *CRYPTO~2021, Part~III*, volume 12827 of *LNCS*, pages 656–686, Virtual Event, August 2021. Springer, Cham.
- DILO22. Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 829–841. ACM Press, November 2022.
- DIO21. Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In Stefano Tessaro, editor, *ITC 2021*, volume 199 of *LIPIcs*, pages 5:1–5:24. Schloss Dagstuhl, July 2021.

- EXY22. Daniel Escudero, Chaoping Xing, and Chen Yuan. More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via galois rings. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 383–412, Cham, 2022. Springer Nature Switzerland.
- GHAK24. Aarushi Goel, Mathias Hall-Andersen, and Gabriel Kaptchuk. Dora: A simple approach to zero-knowledge for RAM programs. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 869–883. ACM Press, October 2024.
- GLH⁺26. Xiaojie Guo, Hanlin Liu, Zhicong Huang, Hongrui Cui, Wenhao Zhang, Cheng Hong, Xiao Wang, Kang Yang, and Yu Yu. Dory: Streaming PCG with Small Memory. In *2026 IEEE Symposium on Security and Privacy (SP)*, pages 1466–1484, Los Alamitos, CA, USA, May 2026. IEEE Computer Society.
- GNS23. Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.
- HHK⁺24. Carmit Hazay, David Heath, Vladimir Kolesnikov, Muthuramakrishnan Venkitasubramaniam, and Yibin Yang. LogRobin++: Optimizing proofs of disjunctive statements in VOLE-based ZK. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT~2024, Part~V*, volume 15488 of *LNCS*, pages 367–401. Springer, Singapore, December 2024.
- JLX⁺25. Yuhao Jia, Songsong Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Polynomial commitments for galois rings and applications to SNARKs over $\mathbb{Z}_{2^k}$. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO~2025, Part~VI*, volume 16005 of *LNCS*, pages 515–548. Springer, Cham, August 2025.
- KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.
- KPR⁺25. Vladimir Kolesnikov, Stanislav Peceny, Rahul Rachuri, Srinivasan Raghuraman, Peter Rindal, and Harshal Shah. Block-accumulate codes: Accelerated linear codes for PCGs and ZK. Cryptology ePrint Archive, Report 2025/1828, 2025.
- KPRR25. Vladimir Kolesnikov, Stanislav Peceny, Srinivasan Raghuraman, and Peter Rindal. Stationary syndrome decoding for improved PCGs. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO~2025, Part~I*, volume 16000 of *LNCS*, pages 284–317. Springer, Cham, August 2025.
- LHLA25. Mengling Liu, Yang Heng, Xingye Lu, and Man Ho Au. JesseQ: Efficient zero-knowledge proofs for circuits over any field. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 3402–3420. IEEE Computer Society Press, May 2025.
- LWYY24. Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part VI*, Lecture Notes in Computer Science, pages 149–179. Springer, 2024.
- LXY24a. Fuchun Lin, Chaoping Xing, and Yizhou Yao. Interactive line-point zero-knowledge with sub-linear communication and linear computation. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT~2024, Part~V*, volume 15488 of *LNCS*, pages 337–366. Springer, Singapore, December 2024.
- LXY24b. Fuchun Lin, Chaoping Xing, and Yizhou Yao. More efficient zero-knowledge protocols over $\mathbb{Z}_{2^k}$ via galois rings. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 424–457, Cham, 2024. Springer Nature Switzerland.
- LXY⁺25. Zhe Li, Chaoping Xing, Yizhou Yao, Chen Yuan, and Mengmeng Zhou. Succinct line-point zero-knowledge arguments from homomorphic secret sharing. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT~2025, Part~V*, volume 16249 of *LNCS*, pages 578–609. Springer, Singapore, December 2025.
- NRS26. Lucien K. L. Ng, Peter Rindal, and Akash Shah. LogVOLE: Succinct and efficient chosen-input VOLE for ZK and beyond. Cryptology ePrint Archive, Paper 2026/925, 2026.
- OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT~2021, Part~I*, volume 12696 of *LNCS*, pages 678–708. Springer, Cham, October 2021.
- Roy22. Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO~2022, Part~I*, volume 13507 of *LNCS*, pages 657–687. Springer, Cham, August 2022.
- RR22. Srinivasan Raghuraman and Peter Rindal. Blazing fast PSI from improved OKVS and subfield VOLE. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2505–2517. ACM Press, November 2022.

- RRT23. Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudo-random correlation generators from LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 602–632. Springer, Cham, August 2023.
- RS21. Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Cham, October 2021.
- SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1055–1072. ACM Press, November 2019.
- vP25. Aron van Baarsen and Sihang Pu. Fuzzy private set intersection from VOLE. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 327–360. Springer, Singapore, December 2025.
- Wan03. Z. Wan. *Lectures On Finite Fields And Galois Rings*. World Scientific Publishing Company, 2003.
- WYKW21. Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE Computer Society Press, May 2021.
- WYX⁺21. Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 501–518. USENIX Association, August 2021.
- WYY⁺22. Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. AntMan: Interactive zero-knowledge proofs with sublinear communication. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 2901–2914. ACM Press, November 2022.
- Yan25. Yibin Yang. Justvengers: Batched VOLE ZK disjunctions in $\mathcal{O}(R + B + C)$ communication. In Benny Applebaum and Huijia (Rachel) Lin, editors, *TCC 2025, Part IV*, volume 16271 of *LNCS*, pages 105–137. Springer, Cham, December 2025.
- YBH⁺25. Yibin Yang, Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Gold OPRF: Post-quantum oblivious power-residue PRF. In Marina Blanton, William Enck, and Cristina Nita-Rotaru, editors, *2025 IEEE Symposium on Security and Privacy*, pages 259–278. IEEE Computer Society Press, May 2025.
- YHH⁺23. Yibin Yang, David Heath, Carmit Hazay, Vladimir Kolesnikov, and Muthuramakrishnan Venkatasubramanian. Batchman and robin: Batched and non-batched branching for interactive ZK. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1452–1466. ACM Press, November 2023.
- YHH⁺24. Yibin Yang, David Heath, Carmit Hazay, Vladimir Kolesnikov, and Muthuramakrishnan Venkatasubramanian. Tight ZK CPU: Batched ZK branching with cost proportional to evaluated instruction. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3095–3109. ACM Press, October 2024.
- YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.
- YWL⁺20. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020.

A More Ideal Functionalities

The ideal functionality of equality test is given in Fig. 15. We note that it can be realized using a UC commitment as noted by [WYKW21]. The OT functionality is given in Fig. 16.

Functionality $\mathcal{F}_{\text{EQ}}$

Upon receiving $V_{\mathcal{P}}$ from $\mathcal{P}$ and $V_{\mathcal{V}}$ from $\mathcal{V}$, do:

1. Send $(V_{\mathcal{P}} \stackrel{?}{=} V_{\mathcal{V}})$ and $V_{\mathcal{P}}$ to $\mathcal{V}$.
2. If $\mathcal{V}$ is honest and $V_{\mathcal{P}} = V_{\mathcal{V}}$, or is corrupted and sends (**continue**), then send $(V_{\mathcal{P}} \stackrel{?}{=} V_{\mathcal{V}})$ to $\mathcal{P}$.
3. If $\mathcal{V}$ is honest and $V_{\mathcal{P}} \neq V_{\mathcal{V}}$, or is corrupted and sends (**abort**), then send (**abort**) to $\mathcal{P}$.

Fig. 15. Ideal Functionality of $\mathcal{F}_{\text{EQ}}$ Borrowed from [WYKW21]

Functionality $\mathcal{F}_{\text{OT}}$

Upon receiving m_0, m_1 from $\mathcal{P}_S$ and $b \in \{0, 1\}$ from $\mathcal{P}_R$, send m_b to $\mathcal{P}_R$.

Fig. 16. Ideal Functionality of OT

B LPN over Galois Rings

B.1 LPN Noise Distribution

There are three kinds of noise distributions that are commonly used.

Bernoulli. Let $\text{Ber}(\mathcal{R}) := \{\text{Ber}_{\mu, n_c}(\mathcal{R})\}_{\mu, n_c}$ be the family of Bernoulli distributions. If a noise vector $\mathbf{e}$ is sampled from $\text{Ber}_{\mu, n_c}(\mathcal{R})$, then $\mathbf{e} \in \mathcal{R}^{n_c}$ and each component of $\mathbf{e}$ is a uniform element in $\mathcal{R}$ with probability μ and 0 otherwise.

Exact. Let $\text{HW}(\mathcal{R}) := \{\text{HW}_{t, n_c}(\mathcal{R})\}_{t, n_c}$ be the family of exact noise distributions. If a noise vector $\mathbf{e}$ is sampled from $\text{HW}_{t, n_c}(\mathcal{R})$, then $\mathbf{e} \in \mathcal{R}^{n_c}$ and it has exactly t uniform non-zero components. The positions of non-zero components of $\mathbf{e}$ are also uniformly random.

Regular. Let $\text{RHW}(\mathcal{R}) := \{\text{RHW}_{t, n_c}(\mathcal{R})\}_{t, n_c}$ be the family of regular noise distributions. Regular noises are a special case of exact noises, which additionally require that the noise vector can be divided into t consecutive unit vectors of size $\lfloor n_c/t \rfloor$.

In the case of Galois rings, we further require all non-zero components of noise vector to be invertible when using exact noise or regular noise, which is a countermeasure used by [BBMHS22, LXY24b] to mitigate the reduction attack in [LWYY24]. We denote these special noise distributions by $\text{HW}^*, \text{RHW}^*$.

B.2 Reductions for LPN over $\text{GR}(p^k, d)$

We adapt the reductions over integer rings in [LWYY24] and reductions over $\text{GR}(2^k, d)$ in [LXY24b] to $\text{GR}(p^k, d)$.

Theorem 9. *If the decisional $(\mathcal{D}_1, \text{GR}(p^k, d))\text{-LPN}(k_c, n_c, t)$ problem is (T, ε) -hard, then the decisional $(\mathcal{D}_2, \mathbb{F}_{p^d})\text{-LPN}(k_c, n_c, t)$ problem is $(T - \text{poly}(k_c, n_c), \varepsilon)$ -hard, where $(\mathcal{D}_1, \mathcal{D}_2) \in \{(\text{Ber}, \text{Ber}), (\text{HW}^*, \text{HW}), (\text{RHW}^*, \text{RHW})\}$.*

Proof. Given LPN samples $(\mathbf{A}, \mathbf{b} = \mathbf{s} \cdot \mathbf{A} + \mathbf{e})$ over $\text{GR}(p^k, d)$, we observe that $(\mathbf{A}' = \mathbf{A} \bmod p, \mathbf{b}' = \mathbf{b} \bmod p)$ constitute exactly the LPN samples over $\mathbb{F}_{p^d}$ for noise $\mathbf{e}' = \mathbf{e} \bmod p$. When $\mathbf{e} \leftarrow \text{Ber}_{\mu, n_c}(\text{GR}(p^k, d))$, $\mathbf{e}' \sim \text{Ber}_{\mu, n_c}(\mathbb{F}_{p^d})$. Furthermore, for a uniform vector $\mathbf{u} \in \text{GR}(p^k, d)^{n_c}$, $\mathbf{u}' = \mathbf{u} \bmod p$ is uniform over $\mathbb{F}_{p^d}^{n_c}$. Therefore, one can use a $(\text{Ber}, \mathcal{G}, \mathbb{F}_{p^d})$ -LPN(k_c, n_c, t) distinguisher to distinguish $(\mathbf{A}, \mathbf{b})$ from uniform samples. The proofs for other cases are very similar. We observe that for $\mathbf{e} \leftarrow \text{HW}_{t, n_c}^*(\text{GR}(p^k, d))$, $\mathbf{e}' \sim \text{HW}_{t, n_c}(\mathbb{F}_{p^d})$, which is also the same for the RHW* case. $\square$

Theorem 10. *If decisional $(\text{Ber}, \mathbb{F}_{p^d})$ -LPN($k_c, n_c, \frac{\mu}{(1-\mu)p^{(k-1)d+\mu}}$) problem is (T, ε) -hard, then decisional $(\text{Ber}, \text{GR}(p^k, d))$ -LPN(k_c, n_c, μ) is $(T - \text{poly}(k_c, n_c), k \cdot \varepsilon)$ -hard.*

Proof. Let $(\mathbf{A}, \mathbf{b} = \mathbf{s} \cdot \mathbf{A} + \mathbf{e})$ be LPN samples over $\text{GR}(p^k, d)$. Let

$$\begin{aligned} (\mathbf{A}^{(0)}, \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(k-1)}) &:= \text{Decomp}(\mathbf{A}) \\ (\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \dots, \mathbf{s}^{(k-1)}) &:= \text{Decomp}(\mathbf{s}) \\ (\mathbf{e}^{(0)}, \mathbf{e}^{(1)}, \dots, \mathbf{e}^{(k-1)}) &:= \text{Decomp}(\mathbf{e}) \\ (\mathbf{b}^{(0)}, \mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k-1)}) &:= \text{Decomp}(\mathbf{b}) \end{aligned}$$

Then we have $\mathbf{b}^{(0)} = \mathbf{s}^{(0)} \cdot \mathbf{A}^{(0)} + \mathbf{e}^{(0)} \bmod p$, and for $i \in [0, k)$:

$$\mathbf{b}^{(i)} = \mathbf{s}^{(i)} \cdot \mathbf{A}^{(0)} + \mathbf{e}^{(i)} + f_i(\mathbf{A}, \mathbf{S}(0, i-1), \mathbf{E}(0, i-1)) \bmod p$$

where $\mathbf{S}(0, i-1) := (\mathbf{s}^{(0)}, \dots, \mathbf{s}^{(i-1)})$, $\mathbf{E}(0, i-1) := (\mathbf{e}^{(0)}, \dots, \mathbf{e}^{(i-1)})$ and f_i sums up all other terms not depending on $\mathbf{s}^{(i)}$ and $\mathbf{e}^{(i)}$. Define the hybrid distributions:

$$\begin{aligned} H_0 &= (\mathbf{A}, \mathbf{u}_0, \dots, \mathbf{u}_{i-1}, \mathbf{u}_i, \dots, \mathbf{u}_{k-1}) \\ &\vdots \\ H_i &= (\mathbf{A}, \mathbf{b}^{(0)}, \dots, \mathbf{b}^{(i-1)}, \mathbf{u}_i, \dots, \mathbf{u}_{k-1}) \\ &\vdots \\ H_k &= (\mathbf{A}, \mathbf{b}^{(0)}, \dots, \mathbf{b}^{(i-1)}, \mathbf{b}^{(i)}, \dots, \mathbf{b}^{(k-1)}) \end{aligned}$$

where $\mathbf{u}_i \leftarrow \mathbb{F}_{p^d}^{n_c}$ for $i \in [0, k)$. Note that all $\mathbf{s}^{(i)}$'s are independent and uniform since $\mathbf{s}$ is uniform over $\text{GR}(p^k, d)$. Therefore, by the decisional $(\text{Ber}, \mathbb{F}_{p^d})$ -LPN assumption, we have the following indistinguishability for all $i \in [0, k)$

$$(\mathbf{A}^{(0)}, \mathbf{u}_i, \mathbf{S}(0, i-1), \mathbf{E}(0, i-1)) \approx_c (\mathbf{A}^{(0)}, \mathbf{s}^{(i)} \cdot \mathbf{A}^{(0)} + \mathbf{e}^{(i)}, \mathbf{S}(0, i-1), \mathbf{E}(0, i-1))$$

where $\mathbf{S}(0, i-1)$ is independent of any other variables, and the noise rate of $\mathbf{e}^{(i)}$ is conditioned on $\mathbf{E}(0, i-1)$. This implies

$$\begin{aligned} &(\mathbf{A}, \mathbf{b}^{(0)}, \dots, \mathbf{b}^{(i-1)}, \mathbf{u}_i + f_i(\mathbf{A}, \mathbf{S}(0, i-1), \mathbf{E}(0, i-1)) \bmod p) \\ &\approx_c (\mathbf{A}, \mathbf{b}^{(0)}, \dots, \mathbf{b}^{(i-1)}, \mathbf{b}^{(i)}) \end{aligned}$$

because $\mathbf{b}^{(0)}, \dots, \mathbf{b}^{(i-1)}, f_i(\mathbf{A}, \mathbf{S}(0, i-1), \mathbf{E}(0, i-1))$ can be efficiently computed given $\mathbf{A}, \mathbf{S}(0, i-1), \mathbf{E}(0, i-1)$. This in turn implies $H_i \approx_c H_{i+1}$.

Therefore, if $H_i \approx_c H_{i+1}$ holds except with probability ε for all $i \in [0, k-1]$, then $H_0 \approx_c H_k$ holds except with probability $k \cdot \varepsilon$. So it remains to estimate the noise rate needed by the LPN assumption. Consider a single noise sample $(e_j^{(0)}, e_j^{(1)}, \dots, e_j^{(k-1)})$. If $(e_j^{(0)}, e_j^{(1)}, \dots, e_j^{(i-1)}) \neq \mathbf{0}$ then $e_j^{(i)}$ is uniformly random and thus perfectly masks $b_j^{(i)}$. Otherwise, we have

$$\Pr[e_j^{(i)} \neq 0 | (e_j^{(0)}, e_j^{(1)}, \dots, e_j^{(i-1)}) = \mathbf{0}] = \frac{\mu \cdot (p^d - 1)}{(1 - \mu)p^{(i+1)d} + \mu \cdot p^d}$$

is the noise rate needed to keep the indistinguishability between H_i and H_{i+1} , which reaches its minimum when $i = k-1$. Therefore, we have $\mu' = \frac{p^d}{p^d - 1} \cdot \frac{\mu(p^d - 1)}{(1 - \mu)p^{kd} + \mu p^d} = \frac{\mu}{(1 - \mu)p^{(k-1)d} + \mu}$. $\square$

B.3 LPN with Static Leakage

Here we only give the definition for regular noise with invertible non-zero entries, which is the noise distribution we use in our construction.

Definition 5 ([BCG⁺19a]). Let $\mathcal{G}$ be a PPT algorithm such that $\mathcal{G}(k_c, n_c, \text{GR}(p^k, d))$ outputs a generator matrix $\mathbf{A} \in \text{GR}(p^k, d)^{k_c \times n_c}$. Let k_c, n_c, t be implicit functions of security parameter λ . We say that the decisional $(\text{RHW}^*, \mathcal{G}, \text{GR}(p^k, d))$ -LPN(k_c, n_c, t) assumption holds if for all PPT algorithms $\mathcal{A}$, we have

$$\left| \Pr[\text{leaky-LPN}_{\mathcal{A}}^{\mathcal{G}}(\lambda) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$$

where the experiment $\text{leaky-LPN}_{\mathcal{A}}^{\mathcal{G}}(\lambda)$ is defined as follows

1. Sample $\mathbf{A} \leftarrow \mathcal{G}(k_c, n_c, t)$, $\mathbf{s} \leftarrow \text{GR}(p^k, d)^{k_c}$, $\mathbf{e} \leftarrow \text{RHW}_{t, n_c}^*(\text{GR}(p^k, d))$. Let $\alpha_1, \dots, \alpha_t \in [n_c/t]$ be the indices of non-zero entries in $\mathbf{e}$.
2. Receive $I_1, \dots, I_t \subseteq [n_c/t]$ from $\mathcal{A}$. If $\alpha_i \in I_i$ for all $i \in [t]$, then send (success) to $\mathcal{A}$. Otherwise, abort the experiment.
3. Sample $b \leftarrow \{0, 1\}$. If $b = 0$, let $\mathbf{x} := \mathbf{s} \cdot \mathbf{A} + \mathbf{e}$. Otherwise, sample $\mathbf{x} \leftarrow \text{GR}(p^k, d)^{n_c}$. Send $\mathbf{x}$ to $\mathcal{A}$ and receive its output bit b' .
4. Output 1 if $b = b'$ otherwise 0.

Algorithm GGM

Parameters: Let $\mathbf{G} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{2\lambda}$ and $\mathbf{G}' : \{0, 1\}^\lambda \rightarrow \text{GR}(p^k, d_1)^2$ be two PRGs. For $\alpha - 1 = \sum_{i \in [h]} 2^{h-i} \alpha_i$, for $\alpha_i \in \{0, 1\}$. Let $\bar{\alpha}_i$ denote the complement of α_i . The GGM algorithms (GGM.KeyGen , GGM.Gen , GGM.PuncEval) are defined as follows:

- $\text{GGM.KeyGen}(1^\lambda)$: output $k^0 \leftarrow \{0, 1\}^\lambda$.
- $\text{GGM.Gen}(\ell, k^0)$: where $\ell = 2^h$, $k^0 \in \{0, 1\}^\lambda$
 - (a) Set $k_0^0 := k^0$.
 - (b) For $i \in [h-1]$ and $j \in [0, 2^{i-1})$, compute $(k_{2j}^i, k_{2j+1}^i) := \mathbf{G}(k_j^{i-1})$.
 - (c) For $j \in [0, 2^{h-1})$, compute $(q_{2j+1}, q_{2j+2}) := \mathbf{G}'(k_j^{h-1})$.
 - (d) For $i \in [h-1]$, compute $K_0^i := \bigoplus_{0 \leq j < 2^{i-1}} k_{2j}^i$ and $K_1^i := \bigoplus_{0 \leq j < 2^{i-1}} k_{2j+1}^i$.
 - (e) Compute $K_0^h := \sum_{0 \leq j < 2^{h-1}} q_{2j+1}$ and $K_1^h := \sum_{0 \leq j < 2^{h-1}} q_{2j+2}$.
 - (f) Output $(\{q_j\}_{j \in [\ell]}, \{(K_0^i, K_1^i)\}_{i \in [h]})$.
- $\text{GGM.PuncEval}(\ell, \alpha, \{K_i\}_{i \in [h]})$: where $\ell = 2^h$, $\alpha \in [\ell]$, $K_i \in \{0, 1\}^\lambda$ for $i \in [h-1]$ and $K_h \in \text{GR}(p^k, d_1)$
 - (a) Set $K_{\bar{\alpha}_1}^1 := K_1$ and $x := 0$.
 - (b) For each $i \in [h-2]$:
 - (i) Update $x := 2x + \alpha_i$.
 - (ii) For $j \in [0, 2^i) \setminus \{x\}$, compute $(K_{2j}^{i+1}, K_{2j+1}^{i+1}) := \mathbf{G}(K_j^i)$.
 - (iii) Compute $K_{2x+\bar{\alpha}_{i+1}}^{i+1} := K_{i+1} \oplus \bigoplus_{j \in [0, 2^i) \setminus \{x\}} K_{2j+\bar{\alpha}_{i+1}}^{i+1}$.
 - (c) Update $x := 2x + \alpha_{h-1}$. For $j \in [0, 2^{h-1}) \setminus \{x\}$, compute $(q_{2j+1}, q_{2j+2}) := \mathbf{G}'(K_j^{h-1})$. Compute $q_{2x+\bar{\alpha}_h+1} := K_h - \sum_{j \in [0, 2^{h-1}) \setminus \{x\}} q_{2j+\bar{\alpha}_h+1}$.
 - (d) Output $\{q_j\}_{j \in [\ell] \setminus \{\alpha\}}$.

Fig. 17. Algorithm GGM

C GGM Algorithm

The details of the GGM algorithms are given in Fig. 17.

Functionality $\mathcal{F}_{\text{VOLE2k}}^{\sigma,s,\ell,n,\mathcal{L}}$

Parameters: $\sigma \geq s$.

Init: Upon receiving (init) from $\mathcal{P}$ and $\mathcal{V}$:

1. If $\mathcal{P}$ is honest, sample $\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_{2^\sigma}^{\ell \times n}$. Otherwise receive $\mathbf{U}, \mathbf{V} \in \mathbb{Z}_{2^\sigma}^{\ell \times n}$ and $L \in \mathcal{L}$ from $\mathcal{A}$.
2. If $\mathcal{V}$ is honest, sample $\Delta \leftarrow \mathbb{Z}_{2^s}^n$ and compute $\mathbf{Q} := \mathbf{V} + \mathbf{U} \cdot \text{diag}(\Delta)$. Otherwise receive $\Delta \in \mathbb{Z}_{2^s}^n, \mathbf{Q} \in \mathbb{Z}_{2^\sigma}^{\ell \times n}$ from $\mathcal{A}$ and recompute $\mathbf{V} := \mathbf{Q} - \mathbf{U} \cdot \text{diag}(\Delta)$.
3. Send $(\mathbf{U}, \mathbf{V})$ to $\mathcal{P}$ and ignore any subsequent (init) commands.

Get: Upon receiving (get) from $\mathcal{V}$, if $\mathcal{P}$ is corrupted and $\Delta \notin L$, send (check failed) to $\mathcal{V}$ and abort. Otherwise send $(\Delta, \mathbf{Q})$ to $\mathcal{V}$.

Fig. 18. Ideal Functionality of VOLE over $\mathbb{Z}_{2^\sigma}$

VOLE Protocol $\Pi_{\text{small-VOLE2k}}^{\sigma,s,\ell,1,\{\mathbb{Z}_{2^s}\}}$ for Small Rings

Parameters:

- $\text{PRG} : \{0, 1\}^\lambda \rightarrow \mathbb{Z}_{2^\sigma}^\ell, N = 2^s, \sigma \geq s$.
- Ideal functionality $\mathcal{F}_{\text{ROT-1}}^N$ defined in Fig. 3.

Init:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\text{ROT-1}}^N$.
2. $\mathcal{P}$ receives $\{s_i\}_{i \in \mathbb{Z}_N}$ from $\mathcal{F}_{\text{ROT-1}}^N$, where $s_i \in \{0, 1\}^\lambda$.
3. For all $i \in \mathbb{Z}_N$, $\mathcal{P}$ computes $\mathbf{t}_i := \text{PRG}(s_i)$.
4. $\mathcal{P}$ computes $\mathbf{u} := \sum_{i \in \mathbb{Z}_N} \mathbf{t}_i \bmod 2^\sigma$, $\mathbf{v} := -\sum_{i \in \mathbb{Z}_N} \mathbf{t}_i \cdot i \bmod 2^\sigma$ and outputs $(\mathbf{u}, \mathbf{v})$.

Get:

1. $\mathcal{V}$ sends (get) to $\mathcal{F}_{\text{ROT-1}}^N$ and receives $\Delta \in \mathbb{Z}_N, \{s_i\}_{i \in \mathbb{Z}_N \setminus \{\Delta\}}$.
2. For all $i \in \mathbb{Z}_N \setminus \{\Delta\}$, $\mathcal{V}$ computes $\mathbf{t}_i := \text{PRG}(s_i)$.
3. $\mathcal{V}$ computes $\mathbf{q} := \sum_{i \in \mathbb{Z}_N \setminus \{\Delta\}} \mathbf{t}_i \cdot (\Delta - i) \bmod 2^\sigma$.
4. $\mathcal{V}$ outputs $(\Delta, \mathbf{q})$.

Fig. 19. VOLE Protocol $\Pi_{\text{small-VOLE2k}}^{\sigma,s,\ell,1,\{\mathbb{Z}_{2^s}\}}$ for Small Rings

D Detailed VOLE Protocols for Moz $\mathbb{Z}_{2^k}$ arella

The ideal functionality of VOLE over $\mathbb{Z}_{2^k}$ is given in Fig. 18. We first give a protocol for small s in Fig. 19.

Theorem 11. *The protocol $\Pi_{\text{small-VOLE}2k}^{\sigma,s,\ell,1,\{\mathbb{Z}_{2^s}\}}$ given in Fig. 19 SUC-realizes $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s,\ell,1,\{\mathbb{Z}_{2^s}\}}$ in the $\mathcal{F}_{\text{ROT-1}}^{2^s}$ -hybrid model, with both semi-honest and malicious security.*

Proof. The proof is basically the same as the proof of Theorem 3. We just need to see $\{1\} \times \mathbb{Z}_{2^s}$ contains a free $\mathbb{Z}_{2^\sigma}$ -basis of $\mathbb{Z}_{2^\sigma} \times \mathbb{Z}_{2^s}$: $(1,0), (1,1)$.

Remark 5. Similarly to Remark 3, the functionality $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s,\ell,n,\{\mathbb{Z}_{2^s}^n\}}$ can be SUC-realized by n parallel executions of $\Pi_{\text{small-VOLE}2k}^{\sigma,s,\ell,1,\{\mathbb{Z}_{2^s}\}}$.

As discussed in Sect. 3.4, we use a $2^{s'}$ -ary decomposition to avoid exponential computation. The details are given in Fig. 20.

VOLE Protocol $\Pi_{\text{large-VOLE}2k}^{\sigma,s,\ell,1,\mathcal{L}}$ for Large Rings

Parameters:

- $s = \tau \cdot s'$. $\mathcal{H} \subseteq \mathbb{Z}_{2^\sigma}^{r \times (\ell+h)}$ is a family of $\mathbb{Z}_{2^\sigma}^\ell$ -hiding, linear ε -almost universal hash functions.
- Ideal functionality $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s',\ell+h,\tau,\{\mathbb{Z}_{2^{s'}}^\tau\}}$ defined in Fig. 18.

Init:

1. $\mathcal{P}$ and $\mathcal{V}$ send (init) to $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s',\ell+h,\tau,\{\mathbb{Z}_{2^{s'}}^\tau\}}$.
2. $\mathcal{P}$ receives $\mathbf{U}, \mathbf{V} \in \mathbb{Z}_{2^\sigma}^{(\ell+h) \times \tau}$ from $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s',\ell+h,\tau,\{\mathbb{Z}_{2^{s'}}^\tau\}}$.
3. Let $\mathbf{u} := \mathbf{U}^1$. $\mathcal{P}$ computes $[0 \ \mathbf{C}] := \mathbf{U} - \mathbf{u} \cdot \mathbf{1}_\tau$. $\mathcal{P}$ sends $\mathbf{C} \in \mathbb{Z}_{2^\sigma}^{(\ell+h) \times (\tau-1)}$.
4. $\mathcal{V}$ samples and sends a uniformly random hash function $\mathbf{H} \leftarrow \mathcal{H}$.
5. $\mathcal{P}$ computes and sends $\tilde{\mathbf{u}} := \mathbf{H}\mathbf{u}, \tilde{\mathbf{V}} := \mathbf{H}\mathbf{V}$.
6. $\mathcal{P}$ computes $\mathbf{v} := \sum_{i \in [\tau]} 2^{s'(i-1)} \cdot \mathbf{V}_{[i]}^\tau \in \mathbb{Z}_{2^\sigma}^\ell$.
7. $\mathcal{P}$ outputs $\mathbf{u}_{[\ell]}, \mathbf{v}$.

Get:

1. $\mathcal{V}$ sends (get) to $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s',\ell+h,\tau,\{\mathbb{Z}_{2^{s'}}^\tau\}}$ and receives $\Delta' \in \mathbb{Z}_{2^{s'}}^\tau, \mathbf{Q}' \in \mathbb{Z}_{2^\sigma}^{(\ell+h) \times \tau}$.
2. $\mathcal{V}$ computes $\mathbf{Q} := \mathbf{Q}' - [0 \ \mathbf{C}] \text{diag}(\Delta')$.
3. $\mathcal{V}$ aborts if $\mathbf{H}\mathbf{Q} \neq \tilde{\mathbf{u}} \cdot \mathbf{1}_\tau \cdot \text{diag}(\Delta') + \tilde{\mathbf{V}}$.
4. $\mathcal{V}$ computes $\Delta := \sum_{i \in [\tau]} 2^{s'(i-1)} \cdot \Delta'_i \in \mathbb{Z}_{2^s}$ and $\mathbf{q} := \sum_{i \in [\tau]} 2^{s'(i-1)} \cdot \mathbf{Q}_{[i]}^\tau \in \mathbb{Z}_{2^\sigma}^\ell$.
5. $\mathcal{V}$ outputs $(\Delta, \mathbf{q})$.

Fig. 20. VOLE Protocol $\Pi_{\text{large-VOLE}2k}^{\sigma,s,\ell,1,\mathcal{L}}$ for Large Rings

Theorem 12. *The protocol $\Pi_{\text{large-VOLE}2k}^{\sigma,s,\ell,1,\mathcal{L}}$ given in Fig. 20 SUC-realizes $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s,\ell,1,\mathcal{L}}$ in the $\mathcal{F}_{\text{VOLE}2k}^{\sigma,s',\ell+h,\tau,\{\mathbb{Z}_{2^{s'}}^\tau\}}$ -hybrid model, with both semi-honest and malicious security. And the distinguisher in malicious model has advantage at most $\varepsilon(\frac{\tau}{2})$.*

The proof is exactly the same as Theorem 4. Finally, we give an efficient construction of $\mathbb{Z}_{2^\sigma}^\ell$ -hiding, linear 2^{-r} -almost universal hash functions, which are essentially r independent linear combinations.

Theorem 13. *Define $\mathcal{H} := \{[\mathbf{H} \ \mathbf{I}_r] : \mathbf{H} \in \{0,1\}^{r \times \ell}\} \subset \mathbb{Z}_{2^\sigma}^{r \times (\ell+r)}$. Then (1) $\mathcal{H}$ is 2^{-r} -almost universal, and (2) $\mathcal{H}$ is $\mathbb{Z}_{2^\sigma}^\ell$ -hiding.*

Proof. By Theorems 1 and 2, it suffices to prove that $\mathcal{H}' = \mathbb{F}_2^{r \times \ell}$ is 2^{-r} -almost uniform over $\mathbb{F}_2$. For any non-zero $\mathbf{x} \in \mathbb{F}_2^\ell$ and any $\mathbf{y} \in \mathbb{F}_2^r$, we have $\Pr_{\mathbf{h} \leftarrow \mathbb{F}_2^r} [\mathbf{h}^\top \cdot \mathbf{x} = y_i] \leq \frac{1}{2}$ for every $i \in [r]$ according to Lemma 2. Additionally, if $\mathbf{H} \leftarrow \mathcal{H}'$, the rows of $\mathbf{H}$ are uniform and mutually independent. Therefore, we have $\Pr_{\mathbf{H} \leftarrow \mathcal{H}'} [\mathbf{H} \cdot \mathbf{x} = \mathbf{y}] \leq 2^{-r}$, which completes the proof.

E Security Proofs for PCG-like Protocols

E.1 Proof of Theorem 5

This proof is adapted from the proof of Theorem 3 in [WYKW21].

Proof. We consider a corrupted prover and a corrupted verifier respectively. In each case, we build a simulator $\mathcal{S}$ that emulates $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$, $\mathcal{F}_{\text{OT}}$, $\mathcal{F}_{\text{EQ}}$, and runs $\mathcal{A}$ as a subroutine.

Corrupted $\mathcal{P}$. Every time the SP-extend procedure is run (on input $\ell = 2^h$), $\mathcal{S}$ interacts with $\mathcal{A}$ as follows

1. $\mathcal{S}$ receives (a, b) from $\mathcal{A}$. Upon receiving a' from $\mathcal{A}$, $\mathcal{S}$ computes $\beta := a' + a \in \text{GR}(p^k, d_0)$ and $\delta := b$.
2. For $1 \leq i < h$, $\mathcal{S}$ samples K^i . $\mathcal{S}$ also samples $K^h \leftarrow \text{GR}(p^k, d_1)$. After receiving $\{\bar{\alpha}_i\}_{i \in [h]}$ from $\mathcal{A}$, $\mathcal{S}$ sets $K_{\bar{\alpha}_i}^i := K^i$ and sends it to $\mathcal{A}$. Then $\mathcal{S}$ computes $\alpha - 1 := \sum_{i \in [h]} 2^{h-i} \alpha_i$ and defines $\mathbf{u} \in \text{GR}(p^k, d_0)^\ell$ such that $u_\alpha = \beta$ and $u_i = 0$ for all $i \neq \alpha$. Next, $\mathcal{S}$ computes $\{q_j\}_{j \in [\ell] \setminus \{\alpha\}} \leftarrow \text{GGM.PuncEval}(\ell, \alpha, \{K_{\bar{\alpha}_i}^i\}_{i \in [h]})$.
3. $\mathcal{S}$ samples $g \leftarrow \text{GR}(p^k, d_1)$ and sends it to $\mathcal{A}$. Then $\mathcal{S}$ defines $\mathbf{v} \in \text{GR}(p^k, d_1)^\ell$ such that $v_\alpha = \delta - g - \sum_{j \in [\ell] \setminus \{\alpha\}} q_j$ and $v_j = q_j$ for all $j \neq \alpha$.
4. $\mathcal{S}$ receives $(\mathbf{x}, \mathbf{y})$ from $\mathcal{A}$.
5. $\mathcal{S}$ receives $(\chi, \mathbf{x}^*)$ from $\mathcal{A}$, and then sets $\mathbf{x}' := \mathbf{x}^* + \mathbf{x} \in \text{GR}(p^k, d_0)^m$ and $x' := \sum_{i \in [m]} x'_i \cdot \xi^{i-1}$.
6. $\mathcal{S}$ receives $V_{\mathcal{P}}$ from $\mathcal{A}$. Then $\mathcal{S}$ computes $V'_{\mathcal{P}} := \sum_{i \in [\ell]} \chi_i \cdot v_i - \sum_{i \in [m]} y_i \cdot \xi^{i-1}$ and does:
 - (a) If $x' = \chi_\alpha \cdot \beta$, then $\mathcal{S}$ checks whether $V_{\mathcal{P}} = V'_{\mathcal{P}}$. If so, $\mathcal{S}$ sends (true) to $\mathcal{A}$, and sends $(\mathbf{u}, \mathbf{v})$ to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. Otherwise, $\mathcal{S}$ sends (abort) to $\mathcal{A}$ and aborts.
 - (b) If $x' \neq \chi_\alpha \cdot \beta$, for any $\theta \in \text{GR}(p^k, d_1)$, let id_θ denote the least index that $a_{id_\theta} \neq 0$, where $\{a_i\}_{0 \leq i < k}$ are coefficients of the p -adic representation of θ . Define $id_0 := k$. Let $\varepsilon := V'_{\mathcal{P}} - V_{\mathcal{P}}$ and $\eta := x' - \chi_\alpha \cdot \beta$, then:
 - If $id_\varepsilon < id_\eta$, $\mathcal{S}$ sends (abort) to $\mathcal{A}$ and aborts.
 - If $id_\varepsilon \geq id_\eta$, $\mathcal{S}$ computes $\Delta' := \frac{\varepsilon}{p^{id_\eta}} \cdot \left(\frac{\eta}{p^{id_\eta}}\right)^{-1} \bmod p^{k-id_\eta}$, and sends a global-key query $(\text{Guess}, \Delta', k - id_\eta)$ to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. If $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ returns (success), $\mathcal{S}$ sends (true) to $\mathcal{A}$, and sends $(\mathbf{u}, \mathbf{v})$ to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. Otherwise, $\mathcal{S}$ sends (abort) to $\mathcal{A}$ and aborts.
7. Whenever $\mathcal{A}$ sends (Guess, Δ') to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ emulated by $\mathcal{S}$, $\mathcal{S}$ sends $(\text{Guess}, \Delta', k)$ to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ and returns the response to $\mathcal{A}$. If it returns (abort), $\mathcal{S}$ also aborts.

We claim that no PPT environment $\mathcal{Z}$ that corrupts $\mathcal{P}$ can distinguish the real execution of the protocol from the simulation described above, except with probability $\text{negl}(\lambda)$. The (init) and (extend) procedures can be perfectly simulated by simply forwarding them to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ and simulating the interaction with $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$. Therefore, we focus on analyzing the simulation of (SP-extend) procedures.

By the standard analysis of the GGM construction, it is not hard to see that g and $\{K_{\bar{\alpha}_i}^i\}_{i \in [h]}$ sent to $\mathcal{A}$ in the above simulation, as well as the vector $\mathbf{q}$ output by $\mathcal{V}$ when it does not abort, are computationally indistinguishable from the corresponding values in the real-world execution. It remains to analyze steps 4-7, where $\mathcal{V}$ could abort.

In the real-world execution, let $\beta = a' + a$, $\mathbf{x}' = \mathbf{x}^* + \mathbf{x}$ and $x' = \sum_{i \in [m]} x'_i \cdot \xi^{i-1}$ as above. Honest $\mathcal{V}$ computes

$$\begin{aligned}
V_{\mathcal{V}} &= \sum_{i \in [\ell]} \chi_i \cdot q_i - \sum_{i \in [m]} z_i \cdot \xi^{i-1} \\
&= \sum_{i \in [\ell] \setminus \{\alpha\}} \chi_i \cdot q_i + \chi_\alpha \cdot q_\alpha - \sum_{i \in [m]} (x'_i \cdot \Delta + y_i) \cdot \xi^{i-1} \\
&= \sum_{i \in [\ell] \setminus \{\alpha\}} \chi_i \cdot v_i + \chi_\alpha \cdot (\delta + \beta \cdot \Delta - g - \sum_{i \in [\ell] \setminus \{\alpha\}} q_i) - x' \cdot \Delta - \sum_{i \in [m]} y_i \cdot \xi^{i-1} \\
&= \sum_{i \in [\ell] \setminus \{\alpha\}} \chi_i \cdot v_i + \chi_\alpha \cdot (\beta \cdot \Delta + v_\alpha) - x' \cdot \Delta - \sum_{i \in [m]} y_i \cdot \xi^{i-1} \\
&= V'_{\mathcal{P}} + (\chi_\alpha \cdot \beta - x') \Delta
\end{aligned}$$

If $x' = \chi_\alpha \cdot \beta$, $\mathcal{F}_{\text{EQ}}$ returns (true) iff $V_{\mathcal{P}} = V_{\mathcal{V}} = V'_{\mathcal{P}}$. Otherwise, it returns (true) iff $V_{\mathcal{V}} = V_{\mathcal{P}}$, which is equivalent to

$$V'_{\mathcal{P}} - V_{\mathcal{P}} + (\chi_\alpha \cdot \beta - x')\Delta = 0 \Leftrightarrow \eta \cdot \Delta = \varepsilon \quad (3)$$

While $\mathcal{S}$ does not know the value of Δ , it can use the global-key query of $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. For Equation (3) to hold, we must have $id_\varepsilon \geq id_\eta$. Then we can divide both sides of Equation (3) by p^{id_η} and get

$$\frac{\eta}{p^{id_\eta}} \cdot \Delta = \frac{\varepsilon}{p^{id_\eta}} \bmod p^{k-id_\eta}$$

Since $\frac{\eta}{p^{id_\eta}}$ must be invertible modulo p^{k-id_η} , $\mathcal{S}$ can solve $\Delta' := \frac{\varepsilon}{p^{id_\eta}} \cdot \left(\frac{\eta}{p^{id_\eta}}\right)^{-1} \bmod p^{k-id_\eta}$. We can thus see that $\mathcal{F}_{\text{EQ}}$ returns (abort) iff $\mathcal{S}$ receives (abort) from $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ in the global-key query.

Corrupted $\mathcal{V}$. $\mathcal{S}$ interacts with $\mathcal{A}$ as follows. $\mathcal{S}$ first receives $\Delta \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$. Then, every time the (SP-extend) procedure is run (on input $\ell = 2^h$), $\mathcal{S}$ does:

1. $\mathcal{S}$ receives $c \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$. Then $\mathcal{S}$ samples $a' \leftarrow \text{GR}(p^k, d_0)$ and sends it to $\mathcal{A}$. Next, $\mathcal{S}$ computes $\gamma := c + a' \cdot \Delta$, samples $\beta \leftarrow \text{GR}(p^k, d_0)^*$ and sets $\delta := \gamma - \beta \cdot \Delta$.
2. $\mathcal{S}$ receives $\{(K_0^i, K_1^i)\}_{i \in [h]}$ from $\mathcal{A}$. Upon receiving $g \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$, for each $\alpha \in [\ell]$, it defines a vector $\mathbf{v}^{(\alpha)} \in \text{GR}(p^k, d_1)^\ell$ as follows:
 - (a) Compute $\{q_j^{(\alpha)}\}_{j \in [\ell] \setminus \{\alpha\}} \leftarrow \text{GGM.PuncEval}(\ell, \alpha, \{K_{\alpha_i}^i\}_{i \in [h]})$, and then set $v_i^{(\alpha)} := q_i^{(\alpha)}$ for all $i \neq \alpha$.
 - (b) Compute $v_\alpha^{(\alpha)} := \delta - g - \sum_{j \in [\ell] \setminus \{\alpha\}} v_j^{(\alpha)}$.
3. After receiving $\mathbf{z}^*$ from $\mathcal{A}$, $\mathcal{S}$ samples $\chi \leftarrow \text{GR}(p^k, d_1)^\ell$, $\mathbf{x}^* \leftarrow \text{GR}(p^k, d_0)^m$ and sends them to $\mathcal{A}$. Then $\mathcal{S}$ computes $\mathbf{z} := \mathbf{z}^* + \mathbf{x}^* \cdot \Delta$.
4. $\mathcal{S}$ computes $Z := \sum_{i \in [m]} z_i \cdot \xi^{i-1}$. Then it receives $V_{\mathcal{V}}$ from $\mathcal{A}$. Next, $\mathcal{S}$ computes I as follows:
 - (a) For $\alpha \in [\ell]$, compute $V_{\mathcal{P}}^\alpha := \sum_{i \in [\ell]} \chi_i \cdot v_i^{(\alpha)} + \chi_\alpha \cdot \beta \cdot \Delta - Z$.
 - (b) Define $I := \{\alpha \in [\ell] \mid V_{\mathcal{P}}^\alpha = V_{\mathcal{V}}\}$.
 $\mathcal{S}$ sends I to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. If it returns (abort), $\mathcal{S}$ picks $\tilde{\alpha} \leftarrow [\ell] \setminus I$, sends (false, $V_{\mathcal{P}}^{\tilde{\alpha}}$) to $\mathcal{A}$, and aborts. Otherwise, $\mathcal{S}$ sends (true, $V_{\mathcal{V}}$) to $\mathcal{A}$.
5. $\mathcal{S}$ picks an arbitrary $\alpha \in I$ and computes $\mathbf{q}$ such that $q_i := v_i^{(\alpha)}$ for $i \in [\ell] \setminus \{\alpha\}$ and $q_\alpha := \gamma - g - \sum_{i \in [\ell] \setminus \{\alpha\}} q_i$. $\mathcal{S}$ sends $\mathbf{q}$ to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$.

Similar to the case of corrupted $\mathcal{P}$, we only focus on analyzing the simulation of (SP-extend) procedures. Note that $\mathcal{S}$ now learns Δ by emulating $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ and receiving it from $\mathcal{A}$.

The distributions of simulated values a' and $\mathbf{x}^*$ are identical to the real execution, because they are both masked by a uniform element output by $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$. The challenges χ are both uniformly random in the real world and ideal world and are thus identical. Therefore, we only need to analyze the simulation for equality test. The set I extracted by $\mathcal{S}$ corresponds to the selective failure attack on the index α^* of the non-zero coordinate of $\mathbf{u}$ output by $\mathcal{P}$. In the ideal world, $\mathcal{S}$ aborts iff $\alpha^* \notin I$, which means $V_{\mathcal{P}}^{\alpha^*} \neq V_{\mathcal{V}}$ and thus the real world execution also aborts. Therefore, the simulation is perfect in the case of aborting.

In the case of no aborting, $\mathcal{A}$ passes the equality check in the real world and in the ideal world, $\alpha^* \in I$ and $\mathcal{S}$ needs to pick an $\alpha \leftarrow I$. When $|I| > 1$, it is possible that $\alpha \neq \alpha^*$ and $\mathcal{Z}$ can thus distinguish two worlds from joint distribution of $\mathcal{A}$'s view and $\mathcal{P}$'s output. Next, we prove that $\mathbf{q}$ output by $\mathcal{S}$ is unique except with probability $1/p^{d_1}$, which completes the proof.

Claim. For any $\alpha, \alpha' \in [\ell]$, let $\mathbf{q}^{(\alpha)}, \mathbf{q}^{(\alpha')}$ be the vectors computed by $\mathcal{S}$ following step 5, then we have

$$\Pr[\mathbf{q}^{(\alpha)} \neq \mathbf{q}^{(\alpha')} \mid V_{\mathcal{P}}^\alpha = V_{\mathcal{P}}^{\alpha'}] \leq \frac{1}{p^{d_1}}$$

Proof. Since $V_{\mathcal{P}}^\alpha = V_{\mathcal{P}}^{\alpha'}$, we have

$$\begin{aligned} \sum_{i \in [\ell]} \chi_i \cdot v_i^{(\alpha)} + \chi_\alpha \cdot \beta \cdot \Delta - Z &= \sum_{i \in [\ell]} \chi_i \cdot v_i^{(\alpha')} + \chi_{\alpha'} \cdot \beta \cdot \Delta - Z \Leftrightarrow \\ \sum_{i \in [\ell] \setminus \{\alpha, \alpha'\}} \chi_i \cdot (v_i^{(\alpha)} - v_i^{(\alpha')}) &+ \chi_\alpha \cdot (v_\alpha^{(\alpha)} - v_\alpha^{(\alpha')} + \beta \cdot \Delta) \\ &+ \chi_{\alpha'} \cdot (v_{\alpha'}^{(\alpha)} - v_{\alpha'}^{(\alpha')} - \beta \cdot \Delta) = 0 \end{aligned}$$

Note that $\Delta, \beta, \mathbf{v}^{(\alpha)}, \mathbf{v}^{(\alpha')}$ are determined before χ are sampled. Furthermore, each coefficient χ_i is uniform. Therefore, by Lemma 2, except with probability $1/p^{d_1}$, we have:

$$\begin{aligned} v_i^{(\alpha)} &= v_i^{(\alpha')} \text{ for all } i \in [\ell] \setminus \{\alpha, \alpha'\} \\ v_{\alpha'}^{(\alpha')} - v_{\alpha}^{(\alpha)} &= v_{\alpha'}^{(\alpha)} - v_{\alpha'}^{(\alpha')} = \beta \cdot \Delta \end{aligned}$$

From the first equation, we obtain that $q_i^{(\alpha)} = q_i^{(\alpha')}$ for all $i \in [\ell] \setminus \{\alpha, \alpha'\}$. Therefore, $q_{\alpha}^{(\alpha)} - v_{\alpha}^{(\alpha)} = \gamma - g - \sum_{i \in [\ell] \setminus \{\alpha\}} q_i^{(\alpha)} - \left(\delta - g - \sum_{j \in [\ell] \setminus \{\alpha\}} v_j^{(\alpha)} \right) = \beta \cdot \Delta$. Together with $v_{\alpha'}^{(\alpha')} - v_{\alpha}^{(\alpha)} = \beta \cdot \Delta$, we have $q_{\alpha}^{(\alpha)} = v_{\alpha}^{(\alpha')} = q_{\alpha'}^{(\alpha')}$. Similarly we have $q_{\alpha'}^{(\alpha')} = v_{\alpha'}^{(\alpha)} = q_{\alpha}^{(\alpha)}$.

We conclude that no PPT $\mathcal{Z}$ can distinguish two worlds except with probability $1/p^{d_1}$. $\square$

E.2 Proof of Theorem 6

We adapt the proof of Theorem 8 from [BBMHS22], the only difference is that we are working on Galois rings.

Proof. We consider a corrupted prover and a corrupted verifier respectively. In each case, we build a simulator $\mathcal{S}$ that emulates $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ and runs $\mathcal{A}$ as a subroutine.

Corrupted $\mathcal{P}$. In **Init**, $\mathcal{S}$ forwards (init) to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ and after receiving (extend, k_c), it receives $\mathbf{u} \in \text{GR}(p^k, d_0)^{k_c}$, $\mathbf{v} \in \text{GR}(p^k, d_1)^{k_c}$ from $\mathcal{A}$. Then every time the **Extend** procedure is run, $\mathcal{S}$ interacts with $\mathcal{A}$ as follows

1. For $i \in [t]$, $\mathcal{S}$ receives $\mathbf{e}_i \in \text{GR}(p^k, d_0)^{n_c/t}$ (with at most one non-zero entry) and $\mathbf{b}_i \in \text{GR}(p^k, d_1)^{n_c/t}$ from $\mathcal{A}$. Then it defines $\mathbf{e} := (\mathbf{e}_1, \dots, \mathbf{e}_t)$ and $\mathbf{b} := (\mathbf{b}_1, \dots, \mathbf{b}_t)$.
2. $\mathcal{S}$ computes $\mathbf{x} := \mathbf{u} \cdot \mathbf{A} + \mathbf{e} \in \text{GR}(p^k, d_0)^{n_c}$ and $\mathbf{y} := \mathbf{v} \cdot \mathbf{A} + \mathbf{b} \in \text{GR}(p^k, d_1)^{n_c}$. $\mathcal{S}$ sends $\mathbf{x}_{[k_c+1, n_c]}$ and $\mathbf{y}_{[k_c+1, n_c]}$ to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$, and updates $\mathbf{u} := \mathbf{x}_{[k_c]}$, $\mathbf{v} := \mathbf{y}_{[k_c]}$.
3. Whenever $\mathcal{A}$ sends (Guess, Δ', k') to $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$ emulated by $\mathcal{S}$, $\mathcal{S}$ sends (Guess, Δ', k') to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$ and returns the response to $\mathcal{A}$. If it returns (abort), $\mathcal{S}$ sends (abort) to $\mathcal{A}$ and aborts.

Note that there is no interaction between parties in the protocol. Furthermore, the outputs of both parties are computed in the same way in two worlds. Therefore, the simulation is perfect.

Corrupted $\mathcal{V}$. In **Init**, $\mathcal{S}$ receives $\Delta \in \text{GR}(p^k, d_1)$ from $\mathcal{A}$ and forwards it to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$. After receiving (extend, k_c), it receives $\mathbf{q} \in \text{GR}(p^k, d_1)^{k_c}$ from $\mathcal{A}$. Then every time the **Extend** procedure is run, $\mathcal{S}$ interacts with $\mathcal{A}$ as follows

1. For $i \in [t]$, $\mathcal{S}$ receives $\mathbf{c}_i \in \text{GR}(p^k, d_1)^{n_c/t}$ from $\mathcal{A}$. Then it defines $\mathbf{c} := (\mathbf{c}_1, \dots, \mathbf{c}_t)$.
2. $\mathcal{S}$ samples $\mathbf{e} := (\mathbf{e}_1, \dots, \mathbf{e}_t) \in \text{GR}(p^k, d_0)^{n_c}$ where each $\mathbf{e}_i$ has a single invertible entry and zeros everywhere else. For $i \in [t]$, $\mathcal{S}$ receives $I_i \subseteq [n_c/t]$ from $\mathcal{A}$ and let α_i be the index of non-zero entry in $\mathbf{e}_i$. If $\alpha_i \in I_i$, $\mathcal{S}$ sends (success) to $\mathcal{A}$. Otherwise, it sends (abort) to $\mathcal{A}$ and aborts.
3. $\mathcal{S}$ computes $\mathbf{z} := \mathbf{q} \cdot \mathbf{A} + \mathbf{c} \in \text{GR}(p^k, d_1)^{n_c}$. $\mathcal{S}$ sends $\mathbf{z}_{[k_c+1, n_c]}$ to $\mathcal{F}_{\text{PCG-srVOLE}}^{p,k,d_0,d_1}$, and updates $\mathbf{q} := \mathbf{z}_{[k_c]}$.

The view of $\mathcal{A}$ is perfectly simulated since $\mathbf{e}$ sampled by $\mathcal{S}$ has the same distribution as that returned by $\mathcal{F}_{\text{sp-srVOLE}}^{p,k,d_0,d_1}$. And in both worlds, the output $(\mathbf{s}, \mathbf{t})$ of $\mathcal{P}$ satisfies $\mathbf{s} \cdot \Delta + \mathbf{t} = \mathbf{z}_{[k_c+1, n_c]}$. The only difference is that in the ideal world $\mathbf{s}$ is uniform in $\text{GR}(p^k, d_0)^{n_c-k_c}$, while in the real world $\mathbf{s} = (\mathbf{u} \cdot \mathbf{A} + \mathbf{e})_{[k_c+1, n_c]}$. From the LPN with static leakage assumption, we know that $\mathbf{s}$ in the real world is indistinguishable from a uniform vector. Therefore, we can conclude that no PPT $\mathcal{Z}$ can distinguish between two worlds except with negligible probability. $\square$

F Security Proofs for ZK protocols

F.1 Proof of Theorem 7

The proof is an adaptation of the proof of Theorem 2 from [YSWW21]. The difference is that we integrate the opening of the output wire into the linear combination of all multiplication gates.

Proof. We consider a corrupted prover and a corrupted verifier respectively. In each case, we build a simulator $\mathcal{S}$.

Corrupted $\mathcal{P}$. $\mathcal{S}$ emulates the functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$ and interacts with $\mathcal{A}$ as follows

1. $\mathcal{S}$ receives $\mathbf{u} \in \text{GR}(p^k, d_0)^{n+t}$, $M[\mathbf{u}] \in \text{GR}(p^k, d)^{n+t}$, $L \in \mathcal{L}$ from $\mathcal{A}$. $\mathcal{S}$ samples $\Delta \leftarrow \mathbb{F}_{p^d}$ and aborts if $\Delta \notin L$. Then it computes $K[\mathbf{u}] := \mathbf{u} \cdot \Delta + M[\mathbf{u}]$.
2. $\mathcal{S}$ receives $A_0^*, A_1^* \in \text{GR}(p^k, d)$ from $\mathcal{A}$ and computes $B^* := A_0^* + A_1^* \cdot \Delta$.
3. After receiving δ from $\mathcal{A}$, $\mathcal{S}$ computes $\mathbf{w} := \delta + \mathbf{u}_{[n]}$.
4. $\mathcal{S}$ executes the rest of the protocol as an honest verifier. If it outputs (true), $\mathcal{S}$ sends (prove, $\mathcal{C}$, $\mathbf{w}$) to $\mathcal{F}_{\text{circuitZK}}$. Otherwise, $\mathcal{S}$ sends (prove, $\mathcal{C}$, $\perp$) to $\mathcal{F}_{\text{circuitZK}}$ and aborts.

It is easy to see that the simulated view of $\mathcal{A}$ is identical to that in the real world. The only difference is that $\mathcal{V}$ can output (true) when $\mathcal{F}_{\text{circuitZK}}$ outputs (false). We first prove that the value of the output wire must be $w_h = \mathcal{C}(\mathbf{w})$ and $w_h = 1$ with high probability. Since Add gates are computed locally, we only need to consider the correctness of Mult gates. For the i -th multiplication gate, let $w_\gamma = w_\alpha \cdot w_\beta + e_i$, where $e_i \in \text{GR}(p^k, d_0)$ is an error chosen by $\mathcal{A}$ by sending incorrect c'_i in step 2 of the online phase. Then we have

$$\begin{aligned} B_i &= K[w_\alpha] \cdot K[w_\beta] - K[w_\gamma] \cdot \Delta \\ &= (w_\alpha \cdot w_\beta - w_\gamma) \cdot \Delta^2 + (w_\alpha \cdot M[w_\beta] + w_\beta \cdot M[w_\alpha] - M[w_\gamma]) \cdot \Delta \\ &\quad + M[w_\alpha] \cdot M[w_\beta] \\ &= A_{0,i} + A_{1,i} \cdot \Delta - e_i \cdot \Delta^2 \end{aligned}$$

In step 4, $\mathcal{A}$ sends $A'_0 = A_0 + E_0$ and $A'_1 = A_1 + E_1$ to $\mathcal{V}$, where $E_0, E_1 \in \text{GR}(p^k, d)$ are errors chosen by $\mathcal{A}$. Then $\mathcal{A}$ passing the check means that

$$\begin{aligned} B - (A'_0 + A'_1 \cdot \Delta) &= \sum_{i \in [t]} \chi_i \cdot (A_{0,i} + A_{1,i} \cdot \Delta - e_i \cdot \Delta^2) + \chi_{t+1} \cdot K[w_h - 1] \cdot \Delta \\ &\quad + B^* - (A_0 + A_1 \cdot \Delta) - (E_0 + E_1 \cdot \Delta) \\ &= \left(\chi_{t+1} \cdot (w_h - 1) - \sum_{i \in [t]} \chi_i \cdot e_i \right) \cdot \Delta^2 - E_1 \cdot \Delta - E_0 \\ &= - \sum_{i \in [t+1]} \chi_i \cdot e_i \cdot \Delta^2 - E_1 \cdot \Delta - E_0 \quad (\text{let } e_{t+1} := 1 - w_h) \\ &= 0 \end{aligned}$$

If $\sum_{i \in [t+1]} \chi_i \cdot e_i \neq 0$, the above equation holds with probability at most $2/p^d$ by Lemma 2. If $\sum_{i \in [t+1]} \chi_i \cdot e_i = 0$, $\mathcal{A}$ can always pass the check by setting $E_0 = E_1 = 0$. So we only need to bound the probability that $\sum_{i \in [t+1]} \chi_i \cdot e_i = 0$. If there exists some $e_j \neq 0$, then we have $\chi_j \cdot e_j = -\sum_{i \in [t+1] \setminus \{j\}} \chi_i \cdot e_i$. Let $(a_0, a_1, \dots, a_{k-1}) := \text{Decomp}(e_j)$ and $(b_0, b_1, \dots, b_{k-1}) := \text{Decomp}(-\sum_{i \in [t+1] \setminus \{j\}} \chi_i \cdot e_i)$. Then there is some $a_{j^*} \neq 0 \in \mathcal{T}$. We get $\chi_j = \frac{b_{j^*}}{a_{j^*}}$, which happens with probability at most $1/p^d$ since χ_j is uniform over $\mathbb{F}_{p^d}$ and independent of every other χ_i and all e_i .

Therefore, the overall probability that an unbounded environment $\mathcal{Z}$ can distinguish between real-world execution and ideal-world execution is at most $3/p^d$.

Corrupted $\mathcal{V}$. If $\mathcal{S}$ receives (false) from $\mathcal{F}_{\text{circuitZK}}$, it simply aborts. Otherwise, it interacts with $\mathcal{A}$ as follows:

1. $\mathcal{S}$ emulates the functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n+t,1,\mathcal{L}}$. It receives $\Delta \in \mathbb{F}_{p^d}$, $K[\mathbf{u}] \in \text{GR}(p^k, d)^{n+t}$ and $B^* \in \text{GR}(p^k, d)$ from $\mathcal{A}$.
2. $\mathcal{S}$ executes step 1-2 of the online phase by sending uniform $\delta \in \text{GR}(p^k, d_0)^n$ and c_i for each multiplication gate to $\mathcal{A}$.
3. After receiving $\chi \in \mathbb{F}_{p^d}^{t+1}$ from $\mathcal{A}$, $\mathcal{S}$ computes B as an honest verifier using Δ , $K[\mathbf{w}] = K[\mathbf{u}] + \delta \cdot \Delta$ and B^* . Then it samples $A_1 \leftarrow \text{GR}(p^k, d)$, computes $A_0 := B - A_1 \cdot \Delta$ and sends (A_0, A_1) to $\mathcal{A}$. Since $\mathbf{u}$ and A_1^* are uniformly random and kept secret from $\mathcal{A}$, we obtain that the simulated view of $\mathcal{A}$ is identical to its view in the real-world execution, which completes the proof. $\square$

F.2 Proof of Theorem 8

The proof is adapted from the proof of Theorem 3 in [YSWW21].

Proof. We consider a corrupted prover and a corrupted verifier respectively. In each case, we build a simulator $\mathcal{S}$.

Corrupted $\mathcal{P}$. The simulator $\mathcal{S}$ emulates the functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$ and interacts with $\mathcal{A}$ as follows

1. $\mathcal{S}$ receives $\mathbf{u} \in \text{GR}(p^k, d_0)^n$, $M[\mathbf{u}] \in \text{GR}(p^k, d)^n$, $L \in \mathcal{L}$ from $\mathcal{A}$. $\mathcal{S}$ samples $\Delta \leftarrow \mathbb{F}_{p^d}$ and aborts if $\Delta \notin L$. Then it computes $K[\mathbf{u}] := \mathbf{u} \cdot \Delta + M[\mathbf{u}]$.
2. $\mathcal{S}$ receives $\{A_h^*\}_{0 \leq h < D}$ from $\mathcal{A}$ and computes $B^* := \sum_{h=0}^{D-1} A_h^* \cdot \Delta^h$.
3. After receiving δ from $\mathcal{A}$, $\mathcal{S}$ extracts the witness $\mathbf{w} := \delta + \mathbf{u}$.
4. $\mathcal{S}$ executes the rest of the protocol as an honest verifier. If it outputs (true), $\mathcal{S}$ sends (prove, $\{f_i\}_{i \in [t]}, \mathbf{w}$) to $\mathcal{F}_{\text{polyZK}}$. Otherwise, $\mathcal{S}$ sends (prove, $\{f_i\}_{i \in [t]}, \perp$) to $\mathcal{F}_{\text{polyZK}}$ and aborts.

One can observe that the distribution of the ideal-world execution simulated by $\mathcal{S}$ is identical to the real-world execution. The only difference is that a real-world verifier can output (true) when $\mathcal{F}_{\text{polyZK}}$ outputs (false).

Let $f_i(\mathbf{w}) = y_i \in \text{GR}(p^k, d)$ for each $i \in [t]$, by Equation (2), we have

$$\begin{aligned} B_i &= \sum_{h=0}^D f_{i,h}(K[w_1], \dots, K[w_n]) \cdot \Delta^{D-h} \\ &= \sum_{h=0}^{D-1} A_{i,h} \cdot \Delta^h + y_i \cdot \Delta^D \end{aligned}$$

In step 3 of the online phase, $\mathcal{S}$ receives $A'_h = A_h + E_h$ for $0 \leq h < D$ from $\mathcal{A}$, where A_h is honestly computed according to the protocol and E_h is the error term chosen by $\mathcal{A}$. If the check passes, we have

$$\begin{aligned} B - \sum_{h=0}^{D-1} A'_h \cdot \Delta^h &= \sum_{i \in [t]} \chi_i \cdot B_i + B^* - \sum_{h=0}^{D-1} (A_h + E_h) \cdot \Delta^h \\ &= \sum_{i \in [t]} \chi_i \cdot \left(\sum_{h=0}^{D-1} A_{i,h} \cdot \Delta^h + y_i \cdot \Delta^D \right) + B^* \\ &\quad - \sum_{h=0}^{D-1} \left(\sum_{i \in [t]} \chi_i \cdot A_{i,h} + A_h^* + E_h \right) \cdot \Delta^h \\ &= \sum_{i \in [t]} \chi_i \cdot y_i \cdot \Delta^D - \sum_{h=0}^{D-1} E_h \cdot \Delta^h \\ &= 0 \end{aligned}$$

If $\sum_{i \in [t]} \chi_i \cdot y_i \neq 0$, then the above equation holds with probability at most D/p^d by Lemma 2. If $\sum_{i \in [t]} \chi_i \cdot y_i = 0$, $\mathcal{A}$ can always pass the check by setting $E_h = 0$ for all $0 \leq h < D$. So we only need to bound the probability that $\sum_{i \in [t]} \chi_i \cdot y_i = 0$, which is at most $1/p^d$.

Therefore, the overall probability that an unbounded environment $\mathcal{Z}$ can distinguish between real-world execution and ideal-world execution is at most $(D+1)/p^d$.

Corrupted $\mathcal{V}$. If $\mathcal{S}$ receives (false) from $\mathcal{F}_{\text{polyZK}}$, it simply aborts. Otherwise, $\mathcal{S}$ interacts with $\mathcal{A}$ as follows

1. $\mathcal{S}$ emulates the functionality $\mathcal{F}_{\text{ext-srVOLE}}^{p,k,d_0,d,n,1,\mathcal{L}}$. It receives $\Delta \in \mathbb{F}_{p^d}$ and $K[\mathbf{u}] \in \text{GR}(p^k, d)^n$ from $\mathcal{A}$. It also receives $B^* \in \text{GR}(p^k, d)$ from $\mathcal{A}$.
2. In step 1 of the online phase, $\mathcal{S}$ sends uniform $\delta \in \text{GR}(p^k, d_0)^n$ to $\mathcal{A}$.
3. After receiving $\chi \in \mathbb{F}_{p^d}^t$ from $\mathcal{A}$, $\mathcal{S}$ computes B using Δ , $K[\mathbf{w}] = K[\mathbf{u}] + \delta \cdot \Delta$ and χ . It samples $A_1, \dots, A_{D-1} \leftarrow \text{GR}(p^k, d)$ and computes $A_0 := B - \sum_{h \in [D-1]} A_h \cdot \Delta^h$. Then $\mathcal{S}$ sends $\{A_h\}_{0 \leq h < D}$ to $\mathcal{A}$.

Since $\mathbf{u}$ and $\{A_h^*\}_{h \in [D-1]}$ are uniform and kept secret from $\mathcal{A}$, the simulated view is identical to that in the real-world execution. $\square$