More Efficient Zero-Knowledge Protocols over $\mathbb{Z}_{2^k}$ via Galois Rings

Fuchun Lin, Chaoping Xing, and Yizhou Yao

Shanghai Jiao Tong University
{linfuchun,xingcp,yaoyizhou0620}@sjtu.edu.cn

Abstract. A recent line of works on zero-knowledge (ZK) protocols with a vector oblivious linear function evaluation (VOLE)-based offline phase provides a new paradigm for scalable ZK protocols featuring fast proving and small prover memory. Very recently, Baum et al. (Crypto'23) proposed the VOLE-in-the-head technique, allowing such protocols to become publicly verifiable. Many practically efficient protocols for proving circuit satisfiability over any Galois field are implemented, while protocols over rings $\mathbb{Z}_{2^k}$ are significantly lagging behind, with only a proof-of-concept pioneering work called Appenzeller to Brie (CCS'21) and a first proposal called Moz$\mathbb{Z}_{2^k}$arella (Crypto'22). The ring $\mathbb{Z}_{2^{32}}$ or $\mathbb{Z}_{2^{64}}$, though highly important (it captures computation in real-life programming and the computer architectures such as CPU words), presents non-trivial difficulties because, for example, unlike Galois fields $\mathbb{F}_{2^k}$, the fraction of units in $\mathbb{Z}_{2^k}$ is 1/2. In this work, we first construct ZK protocols over a high degree Galois ring extension of $\mathbb{Z}_{2^k}$ (fraction of units close to 1) and then convert them to $\mathbb{Z}_{2^k}$ efficiently using amortization techniques. Our results greatly change the landscape of ZK protocols over $\mathbb{Z}_{2^k}$.
(1) We propose a competing ZK protocol that has many advantages over the state-of-the-art Moz$\mathbb{Z}_{2^k}$arella. We remove the undesirable dependence of communication complexity on the security parameter, and achieve communication complexity *strictly* linear in the circuit size. Furthermore, our protocol has better concrete efficiency. For $40, 80$ bits soundness on circuits over $\mathbb{Z}_{2^{32}}$ and $\mathbb{Z}_{2^{64}}$, we offer $1.15\times$–$2.9\times$ improvements in communication.
(2) Inspired by the recently proposed interactive message authentication code technique (Weng et al., CCS'22), we construct a constant round ZK protocol over $\mathbb{Z}_{2^k}$ with sublinear (in the circuit size) communication complexity, which was previously achieved only over fields.
(3) We show that the pseudorandom correlation generator approach can be adapted to efficiently implement VOLE over Galois rings, with analysis of the hardness of underlying LPN assumptions over Galois rings.
(4) We adapt the VOLE-in-the-head technique to make it work for $\mathbb{Z}_{2^k}$, yielding *publicly verifiable* non-interactive ZK protocols over $\mathbb{Z}_{2^k}$ which preserve most of the efficiency metrics of the VOLE-based ZK protocols.

1 Introduction

A proof system (of knowledge) for circuit satisfiability allows a prover to convince a verifier that he holds a witness w for a given circuit $\mathcal{C}$ such that $\mathcal{C}(w) = 1$.

The proof is zero-knowledge (ZK) if no information about w beyond the fact that $\mathcal{C}(w) = 1$ is revealed to the verifier. Typically, the circuit $\mathcal{C}$ can either be a Boolean circuit that consists of `AND` gates and `XOR` gates, or an arithmetic circuit that consists of `Add` gates and `Mult` gates over some ring $\mathcal{R}$.

Over decades of studies, numerous ZK proof systems have been developed, which have various properties and also diverse in efficiency metrics (round complexity, communication complexity, prover/verifier computation complexity, prover/verifier memory, etc.). We briefly review ZK proof systems that admit practically efficient ZK protocols for circuit satisfiability, with special emphasis on scalability to large circuits.

The MPC-in-the-head (MPCitH) paradigm [47], offers a publicly verifiable solution to non-interactive ZK (NIZK), where the prover emulates in his head the evaluation of a circuit with imaginary parties via a multi-party computation (MPC) protocol and proves to the verifier that the circuit is honestly evaluated. The bottleneck of MPCitH is either big proof size [47,40,24,50], or small proof size but large prover time and memory [2]. The garbled circuit [65] ZK (GCZK) [48,37,66,46] paradigm, where the verifier plays the role of garbler who garbles the circuit, admits ZK protocols with small prover time and memory, but large proof size. The interactive oracle proof (IOP)-based ZK protocols admit zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARK) protocols [43,20,9,38,10,26,58]. Most zk-SNARKs achieve short proof size and small verification time simultaneously assuming the existence of a setup phase. However, many of them require that the prover should have sufficient computation power, namely, large prover time and memory. We remark that very recently, there is a line of works [12,14,62,55,13,42] focusing on zk-SNARKs with linear prover time[1], as prover time is a bottleneck for large circuits. To our best knowledge, succinctness for both proof size and verification time can only be achieved when $\mathcal{R}$ is a finite field of size $\Omega(|\mathcal{C}|)$ [14,62,42], and succinct proof size can be achieved for any finite field [55,13]. However, these constructions are not scalable, due to large memory consumption.

VOLE-based ZK. The focus of this work is on a new paradigm of active research usually described as vector oblivious linear function evaluation (VOLE)-based ZK. The (random) VOLE is a primitive that allows the sender to obtain two (random) vectors $\mathbf{M}, \mathbf{x}$ and the receiver to obtain a (random) scalar Δ and a (random) vector $\mathbf{K}$ such that $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$ over some ring $\mathcal{R}$. In general, VOLE-based ZK protocols have main advantages of scalable prover memory, linear prover time, and linear proof size. At a high level, we view a ZK protocol as a special case of secure two-party computation (2-PC), where only the prover (sender) has private inputs. VOLE-based ZK protocols first authenticate the value on each wire using a linearly homomorphic message authentication code (MAC) and then prove to the verifier (receiver) that these authenticated values satisfy the circuit topology. In more detail, VOLE-based ZK protocols have two phases, an

[1] We say a ZK protocol has linear prover time, if the number of ring operations required for the prover is linear in the circuit size.

offline phase that generates random VOLE-based MACs, and an online phase that securely evaluates the circuit by consuming the previously generated MACs.

Boyle et al. [15,17] initiated the study of VOLE-based ZK, by introducing a new cryptographic primitive, the pseudorandom correlation generator (PCG). Generally, PCG is an extension of pseudorandom generator (PRG) from generating a batch of randomness to a batch of correlated randomness between some parties. PCG offers a low-communication candidate for generating random VOLE correlations in the offline phase. The authors of [16] presented a two-round maliciously secure construction of PCG for VOLE, and showed that when combining with a non-interactive online phase, a designated verifier NIZK for circuit satisfiability over arbitrary field can be obtained. The subsequent work Wolverine [60] constructed an efficient constant round online phase over any field (communicates 4 field elements per multiplication gate), and an efficient interactive PCG construction for VOLE. The work [32] introduced line point zero knowledge (LPZK), which essentially admits a more efficient online phase over a sufficiently large field (communicates 1 finite field element per multiplication gate). Concurrent to Wolverine and LPZK, Mac'n'Cheese [7] proposed two different online phase protocols that have sublinear communication complexity when used for disjunction. Follow-up works to the above include QuickSilver [63] and then improved LPZK [31], AntMan [61]. QuickSilver combined the idea of LPZK and Wolverine to achieve one field element per multiplication gate for arbitrary field. Also QuickSilver proposed an online phase for proving low degree polynomials with sublinear communication. Improved LPZK [31] reduced the communication from 1 field element per multiplication gate to 1/2. AntMan [61] proposed an online phase over arbitrary field with sublinear (in the circuit size) communication, by employing a novel authentication technique. We refer to a recent survey [6] for more details of VOLE-base ZK protocols over fields.

Very recently, Baum et al. [5] discovered an interesting way of combining the VOLE-based ZK paradigm and the MPCitH ZK paradigm to achieve the best of the two worlds: an NIZK protocol that inherits some favorable efficiency features from the VOLE-based ZK paradigm, and also achieves *public verifiability*. This approach was dubbed *VOLE-in-the-head* (VOLEitH), as it essentially keeps the structure of a VOLE-based ZK protocol and replaces the PCG-style VOLE protocol with an emulation through first reducing VOLE to $(N-1)$-out-of-N OT as done in [56] and then emulating the underlying OT using a commitment scheme in an MPCitH fashion. This intermediate ZK protocol then is made non-interactive and publicly verifiable via Fiat-Shamir transform, as long as the original ZK is public-coin in the VOLE-hybrid model. Note that in PCG-style VOLE protocols the sender of the underlying OT is playing the role of VOLE receiver (hence the verifier), rendering it not compatible with the MPCitH fashion emulation (see Section 5.2 for more details). The communication complexity of the MPCitH ZK protocols is linear in the circuit size of the statement being proved with usually quite large hidden constant. Recent efforts [45,49] on lowering this hidden constant have allowed MPCitH to shine in settings where a small prover run time is critical, and/or when proving statements of small-to-medium

sized circuits, where the linear proof size may not have a big impact. Compared to these MPCitH ZK protocols, the VOLEitH protocols proposed in [5] have an edge in offering simpler, smaller and faster solutions to practical application scenarios such as signature schemes.

ZK over integer rings. As the models of computation in real-life programming and the computer architectures (such as CPU words) are formulated as operations over the ring $\mathbb{Z}_{2^{32}}$ or $\mathbb{Z}_{2^{64}}$, ZK protocols designed for $\mathbb{Z}_{2^k}$ are more efficient when implemented. However, the fact that half of the ring $\mathbb{Z}_{2^k}$ are zero divisors presents non-trivial technical difficulties. This results in that, for instance, though the ring $\mathbb{Z}_{2^k}$ has size 2^k, its exceptional sets can only contain two elements. To our best knowledge, there are only a few existing works that constructed ZK protocols over $\mathbb{Z}_{2^k}$.

Ganesh et al. [39] proposed Rinocchio by adapting Pinocchio [53], a SNARK for field arithmetic, to work for ring arithmetic. To illustrate the Rinocchio proof system, the authors focus on the ring $\mathbb{Z}_{2^k}$ and use a Galois extension that admits large enough exceptional sets, yielding a designated verifier zk-SNARK with succinct proof size, but not with succinct verification. However, Rinocchio inherits from its predecessors Pinocchio (and most SNARK's) the bottleneck of large prover computation and memory, when scaling up to prove large statements. Following the blueprint of VOLE-based ZK protocols over finite fields, and also inspired by the idea of SPD$\mathbb{Z}_{2^k}$ [27], Baum et al. proposed two online constructions in Appenzeller to Brie [3], and later a more efficient online protocol with a PCG construction for VOLE over $\mathbb{Z}_{2^k}$ in Moz$\mathbb{Z}_{2^k}$arella [4] [2]. These constructions did obtain highly scalable ZK protocols over $\mathbb{Z}_{2^k}$, except that, due to the SPD$\mathbb{Z}_{2^k}$ [27] techniques, their efficiency has an inherent undesirable dependency on the security parameter. More recently, Braun et al. [18] adapted the recent efficient MPCitH protocols [50,8,57,36] to work over the ring $\mathbb{Z}_{2^k}$ through a Galois ring extension with suitable exceptional sets, yielding efficient publicly verifiable NIZK protocols over $\mathbb{Z}_{2^k}$. Same as their MPCitH predecessors for fields, these protocols require heavy prover computation and memory.

In the light of the recent success of Moz$\mathbb{Z}_{2^k}$arella [4] and given the width and depth of the theoretical study on ZK protocols over finite fields, the current state of protocols over rings $\mathbb{Z}_{2^k}$ leaves too much to be desired.

1.1 Our Contributions

On top of making sophisticated use of existing techniques for ZK over $\mathbb{Z}_{2^k}$, we introduce more powerful tools, namely, the reverse multiplication friendly embedding (RMFE) techniques [22,23,28,34], from the MPC literature into the literature of VOLE-based ZK. We focus on optimizing the efficiency of VOLE-based ZK over $\mathbb{Z}_{2^k}$, and obtain the following results.

(1) Targeting the state-of-the-art ZK protocol over $\mathbb{Z}_{2^k}$, Moz$\mathbb{Z}_{2^k}$arella, we propose a competing online phase protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$, which is also public-coin.

[2] Appenzeller to Brie adapted the online phases of Wolverine [60] and Mac'n'Cheese [7]. Moz$\mathbb{Z}_{2^k}$arella adapted the online phase of QuickSilver [63].

Our protocol has the main advantage that the efficiency is independent of the security parameter (see Theorem 2). Thus, in all high security region applications, our protocol has overwhelming advantage over Moz$\mathbb{Z}_{2^k}$arella. We then compare concrete performance between the two statistical security parameter choices $\kappa = 40$ and $\kappa = 80$ over $\mathbb{Z}_{2^{32}}$ and $\mathbb{Z}_{2^{64}}$ in Table 1, assuming the circuit whose satisfiability to be proved is a single instruction multiple data (SIMD) circuit.

Table 1: Concrete (online phase) comparison against Moz$\mathbb{Z}_{2^k}$arella. "Comm." denotes the communication complexity (counted in bits) per multiplication gate, and '$\mathcal{R}$' denotes the ring on which the protocol is running. For $\kappa = 40$, we use $(16, 45)$-RMFEs, while for $\kappa = 80$, we use $(27, 85)$-RMFEs.[3]

k	κ	Moz$\mathbb{Z}_{2^k}$arella		This work ($\Pi_{\mathrm{ZK}}^{m,n,t}$)	
		Comm.	$\mathcal{R}$	Comm.	$\mathcal{R}$
32	40	179	$\mathbb{Z}_{2^{130}}$	93	$\mathtt{GR}(2^{32}, 45)$
	80	302	$\mathbb{Z}_{2^{212}}$	104	$\mathtt{GR}(2^{32}, 85)$
64	40	211	$\mathbb{Z}_{2^{162}}$	183	$\mathtt{GR}(2^{64}, 45)$
	80	334	$\mathbb{Z}_{2^{244}}$	205	$\mathtt{GR}(2^{64}, 85)$

(2) Targeting the ZK protocol over fields with sublinear communication complexity, AntMan, we construct the first VOLE-based ZK protocol $\Pi_{\mathrm{slZK}}^{m,n,t}$ over $\mathbb{Z}_{2^k}$ with the same sublinear communication complexity. We remark that it seems difficult to achieve similar efficiency by the Moz$\mathbb{Z}_{2^k}$arella approach. For concrete efficiency, similar to AntMan, we also require a large circuit size (estimated at least 2^{20}) to allow the computational cost of setting up the new authentication coding scheme to be averaged out. For 40-bit statistical security, we estimate that $\Pi_{\mathrm{slZK}}^{m,n,t}$ outperforms $\Pi_{\mathrm{ZK}}^{m,n,t}$ when computing an SIMD circuit over $\mathbb{Z}_{2^{32}}$ for more than 16×12 copies of data [4].

(3) To complement our VOLE-based ZK protocols, we present efficient constructions for VOLE over Galois rings following the PCG paradigm. The first primal-LPN based construction is constant-round, and has practical efficiency. The second dual-LPN based construction is two-round at a cost of slightly larger computation complexity, and it plays a crucial role in the non-interactive secure computation setting [51]. We analyse the security of LPN over Galois rings. The underlying LPN assumptions over $\mathtt{GR}(2^k, d)$ are weaker than that of Moz$\mathbb{Z}_{2^k}$arella. As indicated in Moz$\mathbb{Z}_{2^k}$arella, they have to carefully select LPN parameters to mitigate the effect of a leakage that the adversary can learn c noise entries with probability $1/2^c$.

(4) By utilizing the algebraic structure of a Galois ring extension to a new level, we manage to adapt the VOLEitH techniques [5] to work for $\mathbb{Z}_{2^k}$. These

[3] We select RMFEs over binary field according to [22], and lift to Galois rings via the approach in [28].

[4] This estimation uses an estimation of the additively homomorphic encryption (AHE) ciphertext size $c < 8920$ bits.

new publicly verifiable NIZK protocols over $\mathbb{Z}_{2^k}$ inherit the majority of the desirable efficiency metrics from VOLE-based ZK protocols. Hence, they provide simpler, faster and possibly smaller solutions compared to the recent MPCitH ZK protocols over $\mathbb{Z}_{2^k}$ in [18].

1.2 Technical Overview

We sketch how we construct VOLE-based ZK protocols over $\mathbb{Z}_{2^k}$. In a high level, the first step is to adapt existing ZK protocols over Galois field to Galois ring $\mathtt{GR}(2^k, d)$. The adaption is straightforward, where we only need to "replace" the field with $\mathtt{GR}(2^k, d)$. We remark that VOLE-based ZK protocols over any field [60,63,61] actually work on a sufficiently large field for security guarantee. Similarly, for the security of Galois ring analogue protocols, the degree d of $\mathtt{GR}(2^k, d)$ is required to be sufficiently large accordingly.

The second step is to modify the above ZK protocols over $\mathtt{GR}(2^k, d)$ to efficiently prove statements over $\mathbb{Z}_{2^k}$. As $\mathtt{GR}(2^k, d)$ is a ring extension of $\mathbb{Z}_{2^k}$, we can simply view circuits over $\mathbb{Z}_{2^k}$ as circuits over $\mathtt{GR}(2^k, d)$, and naively applying ZK protocols over $\mathtt{GR}(2^k, d)$. However, working on $\mathtt{GR}(2^k, d)$ instead of $\mathbb{Z}_{2^k}$ already incurs d times overhead (d needs to be linear in the security parameter), and for malicious security, the prover needs to additionally prove that the witness $\mathbf{w}$ is over $\mathbb{Z}_{2^k}$. Our idea is to use RMFEs.

An RMFE over $\mathbb{Z}_{2^k}$ consists of two $\mathbb{Z}_{2^k}$-linear maps, $\phi : \mathbb{Z}_{2^k}^m \to \mathtt{GR}(2^k, d)$, and $\psi : \mathtt{GR}(2^k, d) \to \mathbb{Z}_{2^k}^m$, such that $\psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{y})) = \mathbf{x} * \mathbf{y}$, for any $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_{2^k}^m$, where m is some positive integer and $*$ denotes the entry-wise multiplication. Intuitively, ϕ "packs" m multiplications over $\mathbb{Z}_{2^k}$ to one multiplication over $\mathtt{GR}(2^k, d)$, while ψ "unpacks" the product over $\mathtt{GR}(2^k, d)$. Since ϕ, ψ are $\mathbb{Z}_{2^k}$-linear maps, m evaluations of a circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ can be simultaneously emulated by $\mathtt{GR}(2^k, d)$ operations, through applying ϕ, ψ iteratively. Existing works [22,23,1,34] that deployed RMFEs in MPC, have spent great efforts to guarantee that ϕ, ψ are applied honestly in order to achieve malicious security.

Observe that in the ZK setting, for proving circuit satisfiability, the prover (of the ZK protocol over $\mathtt{GR}(2^k, d)$) can compute all wire values of the m copies of circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ on his own, and invoke ϕ in parallel after all values are computed. Intuitively, achieving malicious security in this setting can be done more efficiently. Our main innovation in this part is a novel mechanism that allows the prover to efficiently prove honest RMFE encoding to the verifier.

Our re-embedding technique and basic ZK. Let $[x]$ denote that $x \in \mathtt{GR}(2^k, d)$ is authenticated by a linearly homomorphic MAC. In the ZK setting, the problem can be reduced to proving that for a given $[x]$, x belongs to the image of ϕ (denoted by $x \in \mathrm{Im}(\phi)$). Recall that the offline phase produces MACs for random values. Given $[\mu]$, $\mu \xleftarrow{\$} \mathtt{GR}(2^k, d)$, $[x]$ is obtained by the prover sending $\delta := x - \mu$ to the verifier ($[x] := [\mu] + \delta$ by additive homomorphism). Let $\tau = \phi \circ \psi$. According to RMFE properties presented in Section 2, we observe that

$$x = \mu + \delta \Longrightarrow \tau(x) = \tau(\mu + \delta) = \tau(\mu) + \tau(\delta),$$

and

$$x = \tau(x) \iff \mu + \delta = \tau(\mu) + \tau(\delta).$$

Note that for any $x \in \mathtt{GR}(2^k, d)$, $\tau(x) \in \mathrm{Im}(\phi)$ by definition, and assuming $\phi(\mathbf{1}) = 1$, we have $x \in \mathrm{Im}(\phi) \iff x = \tau(x)$ (Lemma 3). From the above observations, if $[\tau(\mu)]$ is also generated in the offline phase, then the two parties can compute $[\tau(x)] := [\tau(\mu)] + \tau(\delta)$. In some sense, $[x]$ is "re-embedded" into $[\tau(x)]$. Thus, we name it the *re-embedding* technique. We formulate the ideal functionality required for the offline phase as the *re-embedding VOLE* (embVOLE), and provide an efficient construction from the sacrifice idea. It is worth highlighting that our re-embedding technique also plays a crucial role in checking multiplications when constructing our first ZK protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$.

ZK with a sublinear online phase. For our second ZK protocol $\Pi_{\mathrm{slZK}}^{m,n,t}$ with sublinear communication complexity, we compile the above RMFE re-embedding with an online phase adapted from AntMan [61]. Consider an SIMD circuit with $m = m_1 \times m_2$ copies of data over $\mathbb{Z}_{2^k}$. We first use RMFEs to map m_1 copies of data over $\mathbb{Z}_{2^k}$ into one copy of data over $\mathtt{GR}(2^k, d)$. Then we apply a Galois ring analogue of the information-theoretic polynomial authentication code (IT-PAC) amortisation technique proposed in AntMan that operates on a batch of m_2 elements in $\mathtt{GR}(2^k, d)$. Our protocol has sublinear communication complexity mainly due to IT-PACs, and the use of RMFEs incurs only a constant ($d/m_1 > 1$) communication overhead. Therefore, in practice we select (m_1, d)-RMFEs with $\frac{d}{m_1}$ as small as possible (m_2 as large as possible) under the premise of $\mathtt{GR}(2^k, d)$ being sufficiently large to satisfy the security requirement. At a high level, an IT-PAC authenticates a polynomial, which is determined by m_2 elements in $\mathtt{GR}(2^k, d)$ via Lagrange interpolation. Namely, an IT-PAC authenticates m_2 elements simultaneously. Note that it seems difficult to obtain a $\mathbb{Z}_{2^{k+s}}$ analogue of IT-PAC, on which Moz$\mathbb{Z}_{2^k}$arella is working, as the maximum size of exceptional sets of $\mathbb{Z}_{2^{k+s}}$ is only 2. Therefore, we are not aware of any approach to obtain a sublinear ZK by combining Moz$\mathbb{Z}_{2^k}$arella with IT-PACs. Tricks of reducing computing a generic circuit to computing an SIMD circuit are proposed in [61] that we postpone to the end of Section 3. We quickly point out some possible disadvantages of the IT-PAC without dwelling on them. The interactive generation of IT-PAC increases the computation complexity considerably and moreover it is not public-coin.

Instantiations of VOLE over $\mathtt{GR}(2^k, d)$. The online phases of our ZK protocols require a single VOLE correlation of sufficiently large length. The similarity of Galois fields and Galois rings makes it natural to seamlessly generalize different VOLE protocols over Galois fields to VOLE protocols over Galois rings with all their different features well-preserved. However, when adapting the SoftSpokenOT [56]-style VOLE construction (which is crucial for applying VOLEitH), the large Galois ring constitutes a bottleneck of computation efficiency. We significantly reduce the computation by restricting the receiver's input to be sampled from its "subfield" and applying the repetition idea of [5].

2 Preliminaries

Notations. In this paper, bold letters (e.g., $\mathbf{a}, \mathbf{b}$) are used to denote vectors. Besides, we use x_i to denote the i-th component of the vector $\mathbf{x}$. We use $[a, b]$ (or $[a, b+1)$ sometimes) to denote the set of integers in the range from a to b, if $a = 1$, it is simplified by $[b]$, which is not to be confused with the MAC notation. We also use $\mathbf{x}[a : b]$ to denote the set $\{x_i \mid i \in [a, b]\}$. We use $x \xleftarrow{\$} \mathcal{R}$ to denote that x is uniformly sampled from a ring $\mathcal{R}$ and denote the uniform distribution over $\mathcal{R}$ by $\mathrm{U}_{\mathcal{R}}$. For a map $\phi : \mathcal{R}_1 \to \mathcal{R}_2$, we naturally extend it to be defined over vector space $\mathcal{R}_1^n$ and matrix space $\mathcal{R}_1^{m \times n}$. Let $\mathrm{Im}(\phi)$ denote the set $\{\phi(x) \mid x \in \mathcal{R}_1\}$ and $\mathrm{Ker}(\phi)$ denote the set $\{x \in \mathcal{R}_1 \mid \phi(x) = 0\}$.

Galois Rings. Let p be a prime, and $k, d \geq 1$ be integers. Let $f(X) \in \mathbb{Z}_{p^k}[X]$ be a monic polynomial of degree d such that $\overline{f(X)} := f(X) \mod p$ is irreducible over $\mathbb{F}_p$. Denote the Galois ring over $\mathbb{Z}_{p^k}$ of degree d by $\mathtt{GR}(p^k, d)$, which is a ring extension $\mathbb{Z}_{p^k}[X]/(f(X))$ of $\mathbb{Z}_{p^k}$. The readers may refer to [59] for a friendly exposition.

We emphasize that Galois rings have a special algebraic structure that, $\mathtt{GR}(p^k, d)/(p) \cong \mathbb{F}_{p^d}$, and every element a of $\mathtt{GR}(p^k, d)$ can be uniquely written as $a_0 + a_1 \cdot p + \ldots + a_{k-1} \cdot p^{k-1}$, where $a_i \in \mathbb{F}_{p^d}$, $i \in [0, k)$. Moreover, zero divisors of $\mathtt{GR}(p^k, d)$ are of the form $a_1 \cdot p + \ldots + a_{k-1} \cdot p^{k-1}$, for all $a_i \in \mathbb{F}_{p^d}$, $i \in [k-1]$. Therefore, $1/p^d$ fraction of elements are zero divisors in $\mathtt{GR}(p^k, d)$, or equivalently, $(1 - 1/p^d)$ fraction of elements are invertible. For polynomials over Galois rings, there is an upper bound on the number of roots.

Lemma 1 ([34]). *A nonzero degree-r polynomial over $\mathtt{GR}(p^k, d)$ has at most $rp^{(k-1)d}$ roots.*

Lemma 1 immediately implies that for any nonzero degree-r polynomial $f(x)$ over $\mathtt{GR}(p^k, d)$, we have that $\Pr\left[f(\alpha) = 0 \;\middle|\; \alpha \xleftarrow{\$} \mathtt{GR}(p^k, d)\right] \leq rp^{-d}$.

Reverse Multiplicative Friendly Embedding. Reverse Multiplicative Friendly Embedding (RMFE) was first introduced by Cascudo et al. [22], which allows packing multiple multiplications over a field $\mathbb{F}_q$ into one multiplication over an extension field $\mathbb{F}_{q^d}$. It was further shown by Cramer et al. [28] that RMFEs over finite fields can be lifted to Galois rings. Very recently, Escudero et al. [33] showed that RMFEs can be extended to have larger multiplication capacity. We recall the definition of RMFE, and then present some of its important properties.

Definition 1 (RMFE [28]). *Let p be a prime, $k, r, m, d \geq 1$ be integers. A pair (ϕ, ψ) is called an (m, d)-RMFE over $\mathtt{GR}(p^k, r)$ if $\phi : \mathtt{GR}(p^k, r)^m \to \mathtt{GR}(p^k, rd)$ and $\psi : \mathtt{GR}(p^k, rd) \to \mathtt{GR}(p^k, r)^m$ are two $\mathtt{GR}(p^k, r)$-linear maps such that*

$$\mathbf{x} * \mathbf{y} = \psi\big(\ \phi(\mathbf{x}) \ \cdot \ \phi(\mathbf{y})\ \big) \tag{1}$$

for all $\mathbf{x}, \mathbf{y} \in \mathtt{GR}(p^k, r)^m$. Here $*$ denotes component-wise product of vectors.

By Proposition 2 of [33], if (ϕ, ψ) is an RMFE, then ϕ is injective while ψ is surjective. Therefore, it is necessary for m less than or equal to d. The following lemma shows the existence of RMFE with a constant ratio $\frac{d}{m}$.

Lemma 2 (Existence of RMFE [22,28]). *There exists a family of (m, d)-RMFEs over Galois ring $\mathtt{GR}(p^k, r)$ with $d = \mathcal{O}(m)$.*

Given an (m, d)-RMFE (ϕ, ψ), we can always assume that $\phi(\mathbf{1}) = 1$. First, we show that $\phi(\mathbf{1})$ is invertible in $\mathtt{GR}(p^k, rd)$ by contradiction. Assume $\phi(\mathbf{1})$ is a zero divisor and hence $p^{k-1} \cdot \phi(\mathbf{1}) = 0$. Due to the linearity of ϕ, we also have $p^{k-1} \cdot \phi(\mathbf{1}) = \phi(p^{k-1} \cdot \mathbf{1})$. This implies that $p^{k-1} \cdot \mathbf{1}$ is another preimage of 0, which leads to a contradiction since ϕ is injective. Then, define $\phi' : \mathtt{GR}(p^k, r)^m \to \mathtt{GR}(p^k, rd)$ as $\phi'(\mathbf{a}) := \phi(\mathbf{a}) \cdot \phi(\mathbf{1})^{-1}$ and $\psi' : \mathtt{GR}(p^k, rd) \to \mathtt{GR}(p^k, r)^m$ as $\psi'(b) := \psi(b \cdot \phi(\mathbf{1}))$. It is straightforward to verify that (ϕ', ψ') is an (m, d)-RMFE with $\phi'(\mathbf{1}) = 1$. From now on, we assume $\phi(\mathbf{1}) = 1$ without explicitly mentioning it.

Lemma 3. *Let (ϕ, ψ) be an (m, d)-RMFE over Galois ring $\mathtt{GR}(p^k, r)$, then $\mathtt{GR}(p^k, rd) = \mathrm{Im}(\phi) \oplus \mathrm{Ker}(\psi)$.*

Proof. As ϕ is injective and ψ is surjective, ψ induces a bijection from the set $\mathrm{Im}(\phi)$ to $\mathtt{GR}(p^k, r)^m$ since $\psi(\phi(\mathbf{x})) = \psi(\phi(\mathbf{x}) \cdot \phi(\mathbf{1})) = \mathbf{x} * \mathbf{1} = \mathbf{x}$. Together with the fact that $\psi : \mathtt{GR}(p^k, rd) \to \mathtt{GR}(p^k, r)^m$ is a $\mathtt{GR}(p^k, r)$-linear map, we have $\mathtt{GR}(p^k, rd) = \mathrm{Im}(\phi) \oplus \mathrm{Ker}(\psi)$. □

We also define $\tau = \phi \circ \psi : \mathtt{GR}(p^k, rd) \to \mathtt{GR}(p^k, rd)$. As ϕ, ψ are $\mathbb{Z}_{p^k}$ linear, so is τ. A simple observation shows that $x \in \mathrm{Im}(\phi)$ if and only if $\tau(x) = x$.

In this work, we mainly consider RMFEs with $r = 1$ and $p = 2$, i.e., a family of (m, d)-RMFEs over $\mathbb{Z}_{2^k}$. As shown in [22], such families of RMFEs exist with $\lim_{m\to\infty} \frac{d}{m} = 4.92$.

VOLE and MAC. (Random) vector oblivious linear function evaluation (VOLE) is a functionality that allows two parties $P_{\mathcal{S}}, P_{\mathcal{R}}$ to obtain random correlated values. In more detail, the sender $P_{\mathcal{S}}$ obtains two vectors $\mathbf{M}, \mathbf{x}$, while the receiver $P_{\mathcal{R}}$ obtains a scalar Δ and a vector $\mathbf{K}$ such that $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$. We formalize the ideal functionality of VOLE over Galois ring $\mathtt{GR}(2^k, d)$ in Figure 1.

The above VOLE correlation can be viewed as Message Authentication Codes (MACs) that authenticate $\mathbf{x}$, denoted by $[\mathbf{x}]$. We then call $\mathbf{M}$ the MAC tags, $\mathbf{K}$ the local keys and Δ the global key. It is easy to see that such MAC is linearly homomorphic. Given authenticated values $[x_1], \ldots, [x_\ell]$ and public coefficients $c, c_1, \ldots, c_\ell \in \mathtt{GR}(p^k, d)$, the two parties can locally compute $[y] = c + \sum_{i\in[\ell]} c_i \cdot [x_i]$ by setting $y = c + \sum_{i\in[\ell]} c_i \cdot x_i$, $M_y = \sum_{i\in[\ell]} c_i \cdot M_{x_i}$, and $K_y = \Delta \cdot c + \sum_{i\in[\ell]} c_i \cdot K_{x_i}$. In particular, we have $[y] = [x] + (y - x)$. Then given $[x]$ for a random x, the two parties can obtain $[y]$ by having $P_{\mathcal{S}}$ send $y - x$ to $P_{\mathcal{R}}$.

Security Model and Functionalities. We prove the security of our protocols in the universal composability (UC) framework [21]. In particular, we consider active adversary and static corruption. More details can be found in Appendix A. The goal of this work is to design secure zero-knowledge protocols realizing functionality $\mathcal{F}_{\mathrm{ZK}}^m$, which allows a prover to prove knowledge of m witnesses

Functionality $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$

Init: Upon receiving $(\texttt{Init})$ from both parties, sample $\Delta \xleftarrow{\$} \texttt{GR}(2^k,d)$ if $P_{\mathcal{R}}$ is honest, and receive $\Delta \in \texttt{GR}(2^k,d)$ from the adversary $\mathcal{A}$ otherwise. Store Δ and send it to $P_{\mathcal{R}}$. All further $(\texttt{Init})$ commands will be ignored.

Extend: Upon receiving $(\texttt{Extend}, n)$ from both parties, proceed as follows:

1. If $P_{\mathcal{R}}$ is honest, sample $\mathbf{K} \xleftarrow{\$} \texttt{GR}(2^k,d)^n$. Otherwise receive $\mathbf{K}$ from $\mathcal{A}$.
2. If $P_{\mathcal{S}}$ is honest, sample $\mathbf{x} \xleftarrow{\$} \texttt{GR}(2^k,d)^n$ and compute $\mathbf{M} := \mathbf{K} - \Delta \cdot \mathbf{x} \in \texttt{GR}(2^k,d)^n$. Otherwise, receive $\mathbf{x} \in \texttt{GR}(2^k,d)^n$ and $\mathbf{M} \in \texttt{GR}(2^k,d)^n$ from $\mathcal{A}$ and then recompute $\mathbf{K} := \mathbf{M} + \Delta \cdot \mathbf{x}$.
3. Send $(\mathbf{x}, \mathbf{M})$ to $P_{\mathcal{S}}$ and $\mathbf{K}$ to $P_{\mathcal{R}}$.

Fig. 1: Ideal functionality for VOLE over $\texttt{GR}(2^k,d)$.

satisfying the same circuit $\mathcal{C}$. Details of $\mathcal{F}_{\text{ZK}}^m$ are in Figure 2. In particular, an interactive ZK protocol is public-coin, if each message from the verifier sent to the prover is a random string.

Functionality $\mathcal{F}_{\text{ZK}}^m$

Upon receiving $(\texttt{prove}, \mathcal{C}, \mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(m)})$ from a prover $\mathcal{P}$ and $(\texttt{verify}, \mathcal{C})$ from a verifier $\mathcal{V}$, where the same circuit $\mathcal{C}$ is input by both parties, send $(\texttt{true})$ to $\mathcal{V}$ if $\mathcal{C}(\mathbf{w}^{(j)}) = 1$ for all $j \in [m]$ and $(\texttt{false})$ otherwise.

Fig. 2: Functionality for zero-knowledge proofs for circuit satisfiability.

We also require some fundamental functionalities for our VOLE constructions. The equality test functionality $\mathcal{F}_{\text{EQ}}$ (see Figure 12, Appendix A.2) allows two parties $\mathcal{P}$ and $\mathcal{V}$ to learn if their inputs are equal and reveals $\mathcal{P}$'s input to $\mathcal{V}$. The oblivious transfer functionality $\mathcal{F}_{\text{OT}}$ (see Figure 13, Appendix A.2) receives a single bit b from the receiver and two strings m_0, m_1 from the sender, and then returns m_b to the receiver.

3 Zero-Knowledge Protocols over $\mathbb{Z}_{2^k}$

Inspired by the methodology of [34], where the authors constructed efficient dishonest majority MPC over $\mathbb{Z}_{2^k}$ by first giving an MPC over $\texttt{GR}(2^k,d)$ and then converting it to work over $\mathbb{Z}_{2^k}$, we make use of RMFEs in the context of VOLE-based ZK protocols, and develop novel, highly efficient techniques.

To obtain efficient zero-knowledge protocols over $\mathbb{Z}_{2^k}$, we first introduce a new functionality $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$ and present a construction that UC-realizes it in the $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$-hybrid model in Section 3.1. In Section 3.2, we present a public-coin ZK protocol over $\mathbb{Z}_{2^k}$. In Section 3.3, we construct a ZK protocol over $\mathbb{Z}_{2^k}$ with communication complexity sublinear in the circuit size. Both ZK constructions are in the $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$-hybrid model.

3.1 Re-embedding VOLE over $\texttt{GR}(2^k, d)$

Jumping ahead, to construct ZK protocols over $\mathbb{Z}_{2^k}$, our first step is to construct ZK protocols over $\texttt{GR}(2^k, d)$. Following the blueprint of ZK protocols over the Galois fields, e.g., [60,32,63], the first step is quite straightforward. The key observation is that the soundness error of these protocols is related to the fraction of zero divisors of the underlying ring. For example, the ZK protocol over a large field $\mathbb{F}_q$ in QuickSilver [63] has soundness error $\mathcal{O}(1/q)$. Therefore, realizing a Galois ring $\texttt{GR}(2^k, d)$ analogue of QuickSilver induces soundness error $\mathcal{O}\big(1/2^d\big)$, which can be set negligible by choosing a sufficiently large parameter $d = \Omega(\kappa)$. Here, κ is the statistical security parameter.

The main obstacle of constructing ZK over $\mathbb{Z}_{2^k}$ lies in the second step, where we need to do the conversion. A naive solution is to run the ZK protocol over $\texttt{GR}(2^k, d)$ from the first step by treating each element in $\mathbb{Z}_{2^k}$ as an element in $\texttt{GR}(2^k, d)$. However, this already incurs $\Omega(\kappa)$ overhead to achieve a negligible soundness error, needless to say that, the prover is additionally required to prove that his inputs (i.e., the witness) are over $\mathbb{Z}_{2^k}$. The above solution essentially uses the naive embedding $\mathbb{Z}_{2^k} \hookrightarrow \texttt{GR}(2^k, d)$, which can be viewed as an *inefficient* RMFE over $\mathbb{Z}_{2^k}$ with the ratio $\frac{d}{m} = \frac{d}{1} = \Omega(\kappa)$. Fortunately, there exist more efficient (m, d)-RMFEs over $\mathbb{Z}_{2^k}$ [22] with an asymptotically constant ratio, which are exactly the RMFEs that we will make use of to accomplish the conversion.

Let $\phi : \mathbb{Z}_{2^k}^m \to \texttt{GR}(2^k, d)$ and $\psi : \texttt{GR}(2^k, d) \to \mathbb{Z}_{2^k}^m$ be an (m, d)-RMFE pair over $\mathbb{Z}_{2^k}$. Suppose the prover has m witnesses over $\mathbb{Z}_{2^k}$, and he will use ϕ to map them to one "witness" over $\texttt{GR}(2^k, d)$. However, there are two issues that we have to overcome if we use such general and more efficient RMFEs for conversion. The first one is that the prover is required to prove that his inputs are over $\text{Im}(\phi)$, as opposed to $\mathbb{Z}_{2^k}$ in the naive embedding case. The second one is how to guarantee honest circuit evaluation. Unlike the naive embedding that has infinite multiplication capacity, (ϕ, ψ) only preserves one time multiplication inherently.

To solve the above issues, we propose a novel, highly efficient technique, the re-embedding VOLE. We show how re-embedding VOLE solves the first issue, and defer the solution to the second issue to Section 3.2. Our key observation is that $\texttt{GR}(2^k, d)$ is the direct sum of $\text{Im}(\phi)$ and $\text{Ker}(\psi)$ (Lemma 3), and the inputs over $\mathbb{Z}_{2^k}$ one-to-one correspond to a vector over $\text{Im}(\phi)$. Therefore, it suffices to find an efficient approach that removes the kernel part of $[\mathbf{x}]$, i.e., re-embed $[\mathbf{x}]$ to $[\tau(\mathbf{x})]$. A direct way is to reveal $\mathbf{x} - \tau(\mathbf{x})$ to the receiver[5], so that they can

[5] This is the mistake that we made in the previous version of this paper.

Functionality $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$

$\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ extends the existing VOLE functionality $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$ (Figure 1). **Init** and **Extend** are identical to those in $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$, respectively. Let (ϕ, ψ) be an (m, d)-RMFE pair over $\mathbb{Z}_{2^k}$, and $\tau := \phi \circ \psi$.

Extend-pair: Upon receiving $(\texttt{Extend-pair}, n)$ from both parties, proceed as follows:

1. If $P_{\mathcal{S}}$ is honest, sample $\mathbf{x}, \mathbf{M}^{(1)}, \mathbf{M}^{(2)} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$; otherwise, receive $\mathbf{x}, \mathbf{M}^{(1)}, \mathbf{M}^{(2)} \in \mathtt{GR}(2^k, d)^n$ from $\mathcal{A}$.
2. If $P_{\mathcal{R}}$ is honest, compute $\mathbf{K}^{(1)} := \mathbf{M}^{(1)} + \Delta \cdot \mathbf{x}, \mathbf{K}^{(2)} := \mathbf{M}^{(2)} + \Delta \cdot \tau(\mathbf{x}) \in \mathtt{GR}(2^k, d)^n$; otherwise, receive $\mathbf{K}^{(1)}, \mathbf{K}^{(2)} \in \mathtt{GR}(2^k, d)^n$ from $\mathcal{A}$ and recompute $\mathbf{M}^{(1)} := \mathbf{K}^{(1)} - \Delta \cdot \mathbf{x}$, $\mathbf{M}^{(2)} := \mathbf{K}^{(2)} - \Delta \cdot \tau(\mathbf{x})$.
3. Send $(\mathbf{x}, \mathbf{M}^{(1)}, \mathbf{M}^{(2)})$ to $P_{\mathcal{S}}$ and $(\mathbf{K}^{(1)}, \mathbf{K}^{(2)})$ to $P_{\mathcal{R}}$.

Fig. 3: Ideal functionality for re-embedding VOLE over $\mathtt{GR}(2^k, d)$.

compute $[\tau(\mathbf{x})] := [\mathbf{x}] - (\mathbf{x} - \tau(\mathbf{x}))$. However, this would leak information, as typically $\mathbf{x}$ on intermediate wires might be the product of two elements of $\text{Im}(\phi)$. To this end, we let them obtain random pairs of $([\boldsymbol{\mu}], [\tau(\boldsymbol{\mu})])$, so that they can compute $[\tau(\mathbf{x})] := [\tau(\boldsymbol{\mu})] + \tau(\boldsymbol{\delta})$, where $\boldsymbol{\delta} := \mathbf{x} - \boldsymbol{\mu}$ is revealed to the receiver. Since $\boldsymbol{\mu}$ is conjectured to be uniformly random, $\boldsymbol{\delta}$ will not leak any information about $\mathbf{x}$. Through this way, $[\mathbf{x}]$ is "automatically" re-embeded into $[\tau(\mathbf{x})]$, as desired. We call $([\boldsymbol{\mu}], [\tau(\boldsymbol{\mu})])$ re-embedding pair MACs.

We define the re-embedding VOLE functionality $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 3. To construct a secure re-embedding VOLE protocol, the two parties generate $n+s$ re-embedding pair MACs and then sacrifice the extra s re-embedding pair MACs. The sacrifice is done by taking s random $\mathbb{Z}_{2^k}$-linear combinations of n re-embedding pair MACs to obtain s equations, with each masked by an extra re-embedding pair MAC. If any one of the n re-embedding pair MACs is not honestly generated, the correctness check will fail, except with probability at most $2^{-s} + 2^{-d}$, which can be made negligible by setting s, d large enough (e.g., $s = d = \kappa + 1$). We give the protocol $\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 4, whose security is guaranteed by Theorem 1. The proof is deferred to Appendix C.1.

Theorem 1. *$\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model. In particular, there exists a* PPT *simulator such that no* PPT *environment $\mathcal{Z}$ can distinguish the real world execution from the ideal world simulation except with advantage at most $2^{-s} + 2^{-d}$.*

To obtain n re-embedding pair MACs, our construction consumes a VOLE correlation of length $(2n+2s)$, and communicates $(n+3s+1)$ Galois ring elements and ns coefficients in $\mathbb{Z}_{2^k}$, or equivalently, on average $(1+(3s+1)/n+s/d)$ Galois

ring elements. As n is sufficiently large for most ZK applications, the overhead for constructing a single re-embedding pair MAC is close to $\mathcal{O}(1)$ Galois ring elements.

Protocol $\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$

Init: Both parties send $(\mathtt{Init})$ to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$, which returns $\Delta \in \mathtt{GR}(2^k,d)$ to $P_{\mathcal{R}}$.

Extend: Both parties send $(\mathtt{Extend}, n)$ to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$, which returns $\mathbf{M}, \mathbf{x} \in \mathtt{GR}(2^k,d)^n$ to $P_{\mathcal{S}}$ and $\mathbf{K} \in \mathtt{GR}(2^k,d)^n$ to $P_{\mathcal{R}}$.

Extend-pair: To generate n authenticated re-embedding pairs, both parties proceed as follows:

1. **Construct:**
   (a) Both parties send $(\mathtt{Extend}, 2n+2s)$ to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $\mathbf{M}^{(1)}, \mathbf{M}', \mathbf{x}, \mathbf{y} \in \mathtt{GR}(2^k,d)^{n+s}$, and $P_{\mathcal{R}}$ receives $\mathbf{K}^{(1)}, \mathbf{K}' \in \mathtt{GR}(2^k,d)^{n+s}$, such that $\mathbf{K}^{(1)} = \mathbf{M}^{(1)} + \mathbf{x} \cdot \Delta$, and $\mathbf{K}' = \mathbf{M}' + \mathbf{y} \cdot \Delta$ hold. Thus, the parties now obtain $[x_i], [y_i], i \in [n+s]$.
   (b) $P_{\mathcal{S}}$ computes $\boldsymbol{\eta} := \tau(\mathbf{x}) - \mathbf{y}$, then sends $\boldsymbol{\eta} \in \mathtt{GR}(2^k,d)^{n+s}$ to $P_{\mathcal{R}}$.
   (c) $P_{\mathcal{S}}$ sets $\mathbf{M}^{(2)} := \mathbf{M}'$, and $P_{\mathcal{R}}$ sets $\mathbf{K}^{(2)} := \mathbf{K}' + \boldsymbol{\eta} \cdot \Delta$. Note that $\mathbf{K}^{(2)} = \mathbf{M}^{(2)} + \tau(\mathbf{x}) \cdot \Delta$ holds, so the parties now obtain $[\tau(x_i)], i \in [n+s]$.
2. **Sacrifice:**
   (a) $P_{\mathcal{R}}$ samples $\boldsymbol{\chi}^{(1)}, ..., \boldsymbol{\chi}^{(s)} \xleftarrow{\$} \mathbb{Z}_{2^k}^n$, and sends them to $P_{\mathcal{S}}$.
   (b) For $i \in [s]$, $P_{\mathcal{S}}$ computes $a_i = x_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot x_j$, $b_i = \tau(x_{n+i}) + \sum_{j\in[n]} \chi_j^{(i)} \cdot \tau(x_j)$, $\hat{M}_i^{(1)} := M_{x_{n+i}}^{(1)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot M_{x_j}^{(1)}$, and $\hat{M}_i^{(2)} := M_{x_{n+i}}^{(2)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot M_{x_j}^{(2)}$. Let $\hat{\mathbf{M}}^{(1)} = (\hat{M}_1^{(1)}, \ldots, \hat{M}_s^{(1)})$, and $\hat{\mathbf{M}}^{(2)} = (\hat{M}_1^{(2)}, \ldots, \hat{M}_s^{(2)})$. Send $\mathbf{a} = (a_1, \ldots, a_s), \mathbf{b} = (b_1, \ldots, b_s)$ to $P_{\mathcal{R}}$.
   (c) $P_{\mathcal{R}}$ checks $\mathbf{b} = \tau(\mathbf{a})$. If the check fails, $P_{\mathcal{R}}$ aborts. Otherwise, $P_{\mathcal{R}}$ computes $\tilde{M}_i^{(1)} := K_{x_{n+i}}^{(1)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_{x_j}^{(1)} - a_i \cdot \Delta$, and $\tilde{M}_i^{(2)} := K_{x_{n+i}}^{(2)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_{x_j}^{(2)} - b_i \cdot \Delta$ for $i \in [s]$. Let $\tilde{\mathbf{M}}^{(1)} = (\tilde{M}_1^{(1)}, \ldots, \tilde{M}_s^{(1)})$ and $\tilde{\mathbf{M}}^{(2)} = (\tilde{M}_1^{(2)}, \ldots, \tilde{M}_s^{(2)})$.
   (d) $P_{\mathcal{S}}$ sends $\hat{\mathbf{M}}^{(1)}, \hat{\mathbf{M}}^{(2)}$ to $P_{\mathcal{R}}$. The latter checks whether $\hat{\mathbf{M}}^{(1)} = \tilde{\mathbf{M}}^{(1)}$, and $\hat{\mathbf{M}}^{(2)} = \tilde{\mathbf{M}}^{(2)}$ and aborts if the check fails.
3. **Output:** Output $([x_i], [\tau(x_i)])$, for $i \in [n]$.

Fig. 4: Protocol for authenticating re-embedding pairs over $\mathtt{GR}(2^k,d)$ in the $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model.

3.2 Public-Coin Zero-Knowledge Protocol over $\mathbb{Z}_{2^k}$

Equipped with our re-embedding VOLE technique and inspired by QuickSilver [63], we construct a highly efficient public-coin ZK protocol over $\mathbb{Z}_{2^k}$ in Figure 5.

Suppose the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ have agreed on a circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ with n inputs and t multiplication gates, and $\mathcal{P}$ has m witnesses $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(m)} \in \mathbb{Z}_{2^k}^n$. By calling $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$, they can obtain $n+t$ re-embedding pair MACs for $\boldsymbol{\mu} \xleftarrow{\$} \mathtt{GR}(2^k,d)^n$ and $\boldsymbol{\nu} \xleftarrow{\$} \mathtt{GR}(2^k,d)^t$, and a MAC $[\pi]$ (will be used as a mask), where $\pi \xleftarrow{\$} \mathtt{GR}(2^k,d)$. $\mathcal{P}$ then computes $\boldsymbol{\omega} := \phi(\mathbf{w}^{(1)}, ..., \mathbf{w}^{(m)}) \in \mathtt{GR}(2^k,d)^n$, and sends $\boldsymbol{\delta} := \boldsymbol{\omega} - \boldsymbol{\mu}$ to $\mathcal{V}$. They compute $[\tau(\boldsymbol{\omega})] := [\tau(\boldsymbol{\mu})] + \tau(\boldsymbol{\delta})$. Next, the two parties evaluate the circuit in a topological order. For `Add` gates, the MAC of the output wire can be computed locally.

However, for `Mul` gates, we face a problem as mentioned in Section 3.1, that RMFEs can only preserve one multiplication. Good news is that in the ZK setting, the prover $\mathcal{P}$ can compute all wire values on his own as he holds witnesses of the circuit. Therefore, the output MAC of a `Mul` gate can be obtained by consuming one random MAC. However, bad news is that since all authenticated values are supposed to be in $\mathrm{Im}(\phi)$, the usual multiplication equality no longer holds. More specifically, for the i-th multiplication gate in $\mathcal{C}$ with inputs $\omega_\alpha, \omega_\beta \in \mathrm{Im}(\phi)$ and output $\omega_\gamma \in \mathrm{Im}(\phi)$, essentially we need to ensure $\psi(\omega_\alpha) * \psi(\omega_\beta) = \psi(\omega_\gamma)$, which is not equivalent to $\omega_\alpha \cdot \omega_\beta = \omega_\gamma$. The former equation is not easy to verify since it is over $\mathbb{Z}_{2^k}$ but the authenticated values are over $\mathtt{GR}(2^k,d)$. We show that this issue can be bypassed by our re-embedding VOLE. Given $([\nu_i], [\tau(\nu_i)])$ from $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$, $\mathcal{P}$ sends $d_i := \omega_\alpha \cdot \omega_\beta - \nu_i$ to $\mathcal{V}$, then they can obtain $[\omega_\gamma] := [\tau(\omega_\alpha \cdot \omega_\beta)] = [\tau(\nu_i)] + \tau(d_i)$. We remark that $\omega_\gamma = \tau(\omega_\alpha \cdot \omega_\beta) \iff \psi(\omega_\gamma) = \psi(\omega_\alpha) * \psi(\omega_\beta)$, as we always assume $\phi(\mathbf{1}) = 1$. To verify that each d_i is computed honestly, the two parties additionally compute $[\hat{\omega}_\gamma] := [\omega_\alpha \cdot \omega_\beta] = [\nu_i] + d_i$. One can observe that

$$\begin{aligned} B_i :&= K_{\omega_\alpha} \cdot K_{\omega_\beta} - K_{\hat{\omega}_\gamma} \cdot \Delta \\ &= (M_{\omega_\alpha} + \Delta \cdot \omega_\alpha) \cdot (M_{\omega_\beta} + \Delta \cdot \omega_\beta) - (M_{\hat{\omega}_\gamma} + (\omega_\alpha \cdot \omega_\beta) \cdot \Delta) \cdot \Delta \\ &= \underbrace{(M_{\omega_\alpha} \cdot M_{\omega_\beta})}_{A_{0,i}} + \underbrace{(\omega_\alpha \cdot M_{\omega_\beta} + \omega_\beta \cdot M_{\omega_\alpha} - M_{\hat{\omega}_\gamma})}_{A_{1,i}} \cdot \Delta \end{aligned}$$

holds if d_i is correct. Therefore, it can be used to detect malicious behaviors by letting $\mathcal{P}$ send $A_{0,i}$, $A_{1,i}$ to $\mathcal{V}$. At a high level, we check multiplications for $\hat{\omega}_\gamma$, and automatically re-embed $\hat{\omega}_\gamma$ to $\omega_\gamma = \tau(\hat{\omega}_\gamma)$ via re-embedding VOLE. Moreover, we can use a random linear combination technique to check all t multiplications simultaneously. Briefly, $\mathcal{V}$ sends uniformly random coefficients $\{\chi_i \in \mathtt{GR}(2^k,d)\}_{i\in[t]}$ to $\mathcal{P}$, who then returns to $\mathcal{V}$ the linear combination $\sum_{i\in[t]} \chi_i \cdot A_{0,i}$ and $\sum_{i\in[t]} \chi_i \cdot A_{1,i}$ masked with M_π and π, respectively. In fact, as $\mathtt{GR}(2^k,d)/(2) \cong \mathbb{F}_{2^d}$, $\boldsymbol{\chi}$ can be sampled from $\mathbb{F}_{2^d}^t$. Intuitively, the entropy of $\boldsymbol{\chi}$ is still sufficient.

Finally, for the output wire ω_h, if both parties follow the protocol honestly, the equation $K_{\omega_h} = M_{\omega_h} + 1 \cdot \Delta$ should hold. Thus, $\mathcal{P}$ opens $[\omega_h]$, which is supposed to be [1], by sending M_{ω_h} to $\mathcal{V}$. We have the following theorem, with its proof deferred to Appendix C.2.

Theorem 2. *Protocol $\Pi_{\text{ZK}}^{m,n,t}$ communicates $(kd+d)/m$ bits per multiplication gate, and UC-realizes $\mathcal{F}_{\text{ZK}}^m$ in the $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model with soundness error $2^{-(d-2)}$ and information-theoretic security.*

Protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$

The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ have agreed on a circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ with n inputs and t multiplication gates, and $\mathcal{P}$ holds m witnesses $\mathbf{w}^{(i)} \in \mathbb{Z}_{2^k}^n$ such that $\mathcal{C}(\mathbf{w}^{(i)}) = 1$, $i \in [m]$.

Offline phase

1. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Init})$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, and $\mathcal{V}$ receives $\Delta \in \mathtt{GR}(2^k, d)$.
2. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend-pair}, n+t)$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, which returns authenticated pairs $([\mu_i], [\tau(\mu_i)])_{i\in[n]}$, $([\nu_j], [\tau(\nu_j)])_{j\in[t]}$, where all μ_i, ν_j are sampled uniformly at random in $\mathtt{GR}(2^k, d)$.
3. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend}, 1)$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, which returns authenticated value $[\pi]$, where π is sampled uniformly at random in $\mathtt{GR}(2^k, d)$.

Online phase

1. For input $W = (\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, ..., \mathbf{w}^{(m)}) \in \mathbb{Z}_{2^k}^{n\times m}$, $\mathcal{P}$ computes $\boldsymbol{\omega} := \phi(W) \in \mathtt{GR}(2^k, d)^n$, and sends $\delta_i := \omega_i - \mu_i$, $i \in [n]$ to $\mathcal{V}$. Both parties locally compute $[\tau(\omega_i)] := [\tau(\mu_i)] + \tau(\delta_i)$, for $i \in [n]$.
2. For each gate $(\alpha, \beta, \gamma, T) \in \mathcal{C}$, in a topological order:
   - If T=`Add`, then $\mathcal{P}$ and $\mathcal{V}$ locally compute $[\omega_\gamma] := [\omega_\alpha] + [\omega_\beta]$.
   - If T=`Mul` and this is the i-th multiplication gate, then $\mathcal{P}$ sends $d_i := \omega_\alpha \cdot \omega_\beta - \nu_i$ to $\mathcal{V}$, and both parties locally compute $[\omega_\gamma] := [\tau(\nu_i)] + \tau(d_i)$, and $[\hat{\omega}_\gamma] := [\nu_i] + d_i$.
3. For the i-th multiplication gate, the parties hold $([\omega_\alpha], [\omega_\beta], [\hat{\omega}_\gamma])$ with $K_{\omega_j} = M_{\omega_j} + \omega_j \cdot \Delta$ for $j \in \{\alpha, \beta\}$, and $K_{\hat{\omega}_\gamma} = M_{\hat{\omega}_\gamma} + \hat{\omega}_\gamma \cdot \Delta$.
   - $\mathcal{P}$ computes $A_{0,i} := M_{\omega_\alpha} \cdot M_{\omega_\beta} \in \mathtt{GR}(2^k, d)$ and $A_{1,i} := \omega_\alpha \cdot M_{\omega_\beta} + \omega_\beta \cdot M_{\omega_\alpha} - M_{\hat{\omega}_\gamma} \in \mathtt{GR}(2^k, d)$.
   - $\mathcal{V}$ computes $B_i := K_{\omega_\alpha} \cdot K_{\omega_\beta} - K_{\hat{\omega}_\gamma} \cdot \Delta \in \mathtt{GR}(2^k, d)$.
4. $\mathcal{P}$ and $\mathcal{V}$ perform the following check.
   (a) $\mathcal{P}$ sets $A_0^* := M_\pi, A_1^* := \pi$, and $\mathcal{V}$ sets $B^* := K_\pi$ so that $B^* = A_0^* + A_1^* \cdot \Delta$.
   (b) $\mathcal{V}$ draws a uniformly random $\boldsymbol{\chi}$ from $\mathbb{F}_{2^d}^t$ and sends it to $\mathcal{P}$.
   (c) $\mathcal{P}$ computes $X := \sum_{i\in[t]} \chi_i \cdot A_{0,i} + A_0^* \in \mathtt{GR}(2^k, d)$ and $Y := \sum_{i\in[t]} \chi_i \cdot A_{1,i} + A_1^* \in \mathtt{GR}(2^k, d)$, and sends (X, Y) to $\mathcal{V}$.
   (d) $\mathcal{V}$ computes $Z := \sum_{i\in[t]} \chi_i \cdot B_i + B^* \in \mathtt{GR}(2^k, d)$, and checks whether $Z = X + Y \cdot \Delta$ holds. If the check fails, $\mathcal{V}$ outputs `false` and aborts.
5. For the single output wire ω_h, both parties hold $[\omega_h]$.
   - $\mathcal{P}$ sends M_{ω_h} to $\mathcal{V}$.
   - $\mathcal{V}$ checks whether $K_{\omega_h} = M_{\omega_h} + 1 \cdot \Delta$. If the check fails, $\mathcal{V}$ outputs `false`. Otherwise, $\mathcal{V}$ outputs `true`.

Fig. 5: Zero-knowledge protocol for circuit satisfiability over $\mathbb{Z}_{2^k}$ in the $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model.

Our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ (Figure 5) transfers a single element from $\mathtt{GR}(2^k,d)$ and a random coefficient in $\mathbb{F}_{2^d}$ per multiplication gate, yielding amortized communication complexity of $(kd+d)/m$-bit, which is independent of the statistical security parameter κ as d/m is constant. Note that both in $\Pi_{\mathrm{ZK}}^{m,n,t}$ and $\Pi_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, the verifier (receiver) only sends random strings to the prover (sender). Thus, we obtain a public-coin ZK protocol over $\mathbb{Z}_{2^k}$ in the $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model. This is critical for constructing a publicly verifiable NIZK over $\mathbb{Z}_{2^k}$ in Section 5.1.

3.3 Sublinear Zero-Knowledge Protocol over $\mathbb{Z}_{2^k}$

In this section, we focus on proofs with communication complexity sublinear in the circuit size. Our construction is initially inspired by AntMan [61], which constructed interactive ZK for any field with sublinear communication complexity via developing a new, powerful technique, the Polynomial Authentication Code (PAC).

Similar to Section 3.2, we follow the methodology of first constructing a sublinear ZK protocol over $\mathtt{GR}(2^k,d)$, and then converting it to one over $\mathbb{Z}_{2^k}$ via RMFEs. The first step involves adapting PAC to work over Galois rings, and for the conversion step, we crucially rely on our re-embedding technique as well.

PAC is essentially an information-theoretic polynomial commitment, which allows to authenticate a polynomial. In more detail, PAC is a generalization of VOLE-based MAC, where the sender $P_{\mathcal{S}}$ holds a MAC tag $M \in \mathtt{GR}(2^k,d)$, and a polynomial $f(\cdot) \in \mathtt{GR}(2^k,d)[X]$, while the receiver $P_{\mathcal{R}}$ holds a polynomial key $\Lambda \in \mathtt{GR}(2^k,d)$, a global key $\Delta \in \mathtt{GR}(2^k,d)$, and a local key $K \in \mathtt{GR}(2^k,d)$ such that $K = M + f(\Lambda)\cdot\Delta$. We remark that the PAC for $f(\cdot)$, denoted by $[\![f(\cdot)]\!]$, can be viewed as a MAC for $f(\Lambda)$. Due to its intrinsic nature, PAC can be used to authenticate a batch of values. This relies on the fact that a batch of (say, m_2) values over $\mathtt{GR}(2^k,d)$ uniquely determine a polynomial $f(\cdot) \in \mathtt{GR}(2^k,d)[X]$ (of degree less than m_2) via Lagrange interpolation, as long as the evaluation points are picked from a large enough exceptional set [6] of $\mathtt{GR}(2^k,d)$. Intuitively, PAC significantly reduces the communication complexity in the sense that it turns m_2 MACs into one MAC $[f(\Lambda)]$.

To construct PAC over Galois rings, we follow the footprint of AntMan [61] by employing an additively homomorphic encryption (AHE) scheme over Galois rings MHz2k [25]. The protocol Π_{PAC} is presented in Figure 18 (Appendix C.3).

Before we illustrate our solutions to the conversion step, let us sketch how the protocol works. Suppose that $\mathcal{P}$ and $\mathcal{V}$ have agreed on a circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ with n inputs and t multiplication gates, and $\mathcal{P}$ holds $m = m_1 m_2$ witnesses. Let (ϕ,ψ) be an (m_1,d)-RMFE pair over $\mathbb{Z}_{2^k}$ and $\zeta \in \mathtt{GR}(2^k,d)$ with order 2^d-1, which admits an exceptional set $T := \{0,1,\zeta,...,\zeta^{2^d-2}\}$. Now $\mathcal{P}$ uses ϕ to map $m_1 m_2$ witnesses over $\mathbb{Z}_{2^k}$ to m_2 vectors $\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(m_2)}$ in $\mathtt{GR}(2^k,d)^n$, and the two parties first authenticate these values via VOLE-based MACs, obtaining $\{[\boldsymbol{\omega}^{(i)}]\}_{i\in[m_2]}$.

[6] Exceptional set is a subset E of $\mathtt{GR}(2^k,d)$ such that the difference of each pair of elements in E is invertible.

Next, the two parties generate the corresponding PACs $[\![u_1(\cdot)]\!], \ldots, [\![u_n(\cdot)]\!]$ such that $u_j(\alpha_i) = \omega_j^{(i)}$ for $i \in [m_2]$ and $j \in [n]$. Here $\{\alpha_1, \ldots, \alpha_{m_2}\} \subset T$ is an evaluation point set. Given such PACs for inputs, they then evaluate the circuit in a topological order.

Procedure `BatchCheck`

Let T be the set $\{0, 1, \zeta, ..., \zeta^{2^d-2}\}$, where $\zeta \in \mathtt{GR}(2^k, d)$ is of order $2^d - 1$. Let d_1, d_2, m, ℓ be parameters. Let $\{\alpha_1, ..., \alpha_m\}$ and $\{\beta_1, ..., \beta_m\}$ be two public subsets of T and $H : \{0,1\}^\kappa \to \mathbb{Z}_{2^k}^\ell$ be a random oracle.

Inputs: $\mathcal{P}$ and $\mathcal{V}$ have the following inputs: two sets of PACs $\{[\![f_1(\cdot)]\!], ..., [\![f_\ell(\cdot)]\!]\}$ and $\{[\![g_1(\cdot)]\!], ..., [\![g_\ell(\cdot)]\!]\}$, where $f_i(\cdot)$ is a polynomial of degree $\leq d_1$ and $g_i(\cdot)$ is a polynomial of degree $\leq d_2$ over $\mathtt{GR}(2^k, d)$ for $i \in [\ell]$.

Consistency Check: $\mathcal{P}$ and $\mathcal{V}$ check $f_j(\alpha_i) = \tau(g_j(\beta_i))$ for all $i \in [m], j \in [\ell]$.

1. **Linear combination phase:** Before the polynomial key Λ is opened, $\mathcal{P}$ and $\mathcal{V}$ proceed as follows:
   (a) $\mathcal{P}$ picks two random polynomial $r(\cdot)$ and $s(\cdot)$ over $\mathtt{GR}(2^k, d)$ with degree d_1 and d_2, respectively, such that $r(\alpha_i) = \tau(s(\beta_i))$, for $i \in [m]$. Then, $\mathcal{P}$ and $\mathcal{V}$ run the `Pre-Gen` procedure of Π_{PAC} to pre-generate two PACs $[\![r(\cdot)]\!]$ and $[\![s(\cdot)]\!]$.
   (b) $\mathcal{V}$ samples a $\mathtt{seed} \leftarrow \{0,1\}^\kappa$ and sends it to $\mathcal{P}$. Then, two parties compute $(\chi_1, ..., \chi_\ell) := H(\mathtt{seed}) \in \mathbb{Z}_{2^k}^\ell$.
   (c) $\mathcal{P}$ and $\mathcal{V}$ locally compute $[\![f(\cdot)]\!] := \sum_{j\in[\ell]} \chi_j \cdot [\![f_j(\cdot)]\!] + [\![r(\cdot)]\!]$ and $[\![g(\cdot)]\!] := \sum_{j\in[\ell]} \chi_j \cdot [\![g_j(\cdot)]\!] + [\![s(\cdot)]\!]$. Then, $\mathcal{P}$ sends the polynomial pair $(f(\cdot), g(\cdot))$ to $\mathcal{V}$, who checks that $f(\cdot), g(\cdot)$ have respective degree d_1 and d_2, and $f(\alpha_i) = \tau(g(\beta_i))$ holds for all $i \in [m]$. If the check fails, $\mathcal{V}$ aborts.
2. **Check phase:** After Λ is opened, $\mathcal{P}$ and $\mathcal{V}$ locally compute $[\mu] := [f(\Lambda)] - f(\Lambda)$ and $[\nu] := [g(\Lambda)] - g(\Lambda)$. Then, $\mathcal{P}$ sends M_μ, M_ν to $\mathcal{V}$ (i.e., $\mathcal{P}$ opens $[\mu]$ and $[\nu]$ to $\mathcal{V}$), and $\mathcal{V}$ checks $M_\mu = K_\mu$ and $M_\nu = K_\nu$. If the check fails, $\mathcal{V}$ aborts.

Fig. 6: Procedure for checking the τ-consistency of polynomial evaluations for two sets of PACs.

To accomplish the conversion step, we need to consider the effects of introducing RMFEs. The first issue is that, we need to guarantee the inputs $\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m_2)}$ are in $\mathrm{Im}(\phi)^n$. Similar to our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$, this can be solved by the re-embedding technique. The second issue is how to guarantee honest circuit evaluation, which is much more complicated, as the circuit is evaluated via PACs rather than MACs. Observe that in the construction of AntMan [61], to evaluate a multiplication gate with input PACs $[\![u_a(\cdot)]\!]$ and $[\![u_b(\cdot)]\!]$, the two parties generate two PACs $[\![u_c(\cdot)]\!]$ and $[\![\widetilde{u}_c(\cdot)]\!]$ such that $\widetilde{u}_c(\cdot) = u_a(\cdot) \cdot u_b(\cdot)$ is a polynomial of degree $\leq 2m_2 - 2$ and $u_c(\cdot)$ is a polynomial of degree $\leq m_2 - 1$ with $u_c(\alpha_i) = \widetilde{u}_c(\alpha_i)$ for $i \in [m_2]$. Next, $\mathcal{P}$ proves to $\mathcal{V}$ that among the four PACed

polynomials: (1) $\widetilde{u}_c(\cdot)$ is indeed the correct multiplication of input polynomials $u_a(\cdot), u_b(\cdot)$, and (2) the polynomials $u_c(\cdot)$ (degree $\leq m_2 - 1$) and $\widetilde{u}_c(\cdot)$ (degree $\leq 2m_2 - 2$) share the same values in the m_2 evaluation points.

For the multiplications in our situation, we need to guarantee that $\psi(u_c(\alpha_i)) = \psi(u_a(\alpha_i)) * \psi(u_b(\alpha_i))$, for all $i \in [m_2]$. For this purpose, we instead define $u_c(\cdot)$ (degree $\leq m_2 - 1$) such that

$$u_c(\alpha_i) = \tau(\widetilde{u}_c(\alpha_i)) = \tau(u_a(\alpha_i) \cdot u_b(\alpha_i)) \quad \text{for all} \quad i \in [m_2]. \tag{2}$$

This can be seen as re-embedding a PAC $[\![\widetilde{u}_c(\cdot)]\!]$ into a PAC $[\![u_c(\cdot)]\!]$ such that (2) holds. All that remain to be shown are to check (1) $\widetilde{u}_c(\cdot) = u_a(\cdot) \cdot u_b(\cdot)$ as in AntMan [61], and (2) the polynomials $u_c(\cdot)$ with degree $\leq m_2 - 1$ and $\widetilde{u}_c(\cdot)$ with degree $\leq 2m_2 - 2$ are "τ-consistent", i.e., $u_c(\alpha_i) = \tau(\widetilde{u}_c(\alpha_i))$ for all $i \in [m_2]$.

The former check crucially relies on an observation that the polynomial key Λ can be revealed to $\mathcal{P}$, after all polynomials needed to be authenticated are authenticated. Then the PACs naturally collapse to MACs for the polynomial evaluations on Λ. Therefore, we can employ the multiplication check procedure of $\Pi_{\text{ZK}}^{m,n,t}$, where $\mathcal{P}$ can only pass the check with probability at most $3/2^d$, if using some incorrect polynomial $\widetilde{u}_c(\cdot)$.

For the latter issue, we remark that if $m_2 = 1$, we can solve it by compiling our re-embedding technique, as shown in Section 3.2. However, it is unclear how to guarantee the τ-consistency of $[\![u_c(\cdot)]\!]$ and $[\![\widetilde{u}_c(\cdot)]\!]$ for a general $m_2 \geq 1$. Therefore, we need to find another strategy. Fortunately, we observe that the `BatchCheck` procedure of AntMan [61] can be adapted to check τ-consistency in batch. In more detail, given $\{[\![v_j(\cdot)]\!]\}_{j\in[m_2]}, \{[\![\widetilde{v}_j(\cdot)]\!]\}_{j\in[m_2]}$, as $\tau = \phi \circ \psi$ is a $\mathbb{Z}_{2^k}$-linear map, $\mathcal{V}$ can check τ-consistency by $\mathcal{P}$ opening random $\mathbb{Z}_{2^k}$-linear combinations of the two sets of polynomials. To avoid potential information leakage from linear combinations, we mask the opened polynomials with a pair of random polynomials that satisfy the degree constraint and τ-consistency. Besides, $\mathcal{P}$ should convince $\mathcal{V}$ that the opened polynomials $f(\cdot), g(\cdot)$ are consistent to their PACs. This can be done by opening $[f(\Lambda)], [g(\Lambda)]$ to $\mathcal{V}$ after $\mathcal{P}$ receives Λ. The details of our `BatchCheck` procedure are presented in Figure 6.

The protocol is given in Figure 7 and Figure 8. We have the following theorem that guarantees security with its proof in Appendix C.3.

Theorem 3. *Protocol $\Pi_{\text{slZK}}^{m,n,t}$ UC-realizes functionality $\mathcal{F}_{\text{ZK}}^{m}$ that proves circuit satisfiability over $\mathbb{Z}_{2^k}$ in the $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model and the random oracle model with soundness error at most $\frac{2m_2+3}{2^d} + \mathsf{negl}(\kappa)$.*

Our protocol $\Pi_{\text{slZK}}^{m,n,t}$ communicates 2 AHE ciphertexts per multiplication gate. The main costs for checking multiplications consist of two parts: transferring random coefficient $\boldsymbol{\chi} \in \mathbb{F}_{2^d}^t$ in Step 5-(e) in Figure 8, and transferring 2 polynomials over $\mathtt{GR}(2^k, d)$ of respective degree $m_2 - 1$ and $2m_2 - 2$ in the Linear combination phase of the `BatchCheck` procedure. Assume the ciphertext size is c, these contribute to $2ct + td + 3m_2kd$ bits in total, which is sublinear in mt. By dividing mt, the amortized complexity per multiplication gate is $(\frac{2c+d}{m} + \frac{3kd}{m_1 t})$-bit.

Protocol $\Pi_{\text{slZK}}^{m,n,t}$-Part I

The prover $\mathcal{P}$ and the verifier $\mathcal{V}$ have agreed on the following public inputs.

- A circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ with n inputs and t multiplication gates;
- An (m_1, d)-RMFE pair (ϕ, ψ) over $\mathbb{Z}_{2^k}$, and an element $\zeta \in \texttt{GR}(2^k, d)$ with order $2^d - 1$;
- A set $T = \{0, 1, \zeta, ..., \zeta^{2^d-2}\}$ and distinct elements $\alpha_1, ..., \alpha_{m_2} \in T$;
- Polynomials $\xi_i(X) = \prod_{j\in[m_2], j\neq i}(X-\alpha_j)/(\alpha_i-\alpha_j) \in \texttt{GR}(2^k, d)[X]$ with degree $(m_2 - 1)$ for $i \in [m_2]$, which are known as the Lagrange basis polynomials.

In addition, $\mathcal{P}$ holds $m = m_1 \cdot m_2$ witnesses $\mathbf{w}^{(i)} \in \mathbb{Z}_{2^k}^n$ such that $\mathcal{C}(\mathbf{w}^{(i)}) = 1$, for all $i \in [m]$.

Offline phase

1. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Init})$ to $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$, and $\mathcal{V}$ receives $\Delta \in \texttt{GR}(2^k, d)$.
2. $\mathcal{P}$ and $\mathcal{V}$ run the `Poly-Key` procedure of Π_{PAC} with input $2m_2 - 2$, and $\mathcal{V}$ receives a uniform polynomial key $\Lambda \in \texttt{GR}(2^k, d)$.
3. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend-pair}, nm_2)$ to $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$, which returns authenticated pairs $\{([\mu_j^{(i)}], [\tau(\mu_j^{(i)})])\}_{i\in[m_2], j\in[n]}$, where all $\mu_j^{(i)}$ are sampled uniformly at random in $\texttt{GR}(2^k, d)$.
4. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend}, n + 2t)$ to $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$, which returns authenticated values $\{[\nu_i]\}_{i\in[n]}$ and $\{[\pi_j]\}_{j\in[2t]}$, where all ν_i, π_j are sampled uniformly at random in $\texttt{GR}(2^k, d)$.

Online phase

1. For input $W^{(i)} := (\mathbf{w}^{((i-1)\cdot m_1+1)}, \mathbf{w}^{((i-1)\cdot m_1+2)}, ..., \mathbf{w}^{((i-1)\cdot m_1+m_1)}) \in \mathbb{Z}_{2^k}^{n\times m_1}$, $\mathcal{P}$ computes $\boldsymbol{\omega}^{(i)} := \phi(W^{(i)}) \in \texttt{GR}(2^k, d)^n$, and sends $\boldsymbol{\delta}^{(i)} := \boldsymbol{\omega}^{(i)} - \boldsymbol{\mu}^{(i)}$, $i \in [m_2]$ to $\mathcal{V}$. Both parties locally compute $[\tau(\omega_j^{(i)})] := [\tau(\mu_j^{(i)})] + \tau(\delta_j^{(i)})$, for $i \in [m_2], j \in [n]$.
2. For $j \in [n]$, for the j-th group of m_2 input gates with input vector $(\omega_j^{(1)}, ..., \omega_j^{(m_2)})$, $\mathcal{P}$ defines a polynomial $u_j(\cdot)$ with degree $\leq (m_2 - 1)$ such that $u_j(\alpha_i) = \omega_j^{(i)}$ for $i \in [m_2]$.
3. $\mathcal{P}$ and $\mathcal{V}$ run the `Pre-Gen` procedure of Π_{PAC} with input $[\boldsymbol{\nu}]$, to pre-generate PACs $[\![u_1(\cdot)]\!], ..., [\![u_n(\cdot)]\!]$.
4. For each gate $(a, b, c, T) \in \mathcal{C}$, in a topological order:
   - If T=`Add`, then $\mathcal{P}$ and $\mathcal{V}$ locally compute $[\![u_c(\cdot)]\!] := [\![u_a(\cdot)]\!] + [\![u_b(\cdot)]\!]$.
   - If T=`Mul` and this is the j-th groups of multiplication gates, where $j \in [t]$, then $\mathcal{P}$ computes a polynomial $\widetilde{v}_j(\cdot) := \widetilde{u}_c(\cdot) := u_a(\cdot) \cdot u_b(\cdot) \in \texttt{GR}(2^k, d)[X]$ with degree $\leq (2m_2 - 2)$ and a polynomial $v_j := u_c$ with degree $\leq (m_2 - 1)$ such that $u_c(\alpha_i) = \tau(\widetilde{u}_c(\alpha_i))$ for all $i \in [m_2]$. Then, $\mathcal{P}$ and $\mathcal{V}$ run the `Pre-Gen` procedure of Π_{PAC} with input $[\pi_{2j}], [\pi_{2j+1}]$, to pre-generate two PACs $[\![u_c(\cdot)]\!]$ and $[\![\widetilde{u}_c(\cdot)]\!]$.

Fig. 7: Zero-knowledge protocol for SIMD circuit satisfiability over $\mathbb{Z}_{2^k}$ with sublinear communication in the $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$-hybrid model-**Part I**.

Protocol $\Pi_{\mathrm{slZK}}^{m,n,t}$-Part II

5. $\mathcal{P}$ and $\mathcal{V}$ perform the following multiplication check.
   (a) $\mathcal{P}$ and $\mathcal{V}$ execute the Linear combination phase of the `BatchCheck` procedure with parameters $(m_2-1), (2m_2-2), m_2, t$ and a common evaluation subset $\{\alpha_1, ..., \alpha_{m_2}\} \subset T$ on inputs $\{[\![v_1(\cdot)]\!], ..., [\![v_t(\cdot)]\!]\}$ and $\{[\![\widetilde{v}_1(\cdot)]\!], ..., [\![\widetilde{v}_t(\cdot)]\!]\}$ to check the degree constraint and τ-consistency, namely, the degree of each $v_j(\cdot) \leq (m_2-1)$, the degree of each $\widetilde{v}_j(\cdot) \leq (2m_2-2)$, and $v_j(\alpha_i) = \tau(\widetilde{v}_j(\alpha_i))$ holds for all $i \in [m_2], j \in [t]$.
   (b) $\mathcal{P}$ and $\mathcal{V}$ run the `Gen` procedure of Π_{PAC}, where Λ is revealed to $\mathcal{P}$. Then $\mathcal{V}$ can compute the local keys on all PACs, and $\mathcal{P}$ can compute the polynomial evaluations on Λ. For the j-th multiplication gate $(a, b, c, \mathtt{Mul}) \in \mathcal{C}$, $\mathcal{P}$ and $\mathcal{V}$ now hold $[u_a(\Lambda)], [u_b(\Lambda)], [\widetilde{u}_c(\Lambda)]$. $\mathcal{P}$ computes $A_{0,j} := M_{u_a(\Lambda)} \cdot M_{u_b(\Lambda)} \in \mathtt{GR}(2^k, d)$ and $A_{1,j} := u_a(\Lambda) \cdot M_{u_b(\Lambda)} + u_b(\Lambda) \cdot M_{u_a(\Lambda)} - M_{\widetilde{u}_c(\Lambda)} \in \mathtt{GR}(2^k, d)$. $\mathcal{V}$ computes $B_j := K_{u_a(\Lambda)} \cdot K_{u_b(\Lambda)} - \Delta \cdot K_{\widetilde{u}_c(\Lambda)} \in \mathtt{GR}(2^k, d)$.
   (c) $\mathcal{P}$ and $\mathcal{V}$ execute the check phase of the `BatchCheck` procedure to complete the above check. If the check fails, $\mathcal{V}$ aborts.
   (d) $\mathcal{P}$ sets $A_0^* := M_\pi, A_1^* := \pi$, and $\mathcal{V}$ sets $B^* := K_\pi$ so that $B^* = A_0^* + \Delta \cdot A_1^*$.
   (e) $\mathcal{V}$ draws a uniformly random $\boldsymbol{\chi}$ from $\mathbb{F}_{2^d}^t$ and sends it to $\mathcal{P}$.
   (f) $\mathcal{P}$ computes $X := \sum_{j\in[t]} \chi_j \cdot A_{0,j} + A_0^* \in \mathtt{GR}(2^k, d)$ and $Y := \sum_{j\in[t]} \chi_j \cdot A_{1,j} + A_1^* \in \mathtt{GR}(2^k, d)$, and sends (X, Y) to $\mathcal{V}$.
   (g) $\mathcal{V}$ computes $Z := \sum_{j\in[t]} \chi_j \cdot B_j + B^* \in \mathtt{GR}(2^k, d)$, and checks whether $Z = X + \Delta \cdot Y$ holds. If the check fails, $\mathcal{V}$ outputs `false` and aborts.
6. $\mathcal{P}$ and $\mathcal{V}$ do the following input and output checks. $\mathcal{P}$ convinces $\mathcal{V}$ that $[\![u_j(\cdot)]\!]$ is consistent to $([\omega_j^{(1)}], ..., [\omega_j^{(m_2)}])$ for $j \in [n]$, and the values on all output gates are 1.
   (i) For $j \in [n]$, $\mathcal{P}$ and $\mathcal{V}$ compute MAC $[z_j] := \sum_{i\in[m_2]} \xi_i(\Lambda) \cdot [\omega_j^{(i)}] - [u_j(\Lambda)]$. Then, $\mathcal{P}$ opens $[z_j]$, $\mathcal{V}$ continues if and only if $z_j = 0$ for all $j \in [n]$. This check makes sense since each $u_j(\cdot)$ can be computed from Lagrange interpolation and MACs are linearly homomorphic.
   (ii) Let $[u(\cdot)]$ be the PAC associated with the output wire of $\mathcal{C}$. $\mathcal{P}$ sends $M_{u(\Lambda)}$ to $\mathcal{V}$, who checks whether $K_{u(\Lambda)} = M_{u(\Lambda)} + 1 \cdot \Delta$. If the check fails, $\mathcal{V}$ outputs `false`. Otherwise, $\mathcal{V}$ outputs `true`.

Fig. 8: Zero-knowledge protocol for SIMD circuit satisfiability over $\mathbb{Z}_{2^k}$ with sublinear communication in the $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model – **Part II**.

We remark that the overall communication complexity of $\Pi_{\text{slZK}}^{m,n,t}$ is linear in the witness size, as the two parties need to obtain MACs for all witnesses at the beginning of the online phase. Besides, $\Pi_{\text{slZK}}^{m,n,t}$ is not public-coin, due to the interactive generation of PACs.

Reduction from Evaluating Arbitrary Circuit to SIMD Circuit. There are two methods in the literature to transform a problem on a given circuit into one on a related SIMD circuit. The first one is to arrange an arbitrary circuit into an SIMD circuit (cf. [29,23]) by dividing gates into layers of addition and multiplication gates, which introduces an overhead that depends on the topology of the given circuit (for a large class of *well formed* circuits, this overhead is quite small). The second method [61] is specific to proving circuit satisfiability. Since the prover $\mathcal{P}$ holds the witness, he can calculate all wire values and authenticate them in batch as evaluating some SIMD circuit. Note that for an arbitrary circuit, the outputs of some gates may be the inputs of other gates. Thus, each wire value now needs to be authenticated twice, once as the output of some gate, and once as the input of another gate. As a result, $\mathcal{V}$ needs to additionally verify the consistency of two authenticated values on the same wire, which may increase the communication complexity considerably.

We remark that the second method is compatible with our PAC-based construction, still yielding an online phase with sublinear communication. The reason is that we can perform the additional verification by slightly modifying the `BatchCheck` procedure to check some specific $\mathbb{Z}_{2^k}$-components rather than τ-consistency, i.e., all $\mathbb{Z}_{2^k}$-components. It is also possible to combine the above two methods, where one can first arrange a generic circuit into an SIMD circuit, viewed as a generic circuit over $\mathtt{GR}(2^k, d)$, and then apply the second method. Readers are advised to perform implementation level optimizations according to the circuit topology.

4 VOLE over Galois Rings

In this section, we present efficient constructions for VOLE over Galois rings following the PCG-style VOLE paradigm. We construct two efficient VOLE protocols assuming the hardness of LPN problems over Galois rings. Specifically, we present an interactive construction based on primal-LPN, and discuss security of LPN over Galois rings. Due to space constraint, we defer the two-round construction based on dual-LPN to Appendix E.

We construct efficient VOLE over $\mathtt{GR}(2^k, d)$, adapting the PCG-style VOLE construction of Wolverine [60]. We also provide a two-round construction adapted from [16] in Appendix E. The construction involves two main steps. The first step is to construct single point VOLE (spVOLE), and the second step is to use LPN to locally convert multiple spVOLE instances of the same length into a VOLE correlation with much longer length. To begin with, we focus on constructing spVOLE over $\mathtt{GR}(2^k, d)$, where $\mathbf{u}$ (held by the sender) in the VOLE correlation $\mathbf{v} = \mathbf{w} + \mathbf{u} \cdot \Delta$ has only one non-zero entry. Similar to that in Moz$\mathbb{Z}_{2^k}$arella [4], we further require each non-zero entry to be a unit of $\mathtt{GR}(2^k, d)$, in order to

mitigate a modulo attack for LPN over Galois rings (Theorem 6). The associated functionality $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$ is given in Figure 17 (Appendix A.2).

Single-point VOLE. We first review puncturable pseudorandom function [15] (PPRF) based on GGM tree [41], which is a main subroutine for constructing spVOLE. Informally, a PPRF allows to distribute n pseudorandom values among the two parties, where the receiver $P_{\mathcal{R}}$ obtains all of them from a key, while the sender $P_{\mathcal{S}}$ can compute them all except for one value on his choice from a punctured key. We recall the GGM algorithm in Figure 19 (Appendix D.2) [7].

Now, suppose the two parties $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ have already obtained random Galois ring elements $\{v_j\}_{j\in[0,n)\setminus\{\alpha\}}$ and $\{v_j\}_{j\in[0,n)}$ from PPRF, respectively, where $\alpha \in [0,n)$ is randomly picked by $P_{\mathcal{S}}$. Then $P_{\mathcal{S}}$ can define two vectors $\mathbf{w}, \mathbf{u} \in \mathtt{GR}(2^k,d)^n$ such that $w_j = v_j, u_j = 0$, for $j \neq \alpha$ (we do not care about w_α, u_α for now). It can be directly observed that $\mathbf{v} = \mathbf{w} + \mathbf{u}\cdot\Delta$ holds, if ignoring the entry indexed by α. To obtain the desired relation at the entry α, we can "inject" a VOLE correlation $\gamma = \delta + \beta\cdot\Delta$ to the target entry. Specifically, let $P_{\mathcal{R}}$ send $g := \gamma - \sum_{j\in[0,n)} v_j$ to $P_{\mathcal{S}}$, who then in turn computes $w_\alpha := \delta - (g + \sum_{i\neq\alpha} w_i)$. One can easily verify that $v_\alpha = w_\alpha + \beta\cdot\Delta$. Thus, setting $u_\alpha := \beta$, the two parties obtain the desired spVOLE correlation $\mathbf{v} = \mathbf{w} + \mathbf{u}\cdot\Delta$.

The above spVOLE construction is only semi-honestly secure. In the malicious setting, the adversary may either cheat in the PPRF procedure or send an incorrect g. As shown in Wolverine [60], to achieve malicious security, it suffices to guarantee that the spVOLE correlation holds in each entry. We remark that this is a major challenge in constructing spVOLE over $\mathbb{Z}_{2^k}$, and Moz$\mathbb{Z}_{2^k}$arella [4] overcame it by using two check mechanisms in a sophisticated way. In our situation, we manage to apply a random-linear combination check, which is much simpler. Let $\chi_i \in \mathtt{GR}(2^k,d)$, $i \in [0,n)$ be uniformly random elements chosen by $P_{\mathcal{S}}$. We first observe that

$$\sum_{i\in[0,n)} \chi_i \cdot v_i = \sum_{i\in[0,n)} \chi_i \cdot w_i + \chi_\alpha \cdot \beta \cdot \Delta. \tag{3}$$

However, we cannot directly use (3) for consistency check, since $\chi_\alpha\cdot\beta\cdot\Delta$ can not be locally computed by any party and it may leak some information. Fortunately, this can be solved by masking (3) with a VOLE correlation $y = z + \chi_\alpha\cdot\beta\cdot\Delta$. Now it is sufficient for the two parties to check (4) instead.

$$\sum_{i\in[0,n)} \chi_i \cdot v_i - y = \sum_{i\in[0,n)} \chi_i \cdot w_i - z. \tag{4}$$

Details of our spVOLE protocol are in Figure 9. We then have the following theorem with proof in Appendix D.2.

[7] We remark that in GGM-based PPRF constructions, $P_{\mathcal{S}}$ plays the role of OT receiver while $P_{\mathcal{R}}$ plays the role of OT sender, resulting in incompatibility of PCG-style VOLE constructions and the VOLEitH technique.

Protocol $\Pi_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$

Init: Both parties send $(\mathtt{Init})$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$, which returns Δ to $P_{\mathcal{R}}$.

SP-Extend: On input $n = 2^h$, the two parties proceeds as follows:

1. **Construct:**
   (a) Both parties send $(\mathtt{Extend}, 1)$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $a, c \in \mathtt{GR}(2^k, d)$ and $P_{\mathcal{R}}$ receives $b \in \mathtt{GR}(2^k, d)$ such that $b = c + a \cdot \Delta$ holds.
   (b) $P_{\mathcal{S}}$ samples $\alpha \xleftarrow{\$} [0, n)$, and draws a uniformly random β in the set of units of $\mathtt{GR}(2^k, d)$. $P_{\mathcal{S}}$ sets a vector $\mathbf{u} \in \mathtt{GR}(2^k, d)^n$ with $u_\alpha = \beta$ and $u_i = 0$ for all $i \neq \alpha$.
   (c) $P_{\mathcal{S}}$ sets $\delta := c$ and sends $a' := \beta - a \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{R}}$. $P_{\mathcal{R}}$ computes $\gamma := b + a' \cdot \Delta \in \mathtt{GR}(2^k, d)$. Note that $\gamma = \delta - \beta \cdot \Delta$, so the two parties now obtain $[\beta]$.
   (d) $P_{\mathcal{R}}$ computes $s \leftarrow \mathtt{GGM.KeyGen}(1^\lambda)$, and runs $\mathtt{GGM.Gen}(1^n, s)$ to obtain $(\{v_j\}_{j\in[0,n)}, \{(K_{0,i}, K_{1,i})\}_{i\in[h]})$. Let $\alpha = \sum_{i\in[h]} 2^{h-i}\alpha_i$, where $\alpha_i \in \{0, 1\}$, and let $\bar{\alpha}_i$ denote the complement of α_i. For $i \in [h]$, $P_{\mathcal{S}}$ sends $\bar{\alpha}_i$ to $\mathcal{F}_{\mathrm{OT}}$ (as OT receiver) while $P_{\mathcal{R}}$ sends $(K_{0,i}, K_{1,i})$ to $\mathcal{F}_{\mathrm{OT}}$ (as OT sender), then $P_{\mathcal{S}}$ receives $K_{\bar{\alpha}_i,i}$. $P_{\mathcal{S}}$ runs $\mathtt{GGM.Eval}(\alpha, \{K_{\bar{\alpha}_i,i}\}_{i\in[h]})$ and gets $\{v_j\}_{j\neq\alpha}$.
   (e) $P_{\mathcal{R}}$ sends $g := \gamma - \sum_{j\in[0,n)} v_j \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ defines a vector $\mathbf{w} \in \mathtt{GR}(2^k, d)^n$ such that $w_i := v_i$ for $i \neq \alpha$ and $w_\alpha := \delta - (g + \sum_{i\neq\alpha} w_i)$. Note that $\mathbf{v} = \mathbf{w} + \mathbf{u} \cdot \Delta$.
2. **Check:**
   (a) Both parties send $(\mathtt{Extend}, 1)$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $x, z \in \mathtt{GR}(2^k, d)$ and $P_{\mathcal{R}}$ receives $y^* \in \mathtt{GR}(2^k, d)$ where $y^* = z + x \cdot \Delta$.
   (b) $P_{\mathcal{S}}$ samples $\chi_i \xleftarrow{\$} \mathtt{GR}(2^k, d)$ for $i \in [0, n)$, and computes $x^* := \chi_\alpha \cdot \beta - x \in \mathtt{GR}(2^k, d)$. $P_{\mathcal{S}}$ sends $(\{\chi_i\}_{i\in[0,n)}, x^*)$ to $P_{\mathcal{R}}$. $P_{\mathcal{R}}$ computes $y := y^* + \Delta \cdot x^* \in \mathtt{GR}(2^k, d)$. Now, we have $y = z + \chi_\alpha \cdot \beta \cdot \Delta$.
   (c) $P_{\mathcal{S}}$ computes $V_{P_{\mathcal{S}}} := \sum_{i\in[0,n)} \chi_i \cdot w_i - z$, and sends $V_{P_{\mathcal{S}}}$ to $\mathcal{F}_{\mathrm{EQ}}$ as $\mathcal{P}$. $P_{\mathcal{R}}$ computes $V_{P_{\mathcal{R}}} := \sum_{i\in[0,n)} \chi_i \cdot v_i - y$, and send $V_{P_{\mathcal{R}}}$ to $\mathcal{F}_{\mathrm{EQ}}$ as $\mathcal{V}$. $P_{\mathcal{R}}$ receives $V_{P_{\mathcal{S}}}$. $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ abort if the equality test fails.
3. **Output:** $P_{\mathcal{S}}$ outputs $(\mathbf{w}, \mathbf{u})$, and $P_{\mathcal{R}}$ outputs $\mathbf{v}$.

Fig. 9: Protocol for single-point VOLE over $\mathtt{GR}(2^k, d)$ in the $(\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{EQ}})$-hybrid model.

Theorem 4. *Assuming the existence of secure* PRG*s,* $\Pi_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ *UC-realizes* $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ *functionality in the* $(\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{EQ}})$*-hybrid model. In particular, no* PPT *environment* $\mathcal{Z}$ *can distinguish the real world execution from an ideal world simulation except with advantage at most* $1/2^d + \mathsf{negl}(\lambda)$.

From spVOLE to VOLE. The extension procedure of constructing VOLE from spVOLE follows prior works, e.g., [64,60]. Let $A \in \mathtt{GR}(2^k,d)^{m\times n}$ be a generating matrix and $\mathcal{D}$ a noise distribution over $\mathtt{GR}(2^k,d)^n$, for which the primal LPN assumption holds. We refer the readers to Appendix B for LPN preliminaries. Suppose $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ have obtained a VOLE correlation of length $(m+n)$, written as $\mathbf{v} = \mathbf{w} + \mathbf{u}\cdot\Delta$ and $\mathbf{b} = \mathbf{c} + \mathbf{e}\cdot\Delta$, where $\mathbf{u} \in \mathtt{GR}(2^k,d)^m$ and $\mathbf{e} \leftarrow \mathcal{D}$. We have

$$\underbrace{(\mathbf{v}\cdot A + \mathbf{b})}_{\mathbf{K}} = \underbrace{(\mathbf{w}\cdot A + \mathbf{c})}_{\mathbf{M}} + \underbrace{(\mathbf{u}\cdot A + \mathbf{e})}_{\mathbf{x}}\cdot\Delta,$$

where $(\mathbf{M},\mathbf{x})$ and $\mathbf{K}$ can be locally computed by $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$, respectively. Moreover, by the primal LPN assumption, $\mathbf{x} = \mathbf{u}\cdot A + \mathbf{e}$ is computationally indistinguishable from a uniformly random vector in $\mathtt{GR}(2^k,d)^n$.

In the literature, $\mathcal{D}$ is usually instantiated with the so-called regular noise distribution, where the error vector with Hamming weight t can be divided into t blocks with each block having only one non-zero entry. Therefore, we can obtain the VOLE correlation $\mathbf{b} = \mathbf{c} + \mathbf{e}\cdot\Delta$ from t repetitions of spVOLE with length n/t. Observe that the spVOLE in Figure 9 requires a VOLE correlation of length two to produce one spVOLE. Thus, the above construction essentially extends a VOLE correlation of length $m+2t$ into one of length n. In addition, as showed in prior works [64,60], we can reserve part of the output VOLE correlation for the next iteration of the extension procedure, to further improve efficiency. Details of the protocol are presented in Figure 10.

LPN with static leakage. Similar to [60,4], our $\Pi_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ essentially allows a corrupted $P_{\mathcal{R}}$ to guess a subset of $[0,n)$ that contains the index of the non-zero, invertible entry of $\mathbf{u}$, and allows a corrupted $P_{\mathcal{S}}$ to guess Δ. Such leakages have been incorporated in $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. The former leakage would in turn affect the LPN assumption. Namely, it allows an LPN adversary to learn if each non-zero position of t noise fragments is contained in $I_1,\ldots,I_t$, respectively, where $I_1,\ldots,I_t \subseteq [n/t]$ are chosen by the adversary. This in general leaks 1-bit information. Below, we formulate a primal LPN assumption over Galois rings with such leakage.

Definition 2 (Primal LPN with static leakage). *We first define the corresponding LPN security game* G_{LPN}.

1. *Let* $A \leftarrow \mathcal{G}(m,n) \in \mathtt{GR}(2^k,d)^{m\times n}$ *be a primal LPN generating matrix,* $\mathbf{x} \leftarrow \mathtt{GR}(2^k,d)^m$ *be the secret, and* $\mathbf{e} = (\mathbf{e}^{(1)},...,\mathbf{e}^{(t)}) \in \mathtt{GR}(2^k,d)^n$ *be the error vector, where* $\mathbf{e}^{(i)}$ *has one invertible entry and zeros anywhere else for* $i \in [t]$.
2. $\mathcal{A}$ *sends* $I_1,...,I_t \subseteq [n/t]$. *If for all* $i \in [t]$, I_i *includes the noisy position of* $\mathbf{e}^{(i)}$, *send* `success` *to* $\mathcal{A}$. *Otherwise, abort.*

3. *Pick* $b \leftarrow \{0,1\}$. *If* $b = 0$, *send* $\mathbf{y} := \mathbf{x} \cdot A + \mathbf{e}$ *to* $\mathcal{A}$, *otherwise, send* $\mathbf{y} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$ *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs a bit* b'. *The game outputs* 1 *if* $b' = b$, *and outputs* 0 *otherwise.*

We say that the decisional $(\mathtt{RG}, \mathcal{G}, \mathcal{R}) - \mathrm{LPN}(m, n, t)$ *with static leakage is* (T, ϵ)*-hard, if for every probabilistic distinguisher* $\mathcal{B}$ *running in time* T, $\mathcal{B}$ *wins the game* G_{LPN} *with advantage, defined as* $|\Pr[G_{\mathrm{LPN}} = 1] - 1/2|$, *at most* ϵ.

Under the primal LPN assumption with static leakage, we have the following theorem with proof in Appendix D.2. In the following, $\mathcal{F}_{\mathrm{qVOLE}}^{\mathtt{GR}(2^k,d)}$ (Figure 16, Appendix A.2) extends $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$, which additionally captures the leakage of Δ.

Protocol $\Pi_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$

Let $A \in \mathtt{GR}(2^k, d)^{m \times n}$ be a generating matrix used for $\mathrm{LPN}(m, n, t)$ over $\mathtt{GR}(2^k, d)$. We always assume that n is a multiple of t.

Init:

- Both parties send $\mathtt{Init}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$, which returns $\Delta \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{R}}$.
- Both parties send $(\mathtt{Extend}, m)$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $\mathbf{w}, \mathbf{u} \in \mathtt{GR}(2^k, d)^m$, and $P_{\mathcal{R}}$ receives $\mathbf{v} \in \mathtt{GR}(2^k, d)^m$, such that $\mathbf{v} = \mathbf{w} + \mathbf{u} \cdot \Delta$ holds.

Extend:

1. For $i \in [t]$, both parties send $(\mathtt{SP\text{-}Extend}, n/t)$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $\mathbf{c}^{(i)}, \mathbf{e}^{(i)}$ and $P_{\mathcal{R}}$ receives $\mathbf{b}^{(i)}$, where $\mathbf{b}^{(i)} = \mathbf{c}^{(i)} + \mathbf{e}^{(i)} \cdot \Delta$ over $\mathtt{GR}(2^k, d)^{n/t}$, and $\mathbf{e}^{(i)} \in \mathtt{GR}(2^k, d)^{n/t}$ has one entry invertible in $\mathtt{GR}(2^k, d)$ and zeros everywhere else.
2. Let $\mathbf{e} := (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$, $\mathbf{c} := (\mathbf{c}^{(1)}, ..., \mathbf{c}^{(t)})$, and $\mathbf{b} := (\mathbf{b}^{(1)}, ..., \mathbf{b}^{(t)}) \in \mathtt{GR}(2^k, d)^n$. $P_{\mathcal{S}}$ locally computes $\mathbf{x} := \mathbf{u} \cdot A + \mathbf{e}$, and $\mathbf{M} := \mathbf{w} \cdot A + \mathbf{c} \in \mathtt{GR}(2^k, d)^n$. $P_{\mathcal{R}}$ locally computes $\mathbf{K} := \mathbf{v} \cdot A + \mathbf{b} \in \mathtt{GR}(2^k, d)^n$.
3. $P_{\mathcal{S}}$ updates $\mathbf{u}, \mathbf{w}$ by setting $\mathbf{u} := \mathbf{x}[1 : m] \in \mathtt{GR}(2^k, d)^m$ and $\mathbf{w} := \mathbf{M}[1 : m] \in \mathtt{GR}(2^k, d)^m$, and outputs $(\mathbf{x}[m + 1 : n], \mathbf{M}[m + 1 : n])$. $P_{\mathcal{S}}$ updates $\mathbf{v}$ by setting $\mathbf{v} := \mathbf{K}[1 : m] \in \mathtt{GR}(2^k, d)^m$ and outputs $\mathbf{K}[m + 1 : n]$.

Fig. 10: Protocol for VOLE over $\mathtt{GR}(2^k, d)$ in the $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model.

Theorem 5. *If the decisional* $(\mathtt{RG}, \mathcal{G}, \mathtt{GR}(2^k, d))$-$\mathrm{LPN}(m, n, t)$ *with static leakage assumption holds, then* $\Pi_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ *presented in Figure 10 UC-realizes* $\mathcal{F}_{\mathrm{qVOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$*-hybrid model.*

Security of LPN. We adapt existing reduction results for LPN over $\mathbb{Z}_{2^k}$ [52] to Galois rings, obtaining the following theorems with proofs in Appendix B.2.

Theorem 6. *If decisional* $(\mathcal{D}, \mathcal{G}, \mathtt{GR}(2^k, d)) - \mathrm{LPN}(m, n, w_1)$ *is* (T, ϵ)*-hard, then decisional* $(\mathcal{D}, \mathcal{G}, \mathbb{F}_{2^d}) - \mathrm{LPN}(m, n, w_2)$ *is* $(T - \mathsf{poly}(m, n), \mathcal{O}(h\epsilon))$*-hard, where* $(\mathcal{D}, w_1, w_2, h) \in \{(\mathtt{HW}, t, \frac{2^{d(k-1)}(2^d-1)}{2^{dk}-1}t, \sqrt{t}), (\mathtt{Ber}, \rho, \rho, 1)\}$.

Theorem 6 implies a modulo attack for LPN over $\mathtt{GR}(2^k, d)$, where the adversary can reduce the noise weight by approximately $1/2^d$ fraction via modulo 2, since $\mathtt{GR}(2^k, d)/(2) \cong \mathbb{F}_{2^d}$. This can be mitigated by requiring that each non-zero entry of the error vector $\mathbf{e}$ is invertible as shown in [4,52], or choosing a sufficiently large d.

Theorem 7. *If decisional* $(\mathtt{Ber}, \mathbb{F}_{2^d}) - \mathrm{LPN}(m, n, \frac{\rho(2^d-1)}{(1-\rho)2^{dk}+\rho 2^d})$ *is* (T, ϵ)*-hard, then decisional* $(\mathtt{Ber}, \mathtt{GR}(2^k, d)) - \mathrm{LPN}(m, n, \rho)$ *is* $(T - \mathsf{poly}(m, n), k\epsilon)$*-hard.*

Theorem 7 indicates a security reduction from LPN problems over Galois fields to LPN over Galois rings. We also analyze some main attacks that may work on LPN over Galois rings in Appendix B.3, following prior works [35,15,16,4,52].

5 Publicly Verifiable NIZK for $\mathbb{Z}_{2^k}$ via VOLEitH

Baum et al. [5] proposed the VOLEitH framework for compiling a public-coin VOLE-based ZK protocol into a publicly verifiable NIZK through bringing the MPCitH techniques into the picture. The framework works generally for a large class of ZK protocols based on OT, where the prover is the OT sender and the verifier the OT receiver. In a bare-bone sketch, given a public-coin VOLE-based ZK protocol, the key is to first apply Fiat-Shamir transform on it to squash interactions, and then construct random VOLE correlations from PRGs in one round communication. To this end, the authors showed an efficient VOLE construction from $(N-1)$-out-of-N OT, which then can be instantiated with random vector commitments from GGM tree PRGs.

We follow this framework and begin with an efficient construction of an interesting variant of VOLE over Galois rings from OT in Section 5.1, where we restrict the receiver's choice value to a specifically designed subset of the Galois ring. Jumping ahead, this subset should have small size to make the construction in Section 5.1 computationally efficient and we also require this subset to have certain structure for making the VOLE support efficient compiling (the latter is discussed in Section 5.2).

5.1 VOLE from $(N-1)$-out-of-N OT

SoftSpokenOT [56] proposed an efficient construction for VOLE over Galois fields, which is based on $(N-1)$-out-of-N OT. Intuitively, an $(N-1)$-out-of-N OT functionality takes as inputs N strings of the same length from the sender and one position $\Delta \in [N]$ from the receiver, and then delivers these strings to the receiver except for the Δ-th one. We formulate a random $(N-1)$-out-of-N OT functionality as $\mathcal{F}^N_{\mathrm{OT}-\bar{1}}$ in Figure 14 (Appendix A.2). We adapt the idea of [56]

to construct VOLE over Galois rings, in which Δ can be viewed as the scalar in a single VOLE correlation.

Without loss of generality, assume $N = 2^{kd}$. Let $\mathbf{s}_1, ..., \mathbf{s}_N \in \mathtt{GR}(2^k, d)^\ell$ be random strings possessed by $P_\mathcal{S}$, and $\Delta \in [N]$ be a random index possessed by $P_\mathcal{R}$. The VOLE construction is induced by the following observation:

$$\sum_{y \in \mathtt{GR}(2^k,d)\backslash\{\Delta\}} \mathbf{s}_y \cdot (\Delta - y) = \sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y \cdot (\Delta - y) = \sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y \cdot \Delta - \sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y \cdot y.$$

Through an invocation to $\mathcal{F}_{\mathrm{OT}-\bar{1}}^N$, $P_\mathcal{R}$ receives $\mathbf{s}_y$, for $y \in \mathtt{GR}(2^k, d)\backslash\{\Delta\}$[8]. Then $P_\mathcal{R}$ can locally compute $\mathbf{K} := \sum_{y \in \mathtt{GR}(2^k,d)\backslash\{\Delta\}} \mathbf{s}_y \cdot (\Delta - y)$ while $P_\mathcal{S}$ can compute $\mathbf{M} := -\sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y \cdot y$ and $\mathbf{x} := \sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y$. It is easy to verify that $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$, i.e., $P_\mathcal{S}$ and $P_\mathcal{R}$ obtain a VOLE correlation over $\mathtt{GR}(2^k, d)$.

The main drawback of the above construction is that the computation costs for $P_\mathcal{S}$ and $P_\mathcal{R}$ are high, since the above construction involves $N = 2^{kd}$ heavy multiplication operations over $\mathtt{GR}(2^k, d)$ per party. To reduce the computation complexity, we restrict Δ to be sampled from a subset $\mathbb{F}_{2^d}$ of $\mathtt{GR}(2^k, d)$ [9] (we explain reasons later in the following section). Let $N = 2^d$, we have the following:

$$\mathbf{K} = \sum_{y \in \mathbb{F}_{2^d}\backslash\{\Delta\}} \mathbf{s}_y \cdot (\Delta - y) = \sum_{y \in \mathbb{F}_{2^d}} \mathbf{s}_y \cdot (\Delta - y) = \sum_{y \in \mathbb{F}_{2^d}} \mathbf{s}_y \cdot \Delta - \sum_{y \in \mathbb{F}_{2^d}} \mathbf{s}_y \cdot y = \mathbf{M} + \mathbf{x} \cdot \Delta. \tag{5}$$

Therefore, to obtain a VOLE relation of length ℓ, now $P_\mathcal{S}$ and $P_\mathcal{R}$ only need to perform 2^d cheaper multiplications of $\mathtt{GR}(2^k, d)$ elements and $\mathbb{F}_{2^d}$ elements. To this end, we define $\mathcal{F}_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 15 (Appendix A.2), where $P_\mathcal{R}$ obtains $\Delta \in \mathbb{F}_{2^d}$. We present the resulting VOLE protocol $\Pi_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 11. The security is guaranteed by the following theorem, whose proof is in Appendix D.1.

Theorem 8. *Protocol $\Pi_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ UC realizes $\mathcal{F}_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ (Figure 15) in the $\mathcal{F}_{\mathrm{OT}-\bar{1}}^N$-hybrid model.*

As shown in [56], the $(N-1)$-out-of-N OT can be efficiently realized by $\mathcal{O}(\log N)$ parallel invocations of OT and PRGs, yielding small communication complexity that is independent of length ℓ.

5.2 Publicly Verifiable NIZK for $\mathbb{Z}_{2^k}$ via VOLEitH

In the VOLEitH framework [5], the $(N-1)$-out-of-N OT-based VOLE protocols play an important role of connecting to the VOLE-based ZK paradigm on one hand, and to the MPCitH paradigm through OT on the other hand. We omit the connection to MPCitH paradigm through OT part as it works the same way

[8] We assume a natural one-to-one correspondence between $\mathtt{GR}(2^k, d)$ and $[N]$ without explicitly defining it, since here $N = 2^{kd}$.

[9] We remark that $\mathbb{F}_{2^d}$ is not closed under the operations inherited from $\mathtt{GR}(2^k, d)$.

Protocol $\Pi_{\text{sfVOLE}}^{\mathtt{GR}(2^k,d)}$

Parameterized by a Galois ring $\mathtt{GR}(2^k, d)$ and a length parameter ℓ. Let $N = 2^d$.

Init: Both parties send $(\mathtt{Init}, \mathtt{GR}(2^k, d)^\ell)$ to $\mathcal{F}_{\text{OT}-\bar{1}}^N$. $P_{\mathcal{R}}$ receives $\Delta \in [N]$.

Extend: Both parties send $(\mathtt{Get})$ to $\mathcal{F}_{\text{OT}-\bar{1}}^N$, and act as follows:

1. Upon receiving $\{\mathbf{s}_y\}_{y \in \mathbb{F}_{2^d} \setminus \{\Delta\}}$ from $\mathcal{F}_{\text{OT}-\bar{1}}^N$, the receiver $P_{\mathcal{R}}$ locally computes $\mathbf{K} = \sum_{y \in \mathbb{F}_{2^d} \setminus \{\Delta\}} \mathbf{s}_y \cdot (\Delta - y)$.
2. Upon receiving $\{\mathbf{s}_y\}_{y \in [n]}$ from $\mathcal{F}_{\text{OT}-\bar{1}}^N$, $P_{\mathcal{S}}$ computes $\mathbf{M} := -\sum_{y \in \mathbb{F}_{2^d}} \mathbf{s}_y \cdot y$ and $\mathbf{x} := \sum_{y \in \mathbb{F}_{2^d}} \mathbf{s}_y$.
3. Output $\mathbf{K} = \mathbf{M} + \Delta \cdot \mathbf{x}$, where $\mathbf{K}, \mathbf{M}, \mathbf{x} \in \mathtt{GR}(2^k, d)^\ell$.

Fig. 11: Protocol for VOLE over $\mathtt{GR}(2^k, d)$ in the $\mathcal{F}_{\text{OT}-\bar{1}}^N$-hybrid model.

for both fields and rings (see Sec 3. and Sec 4. of [5]). In the following, we discuss the former connection.

Recall that when one transforms a $(N-1)$-out-of-N correlation into a VOLE correlation, the computation is dominated by the ring size. Therefore, in [5], they have to consider different VOLE variants that would admit practical computation, according to the field size. For the case of proving a statement over a large prime field $\mathbb{F}_p$, they need a *generalized subspace VOLE*, which, in turn, demands a more complicated modified online phase. For the case of proving a statement over a small field, they show that one can use the simple idea of parallel repetitions to boost the soundness of resulting ZK protocols. Note any Galois ring extension of $\mathbb{Z}_{2^k}$ (even $\mathbb{Z}_{2^k}$ itself) is certainly not small enough. Instead of constructing ZK from developing "large field" techniques of [5], we show that the "small field" techniques actually work with a good efficiency in our setting.

Our main idea is to restrict the Δ obtained by the verifier of our protocol $\Pi_{\text{ZK}}^{m,n,t}$ from the VOLE to be in a subset $\mathbb{F}_{2^d}$ of $\mathtt{GR}(2^k, d)$ (hence in the $\mathcal{F}_{\text{sfVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid). However, this would incur an exponential blow-up of computation, as we originally require $d = \Omega(\kappa)$ for the security of $\Pi_{\text{ZK}}^{m,n,t}$. Similar to [18], we can choose a suitable d' and perform repetitions to amplify the soundness. Then we argue that our ZK protocols from Section 3.2 remain secure, by instantiating the underlying VOLE functionality with $\Pi_{\text{sfVOLE}}^{\mathtt{GR}(2^k,d')}$ and repetitions. Intuitively, since $\mathtt{GR}(2^k, d')/(2) \cong \mathbb{F}_{2^{d'}}$, the entropy of $\Delta \in \mathbb{F}_{2^{d'}}$ induces a soundness error $\mathcal{O}\left(\frac{1}{2^{d'}}\right)$, and through t repetitions, the soundness error is enhanced to $\mathcal{O}\left(\frac{1}{2^{d't}}\right)$. We refer the readers to Appendix D.1 for more detailed discussions. Concretely, let the VOLE length be ℓ and consider t repetitions, the computation would be $t\ell \cdot 2^{d'}$ multiplications of $\mathtt{GR}(2^k, d')$ elements and $\mathbb{F}_{2^{d'}}$ elements, while the communication grows roughly t times due to repetition. Hence, there is a trade-off between communication and computation under the premise of security. For

instance, for 80-bit security, we can use a $(6, 15)$-RMFE (i.e., d' is 15) and do 6 repetitions. Recall that our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ from Section 3.2 admits a public-coin ZK in the VOLE-hybrid model, thus we can apply the VOLEitH technique to obtain a publicly verifiable NIZK protocol over $\mathbb{Z}_{2^k}$, following the road map:

$$\mathcal{F}_{\mathrm{OT}-\bar{1}}^{N} \Rightarrow \Pi_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)} \Rightarrow \Pi_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)} \Rightarrow \Pi_{\mathrm{ZK}}^{m,n,t} \overset{\text{compiler [5]}}{\Longrightarrow} \text{publicly verifiable NIZK.}$$

Finally, developing the "large field" approach of [5] to work for $\mathbb{Z}_{2^k}$ is very interesting and worth exploring, which has the potential of getting better concrete efficiency. We leave it as an interesting problem.

6 Acknowledgements

The authors would like to thank Zhe Li and Yanhong Xu for many helpful discussions of this work. We are also very grateful for the insightful comments from anonymous reviewers. The work was supported in part by the National Key Research and Development (R&D) Program of China under Grant 2022YFA1004900 and in part by the National Natural Science Foundation of China under Grant 12031011, Grant 12361141818, and Grant 12101404.

References

1. Abspoel, M., Cramer, R., Escudero, D., Damgård, I., Xing, C.: Improved single-round secure multiplication using regenerating codes. In: ASIACRYPT 2021. LNCS, vol. 13091, pp. 222–244. Springer (2021)
2. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: CCS 2017. pp. 2087–2104. ACM (2017)
3. Baum, C., Braun, L., Munch-Hansen, A., Razet, B., Scholl, P.: Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and z2k. In: CCS 2021. pp. 192–211. ACM (2021)
4. Baum, C., Braun, L., Munch-Hansen, A., Scholl, P.: Moz$\mathbb{Z}_{2^k}$arella: Efficient vector-ole and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In: CRYPTO 2022. LNCS, vol. 13510, pp. 329–358. Springer (2022)
5. Baum, C., Braun, L., de Saint Guilhem, C.D., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In: CRYPTO 2023. LNCS, vol. 14085, pp. 581–615. Springer (2023)
6. Baum, C., Dittmer, S., Scholl, P., Wang, X.: Sok: vector ole-based zero-knowledge protocols. Des. Codes Cryptogr. **91**(11), 3527–3561 (2023)
7. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In: CRYPTO 2021. LNCS, vol. 12828, pp. 92–122. Springer (2021)
8. Baum, C., Nof, A.: Concretely-efficient zero-knowledge arguments for arithmetic circuits and their application to lattice-based cryptography. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12110, pp. 495–526. Springer (2020)

9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. IACR Cryptol. ePrint Arch. p. 46 (2018), http://eprint.iacr.org/2018/046
10. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: Transparent succinct arguments for R1CS. In: EUROCRYPT 2019. LNCS, vol. 11476, pp. 103–128. Springer (2019)
11. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer (1993)
12. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: ASIACRYPT 2017. LNCS, vol. 10626, pp. 336–365. Springer (2017)
13. Bootle, J., Chiesa, A., Guan, Z., Liu, S.: Linear-time probabilistic proofs with sublinear verification for algebraic automata over every field. IACR Cryptol. ePrint Arch. p. 1056 (2022), https://eprint.iacr.org/2022/1056
14. Bootle, J., Chiesa, A., Liu, S.: Zero-knowledge iops with linear-time prover and polylogarithmic-time verifier. In: EUROCRYPT 2022. LNCS, vol. 13276, pp. 275–304. Springer (2022)
15. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: CCS 2018. pp. 896–912. ACM (2018)
16. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: CCS 2019. pp. 291–308. ACM (2019)
17. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: CRYPTO 2019. LNCS, vol. 11694, pp. 489–518. Springer (2019)
18. Braun, L., de Saint Guilhem, C.D., Jadoul, R., Orsini, E., Smart, N.P., Tanguy, T.: Zk-for-z2k: Mpc-in-the-head zero-knowledge proofs for $\mathbb{Z}_{2^k}$. In: IMACC 2023. LNCS, vol. 14421, pp. 137–157. Springer (2023)
19. Briaud, P., Øygarden, M.: A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In: EUROCRYPT 2023. LNCS, vol. 14008, pp. 391–422. Springer (2023)
20. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: SP 2018. pp. 315–334. IEEE Computer Society (2018)
21. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001. pp. 136–145. IEEE Computer Society (2001)
22. Cascudo, I., Cramer, R., Xing, C., Yuan, C.: Amortized complexity of information-theoretically secure MPC revisited. In: CRYPTO 2018. LNCS, vol. 10993, pp. 395–426. Springer (2018)
23. Cascudo, I., Gundersen, J.S.: A secret-sharing based MPC protocol for boolean circuits with good amortized complexity. In: TCC 2020. LNCS, vol. 12551, pp. 652–682. Springer (2020)
24. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: CCS 2017. pp. 1825–1842. ACM (2017)
25. Cheon, J.H., Kim, D., Lee, K.: Mhz2k: MPC from HE over $\mathbb{Z}_{2^k}$ with new packing, simpler reshare, and better ZKP. In: CRYPTO 2021. LNCS, vol. 12826, pp. 426–456. Springer (2021)

26. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zksnarks with universal and updatable SRS. In: EUROCRYPT 2020. LNCS, vol. 12105, pp. 738–768. Springer (2020)
27. Cramer, R., Damgård, I., Escudero, D., Scholl, P., Xing, C.: SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In: CRYPTO 2018. LNCS, vol. 10992, pp. 769–798. Springer (2018)
28. Cramer, R., Rambaud, M., Xing, C.: Asymptotically-good arithmetic secret sharing over $\mathbb{Z}/p^\ell\mathbb{Z}$ with strong multiplication and its applications to efficient MPC. In: CRYPTO 2021. LNCS, vol. 12827, pp. 656–686. Springer (2021)
29. Damgård, I., Zakarias, S.: Constant-overhead secure computation of boolean circuits using preprocessing. In: TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer (2013)
30. Debris-Alazard, T., Tillich, J.: Statistical decoding. In: ISIT 2017. pp. 1798–1802. IEEE (2017)
31. Dittmer, S., Ishai, Y., Lu, S., Ostrovsky, R.: Improving line-point zero knowledge: Two multiplications for the price of one. In: CCS 2022. pp. 829–841. ACM (2022)
32. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-point zero knowledge and its applications. In: ITC 2021. LIPIcs, vol. 199, pp. 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
33. Escudero, D., Hong, C., Liu, H., Xing, C., Yuan, C.: Degree-d reverse multiplication-friendly embeddings: Constructions and applications. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. LNCS, vol. 14438, pp. 106–138. Springer (2023)
34. Escudero, D., Xing, C., Yuan, C.: More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via galois rings. In: CRYPTO 2022. LNCS, vol. 13507, pp. 383–412. Springer (2022)
35. Esser, A., Kübler, R., May, A.: LPN decoded. In: CRYPTO 2017. LNCS, vol. 10402, pp. 486–514. Springer (2017)
36. Feneuil, T., Rivain, M.: Threshold linear secret sharing to the rescue of mpc-in-the-head. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. LNCS, vol. 14438, pp. 441–473. Springer (2023)
37. Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: EUROCRYPT 2015. LNCS, vol. 9057, pp. 191–219. Springer (2015)
38. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. IACR Cryptol. ePrint Arch. p. 953 (2019), `https://eprint.iacr.org/2019/953`
39. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: Snarks for ring arithmetic. J. Cryptol. **36**(4), 41 (2023)
40. Giacomelli, I., Madsen, J., Orlandi, C.: Zkboo: Faster zero-knowledge for boolean circuits. In: USENIX Security 2016. pp. 1069–1083. USENIX Association (2016)
41. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
42. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for R1CS. In: CRYPTO 2023. LNCS, vol. 14082, pp. 193–226. Springer (2023)
43. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer (2016)
44. Guo, X., Yang, K., Wang, X., Zhang, W., Xie, X., Zhang, J., Liu, Z.: Half-tree: Halving the cost of tree expansion in COT and DPF. In: EUROCRYPT 2023. LNCS, vol. 14004, pp. 330–362. Springer (2023)

45. Gvili, Y., Ha, J., Scheffler, S., Varia, M., Yang, Z., Zhang, X.: Turboikos: Improved non-interactive zero knowledge and post-quantum signatures. In: Sako, K., Tippenhauer, N.O. (eds.) Applied Cryptography and Network Security, ACNS 2021. LNCS, vol. 12727, pp. 365–395. Springer (2021)
46. Heath, D., Kolesnikov, V.: Stacked garbling for disjunctive zero-knowledge proofs. In: EUROCRYPT 2020. LNCS, vol. 12107, pp. 569–598. Springer (2020)
47. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC 2007. pp. 21–30. ACM (2007)
48. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: CCS 2013. pp. 955–966. ACM (2013)
49. Kales, D., Zaverucha, G.: Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. IACR Cryptol. ePrint Arch. p. 588 (2022), `https://eprint.iacr.org/2022/588`
50. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: CCS 2018. pp. 525–537. ACM (2018)
51. Lin, F., Xing, C., Yao, Y., Yuan, C.: Amortized NISC over $\mathbb{Z}_{2^k}$ from RMFE. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023. LNCS, vol. 14438, pp. 38–70. Springer (2023)
52. Liu, H., Wang, X., Yang, K., Yu, Y.: The hardness of LPN over any integer ring and field for PCG applications. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024. LNCS, vol. 14656, pp. 149–179. Springer (2024)
53. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. Commun. ACM **59**(2), 103–112 (2016)
54. Peters, C.: Information-set decoding for linear codes over $\mathbb{F}_q$. In: PQCrypto 2010. LNCS, vol. 6061, pp. 81–94. Springer (2010)
55. Ron-Zewi, N., Rothblum, R.D.: Proving as fast as computing: succinct arguments with constant prover overhead. In: STOC 2022. pp. 1353–1363. ACM (2022)
56. Roy, L.: Softspokenot: Quieter OT extension from small-field silent VOLE in the minicrypt model. In: CRYPTO 2022. LNCS, vol. 13507, pp. 657–687. Springer (2022)
57. de Saint Guilhem, C.D., Orsini, E., Tanguy, T.: Limbo: Efficient zero-knowledge mpcith-based arguments. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS 2021. pp. 3022–3036. ACM (2021)
58. Setty, S.T.V.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: CRYPTO 2020. LNCS, vol. 12172, pp. 704–737. Springer (2020)
59. Wan, Z.X.: Lectures on finite fields and Galois rings. World Scientific Publishing Company (2003)
60. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In: IEEE Symposium on Security and Privacy 2021. pp. 1074–1091. IEEE (2021)
61. Weng, C., Yang, K., Yang, Z., Xie, X., Wang, X.: Antman: Interactive zero-knowledge proofs with sublinear communication. In: CCS 2022. pp. 2901–2914. ACM (2022)
62. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: CRYPTO 2022. LNCS, vol. 13510, pp. 299–328. Springer (2022)
63. Yang, K., Sarkar, P., Weng, C., Wang, X.: Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: CCS 2021. pp. 2986–3001. ACM (2021)

64. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated OT with small communication. In: CCS 2020. pp. 1607–1626. ACM (2020)
65. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: FOCS 1986. pp. 162–167. IEEE Computer Society (1986)
66. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: EUROCRYPT 2015. LNCS, vol. 9057, pp. 220–250. Springer (2015)

Supplementary Material

A More Preliminaries & Functionalities

In this section, we give a brief introduction about the UC framework, commitment scheme, and additively homomorphic encryption in Appendix A.1. Next, we describe several functionalities in Appendix A.2

A.1 More Preliminaries

Security Model. In UC framework [21], security is defined via the comparison of an *ideal* world and a *real* world. In the *real* world, the parties interact with each other following the protocol. In the *ideal* world, the parties interact with an ideal functionality $\mathcal{F}$ that is designed to ideally realize the protocol rather than each other. There exists an environment $\mathcal{Z}$, who lives both in the *real* world and the *ideal* world, provides inputs to the parties and can read the outputs. The corrupted party $\mathcal{A}$, controlled by the environment $\mathcal{Z}$, interacts with honest parties in the *real* world. We consider active adversary and static corruption, namely the adversary $\mathcal{A}$'s behavior is arbitrary and not necessarily according to the protocol specification and corruption occurs before the protocol execution. Further, the environment $\mathcal{Z}$ is allowed to interact with $\mathcal{A}$ at any point throughout the protocol execution. The UC-security is guaranteed, if there is a simulator $\mathcal{S}$ plugged to the *ideal* world that interacts with $\mathcal{A}$ such that the environment $\mathcal{Z}$, who can observe $\mathcal{A}$'s view along with all parties' inputs and outputs, can not distinguish $\mathcal{S}$ and the honest parties. More formally, We say a protocol Π UC-realizes a functionality $\mathcal{F}$ with security parameter κ, if there is a probabilistic polynomial-time (PPT) simulator $\mathcal{S}$ such that no PPT environment $\mathcal{Z}$ can distinguish the *ideal* world and the *real* world with advantage $1/\mathsf{poly}(\kappa)$.

Commitment Scheme. A commitment scheme is a two-party protocol consisting of two algorithms, `Commit` and `Open`. In the commit phase of the protocol, the sender $P_{\mathcal{S}}$ invokes `Commit` to commit some value m, obtaining $(\mathtt{com}, \mathtt{unv}) \leftarrow \mathtt{Commit}(m)$ as the result. Then he sends `com` to the receiver $P_{\mathcal{R}}$. Later on in the unveil phase, $P_{\mathcal{S}}$ is required to send m along with the unveil information `unv` to $P_{\mathcal{R}}$ such that $P_{\mathcal{R}}$ can check whether $\mathtt{Open}(\mathtt{com}, \mathtt{unv}, m) = 1$. Informally, there are two security properties that a commitment scheme should satisfy; 1.$Hiding$: $P_{\mathcal{R}}$ can not learn anything about m from `com` in the commit phase, and 2.$Binding$:

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

On input $V_{\mathcal{P}}$ from $\mathcal{P}$ and $V_{\mathcal{V}}$ from $\mathcal{V}$:

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

Fig. 12: Ideal functionality for equality tests.

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

On input $b \in \{0,1\}$ from $P_{\mathcal{R}}$ and m_0, m_1 from $P_{\mathcal{S}}$: Send m_b to $P_{\mathcal{R}}$.

Fig. 13: Ideal functionality for oblivious transfer.

$P_{\mathcal{S}}$ can not provide a $m' \neq m$ and a $\mathtt{unv}'$ such that $\mathtt{Open}(\mathtt{com}, \mathtt{unv}', m') = 1$ in the unveil phase.

Additively Homomorphic Encryption. We describe the Additively Homomorphic Encryption (AHE) scheme in the private-key setting, which consists of three algorithms, $\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}$. The $\mathtt{KeyGen}$ algorithm outputs a secret key sk, which is determined by its random tape. The $\mathtt{Enc}$ algorithm takes the secret key sk and message m as inputs, and outputs a ciphertext $\langle m \rangle$, while the $\mathtt{Dec}$ algorithm decrypts the ciphertext via sk, denoted by $m := \mathtt{Dec}(\langle m \rangle, sk)$. The additive property indicates that the decryption of a linear combination of ciphertexts equals to the corresponding linear combination of plaintexts. We suppose the AHE scheme works on a Galois ring, and satisfies the chosen plaintext attack (CPA) security. For security of our protocol Π_{PAC} (Figure 18), we additionally require the AHE scheme to satisfy circuit privacy and degree restriction, similar to that in [61]. Such AHE schemes exist as shown in [25].

A.2 More Functionalities

We give the equality test functionality $\mathcal{F}_{\mathrm{EQ}}$ in Figure 12, the oblivious transfer functionality $\mathcal{F}_{\mathrm{OT}}$ in Figure 13, the random $(N-1)$-out-of-N OT functionality $\mathcal{F}^{N}_{\mathrm{OT}-\bar{1}}$ in Figure 14. In addition, two variants of VOLE functionalitiess $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{sfVOLE}}$ and $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{qVOLE}}$ are presented in Figure 15 and Figure 16, respectively. Finally, the spVOLE functionality $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{spVOLE}}$ is presented in Figure 17.

Functionality $\mathcal{F}_{\mathrm{OT}-\bar{1}}^{N}$

This functionality is parameterized by a ring $\mathcal{R}$, and an integer N.

Init: Upon receiving (`Init`) from both parties:

1. If $P_{\mathcal{S}}$ is honest, sample $s_1, ..., s_N \xleftarrow{\$} \mathcal{R}$; otherwise receive $s_1, ..., s_N \in \mathcal{R}$ from $\mathcal{A}$.
2. If $P_{\mathcal{R}}$ is honest, sample $\Delta \xleftarrow{\$} [N]$, and send Δ to $P_{\mathcal{R}}$. Otherwise receive $\Delta \in [N]$ and $\{\hat{s}_i \in \mathcal{R}\}_{i \in [N] \setminus \Delta}$ from the adversary $\mathcal{A}$, then reset $s_i = \hat{s}_i$, for $i \in [N] \setminus \{\Delta\}$.

Get: Upon receiving (`Get`) from both parties:
Send $\{s_i\}_{i \in [N]}$ to $P_{\mathcal{S}}$ and $\{s_i\}_{i \in [N] \setminus \{\Delta\}}$ to $P_{\mathcal{R}}$.

Fig. 14: Ideal functionality for random $(N-1)$-out-of-N oblivious transfer.

Functionality $\mathcal{F}_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$

Init: Upon receiving (`Init`) from both parties, sample $\Delta \xleftarrow{\$} \mathbb{F}_{2^d}$ if $P_{\mathcal{R}}$ is honest, and receive $\Delta \in \mathbb{F}_{2^d}$ from the adversary $\mathcal{A}$ otherwise. Store Δ and send it to $P_{\mathcal{R}}$. All further (`Init`) commands will be ignored.
Extend: Upon receiving $(\mathtt{Extend}, \ell)$ from both parties, proceed as follows:

1. If $P_{\mathcal{R}}$ is honest, sample $\mathbf{K} \xleftarrow{\$} \mathtt{GR}(2^k, d)^{\ell}$. Otherwise receive $\mathbf{K}$ from $\mathcal{A}$.
2. If $P_{\mathcal{S}}$ is honest, sample $\mathbf{x} \xleftarrow{\$} \mathtt{GR}(2^k, d)^{\ell}$ and compute $\mathbf{M} := \mathbf{K} - \Delta \cdot \mathbf{x} \in \mathtt{GR}(2^k, d)^{\ell}$. Otherwise, receive $\mathbf{x} \in \mathtt{GR}(2^k, d)^{\ell}$ and $\mathbf{M} \in \mathtt{GR}(2^k, d)^{\ell}$ from $\mathcal{A}$ and then recompute $\mathbf{K} := \mathbf{M} + \Delta \cdot \mathbf{x}$.
3. Send $(\mathbf{x}, \mathbf{M})$ to $P_{\mathcal{S}}$ and $\mathbf{K}$ to $P_{\mathcal{R}}$.

Fig. 15: Ideal functionality of VOLE over $\mathtt{GR}(2^k, d)$.

Functionality $\mathcal{F}_{\mathrm{qVOLE}}^{\mathtt{GR}(2^k,d)}$

$\mathcal{F}_{\mathrm{qVOLE}}^{\mathtt{GR}(2^k,d)}$ extends the existing VOLE functionality $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$(Figure 1). **Init** and **Extend** are identical to those in $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$, respectively.

Global-key Query: If $P_{\mathcal{S}}$ is corrupted, receive $(\mathtt{Guess}, \Delta')$ from $\mathcal{A}$ with $\Delta' \in \mathtt{GR}(2^k, d)$. If $\Delta' = \Delta$, send `success` to $P_{\mathcal{S}}$ and ignore any subsequent global-key queries. Otherwise, send `abort` to both parties and abort.

Fig. 16: Ideal functionality for VOLE over $\mathtt{GR}(2^k, d)$ with global key query.

Functionality $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$

$\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$ extends the existing VOLE functionality $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$(Figure 1). **Init** and **Extend** are identical to those in $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$, respectively.

SP-Extend: Upon receiving $(\mathtt{SP\text{-}Extend}, n)$ from both parties, proceed as follows:

1. If $P_{\mathcal{R}}$ is honest, sample $\mathbf{v} \xleftarrow{\$} \mathtt{GR}(2^k,d)^n$. Otherwise receive $\mathbf{v}$ from $\mathcal{A}$.
2. If $P_{\mathcal{S}}$ is honest, sample uniform $\mathbf{u}$ from $\mathtt{GR}(2^k,d)^n$ with exactly one entry invertible and zeros everywhere else, and compute $\mathbf{w} := \mathbf{v} - \Delta \cdot \mathbf{u} \in \mathtt{GR}(2^k,d)^n$. Otherwise, receive $\mathbf{u} \in \mathtt{GR}(2^k,d)^n$ (with at most one nonzero entry) and $\mathbf{w} \in \mathtt{GR}(2^k,d)^n$ from $\mathcal{A}$, then recompute $\mathbf{v} := \mathbf{w} + \Delta \cdot \mathbf{u}$.
3. If $P_{\mathcal{R}}$ is corrupted, receive a set $I \subseteq [0,n)$ from $\mathcal{A}$. Let $\alpha \in [0,n)$ be the index of the nonzero entry of $\mathbf{u}$. If $\alpha \in I$, send `success` to $P_{\mathcal{R}}$ and continue. Otherwise, send `abort` to both parties and abort.
4. Send $(\mathbf{u}, \mathbf{w})$ to $P_{\mathcal{S}}$ and $\mathbf{v}$ to $P_{\mathcal{R}}$.

Global-key Query: If $P_{\mathcal{S}}$ is corrupted, receive $(\mathtt{Guess}, \Delta', s)$ from $\mathcal{A}$ with $\Delta' \in \mathtt{GR}(2^k,d)$ and $s \in [k]$. If $\Delta' = \Delta \mod 2^s$, send `success` to $P_{\mathcal{S}}$ and ignore any subsequent global-key queries. Otherwise, send `abort` to both parties and abort.

Fig. 17: Ideal functionality for single-point VOLE over $\mathtt{GR}(2^k,d)$.

B LPN over Galois Rings

B.1 Learning Parity with Noise

The Learning Parity with Noise (LPN) assumption [11] has been studied for decades, and it was adapted to be defined over a finite field $\mathbb{F}_q$ [15] or an integer ring $\mathbb{Z}_n$ [4], recently. We give the definition for (primal) LPN over arbitrary rings below.

Definition 3 (LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{t,n}(\mathcal{R})\}$ be the family of distributions over the ring $\mathcal{R}$ where for any integers $t \le n$, $\mathrm{Im}(\mathcal{D}_{t,n}(\mathcal{R})) \subset \mathcal{R}^n$. Let $\mathcal{G}$ be a probabilistic code generation algorithm such that $\mathcal{G}(m,n,\mathcal{R})$ outputs a generator matrix $A \in \mathcal{R}^{m\times n}$. Let parameters m, n, t be implicit functions of security parameter λ. We say that the decisional $(\mathcal{D},\mathcal{G},\mathcal{R})$-LPN$(m,n,t)$ problem is (T,ϵ)-hard if for every probabilistic distinguisher $\mathcal{B}$ running in time T, we have that*

$$|\Pr[\mathcal{B}(A, \mathbf{x}\cdot A + \mathbf{e}) = 1] - \Pr[\mathcal{B}(A,\mathbf{u}) = 1]| \le \epsilon,$$

where $A \leftarrow \mathcal{G}(m,n,\mathcal{R}), \mathbf{x} \xleftarrow{\$} \mathcal{R}^m, \mathbf{u} \xleftarrow{\$} \mathcal{R}^n$, and $\mathbf{e} \leftarrow \mathcal{D}_{t,n}(\mathcal{R})$.

Informally, the decisional LPN(m,n,t) assumption states that $\mathbf{b} := \mathbf{x}\cdot A + \mathbf{e}$ is pseudorandom, where $A, \mathbf{x}, \mathbf{e}$ are defined as above and A is public. There exists another family of LPN assumptions, i.e., the dual LPN.

Definition 4 (Dual LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{t,n}(\mathcal{R})\}$ be the family of distributions over the ring $\mathcal{R}$ where for any integers $t \leq n$, $\mathtt{Im}(\mathcal{D}_{t,n}(\mathcal{R})) \subset \mathcal{R}^n$. Let $\mathcal{G}^\perp$ be a probabilistic dual code generation algorithm such that $\mathcal{G}^\perp(m, n, \mathcal{R})$ outputs a parity check matrix $H \in \mathcal{R}^{n\times(n-m)}$. Let parameters m, n, t be implicit functions of security parameter λ. We say that the decisional dual $(\mathcal{D}, \mathcal{G}, \mathcal{R})$-LPN$(m, n, t)$ problem is (T, ϵ)-hard if for every probabilistic distinguisher $\mathcal{B}$ running in time T, we have that*

$$|\Pr[\mathcal{B}(H, \mathbf{e} \cdot H) = 1] - \Pr[\mathcal{B}(H, \mathbf{u}) = 1]| \leq \epsilon,$$

where $H \leftarrow \mathcal{G}^\perp(m, n, \mathcal{R}), \mathbf{u} \xleftarrow{\$} \mathcal{R}^m$, and $\mathbf{e} \leftarrow \mathcal{D}_{t,n}(\mathcal{R})$.

We mainly consider three kinds of noise distributions in this paper:
Bernoulli. Let $\mathtt{Ber}(\mathcal{R}) = \{\mathtt{Ber}_{\rho,n}(\mathcal{R})\}_{\rho,n}$ be the family of Bernoulli distributions. $\mathbf{e} \leftarrow \mathtt{Ber}_{\rho,n}(\mathcal{R})$ indicates that each entry of $\mathbf{e}$ is a uniformly random element in $\mathcal{R}$ with probability ρ and zero with probability $1 - \rho$. Thus the expected Hamming weight of $\mathbf{e}$ is $\rho N(|\mathcal{R}| - 1)/|\mathcal{R}|$. We remark that this definition is equivalent to sampling a uniformly non-zero element in $\mathcal{R}$ with probability $\rho N(|\mathcal{R}| - 1)/|\mathcal{R}|$ for each entry.
Fixed Hamming weight. Let $\mathtt{HW}(\mathcal{R}) = \{\mathtt{HW}_{t,n}(\mathcal{R})\}_{t,n}$ be the family of distributions of uniformly random vectors with fixed Hamming weight. Let H denote the set $\{\mathbf{e} \in \mathcal{R}^n \mid \mathtt{wt}(\mathbf{e}) = t\}$. Thus we have $\mathbf{e} \leftarrow \mathtt{HW}_{t,n}(\mathcal{R}) \iff \mathbf{e} \leftarrow \mathrm{U}_H$.
Regular Hamming weight. Let $\mathtt{RG}(\mathcal{R}) = \{\mathtt{RG}_{t,n}(\mathcal{R})\}_{t,n}$ be the family of distributions of uniformly random regular weight vectors. W.o.l.g. we assume $t|n$ and let $\mathbf{e} = (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$, where $\mathbf{e}^{(i)} \in \mathcal{R}^{n/t}, i \in [t]$. Let G denote the set $\{\mathbf{e} \in \mathcal{R}^n \mid \mathtt{wt}(\mathbf{e}^{(i)}) = 1, i \in [t]\}$. Thus we have $\mathbf{e} \leftarrow \mathtt{RG}_{t,n}(\mathcal{R}) \iff \mathbf{e} \leftarrow \mathrm{U}_G$, and $\mathtt{RG}(\mathcal{R})$ can be viewed as a special case of $\mathtt{HW}(\mathcal{R})$.

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

Theorem 9 (Theorem 6, restated). *If decisional $(\mathcal{D}, \mathcal{G}, \mathtt{GR}(2^k, d))-\mathrm{LPN}(m, n, w_1)$ is (T, ϵ)-hard, then decisional $(\mathcal{D}, \mathcal{G}, \mathbb{F}_{2^d})-\mathrm{LPN}(m, n, w_2)$ is $(T-\mathsf{poly}(m, n), \mathcal{O}(h\epsilon))$-hard, where $(\mathcal{D}, w_1, w_2, h) \in \{(\mathtt{HW}, t, \frac{2^{d(k-1)}(2^d-1)}{2^{dk}-1}t, \sqrt{t}), (\mathtt{Ber}, \rho, \rho, 1)\}$.*

Proof. Let $(A, \mathbf{b} := \mathbf{x}\cdot A+\mathbf{e})$ be an LPN instance over $\mathtt{GR}(2^k, d)$. As $\mathtt{GR}(2^k, d)/(2) \cong \mathbb{F}_{2^d}$, we observe that $(A^{(0)} := A \mod 2, \mathbf{b}^{(0)} := \mathbf{b} \mod 2)$ constitute exactly the LPN samples over $\mathbb{F}_{2^d}$ for noise $\mathbf{e}^{(0)} := \mathbf{e} \mod 2$. Since $\mathbf{e} \leftarrow \mathtt{HW}_{t,n}(\mathtt{GR}(2^k, d))$, the noise vector $\mathbf{e}^{(0)}$ follows a Bernoulli-like distribution over $\mathbb{F}_{2^d}$. Further, it can be observed that $\mathbf{e}^{(0)}$ has expected weight $t' = \frac{2^{d(k-1)}(2^d-1)}{2^{dk}-1} \cdot t$. One can naturally generalize Lemma 2 in [52] and obtain that the noise vector $\mathbf{e}^{(0)}$ follows the uniform fixed-weight distribution $\mathtt{HW}_{t',n}(\mathbb{F}_{2^d})$ with probability $\Omega\big(1/\sqrt{t}\big)$. On the other hand, $(A^{(0)}, \mathbf{u}^{(0)})$ are uniformly random as well. Therefore, one can use a $(\mathtt{HW}, \mathcal{G}, \mathbb{F}_{2^d}) - \mathrm{LPN}(m, n, t')$ distinguisher to distinguish $(A^{(0)}, \mathbf{b}^{(0)})$ from uniform samples. The proof for the second statement is similar, except that for $\mathbf{e} \leftarrow \mathtt{Ber}_{\rho,n}(\mathtt{GR}(2^k, d))$, one can immediately gets that $\mathbf{e}^{(0)} \sim \mathtt{Ber}_{\rho,n}(\mathbb{F}_{2^d})$. □

Theorem 10 (Theorem 7, restated). *If decisional* $(\mathtt{Ber}, \mathbb{F}_{2^d})$−LPN$(m, n, \frac{\rho(2^d-1)}{(1-\rho)2^{dk}+\rho 2^d})$ *is* (T, ϵ)*-hard, then decisional* $(\mathtt{Ber}, \mathtt{GR}(2^k, d))$−LPN$(m, n, \rho)$ *is* $(T-\mathsf{poly}(m, n), k\epsilon)$*-hard.*

Proof. Let $(A, \mathbf{b} := \mathbf{x}\cdot A+\mathbf{e})$ *be an LPN instance over* $\mathtt{GR}(2^k, d)$*. As* $\mathtt{GR}(2^k, d)/(2) \cong \mathbb{F}_{2^d}$*, for any* $a \in \mathtt{GR}(2^k, d)$*, it can be uniquely written as the form*

$$a = a^{(0)} + a^{(1)} \cdot 2 + ... + a^{(k-1)} \cdot 2^{k-1},$$

where $a^{(i)} \in \mathbb{F}_{2^d}, i \in [0, k)$*. Thus, we can define a decomposition function* $\mathtt{Decom}$ *such that* $(a^{(0)}, a^{(1)}, ..., a^{(k-1)}) := \mathtt{Decom}(a) \in \mathbb{F}_{2^d}^k$*. We use* $\mathtt{Decom}$ *to decompose the matrix and vectors, and we obtain* $(A^{(0)}, A^{(1)}, ..., A^{(k-1)}) := \mathtt{Decom}(A)$*,* $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, ..., \mathbf{x}^{(k-1)}) := \mathtt{Decom}(\mathbf{x})$*,* $(\mathbf{e}^{(0)}, \mathbf{e}^{(1)}, ..., \mathbf{e}^{(k-1)}) := \mathtt{Decom}(\mathbf{e})$*,* $(\mathbf{b}^{(0)}, \mathbf{b}^{(1)}, ..., \mathbf{b}^{(k-1)}) := \mathtt{Decom}(\mathbf{b})$*. Therefore, we have* $\mathbf{b}^{(0)} = \mathbf{x}^{(0)} \cdot A^{(0)} + \mathbf{e}^{(0)}$*, and for* $i \in [0, k)$*, we have that*

$$\mathbf{b}^{(i)} = (\mathbf{x}^{(i)} \cdot A^{(0)} + \mathbf{e}^{(i)}) + f_i(A, \mathbf{x}^{(0)}, ..., \mathbf{x}^{(i-1)}, \mathbf{e}^{(0)}, ..., \mathbf{e}^{(i-1)}) \mod 2,$$

where f_i *is the sum of all other terms involving the individual* $\mathbf{x}^{(j)}$ *and* $\mathbf{e}^{(j)}$ *with* $j \leq i-1$*. Define the hybrid distributions* $H_0, ..., H_k$ *as follows:*

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

where $\mathbf{u}^{(i)} \xleftarrow{\$} \mathbb{F}_{2^d}^n$ *for* $i = 0, 1, ..., k-1$*. Since* $\mathbf{x}$ *is sampled uniformly at random, its decomposition* $\mathbf{x}^{(i)}$ *are independent and uniformly random as well. To distinguish the adjacent hybrids* H_i *and* H_{i+1}*, it suffices to distinguish* $\mathbf{u}^{(i)}$ *and* $\mathbf{b}^{(i)}$ *with the knowledge of* $(\mathbf{b}^{(0)}, ..., \mathbf{b}^{(i-1)})$*. The* f_i *term can be neglected if the effective noise rate of* $\mathbf{e}^{(i)}$ *conditioned on* $\mathbf{e}^{(0)}, ..., \mathbf{e}^{(i-1)}$ *is sufficient to make* $\mathbf{b}^{(i)}$ *pseudorandom. Consider a single noise sample* $(e_j^{(0)}, e_j^{(1)}, ..., e_j^{(k-1)}) \leftarrow \mathtt{Decom}(\mathtt{Ber}_{\rho,n}(\mathtt{GR}(2^k, d)))$*, where* $e_j^{(i)}$ *is the* j*-th entry of* $e^{(i)}$*. If there exists a non-zero component in* $(e_j^{(0)}, e_j^{(1)}, ..., e_j^{(i-1)})$*,* $e_j^{(i)}$ *must be uniformly random, which makes* $b_j^{(i)}$ *uniformly random. Thus, we have that*

$$\Pr\left[e_j^{(i)} \neq 0 \;\middle|\; (e_j^{(0)}, e_j^{(1)}, ..., e_j^{(i-1)}) = \mathbf{0}\right] = \frac{\rho(2^d-1)(2^{-d(i+1)})}{1-\rho+\rho 2^{-di}}$$

is the noise rate needed to keep the computational indistinguishability between H_i *and* H_{i+1}*, which reaches its minimum when* $i = k-1$*.* □

B.3 Attacks on LPN over Galois Rings

To the best of our knowledge, we are not aware of any advantages for attacks over Galois rings compared to Galois fields. Therefore, we review the main attacks that may work on LPN over Galois rings, following the analysis of previous works [35,15,16,4,52,19].

Pooled Gaussian Elimination This attack takes time $\frac{\binom{n}{t}}{\binom{n-m}{t}} \cdot m^{2.8}$, which was estimated as $(\frac{n}{n-t})^m \cdot m^{2.8}$ in [15]. By the following inequality,

$$\frac{\binom{n}{t}}{\binom{n-m}{t}} = \frac{n!(n-m-t)!}{(n-m)!(n-t)!} = \prod_{i=0}^{t-1}(n-i)/\prod_{i=0}^{t-1}(n-m-i) > (\frac{n}{n-m})^t,$$

we obtain a more accurate estimation for LPN with low noise, i.e., $(\frac{n}{n-m})^t \cdot m^{2.8}$. We give the comparison in Table 2. Therefore, we use this formula to estimate the complexity of Pooled Gaussian Elimination attack.

Table 2: Comparison of two estimations.

LPN parameters			Complexity(bit)		
m	n	t	Gauss	Estimation [15]	Estimation(ours)
108112	358620	215	158.15	140.36	158.11
148912	649590	295	158.96	145.71	158.93

Statistical Decoding (SD) The authors of [15] simplified the complexity of the SD attack [30] by $\log(m+1) + t \cdot \log \frac{n}{n-m-1}$. A recent work [52] pointed out that to succeed with constant advantage, SD attack requires at most

$$\log(m+1) + 2 \cdot \left(\log \binom{n}{t} - \log \binom{n-m-1}{t} + \log \frac{2|\mathbb{F}_q|}{|\mathbb{F}_q| - 1} \right)$$

bits arithmetic operations over $\mathbb{F}_q$. According to the proof of Theorem 10 of [52], we observe that only the term $\log \frac{2|\mathbb{F}_q|}{|\mathbb{F}_q|-1}$ relates to the algebraic structure of the ring (essentially, the fraction of units of the ring). Therefore, we can naturally adapt their results to Galois rings, and we obtain that the bit security of the LPN instance with respect to SD attack is computed as

$$\begin{aligned} &\log(m+1) + 2 \cdot \left(\log \binom{n}{t} - \log \binom{n-m-1}{t} + \log \frac{2p^d}{p^d - 1} \right) \\ &\approx \log(m+1) + 2t \cdot (\log \frac{n}{n-m-1}) + 2. \end{aligned}$$

Information Set Decoding (ISD) The authors of [52] analysed known ISD variants for different field size and show that the generalized SD-ISD attack [54] is equivalent to the pooled Gaussian attack for large fields. Since LPN over $\mathtt{GR}(2^k, d)$ can be reduced to LPN over $\mathbb{F}_{2^d}$ and d is set linear in the statistical security parameter κ in our constructions, the generalized SD-ISD attack is equivalent to the pooled Gaussian attack on LPN instantiations for our VOLE constructions.

Algebraic Geometry Attack [19] To our best knowledge, this is the only attack on LPN that is able to exploit the regular noise distribution. The algebraic

geometry attack performs better than Pooled Gaussian Elimination, SD, ISD attacks, on solving regular LPN problems with a small code rate (i.e., m/n). As most PCG applications suggest the dimension parameter m to be relatively large, the algebraic geometry attack has few advantages in these settings.

From above discussions, we can use the pooled Gaussian attack to estimate the LPN security of our constructions.

C Deferred Proofs of Zero-Knowledge Protocols

C.1 Security of re-embedding VOLE

Theorem 11 (Theorem 1, restated). *$\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model. In particular, no PPT environment $\mathcal{Z}$ can distinguish the real world execution from the ideal world simulation except with advantage at most $2^{-s}+2^{-d}$.*

Proof. We divide our proof into two parts. First, we consider $P_{\mathcal{S}}$ is corrupted and construct a PPT simulator $S_{\mathcal{S}}$, then we consider $P_{\mathcal{R}}$ is corrupted and build a PPT simulator $S_{\mathcal{R}}$ as well. Both Simulators interact with the corrupted party in the ideal world and can read the corrupted party's inputs to the functionality $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$.

***Corrupted** $P_{\mathcal{S}}$**:** Whenever the `Extend-pair` procedure is going to run, $S_{\mathcal{S}}$ acts as follows:*

1. *$S_{\mathcal{S}}$ reads $\mathbf{M}^{(1)},\mathbf{M}',\mathbf{x},\mathbf{y}\in\mathtt{GR}(2^k,d)^{n+s}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$.*
2. *Upon receiving $\boldsymbol{\eta}\in\mathtt{GR}(2^k,d)^{n+s}$ from $\mathcal{A}$, $S_{\mathcal{S}}$ sets $\mathbf{M}^{(2)}:=\mathbf{M}'$.*
3. *$S_{\mathcal{S}}$ samples $\boldsymbol{\chi}^{(1)},...,\boldsymbol{\chi}^{(s)}\xleftarrow{\$}\mathbb{Z}_{2^k}^n$, and sends them to $\mathcal{A}$.*
4. *Upon receiving $\mathbf{a},\mathbf{b}\in\mathtt{GR}(2^k,d)^s$ from $\mathcal{A}$. If $\mathbf{b}\neq\tau(\mathbf{a})$, $S_{\mathcal{S}}$ aborts.*
5. *$S_{\mathcal{S}}$ receives $\hat{M}_i^{(1)},\hat{M}_i^{(2)},i\in[s]$ from $\mathcal{A}$. $S_{\mathcal{S}}$ computes $\tilde{M}_i^{(1)}:=M_{n+i}^{(1)}+\sum_{j\in[n]}\chi_j^{(i)}\cdot M_j^{(1)}$, and $\tilde{M}_i^{(2)}:=M_{n+i}^{(2)}+\sum_{j\in[n]}\chi_j^{(i)}\cdot M_j^{(2)}$ for $i\in[s]$. If $\boldsymbol{\eta}=(\tau(\mathbf{x})-\mathbf{y})$, $\hat{\mathbf{M}}^{(2)}=\tilde{\mathbf{M}}^{(2)}$, and $\hat{\mathbf{M}}^{(2)}=\tilde{\mathbf{M}}^{(2)}$ all hold, $S_{\mathcal{S}}$ sends $(\mathbf{x}[1:n],\mathbf{M}^{(1)}[1:n],\mathbf{M}^{(2)}[1:n])$ to $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$. Otherwise, $S_{\mathcal{S}}$ aborts.*

It can be observed that when $S_{\mathcal{S}}$ aborts in step 2 or step 4 of the simulation, the honest $P_{\mathcal{R}}$ aborts in corresponding step of the protocol as well. Besides, $\boldsymbol{\chi}^{(1)},...,\boldsymbol{\chi}^{(s)}$ are sampled in the same way of the ideal simulation and real execution. Therefore, it remains to consider the check of step 5. Basically, $\mathcal{A}$ can cheat by sending $\boldsymbol{\eta}\in\mathtt{GR}(2^k,d)^{n+s}$ with $\boldsymbol{\eta}\neq\tau(\mathbf{x})-\mathbf{y}$. Let $\boldsymbol{\eta}=\tau(\mathbf{x})-\mathbf{y}+\boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}\in\mathtt{GR}(2^k,d)(\psi)^{n+s}$. Then, $\mathcal{A}$ can compute the following

$$a_i:=x_{n+i}+\sum_{j\in[n]}\chi_j^{(i)}\cdot x_j,\quad b_i:=\tau(x_{n+i})+\sum_{j\in[n]}\chi_j^{(i)}\cdot\tau(x_j),$$

for $i\in[s]$. These a_i,b_i would pass the check of honest $P_{\mathcal{R}}$. Also, $\mathcal{A}$ can compute

$$\hat{M}_i^{(1)}:=M_{x_{n+i}}^{(1)}+\sum_{j\in[n]}\chi_i^{(i)}\cdot M_{x_j}^{(1)},\quad \hat{M}_i^{(2)}:=M_{x_{n+i}}^{(2)}+\sum_{j\in[n]}\chi_i^{(i)}\cdot M_{x_j}^{(2)},$$

for $i \in [s]$. This $\hat{\mathbf{M}}^{(1)}$ can also pass the check of honest $P_{\mathcal{R}}$. However, for $\tilde{\mathbf{M}}^{(2)}$ in the real protocol, we have that

$$\begin{aligned}\tilde{M}_i^{(2)} &= K_{n+i}^{(2)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_j^{(2)} - \Delta \cdot b_i \\ &= (M_{n+i}^{(2)} + (y_{n+i} + \eta_{n+i})\Delta) + \sum_{j\in[n]} \chi_j^{(i)}(M_j^{(2)} + (y_j + \eta_j)\Delta) - \Delta(\tau(x_{n+i}) + \sum_{j\in[n]} \chi_j^{(i)}\tau(x_j)) \\ &= M_{n+i}^{(2)} + (\sum_{j\in[n]} \chi_j^{(i)} \cdot M_j^{(2)}) + \Delta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \varepsilon_j) \\ &= \hat{M}_i^{(2)} + \Delta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \varepsilon_j).\end{aligned}$$

Thus, the check $\hat{\mathbf{M}}^{(2)} = \tilde{\mathbf{M}}^{(2)}$ passes if and only if

$$\Delta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \varepsilon_j) = 0,$$

for all $i \in [s]$. If $\varepsilon_{n+i} + \sum_{i\in[n]} \chi_j^{(i)} \cdot \varepsilon_j \neq 0$, from lemma 1, the above equality holds with probability at most 2^{-d}. Since $\chi_j^{(i)} \in \mathbb{Z}_{2^k}$, for $j \in [n]$, $\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \varepsilon_j = 0$ holds with probability at most $1/2$. Combining together, $\hat{\mathbf{M}} = \hat{\mathbf{M}}'$ holds with probability at most $2^{-s} + 2^{-d}$ in real execution if $\boldsymbol{\eta} \neq \tau(\mathbf{x}) - \mathbf{y}$, while in simulation, this will leads to abort. Note that if $\boldsymbol{\eta}$ is correct, the outputs of honest $P_{\mathcal{R}}$ are computed in the same way in two worlds. Therefore, environment $\mathcal{Z}$ can distinguish the ideal simulation and real execution with advantage at most $2^{-s} + 2^{-d}$.

Corrupted** $P_{\mathcal{R}}$: $S_{\mathcal{R}}$ reads $\Delta \in \mathtt{GR}(2^k, d)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ in the* `Init` *procedure. $S_{\mathcal{R}}$ then sends Δ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$. Every time the* ***`Extend-pair` *procedure is executed, $S_{\mathcal{R}}$ does as follows:*

1. *$S_{\mathcal{R}}$ records $\mathbf{K}^{(1)}, \mathbf{K}' \in \mathtt{GR}(2^k, d)^{n+s}$ sent by $\mathcal{A}$. $S_{\mathcal{R}}$ samples $\hat{\boldsymbol{\eta}} := (\boldsymbol{\eta}, \boldsymbol{\eta}') \xleftarrow{\$} \mathtt{GR}(2^k, d)^{n+s}$ and sends $\hat{\boldsymbol{\eta}}$ to $\mathcal{A}$.*
2. *$S_{\mathcal{R}}$ sets $\mathbf{K}^{(2)} := \mathbf{K}' + \Delta \cdot \hat{\boldsymbol{\eta}}$. Upon receiving $\boldsymbol{\chi}^{(i)} \in \mathbb{Z}_{2^k}^n, i \in [s]$ from $\mathcal{A}$, $S_{\mathcal{R}}$ samples $\mathbf{a} \xleftarrow{\$} \mathtt{GR}(2^k, d)^s$, and computes $\mathbf{b} := \tau(\mathbf{a})$. Then, $S_{\mathcal{R}}$ sends $\mathbf{a}, \mathbf{b}$ to $\mathcal{A}$.*
3. *$S_{\mathcal{R}}$ computes $\hat{M}_i^{(1)} := K_{n+i}^{(1)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_j^{(1)} - \Delta \cdot a_i$, and $\hat{M}_i^{(2)} := K_{n+i}^{(2)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_j^{(2)} - \Delta \cdot b_i$, for $i \in [s]$. $S_{\mathcal{R}}$ sends $\hat{\mathbf{M}}^{(1)}, \hat{\mathbf{M}}^{(2)}$ to $\mathcal{A}$.*
4. *$S_{\mathcal{R}}$ sends $\mathbf{K}^{(1)}[1:n], \mathbf{K}^{(2)}[1:n]$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$.*

The indistinguishability between ideal simulation and real execution for corrupted $P_{\mathcal{R}}$ is simple. We first consider the view of $\mathcal{A}$. In real protocol, $\mathcal{A}$ receives a uniformly random $\boldsymbol{\eta} \in \mathtt{GR}(2^k, d)^{n+s}$, since $\boldsymbol{\eta} := \tau(\mathbf{x}) - \mathbf{y}$ and $\mathbf{y}$ is distributed uniformly at random in $\mathtt{GR}(2^k, d)^{n+s}$. While in simulation, $\hat{\boldsymbol{\eta}}$ are uniformly sampled from $\mathtt{GR}(2^k, d)^{n+s}$ as well. As for $(\mathbf{a}, \mathbf{b})$ in real protocol, $\mathbf{b} := \tau(\mathbf{a})$ and is $\mathbf{a}$ is uniformly random in $\mathtt{GR}(2^k, d)^s$ due to the mask $\mathbf{x}[n+1 : n+s]$. Thus, $(\mathbf{a}, \mathbf{b})$

generated by $S_{\mathcal{R}}$ have the same distribution as that in the real protocol. The final message that $\mathcal{A}$ receives from $S_{\mathcal{R}}$ is $\hat{\mathbf{M}}^{(1)}, \hat{\mathbf{M}}^{(2)}$. It can be easily verified that

$$\tilde{M}_i^{(1)} = K_{n+i}^{(1)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_j^{(1)} - \Delta \cdot a_i = M_{n+i}^{(1)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot M_j^{(1)} = \hat{M}_i^{(1)},$$

and

$$\tilde{M}_i^{(2)} = K_{n+i}^{(2)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot K_j^{(2)} - \Delta \cdot b_i = M_{n+i}^{(2)} + \sum_{j\in[n]} \chi_j^{(i)} \cdot M_j^{(2)} = \hat{M}_i^{(2)},$$

Thus $\hat{\mathbf{M}}^{(1)}, \hat{\mathbf{M}}^{(2)}$ has the same distribution in both worlds. Finally, the output $\mathbf{x}[1:n]$ of the honest $P_{\mathcal{S}}$ are identically distributed uniformly at random in both real protocol and ideal world. Therefore, no PPT *environment $\mathcal{Z}$ can distinguish the ideal simulation and the real execution. This completes the proof.* □

C.2 Security of $\Pi_{\mathrm{ZK}}^{m,n,t}$

Theorem 12 (Theorem 2, restated). *Protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ UC-realizes $\mathcal{F}_{\mathrm{ZK}}^{m}$ in the $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model with soundness error $2^{-(d-2)}$ and information-theoretic security.*

Proof. We divide our proof into two parts. First, we consider $\mathcal{P}$ is corrupted, then we consider $\mathcal{V}$ is corrupted. In each case, we build a PPT simulator $\mathcal{S}$ to interact with the corrupted party in the ideal world, which can read the corrupted party's inputs to functionalities $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$.

Corrupted $\mathcal{P}$: *$\mathcal{S}$ interacts with $\mathcal{A}$ as follows:*

1. *$\mathcal{S}$ samples $\Delta \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and records $(\boldsymbol{\mu}, \mathbf{M}_{\boldsymbol{\mu}}, \mathbf{M}'_{\boldsymbol{\mu}})$, $(\boldsymbol{\nu}, \mathbf{M}_{\boldsymbol{\nu}}, \mathbf{M}'_{\boldsymbol{\nu}})$ and (π, M_π) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$. Thus, $\mathcal{S}$ can immediately obtain the MACs $([\mu_i], [\tau(\mu_i)])_{i\in[n]}$, $([\nu_i], [\tau(\nu_i)])_{i\in[t]}$, $[\pi]$.*
2. *Upon receiving $\boldsymbol{\delta}$ from $\mathcal{A}$, $\mathcal{S}$ locally computes $[\tau(\omega_i)] := [\tau(\mu_i)] + \tau(\delta_i)$, for $i \in [n]$.*
3. *$\mathcal{S}$ runs the rest of the protocol as an honest verifier, using the MACs generated in previous steps. If the honest verifier outputs* `true`*, $\mathcal{S}$ computes $(\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, ..., \mathbf{w}^{(m)}) := \psi(\tau(\boldsymbol{\omega}))$ and then sends them and the circuit $\mathcal{C}$ to $\mathcal{F}_{\mathrm{ZK}}^{m}$. Otherwise, $\mathcal{S}$ sends $\mathbf{w} := \perp$ and $\mathcal{C}$ to $\mathcal{F}_{\mathrm{ZK}}^{m}$ and aborts.*

From the simulation, we can see that $\mathcal{S}$ behaves like an honest verifier towards $\mathcal{A}$, therefore, the environment $\mathcal{Z}$ can not distinguish the ideal simulation and real execution from the adversary $\mathcal{A}$'s view. Note that $\mathcal{Z}$ has access to the output of the honest party, the situation remains to be considered is that honest verifier $\mathcal{V}$ accepts the proof while $\mathcal{A}$ does not hold m witnesses. Below we show the probability that $\mathcal{V}$ accepts a proof of wrong statements (i.e., the soundness error) is upper bounded by $1/2^{d-2}$.

First we claim that $\mathcal{A}$ has to prepare a $\boldsymbol{\omega} \in \mathrm{Im}(\phi)^n$, which can be one to one corresponded to m instances of $\mathbb{Z}_{2^k}^n$. The proof is direct since they compute

$[\tau(\boldsymbol{\omega})] := [\tau(\boldsymbol{\mu})] + \tau(\boldsymbol{\delta})$, *and any* $\boldsymbol{\omega} \notin \mathrm{Im}(\phi)^n$ *will be corrected into* $\tau(\boldsymbol{\omega}) \in \mathrm{Im}(\phi)^n$. *Next we prove that all the values on the wires in the circuit are correct. It can be immediately obtained that the values associated with input wires and the output wires of* `Add` *gates are computed correctly, since* ϕ, ψ, τ *are* $\mathbb{Z}_{2^k}$*-linear. Thus, we need to consider the correctness of values on the output wires of* `Mul` *gates, which is guaranteed by the correctness of* d_i*, for all* $i \in [t]$ *in our protocol* $\Pi_{\mathrm{ZK}}^{m,n,k}$. *Consider that some of components of* $\mathbf{d}$ *are incorrect, e.g., there is an error in the* i*-th* `Mul` *gate. Let* $d_i := \omega_\alpha \cdot \omega_\beta - \nu_i + e_i$, *where* $e_i \in \mathtt{GR}(2^k, d)$. *Thus we have that*

$$\begin{aligned} K_{\hat{\omega}_\gamma} :&= K_{\nu_i} + \Delta \cdot d_i = K_{\nu_i} + \Delta \cdot (\omega_\alpha \cdot \omega_\beta - \nu_i + e_i) \\ &= M_{\nu_i} + \Delta \cdot \nu_i + \Delta \cdot (\omega_\alpha \cdot \omega_\beta - \nu_i + e_i) \\ &= M_{\hat{\omega}_\gamma} + \Delta \cdot (\omega_\alpha \cdot \omega_\beta) + \Delta \cdot e_i, \end{aligned}$$

and

$$\begin{aligned} B_i :&= K_{\omega_\alpha} \cdot K_{\omega_\beta} - \Delta \cdot K_{\hat{\omega}_\gamma} \\ &= (M_{\omega_\alpha} + \Delta \cdot \omega_\alpha) \cdot (M_{\omega_\beta} + \Delta \cdot \omega_\alpha) - \Delta \cdot (M_{\hat{\omega}_\gamma} + \Delta \cdot (\omega_\alpha \cdot \omega_\beta) + \Delta \cdot e_i) \\ &= (M_{\omega_\alpha} \cdot M_{\omega_\beta}) + \Delta \cdot (\omega_\alpha \cdot M_{\omega_\beta} + \omega_\beta \cdot M_{\omega_\alpha} - M_{\hat{\omega}_\gamma}) - \Delta^2 \cdot e_i \\ &= A_{0,i} + \Delta \cdot A_{1,i} - \Delta^2 \cdot e_i, \end{aligned}$$

which leads to

$$\begin{aligned} Z :&= \sum_{i \in [t]} \chi_i \cdot B_i + B^* \\ &= X + \Delta \cdot Y - \Delta^2 \cdot (\sum_{i \in [t]} \chi_i \cdot e_i). \end{aligned}$$

Assume $\mathcal{A}$ *sends* $X' = X + e_X$ *and* $Y' = Y + e_Y$ *to honest verifier, where* $e_X, e_Y \in \mathtt{GR}(2^k, d)$. $\mathcal{V}$ *accepts if and only if*

$$Z = X' + \Delta \cdot Y' \iff 0 = e_X + \Delta \cdot e_Y + \Delta^2 \cdot (\sum_{i \in [t]} \chi_i \cdot e_i).$$

χ_i *is sampled by honest verifier after* e_i *is determined, and* e_X, e_Y *can be picked by* $\mathcal{A}$ *after knowing* $\boldsymbol{\chi}$. *If* $\sum_{i \in [t]} \chi_i \cdot e_i \neq 0$, *from lemma 1, we obtain that the above equation holds with probability at most* $2^{-(d-1)}$. *Otherwise,* $\mathcal{A}$ *can pass the check with probability* 1 *(just sets* $e_X = e_Y = 0$*). Since the coefficients* $\chi_i \in \mathbb{F}_{2^d}$ *are sampled uniformly at random, it suffices to consider there exists a* $j \in [t]$ *such that* $e_j \neq 0$, *and* $e_i = 0$ *for* $i \neq j$. *As* $\Pr[\chi_j = 0] = 1/2^d$, *we obtain that it occurs with probability at most* 2^{-d}.

Finally, we show that if $\mathcal{C}(\mathbf{w}^{(i)}) = 0$, *for some* $i \in [m]$, *and all the values on the wires in the circuit are correct, the probability that* $\mathcal{A}$ *successfully provides a* $M'_{\omega_h} := M_{\omega_h} + e_{\omega_h}$ *such that* $K_{\omega_h} = M'_{\omega_h} + \Delta \cdot \phi(\mathbf{1})$ *is upper bounded by* 2^{-d}. *Let* $\mathbf{r} := (\mathcal{C}(\mathbf{w}^{(1)}), ..., \mathcal{C}(\mathbf{w}^{(m)})) \in \{0,1\}^m$. *After executing the protocol, We have*

$$K_{\omega_h} = M_{\omega_h} + \Delta \cdot \phi(\mathbf{r}).$$

Thus, honest verifier accepts if and only if

$$K_{\omega_h} = M'_{\omega_h} + \Delta \cdot \phi(\mathbf{1}) \iff 0 = e_{\omega_h} + \Delta \cdot \phi(\mathbf{1} - \boldsymbol{r}),$$

which holds for a random $\Delta \in \mathtt{GR}(2^k, d)$ *with probability at most* $1/2^{-d}$ *from lemma 1.*

Thus, the overall soundness error is bounded by $2^{-d} + 2^{-(d-1)} + 2^{-d} = 2^{-(d-2)}$. *Namely, a* PPT $\mathcal{Z}$ *can distinguish between the real world and the ideal world with advantage at most* $2^{-(d-2)}$.

Corrupted $\mathcal{V}$***:*** *If* $\mathcal{S}$ *receives* `false` *from* $\mathcal{F}_{\mathrm{ZK}}^{m}$*, then it just aborts. Otherwise,* $\mathcal{S}$ *interacts with* $\mathcal{A}$ *as follows:*

1. *In the offline phase:* $\mathcal{S}$ *records* $\Delta \in \mathtt{GR}(2^k, d)$ *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* `Init` *procedure, also,* $\mathcal{S}$ *records* $(\mathbf{K}_{\boldsymbol{\mu}}, \mathbf{K}'_{\boldsymbol{\mu}}), (\mathbf{K}_{\boldsymbol{\nu}}, \mathbf{K}'_{\boldsymbol{\nu}})$ *and* K_π *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$.
2. $\mathcal{S}$ *samples* $\boldsymbol{\delta} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$ *and sends* $\boldsymbol{\delta}$ *to* $\mathcal{A}$. $\mathcal{S}$ *computes* $K_{\tau(\omega_i)} := K'_{\mu_i} + \Delta \cdot \delta_i$, *for* $i \in [n]$. *Besides,* $\mathcal{S}$ *samples* $\boldsymbol{\omega} \xleftarrow{\$} \mathrm{Im}(\phi)^n$.
3. *For each gate* $(\alpha, \beta, \gamma, T) \in \mathcal{C}$*, in a topological order:*
   - *If* T=`Add`*,* $\mathcal{S}$ *computes* $K_{\omega_\gamma} := K_{\omega_\alpha} + K_{\omega_\beta}$ *as the honest* $\mathcal{V}$ *would do, and sets* $\omega_\gamma := \omega_\alpha + \omega_\beta$.
   - *If* T=`Mul`*, and this is the* i*-th multiplication gate, then* $\mathcal{S}$ *sends* $d_i := \omega_\alpha \cdot \omega_\beta - \nu_i$ *to* $\mathcal{A}$. $\mathcal{S}$ *computes* $K_{\omega_\gamma} := K'_{\nu_i} + \Delta \cdot \tau(d_i)$, $K_{\hat{\omega}_\gamma} := K_{\nu_i} + \Delta \cdot d_i$ *and* $B_i := K_{\omega_\alpha} \cdot K_{\omega_\beta} - \Delta \cdot K_{\hat{\omega}_\gamma}$ *as the honest verifier would do, and sets* $\omega_\gamma := \tau(\omega_\alpha \cdot \omega_\beta)$.
4. $\mathcal{S}$ *receives* $\boldsymbol{\chi} \in \mathbb{Z}_{2^k}^t$ *from* $\mathcal{A}$.
5. $\mathcal{S}$ *computes* $Z := \sum_{i \in [t]} \chi_i \cdot B_i + B^* \in \mathtt{GR}(2^k, d)$*, where* $B^* := K_\pi$*. Then* $\mathcal{S}$ *samples* $Y \xleftarrow{\$} \mathtt{GR}(2^k, d)$ *and sets* $X := Z - \Delta \cdot Y$. $\mathcal{S}$ *sends* (X, Y) *to* $\mathcal{A}$.
6. *For the single output wire* ω_h*,* $\mathcal{S}$ *already holds* K_{ω_h}. $\mathcal{S}$ *computes* $M_{\omega_h} := K_{\omega_h} - \Delta \cdot \phi(\mathbf{1})$*, and sends* M_{ω_h} *to* $\mathcal{A}$.

Since $\boldsymbol{\mu}$ *is distributed uniformly at random by* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$*,* $\boldsymbol{\delta} := \boldsymbol{\omega} - \boldsymbol{\mu}$ *is uniformly random as well in real protocol. Hence,* $\boldsymbol{\delta}$ *perfectly hides the circuit inputs* $\boldsymbol{\omega} := \phi(W)$ *in* $\mathrm{Im}(\phi)^n$*. Similarly,* d_i *perfectly hides the output value of* i*-th* `Mul` *gate in* $\mathrm{Im}(\phi)$*. Moreover,* X, Y *provided by honest prover are uniformly random thanks to the masks* A_0^*, A_1^**, respectively, under the condition that* $Z = X + \Delta \cdot Y$ *holds. Therefore,* $\mathcal{Z}$*'s view in real execution is indistinguishable to that in simulation. This completes the proof.* □

C.3 Deferred Details on $\Pi_{\mathbf{slZK}}^{m,n,t}$

We first present the protocol Π_{PAC} in Figure 18. Next, we give a sketched proof for Theorem 3.

Theorem 13 (Theorem 3, restated). *Protocol* $\Pi_{\mathrm{slZK}}^{m,n,t}$ *UC-realizes functionality* $\mathcal{F}_{\mathrm{ZK}}^{m}$ *that proves circuit satisfiability over* $\mathbb{Z}_{2^k}$ *in the* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$*-hybrid model and the random oracle model with soundness error at most* $\frac{2m_2+3}{2^d} + \mathsf{negl}(\kappa)$.

Protocol Π_{PAC}

Let $\mathtt{AHE} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be an additively homomorphic encryption scheme over $\mathtt{GR}(2^k, d)$ with CPA security, degree-restriction and circuit privacy. Let $(\mathtt{Commit}, \mathtt{Open})$ be a commitment scheme. Let G be a PRG, and m be the maximum degree of the polynomials to be authenticated.

Init: $\mathcal{P}$ and $\mathcal{V}$ send $(\mathtt{Init})$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, and $\mathcal{V}$ receives $\Delta \in \mathtt{GR}(2^k, d)$.
Poly-Key: On input m:

1. $\mathcal{V}$ samples $\mathtt{seed} \xleftarrow{\$} \{0,1\}^\kappa$, and computes $(\mathtt{com}_1, \mathtt{unv}_1) \leftarrow \mathtt{Commit}(\mathtt{seed})$. Then $\mathcal{V}$ sends $\mathtt{com}_1$ to $\mathcal{P}$.
2. $\mathcal{V}$ samples $\Lambda \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and computes ciphertexts $\langle \Lambda^i \rangle := \mathtt{Enc}(sk, \Lambda^i; r_i)$ for all $i \in [m]$, where $(r_0, r_1, ..., r_m) \leftarrow G(\mathtt{seed})$ and $sk \leftarrow \mathtt{KeyGen}(r_0)$. Then $\mathcal{V}$ sends $\langle \Lambda^1 \rangle, ..., \langle \Lambda^m \rangle$ to $\mathcal{P}$.

Pre-Gen: On input $[\mathbf{u}]$, where $\mathcal{P}$ holds $\mathbf{u}, \mathbf{w} \in \mathtt{GR}(2^k, d)^n$ and $\mathcal{V}$ holds $\mathbf{w} \in \mathtt{GR}(2^k, d)^n$ such that $\mathbf{w} := \mathbf{v} - \Delta \cdot \mathbf{u}$:

1. For each $j \in [n]$, on input the j-th polynomial $f_j(X) = \sum_{i \in [0,m]} f_{j,i} \cdot X^i \in \mathtt{GR}(2^k, d)[X]$, $\mathcal{P}$ computes a ciphertext $\langle b_j \rangle := \sum_{i \in [m]} f_{j,i} \cdot \langle \Lambda^i \rangle + f_{j,0} - u_j$.
2. $\mathcal{P}$ computes $(\mathtt{com}_2, \mathtt{unv}_2) \leftarrow \mathtt{Commit}(\langle b_1 \rangle, ..., \langle b_n \rangle)$, and sends $\mathtt{com}_2$ to $\mathcal{V}$.

Gen:

1. $\mathcal{V}$ sends $(\mathtt{unv}_1, \mathtt{seed})$ and Λ to $\mathcal{P}$, who then checks if $\mathtt{Open}(\mathtt{com}_1, \mathtt{unv}_1, \mathtt{seed}) = 1$. If the check fails, $\mathcal{P}$ aborts. Otherwise, $\mathcal{P}$ computes $(r_0, r_1, ..., r_m) \leftarrow G(\mathtt{seed})$ and $sk \leftarrow \mathtt{KeyGen}(r_0)$. $\mathcal{P}$ checks that $\langle \Lambda^i \rangle = \mathtt{Enc}(sk, \Lambda^i; r_i)$ for all $i \in [m]$, and aborts if the check fails. For each $j \in [n]$, $\mathcal{P}$ sets $M_j := w_j$.
2. $\mathcal{P}$ sends $(\mathtt{unv}_2, \langle b_1 \rangle, ..., \langle b_n \rangle)$ to $\mathcal{V}$. $\mathcal{V}$ checks $\mathtt{Open}(\mathtt{com}_2, \mathtt{unv}_2, \langle b_1 \rangle, ..., \langle b_n \rangle) = 1$. If the check fails, $\mathcal{V}$ aborts. Then, for $j \in [n]$, $\mathcal{V}$ computes $b_j := \mathtt{Dec}(sk, \langle b_j \rangle)$, and sets $K_j := v_j + \Delta \cdot b_j$. Thus, $\mathcal{P}$ and $\mathcal{V}$ obtain a PAC $[f_j(\cdot)]$, for each $j \in [n]$.

Fig. 18: Protocol for generating PACs over $\mathtt{GR}(2^k, d)$.

Proof. Since the proof has the similar structure to that in AntMan [61], hence we only explain the difference here. The readers may refer to [61] for more details. Note that for the `BatchCheck` *procedure, we let* $\mathcal{V}$ *check that* $f(\alpha_i) = \tau(g(\beta_i))$*, for* $i \in [t]$*, which in fact is an equality constraint over* $\mathbb{Z}_{2^k}$*. Apparently, this modification would not raise any more considerations of the proof. Another main difference is that we use an ideal functionality* $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ *which delivers MACs of random re-embedding pairs* $(\boldsymbol{\mu}, \tau(\boldsymbol{\mu}))$ *over Galois rings. Recall that we let* $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ *additionally send* $\boldsymbol{\mu} - \tau(\boldsymbol{\mu})$ *to* $\mathcal{V}$*, as explained before in the proof of Theorem 2, this leakage would not influence the privacy of witnesses over* $\mathbb{Z}_{2^k}$ *and it guarantees* $\mathcal{S}$ *can extract SIMD witnesses correctly if the sender is corrupted. Specifically, we instantiate the* DVZK *procedure in [61] by the* LPZK*-based approach, namely the multiplication check procedure in protocol* $\Pi_{\text{ZK}}^{m,n,t}$*. According to Lemma 1 and the upper bound in the Galois field case, the soundness error in the Galois ring case is upper bounded by* $\frac{2m_2+3}{2^d} + \mathsf{negl}(\kappa)$*. This concludes the proof.* □

D Deferred Proofs of VOLE Protocols

D.1 Security of VOLE based on $(N-1)$-out-of-N OT

We claim that sampling Δ from $\mathbb{F}_{2^d}$ makes sense for security of our ZK protocols. Consider a simple game G_0 where the challenger samples a random $\Delta_0 \in \mathbb{F}_{2^d}$ and asks the adversary to output $a, b \in \mathtt{GR}(2^k, d)$ such that $\alpha \cdot \Delta_0 + \beta = 0$, and a similar game G_1 except that the challenger randomly samples Δ from $\mathtt{GR}(2^k, d)$. For an adversary $\mathcal{A}$ that wins G_1 with advantage a_1, we have $\mathcal{A}$ wins G_0 with advantage at least a_1. On the other hand, for an adversary $\mathcal{A}$ that wins G_0 with advantage a_0, there exists an adversary $\mathcal{B}$ (involved in G_1) who invokes $\mathcal{A}$ and outputs $\alpha' := \alpha + 2^{k-1}, \beta' := \beta$, where α, β are $\mathcal{A}$'s outputs. It can be observed that $\mathcal{B}$ wins G_1 (i.e., $\alpha' \cdot \Delta + \beta = 0$) as long as $\alpha \cdot \Delta_0 + \beta = 0$, since Δ can be uniquely written as $\Delta = \Delta_0 + \Delta_1 \cdot 2 + ... + \Delta_{k-1} \cdot 2^{k-1}$, where $\Delta_i \in \mathbb{F}_{2^d}$, for $i \in [0, k)$. Therefore, $\mathcal{B}$ wins G_1 with advantage a_0. Combining together, G_0 and G_1 are equivalent. Note that in our protocol, e.g., $\Pi_{\text{ZK}}^{m,n,t}$, the malicious sender is allowed to guess $\Delta \in \mathtt{GR}(2^k, d)$, who is essentially participating in G_1.

We remark that the above discussion also indicates why we can sample random coefficients from $\mathbb{F}_{2^d}$ rather than $\mathtt{GR}(2^k, d)$ for the random linear combination check.

Theorem 14 (Theorem 8, restated). *Protocol* $\Pi_{\text{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ *UC realizes* $\mathcal{F}_{\text{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ *(Figure 15) in the* $\mathcal{F}_{\text{OT}-\bar{1}}^{N}$*-hybrid model.*

Proof. If the sender $P_{\mathcal{S}}$ *is corrupted. The simulator* $S_{\mathcal{S}}$ *records* $\{\mathbf{s}_y\}_{y\in[N]}$ *sent by* $\mathcal{A}$*. Then* $S_{\mathcal{S}}$ *computes* $\mathbf{M} := -\sum_{y\in\mathbb{F}_{2^d}} \mathbf{s}_y \cdot y$ *and* $\mathbf{x} := \sum_{y\in\mathbb{F}_{2^d}} \mathbf{s}_y$*. Finally,* $S_{\mathcal{S}}$ *sends* $\mathbf{M}, \mathbf{x} \in \mathtt{GR}(2^k, d)^\ell$ *to the ideal functionality* $\mathcal{F}_{\text{sfVOLE}}^{\mathtt{GR}(2^k,d)}$*. The indistinguishability between two worlds is straightforward.*

If the receiver $P_{\mathcal{R}}$ is corrupted. The simulator $S_{\mathcal{R}}$ records $\Delta \in [N]$ and $\{\mathbf{s}_y\}_{y\in[N]\setminus\Delta}$ sent by $\mathcal{A}$ in the `Init` *phase, and forwards Δ to the ideal functionality $\mathcal{F}_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$. Then $S_{\mathcal{R}}$ computes $\mathbf{K} = \sum_{y\in\mathbb{F}_{2^d}\setminus\{\Delta\}} \mathbf{s}_y \cdot (\Delta - y)$ and sends $\mathbf{K} \in \mathtt{GR}(2^k,d)^\ell$ to $\mathcal{F}_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$. The indistinguishability between two worlds is straightforward as well. This concludes the proof.* □

D.2 Security of VOLE based on primal LPN

We first describe the GGM algorithm in Figure 19, then prove Theorem 4 and Theorem 5.

Theorem 15 (Theorem 4, restated). *If G and G' are* PRG*s, then $\Pi_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ functionality in the $(\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{EQ}})$-hybrid model. In particular, no* PPT *environment $\mathcal{Z}$ can distinguish the real world execution from the ideal world simulation except with advantage at most $1/2^d + \mathsf{negl}(\lambda)$.*

Proof. We divide our proof into two parts. First, we consider $P_{\mathcal{S}}$ is corrupted and construct a PPT *simulator $S_{\mathcal{S}}$, then we consider $P_{\mathcal{R}}$ is corrupted and build a* PPT *simulator $S_{\mathcal{R}}$ as well. Both Simulators interact with the corrupted party in the ideal world and can read the corrupted party's inputs to functionalities $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{OT}}, \mathcal{F}_{\mathrm{EQ}}$.*

***Corrupted** $P_{\mathcal{S}}$**:** Each time the* `SP-Extend` *procedure is going to run, $S_{\mathcal{S}}$ acts as follows:*

1. *$S_{\mathcal{S}}$ reads the values (a, c) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. Upon receiving $a' \in \mathtt{GR}(2^k,d)$ from $\mathcal{A}$, $S_{\mathcal{S}}$ sets $\beta := a' + a \in \mathtt{GR}(2^k,d)$ and $\delta := c$.*
2. *For $i \in [1,h)$, $S_{\mathcal{S}}$ samples $K_i \xleftarrow{\$} \{0,1\}^\lambda$, and $S_{\mathcal{S}}$ samples $K_h \xleftarrow{\$} \mathtt{GR}(2^k,d)$. Then $S_{\mathcal{S}}$ reads the choices $\bar{\alpha}_i \in \{0,1\}, i \in [h]$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{OT}}$, and sends $K_{\bar{\alpha}_i,i} := K_i$ to $\mathcal{A}$. $S_{\mathcal{S}}$ computes $\alpha := \sum_{i\in[h]} 2^{h-i} \cdot \alpha_i$ and defines a vector $\mathbf{u} \in \mathtt{GR}(2^k,d)^n$ such that $u_\alpha = \beta$ and $u_i = 0$ for $i \neq \alpha$. Next, $S_{\mathcal{S}}$ runs $\mathtt{GGM.Eval}(\alpha, \{K_{\bar{\alpha}_i,i}\}_{i\in[h]})$ and obtains $\{v_j\}_{j\neq\alpha}$.*
3. *$S_{\mathcal{S}}$ samples $g \xleftarrow{\$} \mathtt{GR}(2^k,d)$ and sends it to $\mathcal{A}$. Then $S_{\mathcal{S}}$ defines a vector $\mathbf{w} \in \mathtt{GR}(2^k,d)^n$ such that $w_\alpha = \delta - (g + \sum_{i\neq\alpha} v_i)$ and $w_i = v_i$ for $i \neq \alpha$.*
4. *$S_{\mathcal{S}}$ reads the values (x, z) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$.*
5. *Upon receiving $\{\chi_i\}_{i\in[0,n)}$ and $x^* \in \mathtt{GR}(2^k,d)$ from $\mathcal{A}$, $S_{\mathcal{S}}$ sets $x' := x^* + x \in \mathtt{GR}(2^k,d)$.*
6. *$S_{\mathcal{S}}$ reads the values $V_{P_{\mathcal{S}}}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{EQ}}$. Then $S_{\mathcal{S}}$ computes $V'_{P_{\mathcal{S}}} := \sum_{i\in[0,n)} \chi_i \cdot w_i - z$ and does as follows:*
   (a) *If $x' = \chi_\alpha \cdot \beta$, then $S_{\mathcal{S}}$ checks whether $V_{P_{\mathcal{S}}} = V'_{P_{\mathcal{S}}}$. If so, $S_{\mathcal{S}}$ sends* `true` *to $\mathcal{A}$, and sends $\mathbf{u}, \mathbf{w}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Otherwise, $S_{\mathcal{S}}$ sends* `abort` *to $\mathcal{A}$ and aborts.*
   (b) *If $x' \neq \chi_\alpha \cdot \beta$, since every $q \in \mathtt{GR}(2^k,d)$ can be represented as $q = \sum_{i\in[0,k)} q_i \cdot 2^i, q_i \in \mathbb{F}_{2^d}$. Let id_q denote the least index such that $q_{id_q} \neq 0$ (define $id_0 := k$). It can be observed that q is invertible in $\mathtt{GR}(2^k,d)$ if and only if $id_q = 0$. Let $\varepsilon := V'_{P_{\mathcal{S}}} - V_{P_{\mathcal{S}}}$ and $\eta := x' - \chi_\alpha \cdot \beta$, we have that*

Algorithm $\mathtt{GGM}$

Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$, and $G' : \{0,1\}^\lambda \to \mathtt{GR}(2^k, d)^2 \times \{0,1\}^\lambda$ be two pseudo-random generators (PRGs). For $\alpha \in [0, 2^h)$, we write $\alpha = \sum_{i\in[h]} 2^{h-i}\alpha_i$, where $\alpha_i \in \{0,1\}$ and we denote the complement of α_i by $\bar{\alpha}_i$. The GGM algorithms $\mathtt{GGM.KeyGen}, \mathtt{GGM.Gen}, \mathtt{GGM.Eval}$ are defined as follows:

- $\mathtt{GGM.KeyGen}$: on input 1^λ, output $s \xleftarrow{\$} \{0,1\}^\lambda$.
- $\mathtt{GGM.Gen}$: on input n and $s \in \{0,1\}^\lambda$, where $n = 2^h$,
  (a) Set $S_{0,0} := s$.
  (b) For $i \in [h-1]$ and $j \in \{0, 1, ..., 2^{i-1} - 1\}$, compute $(S_{2j,i}, S_{2j+1,i}) := G(S_{j,i-1})$, where $S_{2j,i}$ ($S_{2j+1,i}$ resp.) is the left (right resp.) child of $S_{j,i-1}$.
  (c) For $j \in [0, 2^{h-1})$, compute $(v_{2j,h}, v_{2j+1,h}, \Gamma_j) := G'(S_{j,h-1})$.
  (d) For $i \in [h-1]$, set $K_{0,i} := \bigoplus_{j\in[0,2^{i-1})} S_{2j,i}$ and $K_{1,i} := \bigoplus_{j\in[0,2^{i-1})} S_{2j+1,i}$, i.e., XOR of left (right) nodes in layer i, respectively.
  (e) Set $K_{0,h} := \sum_{j\in[0,2^{h-1})} S_{2j,h}$ and $K_{1,h} := \sum_{j\in[0,2^{h-1})} S_{2j+1,h}$, i.e., sum of left (right) leaves, respectively.
  (f) Output $(\{v_j\}_{j\in[0,n)}, \{(K_{0,i}, K_{1,i})\}_{i\in[h]})$, $\{\Gamma_j\}_{j\in[0,2^{h-1})}$.
- $\mathtt{GGM.Eval}$: On input $\alpha, \{K_i\}_{i\in[h]}$, where $\alpha \in [0, n)$,
  (a) Set $S^1_{\bar{\alpha}_1} := K_1$, and $x := 0$.
  (b) For $i \in [h-2]$,
    i. Update x by $2x + \alpha_i$.
    ii. For $j \in [0, 2^i)\backslash\{x\}$, compute $(S_{2j,i+1}, S_{2j+1,i+1}) := G(S_{j,i})$.
    iii. Compute $S_{2x+\bar{\alpha}_{i+1},i+1} := K_{i+1} \oplus \bigoplus_{j\in[0,2^i)\backslash\{x\}} S_{2j+\bar{\alpha}_{i+1},i+1}$.
  (c) Update x by $2x + \alpha_{h-1}$.
    i. For $j \in [0, 2^{h-1})\backslash\{x\}$, compute $(v_{2j}, v_{2j+1}, \Gamma_j) := G'(S_{j,h-1})$.
    ii. Compute $v_{2x+\bar{\alpha}_h} := K_h - \sum_{j\in[0,2^{h-1})\backslash\{x\}} v_{2j+\bar{\alpha}_h}$.
  (d) Output $(\{v_j\}_{j\in[0,n)\backslash\{\alpha\}})$.
- $\mathtt{GGM.Eval}'$: On input $\alpha, \{K_i\}_{i\in[h]}$ and $K_{1,h+1}$, where $\alpha \in [0, n)$,
  (a) Set $S^1_{\bar{\alpha}_1} := K_1$, and $x := 0$.
  (b) For $i \in [h-1]$,
    i. Update x by $2x + \alpha_i$.
    ii. For $j \in [0, 2^i)\backslash\{x\}$, compute $(S_{2j,i+1}, S_{2j+1,i+1}) := G(S_{j,i})$.
    iii. Compute $S_{2x+\bar{\alpha}_{i+1},i+1} := K_{i+1} \oplus \bigoplus_{j\in[0,2^i)\backslash\{x\}} S_{2j+\bar{\alpha}_{i+1},i+1}$.
  (c) i. For $j \in [0, 2^h)\backslash\{\alpha\}$, compute $(v_{2j}, v_{2j+1}, \Gamma_j) := G'(S_{j,h-1})$.
    ii. Compute $\Gamma_\alpha := K_{1,h+1} \oplus \bigoplus_{j\in[0,2^h)\backslash\{\alpha\}} \Gamma_j$.
  (d) Output $(\{(v_{2j}, v_{2j+1})\}_{j\in[0,n)\backslash\{\alpha\}}, \{\Gamma_j\}_{j\in[0,n)})$.

Fig. 19: Algorithms for GGM tree based PPRF.

– *If* $id_\varepsilon < id_\eta$, $S_\mathcal{S}$ *sends* `abort` *to* $\mathcal{A}$ *and aborts.*
– *If* $id_\varepsilon \geq id_\eta$, $S_\mathcal{S}$ *computes* $\Delta' := \frac{\varepsilon}{2^{id_\eta}} \cdot (\frac{\eta}{2^{id_\eta}})^{-1} \mod 2^{k-id_\eta}$, *and sends a global-key query* $(\texttt{Guess}, \Delta', k - id_\eta)$ *to* $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$. *If* $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$ *returns* `success`, $S_\mathcal{S}$ *sends* `true` *to* $\mathcal{A}$, *and sends* $\boldsymbol{u}, \boldsymbol{w}$ *to* $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$. *Otherwise,* $S_\mathcal{S}$ *sends* `abort` *to* $\mathcal{A}$ *and aborts.*

The `Init` *procedure can be called only once, and it can be perfectly simulated by forwarding* `Init` *from* $\mathcal{A}$ *to* $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$. *Thus, we focus on analysing indistinguishability between the* ***SP-Extend*** *procedures.*

$S_\mathcal{S}$ *can record* a, c *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$. *From the construction of GGM tree, it can be observed that* $K_{0,1}, K_{1,1}$ *are pseudorandom. Further, we have that* $K_{0,2}, K_{1,2}$ *are pseudorandom as well conditioned on* $K_{0,1}$ *or* $K_{1,1}$. *By induction, we obtain that* $K_{\bar{\alpha}_i,i}$ *is pseudorandom conditioned on* $\{K_{\bar{\alpha}_j,j}\}_{j<i}$, *for* $i \in [h]$. *Therefore, we claim that* $\{K_{\bar{\alpha}_i,i}\}_{i\in[h]}$ *are pseudorandom, which indicates that* $K_i, i \in [h]$ *provided by* $S_\mathcal{S}$ *are computationally indistinguishable to those in real execution. In real protocol,* γ *is uniformly random in* $\mathcal{A}$*'s view, since* Δ *is uniformly random and* β *is a unit. Therefore,* g *sampled uniformly at random by* $S_\mathcal{S}$ *is of the same distribution to that masked with* γ *in* $\mathcal{A}$*'s view. Now it remains to consider the check step.*

The second call $(\texttt{Extend}, 1)$ *enables* $S_\mathcal{S}$ *to record* x, z. *In real protocol, honest verifier* $P_\mathcal{R}$ *computes*

$$\begin{aligned}
V_{P_\mathcal{R}} :&= \sum_{i\in[0,n)} \chi_i \cdot v_i - y \\
&= \sum_{i\in[0,n)} \chi_i \cdot w_i + \sum_{i\in[0,n)} \chi_i \cdot (v_i - w_i) - y \\
&= \sum_{i\in[0,n)} \chi_i \cdot w_i + \chi_\alpha \cdot (v_\alpha - w_\alpha) - (y^* + \Delta \cdot x^*) \\
&= \sum_{i\in[0,n)} \chi_i \cdot w_i - (y^* - \Delta \cdot x) - \Delta \cdot (x^* + x - \chi_\alpha \cdot \beta) \\
&= V'_{P_\mathcal{S}} - \Delta \cdot (x' - \chi_\alpha \cdot \beta).
\end{aligned}$$

If $\mathcal{A}$ *behaves honestly, then* $x' = \chi_\alpha \cdot \beta$ *will hold and* $\mathcal{F}_{\text{EQ}}$ *will return* `true`. *Note that* $S_\mathcal{S}$ *can extract* α *from* $\mathcal{A}$*'s inputs to* $\mathcal{F}_{\text{OT}}$, *thus* $S_\mathcal{S}$ *can check* $x' = \chi_\alpha \cdot \beta$. *If the equation holds,* $\mathcal{F}_{\text{EQ}}$ *can be emulated by sending* `true` (`abort`) *to* $\mathcal{A}$ *when* $V_{P_\mathcal{S}} = V'_{P_\mathcal{S}}$ ($V_{P_\mathcal{S}} \neq V'_{P_\mathcal{S}}$), *which is the same as in the real protocol.*

Otherwise, x^* *sent by* $\mathcal{A}$ *must be incorrect. Let* $\eta := x' - \chi_\alpha \cdot \beta$ *and* $\varepsilon := V'_{P_\mathcal{S}} - V_{P_\mathcal{S}}$. *Therefore,* $\mathcal{A}$ *passes the equality test if and only if*

$$V_{P_\mathcal{S}} + \Delta \cdot \eta = V'_{P_\mathcal{S}} \iff \Delta \cdot \eta = \varepsilon,$$

where $\Delta, \eta, \epsilon \in \texttt{GR}(2^k, d)$. *Although* $S_\mathcal{S}$ *does not hold* Δ, *he can query the global key to* $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$. *Since the above equation is over* $\texttt{GR}(2^k, d)$, *there may be no solutions for* Δ, *or more than one solutions for* Δ. *Thus the* ***Global-key Query***

is extended to allow queries of "lower bit" of Δ. It is not hard to see that the simulation matches the real execution in this case.

Thus, we conclude that $\mathcal{Z}$ can not computationally distinguish the ideal simulation and the real execution from joint view of $\mathcal{A}$ and the output of $P_{\mathcal{R}}$.

Corrupted** $P_{\mathcal{R}}$**:** In the* `Init` *procedure, $S_{\mathcal{R}}$ reads the global key $\Delta \in \mathtt{GR}(2^k, d)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. Then whenever the* ***SP-Extend *procedure is going to run, $S_{\mathcal{R}}$ acts as follows:*

1. *$S_{\mathcal{R}}$ reads $b \in \mathtt{GR}(2^k, d)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ samples $a' \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and sends a' to $\mathcal{A}$. Then, $S_{\mathcal{R}}$ draws a uniformly random β in the set of units of $\mathtt{GR}(2^k, d)$. Next, $S_{\mathcal{R}}$ computes $\gamma := b + \Delta \cdot a'$ and $\delta := \gamma - \Delta \cdot \beta$.*
2. *$S_{\mathcal{R}}$ reads the values $(K_{0,i}, K_{1,i})_{i\in[h]}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{OT}}$. Upon receiving $g \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$, for each $\alpha \in [0, n)$, $S_{\mathcal{R}}$ defines a vector $\mathbf{w}^{(\alpha)}$ such that $\{w_j^{(\alpha)}\}_{j\neq\alpha} := \mathtt{GGM.Eval}(\alpha, \{K_{\bar{\alpha}_i,i}\}_{i\in[h]})$ and $w_\alpha^{(\alpha)} := \delta - (g + \sum_{i\neq\alpha} w_i^{(\alpha)})$.*
3. *$S_{\mathcal{R}}$ reads the value y^* that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ samples $\chi_i \xleftarrow{\$} \mathtt{GR}(2^k, d)$ for $i \in [0, n)$ and $x^* \xleftarrow{\$} \mathtt{GR}(2^k, d)$. $S_{\mathcal{R}}$ sends $(\{\chi_i\}_{i\in[0,n)}, x^*)$ to $\mathcal{A}$. Then, $S_{\mathcal{R}}$ computes $y := y^* + \Delta \cdot x^*$.*
4. *$S_{\mathcal{R}}$ reads $\mathcal{V}_{P_{\mathcal{R}}}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{EQ}}$. Then $S_{\mathcal{R}}$ constructs a set $I \subseteq [0, n)$ as follows:*
   (a) *For $\alpha \in [0, n)$, compute $V_{P_{\mathcal{S}}}^\alpha := \sum_{i\in[0,n)} \chi_i \cdot w_i^{(\alpha)} + \Delta \cdot \chi_\alpha \cdot \beta - y$.*
   (b) *Append α satisfying $V_{P_{\mathcal{S}}}^\alpha = V_{P_{\mathcal{R}}}$ to set I, i.e., $I := \{\alpha \in [0, n) \mid V_{P_{\mathcal{S}}}^\alpha = V_{P_{\mathcal{R}}}\}$.*

   $S_{\mathcal{R}}$ sends I to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. If it returns `abort`*, $S_{\mathcal{R}}$ samples $\hat{\alpha} \xleftarrow{\$} [0, n)\backslash I$, sends $(\mathtt{false}, V_{P_{\mathcal{S}}}^{\hat{\alpha}})$ to $\mathcal{A}$ (emulating $\mathcal{F}_{\mathrm{EQ}}$), and then aborts. Otherwise, $S_{\mathcal{R}}$ sends $(\mathtt{true}, V_{P_{\mathcal{R}}})$ to $\mathcal{A}$.*
5. *$S_{\mathcal{R}}$ picks an arbitrary $\alpha \in I$ and defines $\mathbf{v}$ such that $v_i := w_i^{(\alpha)}$, for $i \neq \alpha$ and $v_\alpha := \gamma - g - \sum_{i\neq\alpha} v_i$. $S_{\mathcal{R}}$ sends $\mathbf{v}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$.*

The `Init` *procedure can be called only once, and $S_{\mathcal{R}}$ learns Δ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. Now we consider the simulation for the* ***SP-Extend*** *procedure.*

In real execution, a is uniformly random in $\mathcal{A}$'s view, which perfectly masks a' in $\mathtt{GR}(2^k, d)$. Thus, a' provided by $S_{\mathcal{R}}$ has the same distribution as that in the real execution. $S_{\mathcal{R}}$ learns $\{(K_{0,i}, K_{1,i})\}_{i\in[h]}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{OT}}$, therefore, $S_{\mathcal{R}}$ can evaluate n punctured GGM trees with one value of the α-th leaf missed, for each $\alpha \in [0, n)$. $S_{\mathcal{R}}$ learns y^* that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. The indistinguishability of $\boldsymbol{\chi} \in \mathtt{GR}(2^k, d)^n$ is obvious and the next message that $\mathcal{A}$ receives from $P_{\mathcal{S}}$ or $S_{\mathcal{R}}$ is x^*. Similarly, x^* is perfectly masked with x since x is uniformly random in $\mathcal{A}$'s view. Therefore, we obtain the indistinguishability of x^* between two worlds. What remains to consider is the simulation for equality test. The set I constructed by $S_{\mathcal{R}}$ corresponds to the selective failure attack on the α^* of honest $P_{\mathcal{S}}$. This attack can be formalized as follows: (1) $\mathcal{A}$ generates a GGM tree correctly and obtains $(\{v_j\}_{j\in[0,n)}, \{(K_{0,i}, K_{1,i})\}_{i\in[h]})$, (2) $\mathcal{A}$ guesses a set $I \subseteq [0, n)$, (3) Let U denote the union of the sets $\{K_{\bar{\alpha}_i,i}\}_{i\in[h]}$, for $\alpha \in I$. $\mathcal{A}$ keeps the values of U

unchanged and randomizes the remaining values in $\{K_{0,i}, K_{1,i}\}_{i\in[h]}$. *(4)* $\mathcal{A}$ *runs the rest program as an honest* $P_{\mathcal{R}}$. *It can be observed that if* $\alpha^* \in I$, *then the equality check will pass, i.e.,* $V_{P_{\mathcal{S}}}^{\alpha^*} = V_{P_{\mathcal{R}}}$. *This observation guarantees that the set* I *reconstructed by* $S_{\mathcal{R}}$ *is identical to that generated by* $\mathcal{A}$. *Therefore,* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$ *aborts if and only if the equality check fails in the real execution. Note that if* $|I| > 1$, $S_{\mathcal{R}}$ *does not know the actual* α^*. *However,* $S_{\mathcal{R}}$ *is required to send* $\mathbf{v}^{(\alpha^*)}$ *to* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$, *otherwise,* $\mathcal{Z}$ *may distinguish two worlds from joint view of* $\mathcal{A}$ *and output of* $P_{\mathcal{S}}$. *We claim that*

$$\Pr\left[\mathbf{v}^{(\alpha)} \neq \mathbf{v}^{(\alpha')} \;\middle|\; V_{P_{\mathcal{S}}}^{\alpha} = V_{P_{\mathcal{S}}}^{\alpha'}\right] \leq \frac{1}{2^d}.$$

We have that

$$V_{P_{\mathcal{S}}}^{\alpha} = V_{P_{\mathcal{S}}}^{\alpha'} \iff$$
$$\sum_{i\in[0,n)} \chi_i \cdot w_i^{(\alpha)} + \Delta \cdot \chi_\alpha \cdot \beta - y = \sum_{i\in[0,n)} \chi_i \cdot w_i^{(\alpha')} + \Delta \cdot \chi_{\alpha'} \cdot \beta - y \iff$$

$$\sum_{i\in[0,n), i\neq\alpha,\alpha'} \chi_i\cdot(w_i^{(\alpha)} - w_i^{(\alpha')}) + \chi_\alpha(\Delta\cdot\beta + w_\alpha^{(\alpha)} - w_\alpha^{(\alpha')}) + \chi_{\alpha'}(w_{\alpha'}^{(\alpha)} - \Delta\cdot\beta - w_{\alpha'}^{(\alpha')}) = 0$$

Note that $\Delta, \beta, \mathbf{w}^{(\alpha)}, \mathbf{w}^{(\alpha')}$ *are determined before* $\boldsymbol{\chi}$ *sampled. Therefore, from lemma 1, we have that*

$$w_i^{(\alpha)} = w_i^{(\alpha')}, \text{for } i \in [0,n)\backslash\{\alpha,\alpha'\},$$

and

$$\Delta \cdot \beta = w_\alpha^{(\alpha')} - w_\alpha^{(\alpha)} = w_{\alpha'}^{(\alpha)} - w_{\alpha'}^{(\alpha')}$$

hold except with probability 2^{-d}. *Thus we immediately obtain that* $\mathbf{v}^{(\alpha)} = \mathbf{v}^{(\alpha')}$ *holds except with probability* 2^{-d} *under the condition that* $V_{P_{\mathcal{S}}}^{\alpha} = V_{P_{\mathcal{S}}}^{\alpha'}$. *Thus, from above discussion, we conclude that* $\mathcal{Z}$ *can distinguish the ideal simulation and real execution with advantage at most* 2^{-d}. *This completes the whole proof.* □

Theorem 16 (Theorem 5, restated). *If the decisional* $(\mathtt{RG}, \mathcal{G}, \mathtt{GR}(2^k,d))$-LPN$(m,n,t)$ *with static leakage assumption holds, then* $\Pi_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$ *presented in Figure 10 UC-realizes* $\mathcal{F}_{q\text{VOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$*-hybrid model.*

Proof. We divide our proof into two parts. First, we consider $P_{\mathcal{S}}$ *is corrupted and construct a* PPT *simulator* $S_{\mathcal{S}}$, *then we consider* $P_{\mathcal{R}}$ *is corrupted and build a* PPT *simulator* $S_{\mathcal{R}}$ *as well. Both Simulators interact with the corrupted party in the ideal world and can read the corrupted party's inputs to functionalities* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$.

Corrupted $P_{\mathcal{S}}$***:*** $S_{\mathcal{S}}$ *reads the vectors* $\mathbf{u}, \mathbf{w} \in \mathtt{GR}(2^k,d)^m$ *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* `Init` *procedure. Then whenever* `Extend` *procedure is going to run,* $S_{\mathcal{S}}$ *acts as follows:*

1. *For $i \in [t]$, $S_{\mathcal{S}}$ reads $\mathbf{e}^{(i)} \in \mathtt{GR}(2^k,d)^m$ (with at most one invertible entry and zeros everywhere else) and $\mathbf{c}^{(i)} \in \mathtt{GR}(2^k,d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Let $\mathbf{e} := (\mathbf{e}^{(1)},...,\mathbf{e}^{(t)}) \in \mathtt{GR}(2^k,d)^n$ and $\mathbf{c} := (\mathbf{c}^{(1)},...,\mathbf{c}^{(t)}) \in \mathtt{GR}(2^k,d)^n$.*
2. *$S_{\mathcal{S}}$ computes $\mathbf{x} := \mathbf{u} \cdot A + \mathbf{e} \in \mathtt{GR}(2^k,d)^n$ and $\mathbf{M} := \mathbf{w} \cdot A + \mathbf{c} \in \mathtt{GR}(2^k,d)^n$. Then $S_{\mathcal{S}}$ updates $\mathbf{u} := \mathbf{x}[1:m]$ and $\mathbf{z} := \mathbf{M}[1:m]$. Next, $S_{\mathcal{S}}$ sends $\mathbf{x}[m+1,n]$ and $\mathbf{M}[m+1,n]$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$.*
3. *Whenever $\mathcal{A}$ sends a global-key query $(Guess, \Delta', s')$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$, $S_{\mathcal{S}}$ sends $(Guess, \Delta')$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ and forwards the answer from $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ to $\mathcal{A}$. If the answer is* `abort`*, $S_{\mathcal{S}}$ aborts.*

The indistinguishability between simulation and execution is obvious, since the outputs of both parties in two worlds are computed in the same way.
***Corrupted** $P_{\mathcal{R}}$**:** $S_{\mathcal{R}}$ first receives $\Delta \in \mathtt{GR}(2^k,d)$ from $\mathcal{A}$ and reads $\mathbf{v} \in \mathtt{GR}(2^k,d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ in the* `Init` *procedure. $S_{\mathcal{R}}$ sends Δ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. Then whenever* `Extend` *procedure is going to run, $S_{\mathcal{R}}$ acts as follows:*

1. *For $i \in [t]$, $S_{\mathcal{R}}$ reads the vector $\mathbf{b}^{(i)} \in \mathtt{GR}(2^k,d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Let $\mathbf{b} := (\mathbf{b}^{(1)},...,\mathbf{b}^{(t)}) \in \mathtt{GR}(2^k,d)^n$.*
2. *$S_{\mathcal{R}}$ samples a random $\mathbf{e} := (\mathbf{e}^{(1)},...,\mathbf{e}^{(t)}) \in \mathtt{GR}(2^k,d)^n$, where each $\mathbf{e}^{(i)}$ has one invertible entry and zeros everywhere else. Let $\{\alpha_1,...,\alpha_t\}$ be the invertible entry in $\{\mathbf{e}^{(1)},...,\mathbf{e}^{(t)}\}$, respectively. For $i \in [t]$, $S_{\mathcal{R}}$ reads the set $I_i \subseteq [0,m)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. If $\alpha_i \in I_i$, then $S_{\mathcal{R}}$ continues and sends* `success` *to $\mathcal{A}$. Otherwise, $S_{\mathcal{R}}$ sends* `abort` *to $\mathcal{A}$ and aborts.*
3. *$S_{\mathcal{R}}$ computes $\mathbf{K} := \mathbf{v} \cdot A + \mathbf{b} \in \mathtt{GR}(2^k,d)^n$, and updates $\mathbf{v} := \mathbf{K}[1:m]$. $S_{\mathcal{R}}$ sends $\mathbf{K}[m+1:n]$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$.*

It is not hard to see that $\mathbf{e}$ sampled by $S_{\mathcal{R}}$ has the same distribution to that provided by $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$, therefore, the probability that $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ returns `success` *or* `abort` *is identical in real execution and ideal simulation. For the executions of the* `Extend` *procedure, the view of $\mathcal{A}$ is simulated perfectly. However, the distribution of the output $\mathbf{x}[m+1:n]$ of the honest $P_{\mathcal{S}}$ is different to that in ideal simulation. Thus, environment $\mathcal{Z}$ can not distinguish between two worlds if the the decisional $(\mathtt{RG}, \mathcal{G}, \mathtt{GR}(2^k,d))$-LPN$(m,n,t)$ with static leakage assumption holds. This completes the proof.* □

E VOLE based on dual LPN

Here we focus on reducing the round complexity of the VOLE extension protocol. Our (primal LPN-based) construction described in Section 4 can not be made two-round, since the two parties need to obtain an additional VOLE correlation for the check. Following the approach of [16], we construct a two-round VOLE extension protocol that relies on the ideal functionality for generalized VOLE over Galois rings (see $\mathcal{F}_{\mathrm{gVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 20). Informally in the generalized VOLE,

the two parties obtain two VOLE correlations with the same length for different global keys, e.g., $\mathbf{K} = \mathbf{M} + \mathbf{y} \cdot \Delta$ and $\mathbf{a} = \mathbf{c} + \mathbf{y} \cdot \delta$. The functionality $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$ can be naturally realized by OTs. To achieve malicious security, as shown in [51] we can apply a Galois ring analogue of the eVOLE technique [32].

We provide a multi-point VOLE construction in Figure 22, which can be viewed as a combination of multiple single-point VOLEs intuitively. The ideal functionality for multi-point VOLE is presented in Figure 21.

Suppose the two parties want to obtain a t-point VOLE correlation, and w.l.o.g. we assume the length n has the form $t \cdot 2^h$. We still use the GGM tree algorithms (Figure 19) to let the two parties obtain PPRF outputs. But, this time we let $P_{\mathcal{R}}$ calculate one more layer for each GGM tree, and view $\{\Gamma_i\}_{i\in[0,2^h)}$ as the right leaves of the last layer. For the j-th GGM tree, $j \in [t]$, we let $P_{\mathcal{S}}$ always learn $\{\Gamma_i^{(j)}\}_{i\in[0,2^h)}$. Thus, $P_{\mathcal{S}}$ can check the consistency of each GGM tree independently, by hashing the right leaves. We remark that there is a recent work [44] showing how to reduce the cost of generating GGM trees by half. However, their constructions are only semi-honestly secure and it remains unclear how to construct multi-point VOLE protocols with malicious security basing on their techniques.

We slightly modify the $\mathtt{GGM.Eval}$ algorithm and obtain $\mathtt{GGM.Eval}'$ in Figure 19. To ensure all the right leaves fix a unique tree, we require that G' has the right half injective property [10].

Functionality $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$

Init: Upon receiving $(\mathtt{Init})$ from both parties, sample $\Delta, \delta \xleftarrow{\$} \mathtt{GR}(2^k, d)$ if receiver $P_{\mathcal{R}}$ is honest, and receive $\Delta, \delta \in \mathtt{GR}(2^k, d)$ from the adversary $\mathcal{A}$ otherwise. Store Δ, δ and send them to $P_{\mathcal{R}}$. All further $(\mathtt{Init})$ commands will be ignored.
Extend: Upon receiving $(\mathtt{Extend}, t)$ from both parties, proceed as follows:

1. If $P_{\mathcal{R}}$ is honest, sample $\mathbf{K}, \mathbf{a} \xleftarrow{\$} \mathtt{GR}(2^k, d)^t$. Otherwise, receive $\mathbf{K}, \mathbf{a}$ from $\mathcal{A}$.
2. If $P_{\mathcal{S}}$ is honest, sample $\mathbf{y} \xleftarrow{\$} \mathtt{GR}(2^k, d)^t$ with each component invertible, and compute $\mathbf{M} := \mathbf{K} - \Delta \cdot \mathbf{y}$ and $\mathbf{c} := \mathbf{a} - \delta \cdot \mathbf{y}$. Otherwise, receive $\mathbf{M}, \mathbf{c}, \mathbf{y}$ from $\mathcal{A}$, and then recompute $\mathbf{K} := \mathbf{M} + \Delta \cdot \mathbf{y}$ and $\mathbf{a} := \mathbf{c} + \delta \cdot \mathbf{y}$.
3. Send $(\mathbf{M}, \mathbf{c}, \mathbf{y})$ to $P_{\mathcal{S}}$ and send $(\mathbf{K}, \mathbf{a})$ to $P_{\mathcal{R}}$.

Fig. 20: Ideal functionality for generalized VOLE over $\mathtt{GR}(2^k, d)$.

[10] As noted in [16], the right-half injectivity requirement can be relaxed to right-half collision resistance, if G' is sampled uniformly at random from a family of hash functions that are collision-resistant in their right-half output.

Definition 5. *We say a function $f = (f_0, f_1) : \{0,1\}^\lambda \to \mathtt{GR}(2^k, d)^2 \times \{0,1\}^\lambda$ has the right half injective property, if and only if $f_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is injective.*

Then, from the j-th GGM tree, the two parties can obtain two single-point VOLE correlations of length 2^h, $\mathbf{V}_0^{(j)} = \mathbf{W}_0^{(j)} + \mathbf{e}^{(j)} \cdot \Delta$ and $\mathbf{V}_1^{(j)} = \mathbf{W}_1^{(j)} + \mathbf{e}^{(j)} \cdot \delta$. To guarantee the VOLE correlation on the position $\alpha^{(j)}$, we use a random linear combination check that sacrifices half of the PPRF outputs, i.e., $\mathbf{W}_1^{(j)}, \mathbf{V}_1^{(j)}$. We remark that these two checks allow the corrupted receiver to query for the noisy positions twice, while the corrupted sender learns nothing about Δ. We have the following theorem.

Functionality $\mathcal{F}_{\text{mpVOLE}}^{\mathtt{GR}(2^k,d)}$

Init: Upon receiving (`Init`) from both parties, sample $\Delta \xleftarrow{\$} \mathtt{GR}(2^k, d)$ if receiver $P_\mathcal{R}$ is honest, and receive $\Delta \in \mathtt{GR}(2^k, d)$ from the adversary $\mathcal{A}$ otherwise. Store Δ and send it to $P_\mathcal{R}$. All further (`Init`) commands will be ignored.
MP-Extend: Upon receiving ($\mathtt{MP\text{-}Extend}, n = t \cdot 2^h$) from both parties, proceed as follows:

1. If $P_\mathcal{R}$ is honest, sample $\mathbf{v} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$. Otherwise, receive $\mathbf{v}$ from $\mathcal{A}$.
2. If $P_\mathcal{S}$ is honest, sample $\alpha_1, ..., \alpha_t \xleftarrow{\$} [0, 2^h)$, and $y_1, ..., y_t \xleftarrow{\$} \mathtt{GR}(2^k, d)^*$, and define $\mathbf{e}^{(j)} \in \mathtt{GR}(2^k, d)^{2^h}$ such that $e_{\alpha_j}^{(j)} = y_j$ and $e_i^{(j)} = 0, i \neq \alpha_j$, for $j \in [t]$, then computes $\mathbf{w} := \mathbf{v} - \Delta \cdot (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$. Otherwise, receive $\mathbf{e}^{(j)}$ with at most one invertible entry and zeros everywhere else, for $j \in [t]$, and $\mathbf{w}$ from $\mathcal{A}$, then recompute $\mathbf{v} := \mathbf{w} + \Delta \cdot (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$.
3. If $P_\mathcal{R}$ is corrupted, receive a series of sets $I^{(j)} \subseteq [0, 2^h)$ from $\mathcal{A}$. If $\alpha_j \in I^{(j)}$ for all $j \in [t]$, send `success` to $\mathcal{A}$ and continue. Otherwise, send `abort` to both parties and abort. Construct a set $J := \{j \in [t] \mid |I^{(j)}| = 1\}$.
4. If $P_\mathcal{R}$ is corrupted, receive a series of elements $\hat{\alpha}_j$, where $\hat{\alpha}_j \in I^{(j)}$ and $j \in [t] \backslash J$. If $\hat{\alpha}_j = \alpha_j$ for all $\hat{\alpha}_j$ received from $\mathcal{A}$, send `success` to $\mathcal{A}$ and continue. Otherwise, send `abort` to both parties and abort.
5. Send $\mathbf{w}, (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$ to $P_\mathcal{S}$ and $\mathbf{v}$ to $P_\mathcal{R}$.

Fig. 21: Ideal functionality for multi-point VOLE over $\mathtt{GR}(2^k, d)$.

Theorem 17. *If G and G' are PRGs with G' having the right half injective property, and $\mathtt{Hash} : \{0,1\}^{2^h\lambda} \to \{0,1\}^\lambda$ is a collision resistant hash function, $\Pi_{\text{mpVOLE}}^{\mathtt{GR}(2^k,d)}$ realizes the functionality $\mathcal{F}_{\text{mpVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $(\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\text{OT}})$-hybrid model with malicious security.*

Proof. ***Malicious Sender.*** *The simulator $S_\mathcal{S}$ acts as follows.*

1. *$S_\mathcal{S}$ records the inputs $\mathbf{M}, \mathbf{c}, \mathbf{y} \in \mathtt{GR}(2^k, d)^t$ sent to $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$ by $\mathcal{A}$.*

Protocol $\Pi^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$

Init: Both parties send $\mathtt{Init}$ to $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}$, which returns Δ, δ to $P_{\mathcal{R}}$.

MP-Extend: On input $n = t \cdot 2^h$, the parties do as follows:
Construct:

(a) Both parties send $(\mathtt{Extend}, t)$ to $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}$, which returns $(\mathbf{M}, \mathbf{c}, \mathbf{y})$ to $P_{\mathcal{S}}$ and $(\mathbf{K}, \mathbf{a})$ to $P_{\mathcal{R}}$ such that $\mathbf{K} = \mathbf{M} + \mathbf{y} \cdot \Delta$ and $\mathbf{a} = \mathbf{c} + \mathbf{y} \cdot \delta$, where each entry of $\mathbf{y}$ is invertible.

(b) For $j \in [t]$, both parties do as follows. The receiver $P_{\mathcal{R}}$ computes $s^{(j)} \leftarrow \mathtt{GGM.KeyGen}(1^\lambda)$, and runs $\mathtt{GGM.Gen}(2^{h+1}, s^{(j)})$ to obtain $(\{(v^{(j)}_{2i}, v^{(j)}_{2i+1}, \Gamma^{(j)}_i)\}_{i\in[0,2^h)}, \{(K^{(j)}_{0,i}, K^{(j)}_{1,i})\}_{i\in[h+1]})$. Recompute $K^{(j)}_{1,h+1} := \bigoplus_{i\in[0,2^h)} \Gamma^{(j)}_i$. Let $\alpha^{(j)} = \sum_{i\in[h]} 2^{h-i}\alpha^{(j)}_i$, where $\alpha^{(j)}_i \xleftarrow{\$} \{0,1\}$, and let $\bar{\alpha}^{(j)}_i$ denote the complement of $\alpha^{(j)}_i$. For $i \in [h]$, $P_{\mathcal{S}}$ sends $\bar{\alpha}^{(j)}_i$ to $\mathcal{F}_{\mathrm{OT}}$ (as OT receiver) while $P_{\mathcal{R}}$ sends $(K^{(j)}_{0,i}, K^{(j)}_{1,i})$ to $\mathcal{F}_{\mathrm{OT}}$ (as OT sender), then $P_{\mathcal{S}}$ receives $K^{(j)}_{\bar{\alpha}^{(j)}_i, i}$. $P_{\mathcal{R}}$ sends $K^{(j)}_{1,h+1}$ to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ runs $\mathtt{GGM.Eval'}(\alpha^{(j)}, \{K^{(j)}_{\bar{\alpha}^{(j)}_i, i}\}_{i\in[h]}, K^{(j)}_{1,h+1})$ and gets $(\{(v^{(j)}_{2i}, v^{(j)}_{2i+1})\}_{i\neq\alpha^{(j)}}, \{\Gamma^{(j)}_i\}_{i\in[0,2^h)})$. Additionally, $P_{\mathcal{R}}$ sends $g^{(j)}_0 := K_j - \sum_{i\in[0,2^h)} v^{(j)}_{2i}$ and $g^{(j)}_1 := a_j - \sum_{i\in[0,2^h)} v^{(j)}_{2i+1} \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{S}}$.

(c) $P_{\mathcal{S}}$ defines vectors $\mathbf{W}^{(j)}_0, \mathbf{W}^{(j)}_1 \in \mathtt{GR}(2^k, d)^{2^h}$ such that $W^{(j)}_{0,i} = v^{(j)}_{2i}$, $W^{(j)}_{1,i} = v^{(j)}_{2i+1}$ for $i \neq \alpha^{(j)}$ and $W^{(j)}_{0,\alpha^{(j)}} := M_j - (g^{(j)}_0 + \sum_{i\neq\alpha^j} W^{(j)}_{0,i})$, $W^{(j)}_{1,\alpha^{(j)}} := c_j - (g^{(j)}_1 + \sum_{i\neq\alpha^{(j)}} W^{(j)}_{1,i})$. $P_{\mathcal{R}}$ defines vectors $\mathbf{V}^{(j)}_0 := (v^{(j)}_0, v^{(j)}_2, ..., v^{(j)}_{2^{h+1}-2})$ and $\mathbf{V}^{(j)}_1 := (v^{(j)}_1, v^{(j)}_3, ..., v^{(j)}_{2^{h+1}-1})$. Note that $\mathbf{V}^{(j)}_0 = \mathbf{W}^{(j)}_0 + \mathbf{e}^{(j)} \cdot \Delta$ and $\mathbf{V}^{(j)}_1 = \mathbf{W}^{(j)}_1 + \mathbf{e}^{(j)} \cdot \delta$, where $\mathbf{e}^{(j)} \in \mathtt{GR}(2^k, d)^{2^h}$ has only one entry invertible in $\mathtt{GR}(2^k, d)$ and zeros everywhere else, i.e., $e^{(j)}_{\alpha^j} = y_j \in \mathtt{GR}(2^k, d)^*$ and $e^{(j)}_i = 0$ for $i \neq \alpha^j$.

(d) $P_{\mathcal{R}}$ computes $\Gamma^{(j)} := \mathtt{Hash}(\Gamma^{(j)}_0, ..., \Gamma^{(j)}_{2^h-1})$ and sends it to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ computes $\hat{\Gamma}^{(j)} := \mathtt{Hash}(\Gamma^{(j)}_0, ..., \Gamma^{(j)}_{2^h-1})$ and checks $\Gamma^{(j)} = \hat{\Gamma}^{(j)}$, for all $j \in [t]$. If the check fails, $P_{\mathcal{S}}$ aborts.

(e) $P_{\mathcal{S}}$ samples random $\chi, \chi_0, ..., \chi_{2^h-1} \in \mathbb{F}_{2^d}$ and sends them to $P_{\mathcal{R}}$. For $j \in [t]$, $P_{\mathcal{R}}$ computes $V_j := \sum_{i\in[0,2^h)} \chi_i(v^{(j)}_{2i} + \chi \cdot v^{(j)}_{2i+1})$. $P_{\mathcal{R}}$ computes $X := \Delta + \chi \cdot \delta$. $P_{\mathcal{R}}$ sends $\{V_j\}_{j\in[t]}$ and X to $P_{\mathcal{S}}$.

(f) $P_{\mathcal{S}}$ computes $W_j := \sum_{i\in[0,2^h)} \chi_i(W^{(j)}_{0,i} + \chi \cdot W^{(j)}_{1,i})$. $P_{\mathcal{S}}$ checks that $V_j - W_j = X \cdot \chi_{\alpha^{(j)}} \cdot y_j$ holds for all $j \in [t]$. If the check fails, $P_{\mathcal{S}}$ aborts.

Output: $P_{\mathcal{S}}$ outputs $(\mathbf{W}^{(1)}_0, ..., \mathbf{W}^{(t)}_0) \in \mathtt{GR}(2^k, d)^n, (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)}) \in \mathtt{GR}(2^k, d)^n$. $P_{\mathcal{R}}$ outputs $(\mathbf{V}^{(1)}_0, ..., \mathbf{V}^{(t)}_0) \in \mathtt{GR}(2^k, d)^n$.

Fig. 22: Protocol for multi-point VOLE over $\mathtt{GR}(2^k, d)$ in the $(\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}, \mathcal{F}_{\mathrm{OT}})$-hybrid model.

2. *For $j \in [t]$ and $i \in [h]$, $S_{\mathcal{S}}$ records the input $\bar{\alpha}_i^{(j)}$ sent to $\mathcal{F}_{\mathrm{OT}}$ by $\mathcal{A}$, then $S_{\mathcal{S}}$ can recover the values $\alpha^{(j)}, j \in [t]$. For $j \in [t]$, $S_{\mathcal{S}}$ computes $s^{(j)} \leftarrow \mathtt{GGM.KeyGen}(1^\lambda)$, and gets $(\{(v_{2i}^{(j)}, v_{2i+1}^{(j)}, \Gamma_i^{(j)})\}_{i\in[0,2^h)}, \{(K_{0,i}^{(j)}, K_{1,i}^{(j)})\}_{i\in[h+1]}) \leftarrow \mathtt{GGM.Gen}(2^{h+1}, s^{(j)})$. $S_{\mathcal{S}}$ resets $K_{1,h+1}^{(j)} := \bigoplus_{i\in[0,2^h)} \Gamma_i^{(j)}$. $S_{\mathcal{S}}$ emulates $\mathcal{F}_{\mathrm{OT}}$ by sending $K_{\bar{\alpha}_i^{(j)},i}^{(j)}$ to $\mathcal{A}$. $S_{\mathcal{S}}$ sends $g_0^{(j)}, g_1^{(j)} \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and $K_{1,h+1}^{(j)}$ to $\mathcal{A}$.*
3. *For $j \in [t]$, $S_{\mathcal{S}}$ computes $\Gamma^{(j)} := \mathtt{Hash}(\Gamma_0^{(j)}, ..., \Gamma_{2^h-1}^{(j)})$ and sends it to $\mathcal{A}$.*
4. *Upon receiving $\chi, \chi_0, ..., \chi_{2^h-1} \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$, for $j \in [t]$, $S_{\mathcal{S}}$ computes $W_{0,\alpha^{(j)}}^{(j)} := M_j - g_0^{(j)} - \sum_{i\neq\alpha^{(j)}} v_{2i}^{(j)}$ and $W_{1,\alpha^{(j)}}^{(j)} := c_j - g_1^{(j)} - \sum_{i\neq\alpha^{(j)}} v_{2i+1}^{(j)}$. Next, $S_{\mathcal{S}}$ computes $W_j := \sum_{i\neq\alpha^{(j)}} \chi_i(v_{2i}^{(j)} + \chi \cdot v_{2i+1}^{(j)}) + \chi_{\alpha^{(j)}}(W_{0,\alpha^{(j)}}^{(j)} + \chi \cdot W_{1,\alpha^{(j)}}^{(j)})$. Finally, $S_{\mathcal{S}}$ samples $X \xleftarrow{\$} \mathtt{GR}(2^k, d)$, and sends $X, \{V_j := W_j + X \cdot \chi_{\alpha^{(j)}} \cdot y_j\}_{j\in[t]}$ to $\mathcal{A}$.*
5. *$S_{\mathcal{S}}$ sets $W_{0,i}^{(j)} := v_{2i}^{(j)}$, for $j \in [t], i \in [0, 2^h), i \neq \alpha^{(j)}$. $S_{\mathcal{S}}$ sets $\mathbf{e}^{(j)} \in \mathtt{GR}(2^k, d)^{2^h}$ such that $e_{\alpha^{(j)}}^{(j)} := y_j$, and $e_i^{(j)} = 0$ for $i \neq \alpha^{(j)}$. $S_{\mathcal{S}}$ sends $(\mathbf{W}_0^{(1)}, ..., \mathbf{W}_0^{(t)})$, $(\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$ to $\mathcal{F}_{\mathrm{mpVOLE}}^{\mathtt{GR}(2^k,d)}$.*

The messages produced by $S_{\mathcal{S}}$ are generated in the same way to that in the real execution except for $\{g_0^{(j)}, g_1^{(j)}\}_{j\in[t]}$ and V, X. It is not hard to see that these messages (except V, since V is dependent on X) are uniformly random in $\mathcal{A}$'s view in the real execution. Besides, $S_{\mathcal{S}}$ can correctly extract $(\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$ and compute $(\mathbf{W}_0^{(1)}, ..., \mathbf{W}_0^{(t)})$, which guarantees the indistinguishability of the outputs. Thus, the real execution and the ideal simulation are indistinguishable.

Malicious Receiver. *The simulator $S_{\mathcal{R}}$ acts as follows.*

1. *$S_{\mathcal{R}}$ records the input Δ, δ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{gVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $\mathtt{Init}$ phase. Later in the $\mathtt{Extend}$ phase, $S_{\mathcal{R}}$ records the input $\mathbf{K}, \mathbf{a} \in \mathtt{GR}(2^k, d)^t$ sent to $\mathcal{F}_{\mathrm{gVOLE}}^{\mathtt{GR}(2^k,d)}$ by $\mathcal{A}$.*
2. *For $j \in [t], i \in [0, 2^h)$, $S_{\mathcal{R}}$ records the input $(K_{0,i}^{(j)}, K_{1,i}^{(j)})$ sent to $\mathcal{F}_{\mathrm{OT}}$ by $\mathcal{A}$. $S_{\mathcal{R}}$ sets $\alpha^{(l)} = l$, for $l \in [0, 2^h)$. Upon receiving $\{K_{1,h+1}^{(j)}, \Gamma^{(j)}\}_{j\in[t]}$, $S_{\mathcal{R}}$ runs $\mathtt{GGM.Eval}'(\alpha^{(l)}, \{K_{\bar{\alpha}_i^{(l)},i}^{(j)}\}_{i\in[h]}, K_{1,h+1}^{(j)})$ and gets $(\{(v_{2i}^{(j,l)}, v_{2i+1}^{(j,l)})\}_{i\neq\alpha^{(l)}}, \{\Gamma_i^{(j,l)}\}_{i\in[0,2^h)})$, for $l \in [0, 2^h)$. $S_{\mathcal{R}}$ computes $\Gamma_l^{(j)} := \mathtt{Hash}(\Gamma_0^{(j,l)}, ..., \Gamma_{2^h-1}^{(j,l)})$.*
3. *For $j \in [t]$, $S_{\mathcal{R}}$ builds a series of sets $I^{(j)} := \{l \in [0, 2^h) \mid \Gamma_l^{(j)} = \Gamma^{(j)}\} \subseteq [0, 2^h)$. If there exists a $j \in [t]$ such that $I^{(j)} = \emptyset$, $S_{\mathcal{R}}$ aborts. $S_{\mathcal{R}}$ sends $\{I^{(j)}\}_{j\in[t]}$ to $\mathcal{F}_{\mathrm{mpVOLE}}^{\mathtt{GR}(2^k,d)}$. If it aborts, then $S_{\mathcal{R}}$ aborts.*
4. *$S_{\mathcal{R}}$ builds a set $J := \{j \in [t] \mid |I^{(j)}| = 1\}$. If there exists some $l \neq l' \in I^{(j)}$, $j \in [t]\backslash J$ and $i \neq l, l'$ such that $v_{2i}^{(j,l)} \neq v_{2i}^{(j,l')}$, $S_{\mathcal{R}}$ aborts. Thus, for $j \in [t]\backslash J$, $S_{\mathcal{R}}$ can obtain consistent $\{(v_{2i}^{(j)}, v_{2i+1}^{(j)})\}_{i\in[0,2^h)}$. Besides, for $j \in J$, suppose $I^{(j)} = \{\alpha^{(j)}\}$, then $S_{\mathcal{R}}$ can set $v_{2\alpha^{(j)}}^{(j)} = v_{2\alpha^{(j)}+1}^{(j)} = 0$ to obtain $\mathbf{V}_0^{(j)}, \mathbf{V}_1^{(j)}$.*
5. *Upon receiving $\{g_0^{(j)}, g_1^{(j)}\}_{j\in[t]}$ from $\mathcal{A}$, $S_{\mathcal{R}}$ samples random $\chi, \chi_0, ..., \chi_{2^h-1} \xleftarrow{\$} \mathbb{F}_{2^d}$ and sends them to $\mathcal{A}$. For $j \in [t]$, $S_{\mathcal{R}}$ computes $\beta_0^{(j)} := g_0^{(j)} - (K_j -$*

$\sum_{i\in[0,2^h)} v_{2i}^{(j)}$) *and* $\beta_1^{(j)} := g_1^{(j)} - (a_j - \sum_{i\in[0,2^h)} v_{2i+1}^{(j)})$. *If there exists a* $j \in [t]$ *such that* $\beta_1^{(j)} \neq 0$, *but* $\beta_0^{(j)} + \chi \cdot \beta_1^{(j)} = 0$, $S_{\mathcal{R}}$ *aborts.* $S_{\mathcal{R}}$ *computes* $\hat{V}_j := \sum_{i\in[0,2^h)} \chi_i(v_{2i}^{(j)} + \chi \cdot v_{2i+1}^{(j)})$.

6. *Upon receiving* $\{V_j\}_{j\in[t]}$ *and* X *from* $\mathcal{A}$, *for* $j \in [t]$ *with* $\beta_1^{(j)} \neq 0$, $S_{\mathcal{R}}$ *tries to find* $\hat{\alpha}^{(j)} \in I^{(j)}$ *such that* $\hat{V}_j - V_j = \chi_{\hat{\alpha}^{(j)}}(\beta_0^{(j)} + \chi \cdot \beta_1^{(j)})$. *If such an* $\hat{\alpha}^{(j)}$ *does not exist or is not unique,* $S_{\mathcal{R}}$ *aborts.*
7. *For* $j \in [t]\backslash J$ *with* $\beta_1^{(j)} \neq 0$, $S_{\mathcal{R}}$ *resets* $I^{(j)} := \{\hat{\alpha}^{(j)}\}$, *and sends* $I^{(j)}$ *to* $\mathcal{F}_{\text{mpVOLE}}^{\mathtt{GR}(2^k,d)}$. *If it aborts, then* $S_{\mathcal{R}}$ *aborts.*
8. $S_{\mathcal{R}}$ *sends* $(\mathbf{V}_0^{(1)}, ..., \mathbf{V}_0^{(t)})$ *to* $\mathcal{F}_{\text{mpVOLE}}^{\mathtt{GR}(2^k,d)}$.

We show the probability that honest sender aborts in a real execution is negligibly close to aborting in the ideal simulation. If $P_{\mathcal{S}}$ *aborts in the real execution due to some* $j \in [t]$ *such that* $\hat{\Gamma}^{(j)} \neq \Gamma^{(j)}$, *this will lead to that* $\alpha^{(j)} \notin I^{(j)}$ *in the ideal simulation, which means that* $S_{\mathcal{R}}$ *will abort as well.*

Next, we claim the probability that $S_{\mathcal{R}}$ *aborts in the step 4 of the simulation is negligible. We prove it by contradiction. If such* $l \neq l' \in I^{(j)}$, $j \in [t]\backslash J$ *exist, from the construction of GGM tree, there exists* $\rho, \rho' \in \{0,1\}^\lambda$ *such that* $(v_{2i}^{(j,l)}, v_{2i+1}^{(j,l)}, \Gamma_i^{(j,l)}) = G'(\rho)$ *and* $(v_{2i}^{(j,l')}, v_{2i+1}^{(j,l')}, \Gamma_i^{(j,l')}) = G'(\rho')$. *Thus, we have that* $\rho \neq \rho'$. *By the right-half injectivity of* G', *we have* $\Gamma_i^{(j,l)} \neq \Gamma_i^{(j,l')}$, *which leads to* $\Gamma_l^{(j)} \neq \Gamma_{l'}^{(j)}$ *overwhelmingly. However,* $\Gamma_l^{(j)} = \Gamma_{l'}^{(j)} = \Gamma^{(j)}$ *since* $l, l' \in I^{(j)}$. *This completes the proof of the above claim.*

If $\mathcal{A}$ *behaves honestly in the* j*-th iteration, we have* $\beta_0^{(j)} = \beta_1^{(j)} = 0$. *Since* χ *is picked uniformly at random in* $\mathbb{F}_{2^d}$ *by* $S_{\mathcal{R}}$, *the probability that* $\beta_0^{(j)} + \chi \cdot \beta_1^{(j)} = 0$ *with* $\beta_1^{(j)} \neq 0$ *is equal to* $1/2^d$ *form Lemma 1. By a union bound,* $S_{\mathcal{R}}$ *aborts in the step 5 of the simulation with probability at most* $t/2^d$.

If $\mathcal{A}$ *sends* $(\hat{g}_0^{(j)} := g_0^{(j)} + \beta_0^{(j)}, \hat{g}_1^{(j)} := g_1^{(j)} + \beta_1^{(j)})$ *instead of correct* $(g_0^{(j)}, g_1^{(j)})$, *there will be a bias* $\chi_{\alpha^{(j)}}(\beta_0^{(j)} + \chi \cdot \beta_1^{(j)})$ *for* W_j *computed by honest* $P_{\mathcal{S}}$. *Therefore, it can be observed that if* $\mathcal{A}$ *successfully guesses the* $\alpha^{(j)}$ *chosen by honest* $P_{\mathcal{S}}$, $\mathcal{A}$ *can pass the check of* $P_{\mathcal{S}}$ *by sending* $V_j := \hat{V}_j - \chi_{\alpha^{(j)}}(\beta_0^{(j)} + \chi \cdot \beta_1^{(j)})$. *On the other hand, if* $S_{\mathcal{R}}$ *can not find a solution* $\hat{\alpha}^{(j)}$, *the honest* $P_{\mathcal{S}}$ *will abort, no matter what* $\alpha^{(j)}$ *he picks. As the coefficients* $\chi_0, ..., \chi_{2^h-1}$ *are sampled uniformly at random by* $S_{\mathcal{R}}$, *they are pair-wise distinct except with probability at most* $1/2^{d-h}$.

Therefore, from above discussions, the probability that $S_{\mathcal{R}}$ *aborts while honest* $P_{\mathcal{S}}$ *does not abort is bounded by* $(t/2^d + 1/2^{d-h} + \mathsf{negl}(\lambda))$. *This concludes the whole proof.* □

$P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ can naturally obtain a pseudorandom VOLE correlation with the help of a multi-point VOLE, just multiplying the functionality outputs by a parity check matrix H for which a dual variant of the LPN assumption holds. Here we can use the dual variant to further reduce the communication complexity, which is formally defined as follows.

Definition 6 (Dual LPN with static leakage). *We first describe the corresponding dual LPN security game* G_{Dual} *as follows.*

1. *Let* $H \leftarrow \mathcal{G}^{\perp}(m,n) \in \mathtt{GR}(2^k,d)^{n\times(n-m)}$ *be a dual LPN matrix, and* $\mathbf{e} = (\mathbf{e}^{(1)},...,\mathbf{e}^{(t)}) \in \mathtt{GR}(2^k,d)^n$ *be the error vector, where each* $\mathbf{e}^{(i)}$ *has only one entry invertible in* $\mathtt{GR}(2^k,d)$ *and zeros everywhere else.*
2. $\mathcal{A}$ *sends* $I_1,...,I_t \subset [n/t]$*. If for all* $i \in [t]$*,* I_i *includes the noisy position of* $\mathbf{e}^{(i)}$*, send* `success` *to* $\mathcal{A}$*. Otherwise, abort. Construct a set* $J := \{j \in [t] \mid |I_j| = 1\}$*.*
3. $\mathcal{A}$ *sends* $\hat{\alpha}_i$*, where* $\hat{\alpha}_i \in I_i$ *and* $i \in [t]\backslash J$*. If for all* $i \in [t]\backslash J$*,* $\hat{\alpha}_i$ *is the exact noisy position of* $\mathbf{e}^{(i)}$*, send* `success` *to* $\mathcal{A}$*. Otherwise, abort.*
4. *Pick* $b \leftarrow \{0,1\}$*. If* $b = 0$*, send* $\mathbf{y} := \mathbf{e} \cdot H$ *to* $\mathcal{A}$*, otherwise, send* $\mathbf{y} \xleftarrow{\$} \mathtt{GR}(2^k,d)^{(n-m)}$ *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *outputs a bit* b'*. The game outputs* 1 *if* $b' = b$*, and outputs* 0 *otherwise.*

We say that the decisional dual $(\mathtt{RG},\mathcal{G},\mathcal{R}) - \mathrm{LPN}(m,n,t)$ *with static leakage is* (T,ϵ)*-hard, if for every probabilistic distinguisher* $\mathcal{B}$ *running in time* T*,* $\mathcal{B}$ *wins the game* G_{Dual} *with advantage at most* ϵ*, i.e.,*

$$|\Pr[G_{\mathrm{Dual}} = 1] - 1/2| \leq \epsilon.$$

By Fiat-Shamir transform, we immediately obtain a two-round multi-point VOLE protocol, which yields a round optimal VOLE extension protocol. We remark that when the length of required VOLE correlation is small, or namely the "bootstrapping" in the primal LPN-based construction is not activated, the two-round variant has lower communication complexity, as dual-LPN allows for polynomial stretch.