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, in particular a prover memory footprint almost the same as computing the circuit in the clear. 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 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 rings $\mathbb{Z}_{2^k}$ 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 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: our efficiency is independent of the security parameter (so increasing superiority as the security parameter grows); even for concrete $40, 80$ bits soundness on $32, 64$-bit CPUs, we all offer savings (up to $3\times$ at best).
(2) Inspired by the recently proposed *interactive* message authentication code technique (CCS'22), we construct a constant round VOLE-based ZK protocol over $\mathbb{Z}_{2^k}$ with sublinear (in the circuit size) communication complexity, which was previously achieved only over fields.
(3) We adapt the VOLE-in-the-head technique, and apply it to our first ZK protocol, yielding the first publicly verifiable *non-interactive ZK* (NIZK) over $\mathbb{Z}_{2^k}$ with linear communication complexity. Also, we show that the *pseudorandom correlation generator* (PCG) approach (incompatible with VOLE-in-the-head) can be adapted to efficiently implement VOLE over Galois rings, with analysis of the hardness of underlying assumptions.

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 different 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 [42], offers a *publicly verifiable* solution to *non-interactive ZK* (NIZK), where the prover emulates in his head the evaluation of the 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 [42,36,22,44], or small proof size but large prover time and memory [2]. The garbled circuit [57] ZK (GCZK) [43,33,58,41] paradigm admits ZK protocols with small prover time and memory, but large proof size, where the verifier plays the role of garbler (who garbles the circuit). The interactive oracle proof (IOP)-based ZK proofs admit *zero-knowledge succinct non-interactive arguments of knowledge* (zk-SNARK) protocols [39,18,8,34,9,24,51]. Most zk-SNARKs achieve short proof size and small verification time simultaneously (assuming a setup). However, many of them require the prover has sufficient computation power (namely, large prover time and memory). We remark that very recently, there is a line of works [11,13,55,49,12,38] focusing on zk-SNARKs with linear prover time [1], as proving time becomes 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 $\Omega(|\mathcal{C}|)$ [13,55,38], and succinct proof size can be achieved for any field [49,12]. These constructions are still not scalable, due to the large memory requirement.

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 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. In a high level, by viewing a ZK proof as a special case of secure two-party computation (2-PC), where only the prover (sender) has inputs, each value on the wire is authenticated by a linearly homomorphic message authentication code (MAC), and the prover proves to the verifier that the authenticated values satisfy the circuit topology. In more detail, VOLE-based ZK protocols have two phases, an offline phase that generates random VOLE-based MACs, and an online phase that securely evaluates the circuit.

[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.

Boyle et al. [14,16] 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 locally to a batch of correlated randomness between some parties. PCG offers a lightweight candidate for generating random VOLE correlations in the offline phase. The authors of [15] 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 [53] 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 [30] 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 [56] and then improved LPZK [29], AntMan [54]. 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 [29] reduced the communication from 1 field element per multiplication gate to 1/2. AntMan [54] 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 the technical details. Very recently, Baum et al. [5] proposed the VOLE-in-the-head technique, which compiles a *public coin* VOLE-based ZK protocol into a *publicly verifiable* NIZK, by using a SoftSpokenOT [50] style VOLE construction instead of a PCG-style VOLE construction. On the other hand, VOLE-in-the-head can be viewed as an optimization of MPCitH.

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 over $\mathbb{Z}_{2^k}$ 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. To our best knowledge, there are only three existing works that constructed ZK protocols over $\mathbb{Z}_{2^k}$.

Ganesh et. al [35] proposed Rinocchio by adapting Pinocchio [47], a SNARK for field arithmetic, to work for ring arithmetic, while other SNARKs seems very hard to adapt. They introduced the quadratic ring program (QRP) problem and achieved a *designated verifier* zk-SNARK with succinct proof size (not succinct verification) for proving QRP over a general ring $\mathcal{R}$ that contains sufficiently large exceptional sets. When used for $\mathbb{Z}_{2^k}$, Rinocchio suffers from a significant efficiency loss, as exceptional sets of $\mathbb{Z}_{2^k}$ are small. Following the blueprint of VOLE-based ZK protocols over finite fields, and also inspired by the idea of SPD$\mathbb{Z}_{2^k}$ [25], 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]. The main drawback of these constructions, is that the efficiency has an inherent undesirable dependency on the security parameter.

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

We introduce more powerful tools (namely, the *reverse multiplication friendly embedding* (RMFE) techniques [20,21,26,31]) 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. Our protocol has the main advantage that the efficiency is independent on 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	4	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 asymptotic efficiency. We remark that it seems impossible 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

[2] *Appenzeller to Brie* adapted the online phases of Wolverine [53] and Mac'n'Cheese [7], respectively. Moz$\mathbb{Z}_{2^k}$arella adapted the online phase of QuickSilver [56].

[3] We select RMFEs over binary field according to [20], and lift to Galois rings via the approach in [26].

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 a 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 several efficient constructions for VOLE over Galois rings. We first present a construction via the SoftSpokenOT [50] idea, which has fascinating small communication complexity. And in particular, instantiating the VOLE functionality of our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ with this construction, allows us to apply the VOLE-in-the-head [5] technique, then we obtain the first publicly verifiable NIZK over $\mathbb{Z}_{2^k}$ with linear (in the circuit size, independent of the security parameter) communication complexity.

To achieve better concrete efficiency (especially smaller computation complexity), we present two constructions following the PCG paradigm. The first primal-LPN based construction is constant-round, and has generally best performance (among the three). The second dual-LPN based construction is two-round, at the cost of slightly larger computation complexity. In the end, we analyse the security of LPN over Galois rings. We remark that PCG-style VOLE constructions are incompatible with the VOLE-in-the-head technique, and therefore, one may do some trade-offs in real-life applications.

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 [53,56,54] actually work on a sufficiently large field for security guarantee. For the Galois ring analogue protocols, the degree d of $\mathtt{GR}(2^k, d)$ is required to be sufficiently large accordingly (this is essentially due to Lemma 1).

The second step is to modify the above ZK protocols over $\mathtt{GR}(2^k, d)$ to efficiently "simulate" ZK protocols 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 $\mathbf{w}$ are over $\mathbb{Z}_{2^k}$. Our idea for the efficient "simulation" 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. Intuitively, ϕ "packs" m multiplications over $\mathbb{Z}_{2^k}$ to one multiplication over $\mathtt{GR}(2^k, d)$, and 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 [20,21,1,31] that applied RMFEs in MPC, have paid

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

great efforts to guarantee that ϕ, ψ are applied honestly (to achieve malicious security).

Observing that in the ZK for proving circuit satisfiability setting, the prover (of the ZK protocol over $\mathtt{GR}(2^k, d)$) can compute all values on the wires of the circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ on his own, ϕ, ψ only need to be invoked for one time (after all values of the circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ are computed). Thus, for malicious security, the prover needs to additionally prove to the verifier that he follows RMFE encoding honestly. Our main innovation of this paper is a novel mechanism that solves the above issue efficiently.

Our re-embedding technique. Let $\tau = \phi \circ \psi$ and $[x]$ 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 prove 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 $d := x - \mu$ to the verifier ($[x] := [\mu] + d$ by additive homomorphism). According to RMFE properties presented in Section 2, we observe that

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

and

$$x = \tau(x) \iff \mu + d = \tau(\mu) + \tau(d) \iff d - \tau(d) = \tau(\mu) - \mu.$$

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 above observations, we let $\tau(\mu) - \mu$ be revealed to the verifier (in the offline phase), and in the online phase, the verifier can check $d - \tau(d) = \tau(\mu) - \mu$ upon receiving d. If the check passes, $[\tau(x)]$ can be computed by $[\tau(\mu)] + \tau(d) = [\mu] + d$. In some sense, $[x]$ is "re-embedded" into $[\tau(x)]$. Thus, we name it by *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 sacrifice. We remark 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 sublinear ZK protocol $\Pi_{\mathrm{slZK}}^{m,n,t}$, we compile with an online phase adapted from AntMan [54]. 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 the *information-theoretic polynomial authentication code* (IT-PAC) amortisation technique proposed in AntMan that operates on a batch of m_2 elements in $\mathtt{GR}(2^k, d)$. We remark that the use of RMFEs in fact incurs a constant ($d/m_1 > 1$) communication overhead, while the sublinear communication complexity is due to IT-PACs. 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. In 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). We note here that IT-PACs are incompatible with the

Moz$\mathbb{Z}_{2^k}$arella [4] approach, since the maximum size of exceptional sets of $\mathbb{Z}_{2^{k+s}}$ is only 2, on which Moz$\mathbb{Z}_{2^k}$arella is working. Tricks of reducing computing a generic circuit to computing an SIMD circuit are proposed in [54] that we postpone to the end of Section 3. We quickly point out some possible disadvantages (without dwelling on them) of the IT-PAC. 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. To speed-up the offline phase, we present three constructions for VOLE over $\mathtt{GR}(2^k, d)$. Comparing to the primal-LPN based construction for VOLE over $\mathbb{Z}_{2^k}$ [4] (i.e. the offline phase of Moz$\mathbb{Z}_{2^k}$arella), our constructions have advantages in the following aspects.

(1) The algebraic structure of 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. We include generalizations of the three most representative constructions of VOLE, a SoftSpokenOT-style one, a primal-LPN based one, and a dual-LPN based one. We remark that there might exist a SoftSpokenOT-style construction for VOLE over $\mathbb{Z}_{2^k}$, while it is unclear how to construct a dual-LPN based variant with similar performance.

(2) The underlying LPN assumptions over $\mathtt{GR}(2^k, d)$ (of the latter two constructions) are more secure than that of Moz$\mathbb{Z}_{2^k}$arella. We show a reduction relating to LPN over the field $\mathbb{F}_{2^d}$, by generalizing the approach of [46]. A similar reduction severely affects the security of an earlier version of Moz$\mathbb{Z}_{2^k}$arella as reported in [46] and, to evade this attack, each non-zero entry of the LPN error vector should be invertible in $\mathbb{Z}_{2^k}$. Moreover, 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 $1/2^c$.

2 Preliminaries

Notations. In this paper, bold letters (e.g. $\mathbf{a}, \mathbf{b}$) are used to denote vectors. Besides, we use $\mathbf{x}_i$ 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 $\{\mathbf{x}_i \mid i \in [a, b]\}$. 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 [52] for a friendly exposition.

We emphasize that Galois rings have a special algebraic structure that, 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 ([31]). *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 $\Pr\Big[f(\alpha)=0 \;\Big|\; \alpha \xleftarrow{\$} \mathtt{GR}(p^k,d)\Big] \le rp^{-d}$.

Reverse Multiplicative Friendly Embedding. Reverse Multiplicative Friendly Embedding (RMFE) was first introduced by Cascudo et al. [20], which allows packing multiple multiplications over a field $\mathbb{F}_q$ to one multiplication over an extension field $\mathbb{F}_{q^d}$. It was further showed by Cramer et al. [26] that RMFEs over finite fields can be lifted to Galois rings. We first recall the definition of RMFE, and then present some of its important properties.

Definition 1 (RMFE [26]). *Let p be a prime, $k,r,m,d \ge 1$ 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.

From Definition 1, 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 $\frac{m}{d}$.

Lemma 2 (Existence of RMFE [20,26]). *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 $\phi(\mathbf{1})$ 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 makes 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 $\psi'(b) := \psi(b\cdot\phi(1))$. 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 the "re-embed" map $\tau = \phi \circ \psi : \mathtt{GR}(p^k, r) \to \mathtt{GR}(p^k, r)$. Looking ahead, Lemma 3 is critical for our ZK protocols.

In this manuscript, we mainly consider RMFEs with $r = 1$ and $p = 2$, i.e., a family of (m, d)-RMFEs over $\mathbb{Z}_{2^k}$. As showed in [20], 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 Fig. 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 $[\mathbf{x}_1], \ldots, [\mathbf{x}_l]$ and public coefficients $c, c_1, \ldots, c_l \in \mathtt{GR}(p^k, d)$, the two parties can locally compute $[y] = c + \sum_{i\in[l]} c_i \cdot [\mathbf{x}_i]$ by setting $y = c + \sum_{i\in[l]} c_i \cdot \mathbf{x}_i$, $\mathbf{M}_y = \sum_{i\in[l]} \mathbf{M}_{\mathbf{x}_i}$, and $\mathbf{K}_y = \Delta \cdot c + \sum_{i\in[l]} c_i \cdot \mathbf{K}_{\mathbf{x}_i}$. 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 $P_\mathcal{R}$.

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

Init: Upon receiving $(\mathtt{Init})$ from both parties, sample $\Delta \xleftarrow{\$} \mathtt{GR}(2^k, d)$ if $P_\mathcal{R}$ 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 $(\mathtt{Init})$ commands will be ignored.

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

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

Security Model and Functionalities. We prove the security of our protocols in the universal composability (UC) framework [19]. In particular, we consider active adversary and static corruption. We refer to Appendix A for a friendly exposition. 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 satisfying the same circuit $\mathcal{C}$. Details of $\mathcal{F}_{\mathrm{ZK}}^m$ are in Fig. 2. In particular, we say an interactive ZK protocol is *public coin*, if each message (from the verifier) sent to the prover is a random string.

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

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

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

We also require some fundamental functionalities for our VOLE constructions, e.g., the equality test functionality ($\mathcal{F}_{\mathrm{EQ}}$, Figure 12), which allows two parties $\mathcal{P}$ and $\mathcal{V}$ to check their inputs are equivalent with $\mathcal{P}$'s input revealed to $\mathcal{V}$, the oblivious transfer functionality ($\mathcal{F}_{\mathrm{OT}}$, Figure 13), which receives a bit from $P_{\mathcal{S}}$ and two strings from $P_{\mathcal{R}}$, and then sends one of the strings to $P_{\mathcal{S}}$ according to the choice bit [5].

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

Inspired by the methodology of [31], where the authors constructed efficient dishonest majority MPC over $\mathbb{Z}_{2^k}$ by first giving an MPC over $\mathtt{GR}(2^k, d)$ and then converting it to work over $\mathbb{Z}_{2^k}$, we introduce RMFEs into 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 $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ and present a construction that UC-realizes it in the $(\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{EQ}})$-hybrid model. Next we show how to construct ZK protocols over $\mathbb{Z}_{2^k}$ basing on our $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ functionality. In Section 3.2, we present a public coin ZK protocol over $\mathbb{Z}_{2^k}$ and in Section 3.3, we construct a ZK protocol over $\mathbb{Z}_{2^k}$ with communication complexity sublinear in the circuit size.

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

Jumping ahead, to construct ZK protocols over $\mathbb{Z}_{2^k}$, our first step is to construct ZK protocols over $\mathtt{GR}(2^k, d)$. Following the blueprint of ZK protocols over the Galois fields, e.g., [53,30,56], this step is quite straightforward. The key observation is that the soundness error of these protocols is related to the fraction of units of the underlying ring. For example, the ZK protocol over $\mathbb{F}_q$ in QuickSilver [56] has soundness error $\mathcal{O}(1/q)$. By Lemma 1, realizing a Galois ring $\mathtt{GR}(2^k, d)$ analogue ZK of QuickSilver induces soundness error $\mathcal{O}\big(1/2^d\big)$, which can be set negligible in the security parameter κ by choosing a sufficiently large parameter d.

The main obstacle lies in the second step, where we need to do the conversion. A naive conversion is to run the above ZK protocol over $\mathtt{GR}(2^k, d)$ by treating each element in $\mathbb{Z}_{2^k}$ as an element in $\mathtt{GR}(2^k, d)$. However, this already incurs

[5] We remark that roles are reversed, compared to the standard OT definition.

$\mathcal{O}(\kappa)$ overhead for a negligible soundness error, needless to say that, the prover is additionally required to prove that his inputs (i.e. the witness) are over $\mathbb{Z}_{2^k}$. In fact, the above naive conversion uses the naive embedding $\mathbb{Z}_{2^k} \hookrightarrow \mathtt{GR}(2^k, d)$, which can be viewed as an inefficient RMFE over $\mathbb{Z}_{2^k}$ in some sense. Therefore, we instead use efficient (m, d)-RMFEs over $\mathbb{Z}_{2^k}$ to accomplish the conversion, where the ratio d/m is asymptotically constant.

Let $\phi : \mathbb{Z}_{2^k}^m \to \mathtt{GR}(2^k, d)$ and $\psi : \mathtt{GR}(2^k, d) \to \mathbb{Z}_{2^k}^m$ be an (m, d)-RMFE pair over $\mathbb{Z}_{2^k}$. There are two issues that we have to overcome, when we use RMFEs. The first thing is that the prover is additionally required to prove that his inputs (i.e. the witness) are over $\mathrm{Im}(\phi)$ (as opposed to $\mathbb{Z}_{2^k}$ in the naive embedding case). The second thing is to guarantee the honest circuit evaluation, since (ϕ, ψ) only preserves one time multiplication by definition, unlike the naive embedding which has infinite multiplication capacity.

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

$\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ 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. Let (ϕ, ψ) be an (m, d)-RMFE pair over $\mathbb{Z}_{2^k}$, and $\tau := \phi \circ \psi$.

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

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

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

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 $\mathtt{GR}(2^k, d)$ is the direct sum of $\mathrm{Im}(\phi)$ and $\mathrm{Ker}(\psi)$ (Lemma 3), and the inputs over $\mathbb{Z}_{2^k}$ one-to-one correspond to a vector over $\mathrm{Im}(\phi)$ (we use ϕ to map m witnesses to a vector over $\mathtt{GR}(2^k, d)$). Thus, we can allow the projection of $\mathbf{x}$ on $\mathrm{Ker}(\psi)$ (i.e., $\mathbf{x} - \tau(\mathbf{x})$, where $\tau := \phi \circ \psi : \mathtt{GR}(2^k, d) \to \mathtt{GR}(2^k, d)$ is a $\mathbb{Z}_{2^k}$-linear map as well) to be revealed to the verifier $\mathcal{V}$. Recall that VOLE can be viewed as linearly homomorphic MACs, this enables the two parties to obtain the re-embedding pair MACs of $\mathbf{x}$, namely, $([\mathbf{x}], [\tau(\mathbf{x})])$.

We define the re-embedding VOLE functionality $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 3, which essentially allows the two parties to obtain random $([\boldsymbol{\mu}], [\tau(\boldsymbol{\mu})])$ through revealing

$\tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ to the receiver. Now we show that building upon $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$, the two parties can obtain $[\mathbf{x}]$, where $\mathbf{x}$ is destined to be over $\text{Im}(\phi)$. Given $([\boldsymbol{\mu}], [\tau(\boldsymbol{\mu})])$ (by $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$), where $\boldsymbol{\mu} \xleftarrow{\$} \texttt{GR}(2^k,d)^n$, the prover sends $\boldsymbol{\delta} = \mathbf{x} - \boldsymbol{\mu}$ to the verifier, the verifier can check

$$\boldsymbol{\delta} - \tau(\boldsymbol{\delta}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}, \tag{2}$$

which is equivalent to check

$$(\mathbf{x} - \boldsymbol{\mu}) - \tau(\mathbf{x} - \boldsymbol{\mu}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu} \iff \mathbf{x} - \tau(\mathbf{x}) = 0 \iff \mathbf{x} \in \text{Im}(\phi)^n.$$

Therefore, if the above equation (2) holds, $[\mathbf{x}]$ computed by $[\boldsymbol{\mu}] + \boldsymbol{\delta}$ are MACs for elements in $\text{Im}(\phi)$, as desired.

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

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

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

1. **Construct:**
   (a) Both parties send $(\texttt{Extend}, n+s)$ to $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $\mathbf{M}, \mathbf{x} \in \texttt{GR}(2^k,d)^{n+s}$, and $P_{\mathcal{R}}$ receives $\mathbf{K} \in \texttt{GR}(2^k,d)^{n+s}$, such that $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$ holds. Thus, the parties now obtain $[\mathbf{x}_i], i \in [n+s]$.
   (b) $P_{\mathcal{S}}$ computes $\boldsymbol{\eta} := \tau(\mathbf{x}) - \mathbf{x}$, then sends $\boldsymbol{\eta} \in \texttt{Ker}(\psi)^{n+s}$ to $P_{\mathcal{R}}$. If $\boldsymbol{\eta} \notin \texttt{Ker}(\psi)^{n+s}$, $P_{\mathcal{R}}$ aborts.
   (c) $P_{\mathcal{S}}$ sets $\mathbf{M}' := \mathbf{M}$, and $P_{\mathcal{R}}$ sets $\mathbf{K}' := \mathbf{K} + \Delta \cdot \boldsymbol{\eta}$. Note that $\mathbf{K}' = \mathbf{M}' + \tau(\mathbf{x}) \cdot \Delta$ holds, so the parties now obtain $[\tau(\mathbf{x}_i)], i \in [n+s]$.
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 $\mathbf{a}_i = \mathbf{x}_{n+i} + \sum_{j \in [n]} \chi_j^{(i)} \cdot \mathbf{x}_j$ and $\mathbf{b}_i = \tau(\mathbf{x}_{n+i}) + \sum_{j \in [n]} \chi_j^{(i)} \cdot \tau(\mathbf{x}_j)$. $P_{\mathcal{S}}$ sends $\mathbf{a}, \mathbf{b}$ to $P_{\mathcal{R}}$. $P_{\mathcal{S}}$ computes $\hat{\mathbf{M}}_i := \mathbf{M}_{n+i} + \sum_{j \in [n]} \chi_j^{(i)} \cdot \mathbf{M}_j$, for $i \in [s]$.
   (c) $P_{\mathcal{R}}$ checks $\mathbf{b}_i - \mathbf{a}_i = \eta_{n+i} + \sum_{j \in [n]} \chi_j^{(i)} \cdot \eta_j$, for $i \in [s]$ and $\mathbf{b} \in \texttt{Im}(\phi)^s$. If the check fails, $P_{\mathcal{R}}$ aborts. $P_{\mathcal{R}}$ computes $\hat{\mathbf{M}}_i' := \mathbf{K}_{n+i} + \sum_{j \in [n]} \chi_j^{(i)} \cdot \mathbf{K}_j - \mathbf{a}_i \cdot \Delta$, for $i \in [s]$.
   (d) $P_{\mathcal{S}}$ sends $\hat{\mathbf{M}}$ to $P_{\mathcal{R}}$. $P_{\mathcal{R}}$ checks whether $\hat{\mathbf{M}} = \hat{\mathbf{M}}'$, and aborts if the check fails.
3. **Output:** Output $([\mathbf{x}_i], [\tau(\mathbf{x}_i)])$ for $i \in [n]$.

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

To construct a secure re-embedding VOLE protocol, We let the two parties generate $n+s$ re-embedding pair MACs and sacrifice the extra s re-embedding pair MACs. The sacrifice is done by taking s random $\mathbb{Z}_{2^k}$-linear combinations of n remaining re-embedding pair MACs to obtain s equations, with each masked with an extra re-embedding pair MAC. If there is at least one of the remaining n re-embedding pair MACs that is not honestly generated, the correctness check will fail, except with probability at most $2^{-s} + 2^{-d}$, which can be negligible in the security parameter κ 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, and we have the following theorem with proof deferred to Appendix C.1. To obtain n re-embedding pair MACs, our construction consumes a VOLE correlation of length $(n + s)$, and communicates $(n + 3s)$ Galois ring elements and ns coefficients over $\mathbb{Z}_{2^k}$, or equivalently, on average $(1 + 3s/n + s/d)$ Galois ring elements. As n is sufficiently large for most ZK applications, the overhead for constructing re-embedding pairs is close to a small constant.

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, no* PPT *environment $\mathcal{Z}$ can distinguish the real world execution from the ideal world simulation except with advantage at most* $2^{-s} + 2^{-d}$.

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

Equipped with our re-embedding VOLE technique and inspired by QuickSilver [56], we construct a highly efficient public coin ZK protocol over $\mathbb{Z}_{2^k}$.

Suppose the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ have agreed on an SIMD circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ with n inputs and t multiplication gates, and $\mathcal{P}$ has m witnesses. 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 $[\pi]$, where $\pi \xleftarrow{\$} \mathtt{GR}(2^k, d)$. $\mathcal{P}$ then computes $\boldsymbol{\omega} := \phi(\mathbf{w}^{(1)}, ..., \mathbf{w}^{(m)})$, and sends $\boldsymbol{\delta} := \boldsymbol{\omega} - \boldsymbol{\mu}$ to $\mathcal{V}$. They can obtain $[\tau(\boldsymbol{\omega})] := [\tau(\boldsymbol{\mu})] + \tau(\boldsymbol{\delta})$. Recall that if $\boldsymbol{\omega} \in \text{Im}(\phi)^n$, we have that $\tau(\boldsymbol{\omega}) = \boldsymbol{\omega}$ and $\boldsymbol{\delta} - \tau(\boldsymbol{\delta}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ should hold. Next, the two parties are going to 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. One good news is that in the ZK setting, the prover $\mathcal{P}$ can compute all values in the circuit on his own (as he holds the witness of the circuit). Therefore, the output MAC of a `Mul` gate can be obtained by consuming one random MAC. The bad thing is that all authenticated values in the circuit should be in $\text{Im}(\phi)$, but the multiplication equality no longer holds! More specifically, for the i-th multiplication gate in $\mathcal{C}$ with inputs $\omega_\alpha, \omega_\beta \in \text{Im}(\phi)$ and output $\omega_\gamma \in \text{Im}(\phi)$, we have $\psi(\omega_\alpha) * \psi(\omega_\beta) = \psi(\omega_\gamma)$, which is not equivalent to $\omega_\alpha \cdot \omega_\beta = \omega_\gamma$. The former equation is not easy to verify since values are evaluated over $\mathtt{GR}(2^k, d)$ rather than $\mathbb{Z}_{2^k}$. We show that this issue can be bypassed by our re-embedding VOLE. Given $([\nu_i], [\tau(\nu_i)])$ (by $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$), $\mathcal{P}$ sends d_i to $\mathcal{V}$, which is supposed to equal $\omega_\alpha \cdot \omega_\beta - \nu_i$ if $\mathcal{P}$ is honest, 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

$$\begin{aligned} B_i :&= \mathbf{K}_{\omega_\alpha} \cdot \mathbf{K}_{\omega_\beta} - \mathbf{K}_{\hat{\omega}_\gamma} \cdot \Delta \\ &= (\mathbf{M}_{\omega_\alpha} + \Delta \cdot \omega_\alpha) \cdot (\mathbf{M}_{\omega_\beta} + \Delta \cdot \omega_\alpha) - (\mathbf{M}_{\hat{\omega}_\gamma} + (\omega_\alpha \cdot \omega_\beta) \cdot \Delta) \cdot \Delta \\ &= (\mathbf{M}_{\omega_\alpha} \cdot \mathbf{M}_{\omega_\beta}) + (\omega_\alpha \cdot \mathbf{M}_{\omega_\beta} + \omega_\beta \cdot \mathbf{M}_{\omega_\alpha} - \mathbf{M}_{\hat{\omega}_\gamma}) \cdot \Delta \end{aligned}$$

holds if d_i is correct. Therefore, it can be used to detect malicious behaviors by letting $\mathcal{P}$ send $A_{0,i} := \mathbf{M}_{\omega_\alpha} \cdot \mathbf{M}_{\omega_\beta}, A_{1,i} := \omega_\alpha \cdot \mathbf{M}_{\omega_\beta} + \omega_\beta \cdot \mathbf{M}_{\omega_\alpha} - \mathbf{M}_{\hat{\omega}_\gamma}$ to $\mathcal{V}$. In 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 equations simultaneously. Briefly, $\mathcal{V}$ sends uniformly random coefficients $\{\chi_i \in \mathtt{GR}(2^k, d)\}_{i\in[t]}$ to $\mathcal{P}$, and $\mathcal{P}$ returns to $\mathcal{V}$ the linear combination of $\{A_{0,i}, A_{1,i}\}_{i\in[t]}$ masked with $\mathbf{M}_\pi, \pi$, respectively. In fact, as $\mathtt{GR}(2^k, d)$ contains a subfield $\mathbb{F}_{2^d}$, $\boldsymbol{\chi}$ 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 $\mathbf{K}_{\omega_h} = \mathbf{M}_{\omega_h} + \phi(\mathbf{1}) \cdot \Delta$ should hold. Thus, we let $\mathcal{P}$ open $[\omega_h]$ by sending $\mathbf{M}_{\omega_h}$ to $\mathcal{V}$. We have the following theorem, with security proof deferred to Appendix C.2.

Theorem 2. *Protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ communicates $(kd+d)/m$ bits per multiplication gate, and 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.*

Our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ transfers one $\mathtt{GR}(2^k, d)$ element and one random coefficient over $\mathbb{F}_{2^d}$ 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 $\Pi_{\mathrm{ZK}}^{m,n,t}$, $\Pi_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, 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 VOLE-hybrid model, which allows us to construct a publicly verifiable NIZK over $\mathbb{Z}_{2^k}$ in Section 4.1.

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

Achieving *succinctness* is always a fascinating challenge of constructing ZK protocols. For proving circuit satisfiability over finite fields, there are plenty of practical candidates (e.g. zk-SNARKs [9,24,51]). However, it is extremely hard to adapt such ZK protocols to work efficiently over integer rings in general. In this section, we focus on *succinct* proof size (i.e., sublinear communication complexity). Note that our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ has linear communication complexity. We are initially inspired by AntMan [54], which constructed interactive ZK for any field with sublinear communication complexity via developing a new, powerful technique, the Polynomial Authentication Code (PAC).

We sketch how we construct sublinear ZK over $\mathbb{Z}_{2^k}$. Our starting point is the sublinear ZK protocol of AntMan [54]. Similar to Section 3.2, we follow the

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 $\mathcal{F}_{\mathrm{embVOLE}}^{\texttt{GR}(2^k,d)}$, and $\mathcal{V}$ receives $\Delta \in \texttt{GR}(2^k, d)$.
2. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend-pair}, n+t)$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\texttt{GR}(2^k,d)}$, 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 $\texttt{GR}(2^k, d)$. Note that $\mathcal{V}$ also learns $\tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ and $\tau(\boldsymbol{\nu}) - \boldsymbol{\nu}$ from $\mathcal{F}_{\mathrm{embVOLE}}^{\texttt{GR}(2^k,d)}$.
3. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend}, 1)$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\texttt{GR}(2^k,d)}$, which returns authenticated value $[\pi]$, where π is sampled uniformly at random in $\texttt{GR}(2^k, d)$.

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 $\boldsymbol{\omega} := \phi(W)$, and sends $\delta_i := \omega_i - \mu_i$, $i \in [n]$ to $\mathcal{V}$. $\mathcal{V}$ checks whether $\boldsymbol{\delta} - \tau(\boldsymbol{\delta}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ holds. If the check fails, aborts. Both parties can locally compute $[\tau(\omega_i)] := [\tau(\mu_i)] + \tau(\delta_i)$.
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 $\mathbf{K}_{\omega_j} = \mathbf{M}_{\omega_j} + \omega_j \cdot \Delta$ for $j \in \{\alpha, \beta\}$, and $\mathbf{K}_{\hat{\omega}_\gamma} = \mathbf{M}_{\hat{\omega}_\gamma} + \hat{\omega}_\gamma \cdot \Delta$.
   – $\mathcal{P}$ computes $A_{0,i} := \mathbf{M}_{\omega_\alpha} \cdot \mathbf{M}_{\omega_\beta} \in \texttt{GR}(2^k, d)$ and $A_{1,i} := \omega_\alpha \cdot \mathbf{M}_{\omega_\beta} + \omega_\beta \cdot \mathbf{M}_{\omega_\alpha} - \mathbf{M}_{\hat{\omega}_\gamma} \in \texttt{GR}(2^k, d)$.
   – $\mathcal{V}$ computes $B_i := \mathbf{K}_{\omega_\alpha} \cdot \mathbf{K}_{\omega_\beta} - \mathbf{K}_{\hat{\omega}_\gamma} \cdot \Delta \in \texttt{GR}(2^k, d)$.
4. $\mathcal{P}$ and $\mathcal{V}$ do the following check.
   (a) $\mathcal{P}$ sets $A_0^* := \mathbf{M}_\pi$, $A_1^* := \pi$, and $\mathcal{V}$ sets $B^* := \mathbf{K}_\pi$ 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 $X := \sum_{i\in[t]} \chi_i \cdot A_{0,i} + A_0^* \in \texttt{GR}(2^k, d)$ and $Y := \sum_{i\in[t]} \chi_i \cdot A_{1,i} + A_1^* \in \texttt{GR}(2^k, d)$, and sends (X, Y) to $\mathcal{V}$.
   (d) $\mathcal{V}$ computes $Z := \sum_{i\in[t]} \chi_i \cdot B_i + B^* \in \texttt{GR}(2^k, d)$, 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 $\mathbf{M}_{\omega_h}$ to $\mathcal{V}$.
   – $\mathcal{V}$ checks whether $\mathbf{K}_{\omega_h} = \mathbf{M}_{\omega_h} + \phi(\mathbf{1}) \cdot \Delta$. 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 $\mathcal{F}_{\mathrm{embVOLE}}^{\texttt{GR}(2^k,d)}$-hybrid model.

methodology that first constructing a sublinear ZK protocol over $\mathtt{GR}(2^k, d)$, and then converting it to a ZK protocol 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.

The 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 $f(\Lambda)$. On the other hand, a PAC also authenticates a batch of values, simultaneously. 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 appropriately (i.e., in some exceptional set of $\mathtt{GR}(2^k, d)$). Intuitively, PAC significantly reduces the communication complexity in the sense that turning m_2 MACs into one MAC $[f(\Lambda)]$.

In a high level, our protocol starts with the prover $\mathcal{P}$ (as PAC sender) and the verifier $\mathcal{V}$ (as PAC receiver) obtaining PACs for the SIMD witnesses, then they evaluate the circuit using these PACs, and finally $\mathcal{P}$ opens the output PAC to $\mathcal{V}$. Thus, to guarantee malicious security, $\mathcal{V}$ should be convinced that the PACs for inputs are corresponding to the witnesses, and the circuit evaluation is done correctly. We next briefly summarize how the protocol works, with special emphasis on how the RMFE and PAC amortisation techniques collaborate.

Suppose that $\mathcal{P}$ and $\mathcal{V}$ have agreed on a SIMD 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 $T = \{0, 1, \zeta, ..., \zeta^{2^d-2}\}$, where $\zeta \in \mathtt{GR}(2^k, d)$ is of order $2^d - 1$. Now $\mathcal{P}$ uses ϕ to map $m_1 m_2$ witnesses over $\mathbb{Z}_{2^k}$ to m_2 vectors over $\mathtt{GR}(2^k, d)$, and the two parties first authenticate these values via VOLE based MACs. Similar to that in our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$, we use re-embedding pair MACs to guarantee that the authenticated values are in the image of ϕ. Next, the two parties generate the corresponding PACs. We follow the interactive approach of AntMan [54] that generates a batch of PACs over Galois fields, using an additively homomorphic encryption (AHE) scheme. For the Galois ring case, we can instead use the AHE scheme of MHz2k [23]. The sub-protocol Π_{PAC} for generating PAC is presented in Figure 19 due to limited space. Then, the two parties can locally evaluate `Add` gates in the circuit. For the j-th `Mul` gate with input polynomials $u_a(\cdot), u_b(\cdot)$, $\mathcal{P}$ and $\mathcal{V}$ generates two PACs $[\![\hat{v}_j(\cdot)]\!]$ and $[\![v_j(\cdot)]\!]$, where $\hat{v}_j := u_a(\cdot) \cdot u_b(\cdot)$ with degree $2m_2 - 2$ and $v_j(\cdot)$ with degree $m_2 - 1$ is "τ-consistent" to $\hat{v}_j(\cdot)$, i.e. $v_j(\alpha_i) = \tau(\hat{v}_j(\alpha_i))$, for some fixed points $\alpha_i \in T$, $i \in [m_2]$. After the circuit evaluation is completed, $\mathcal{P}$ and $\mathcal{V}$ do a series of checks.

(1) The first check is for the τ-consistency. We remark that the τ-consistency of our protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$ is guaranteed *for free* due to the use of re-embedding VOLE. However, now the τ-consistency is defined between two polynomials (rather than two elements). Therefore, we need to find another method. Fortunately, it can

be observed that the `BatchCheck` procedure of AntMan [54], which is originally used to check $v_j(\alpha_i) = \hat{v}_j(\alpha_i)$ in a batch, can be directly adapted to check τ-consistency. For a better illustration, we briefly explain how our `BatchCheck` procedure works.

Recall that $\tau = \phi \circ \psi$ is a $\mathbb{Z}_{2^k}$-linear map. Thus, $\mathcal{V}$ can check 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, $\mathcal{V}$ should be convinced that the opened polynomials are consistent to their PACs. It can be observed that the polynomial key Λ can be revealed to $\mathcal{P}$, after all polynomials needed to be authenticated are authenticated. Then the PACs are turned into MACs for the polynomial evaluations on Λ naturally, and $\mathcal{V}$ can check this by $\mathcal{P}$ opening the MACs.

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. Let $H : \{0,1\}^\kappa \to \mathbb{Z}_{2^k}^l$ be a random oracle.

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

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

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

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

(2) The next check is for multiplication. Since PACs collapse to MACs automatically after the polynomial key is revealed to $\mathcal{P}$, we can use the multiplication check procedure of $\Pi_{\mathrm{ZK}}^{m,n,t}$, where $\mathcal{P}$ can only pass the check with probability at most $3/2^d$, if using some incorrect polynomials $\hat{v}_j(\cdot), j \in [t]$.

(3) The final check is for inputs and outputs. After Λ is opened to $\mathcal{P}$, the two parties can obtain MACs $[u_j(\Lambda)]$ from the PACs $[\![u_j(\cdot)]\!]$, $j \in [n]$. Note that they have already obtained MACs for inputs (which are restricted in the $\mathrm{Im}(\phi)$ by re-embedding VOLE). Let $\xi_i(\cdot), i \in [m_2]$ be the Lagrange polynomials defined by the evaluation point set $\{\alpha_1, ..., \alpha_{m_2}\} \subset T$. As $u_j(\cdot)$, $j \in [n]$ can be computed by Lagrange interpolation, the formula still holds in a MAC representation by taking $\xi_i(\Lambda)$ as the coefficients, which enables $\mathcal{V}$ to check input PACs $[\![u_j(\cdot)]\!]$ are consistent with corresponding input MACs. Checking outputs can be done by opening the PAC associated with the output wire, which should collapse to [1] with an honest execution of the protocol.

The protocol is given in Figure 7,8, which takes the `BatchCheck` procedure in Figure 6 and Π_{PAC} in Figure 19 as sub-protocols. We have the following theorem that guarantees security with a sketched proof in Appendix C.3.

Theorem 3. *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 $\frac{2m_2+3}{2^d} + \mathsf{negl}(\kappa)$.*

Our protocol $\Pi_{\mathrm{slZK}}^{m,n,t}$ communicates 2 AHE ciphertexts per multiplication gate. The main costs for checking multiplications consist of two parts: transferring random coefficients $\boldsymbol{\chi}$ in Step 5.e in Figure 8, and transferring 2 polynomials over $\mathtt{GR}(2^k, d)$ of respective degree $m_2 - 1, 2m_2 - 1$ in the *Linear combination phase* of the `BatchCheck` procedure. Assume the ciphertext size is c, these yield $2ct+td+3m_2kd$ bits in total, which is sublinear in the circuit size mt. By dividing mt, the amortized complexity per multiplication gate is $(\frac{2c+d}{m} + \frac{3kd}{m_1 t})$-bit. We remark that the overall communication complexity of $\Pi_{\mathrm{slZK}}^{m,n,t}$ is linear in the witness size, as the two parties need to obtain MACs for all witnesses at first. Besides, $\Pi_{\mathrm{slZK}}^{m,n,t}$ 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 a generic circuit into an SIMD circuit (cf. [27,21]) 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 [54] is specific to proving circuit satisfiability. Since the prover $\mathcal{P}$ holds the witness, he can calculate all values on the wires in the circuit and authenticate them in batch as evaluating some SIMD circuit. But in the generic circuit, the outputs of some gates may be the inputs of other gates, thus each value on wires now needs to be authenticated twice, one as the output of some gate, and the other as the input of some other gate. Therefore, $\mathcal{V}$ need to check not only the correctness of gate evaluations, but also the consistency of every

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 $\zeta \in \mathtt{GR}(2^k, d)$ 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 $\xi_i(X) = \prod_{j\in[m_2], j\neq i}(X - \alpha_j)/(\alpha_i - \alpha_j) \in \mathtt{GR}(2^k, d)[X]$ 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 $(\mathtt{Init})$ to $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$, and $\mathcal{V}$ receives $\Delta \in \mathtt{GR}(2^k, d)$.
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 $\Lambda \in \mathtt{GR}(2^k, d)$.
3. $\mathcal{P}$ and $\mathcal{V}$ send $(\mathtt{Extend\text{-}pair}, nm_2)$ to $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$, 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 $\mathtt{GR}(2^k, d)$. Note that $\mathcal{V}$ also learns $\tau(\boldsymbol{\mu}^{(i)}) - \boldsymbol{\mu}^{(i)}$ from $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$.
4. $\mathcal{P}$ and $\mathcal{V}$ send $(\mathtt{Extend}, n + 2t)$ to $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$, 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 $\mathtt{GR}(2^k, d)$.

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 $\boldsymbol{\omega}^{(i)} := \phi(W^{(i)}) \in \mathtt{GR}(2^k, d)^n$, and sends $\boldsymbol{\delta}^{(i)} := \boldsymbol{\omega}^{(i)} - \boldsymbol{\mu}^{(i)}$, $i \in [m_2]$ to $\mathcal{V}$. $\mathcal{V}$ checks whether $\boldsymbol{\delta}^{(i)} - \tau(\boldsymbol{\delta}^{(i)}) = \tau(\boldsymbol{\mu}^{(i)}) - \boldsymbol{\mu}^{(i)}$ holds for all $i \in [m_2]$. If the check fails, aborts. Both parties can 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 inputs gates with input vector $(\omega_j^{(1)}, ..., \omega_j^{(m_2)})$, $\mathcal{P}$ defines a degree $\leq (m_2 - 1)$ polynomial $u_j(\cdot)$ 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 multiplication gate, where $j \in [t]$, then $\mathcal{P}$ computes a degree $\leq (2m_2 - 2)$ polynomial $\widetilde{v}_j(\cdot) = \widetilde{u}_c(\cdot) := u_a(\cdot) \cdot u_b(\cdot) \in \mathtt{GR}(2^k, d)[X]$ and a degree $\leq (m_2 - 1)$ polynomial $v_j = u_c$ such that $v_j(\alpha_i) = u_c(\alpha_i) = \tau(\widetilde{u}_c(\alpha_i))$ 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 $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model-**Part I**.

Protocol $\Pi_{\text{slZK}}^{m,n,t}$-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 that $v_j(\alpha_i) = \tau(\widetilde{v}_j(\alpha_i))$ holds for all $i \in [m_2], j \in [t]$.
   (b) $\mathcal{P}$ and $\mathcal{V}$ run the `Gen` procedure of Π_{PAC} to open Λ to $\mathcal{P}$, and then $\mathcal{V}$ can compute the local keys on all PACs. For the j-th multiplication gate $(a, b, c, \texttt{Mul}) \in \mathcal{C}$, $\mathcal{P}$ and $\mathcal{V}$ now hold $[u_a(\Lambda)], [u_b(\Lambda)], [\widetilde{u}_c(\Lambda)]$. $\mathcal{P}$ computes $A_{0,j} := \mathbf{M}_{u_a(\Lambda)} \cdot \mathbf{M}_{u_b(\Lambda)} \in \texttt{GR}(2^k, d)$ and $A_{1,j} := u_a(\Lambda) \cdot \mathbf{M}_{u_b(\Lambda)} + u_a(\Lambda) \cdot \mathbf{M}_{u_a(\Lambda)} - \mathbf{M}_{\widetilde{u}_c(\Lambda)} \in \texttt{GR}(2^k, d)$. $\mathcal{V}$ computes $B_j := \mathbf{K}_{u_a(\Lambda)} \cdot \mathbf{K}_{u_b(\Lambda)} - \Delta \cdot \mathbf{K}_{\widetilde{u}_c(\Lambda)} \in \texttt{GR}(2^k, d)$.
   (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 $A_0^* := \mathbf{M}_\pi, A_1^* := \pi$, and $\mathcal{V}$ sets $B^* := \mathbf{K}_\pi$ 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 $X := \sum_{j\in[t]} \chi_j \cdot A_{0,j} + A_0^* \in \texttt{GR}(2^k, d)$ and $Y := \sum_{j\in[t]} \chi_j \cdot A_{1,j} + A_1^* \in \texttt{GR}(2^k, d)$, and sends (X, Y) to $\mathcal{V}$.
   (g) $\mathcal{V}$ computes $Z := \sum_{j\in[t]} \chi_j \cdot B_j + B^* \in \texttt{GR}(2^k, d)$, 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 output check. $\mathcal{P}$ 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 each $j \in [n]$, $\mathcal{P}$ and $\mathcal{V}$ locally compute an MAC $[\mathbf{z}_j] := \sum_{i\in[m_2]} \xi_i(\Lambda) \cdot [\omega_j^{(i)}] - [u_j(\Lambda)]$. Then, $\mathcal{P}$ opens $[\mathbf{z}_j]$, $\mathcal{V}$ continues if and only if $\mathbf{z}_j = 0$ holds for all $j \in [n]$. Otherwise, $\mathcal{V}$ aborts.
   (ii) Let $[u(\cdot)]$ be the PAC associated with the output wire of $\mathcal{C}$. $\mathcal{P}$ sends $\mathbf{M}_{u(\Lambda)}$ to $\mathcal{V}$.
   (iii) $\mathcal{V}$ checks whether $\mathbf{K}_{u(\Lambda)} = \mathbf{M}_{u(\Lambda)} + \phi(\mathbf{1}) \cdot \Delta$. 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 $\mathcal{F}_{\text{embVOLE}}^{\texttt{GR}(2^k,d)}$-hybrid model – **Part II**.

two authenticated values on a wire, which would increase the communication complexity considerably, depending on the topology of the given circuit.

We remark that the second method is compatible with our PAC-based construction, still yielding a sublinear communication online phase, where we can do the additional check by slightly modifying the `BatchCheck` procedure to allow checks on some specific $\mathbb{Z}_{2^k}$-components (rather than to check 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 (m_1 copies), viewed as a generic circuit over $\mathtt{GR}(2^k, d)$, and then apply the second method. Users are advised to pick the most efficient strategy according to the circuit topology.

4 VOLE over Galois Rings

In this section, we show efficient constructions for VOLE over Galois rings. The first construction adapts the idea of SoftSpokenOT [50], and allows us to construct publicly verifiable ZK protocols. Following the PCG paradigm, we construct concretely more efficient VOLE protocols with security under LPN over Galois rings. We present an interactive construction (better computation complexity) based on primal LPN in Section 4.2, and a two-round construction (small communication complexity) based on dual LPN in Appendix E. We discuss security of LPN over Galois rings in the end of this section.

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

SoftSpokenOT [50] showed 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 functionality (formulated as $\mathcal{F}^N_{\mathrm{OT}-\bar{1}}$ in Figure 14), takes as inputs N strings (of the same length) from the sender and one position $\varDelta \in [N]$ from the receiver, then delivers these strings to the receiver except for the $\varDelta$-th one. As showed in [50], $\mathcal{F}_{\mathrm{OT}-\bar{1}}$ can be efficiently instantiated with $\log N$ OTs. We adapt the SoftSpokenOT idea to construct VOLE protocols over Galois rings. W.l.o.g., assume $N = 2^{kd}$. Let $\mathbf{s}_1, ..., \mathbf{s}_N \in \mathtt{GR}(2^k, d)^l$ be random strings held by $P_\mathcal{S}$, and $\varDelta \in [N]$ be a random index selected by $P_\mathcal{R}$. The VOLE construction is induced by the following observation:

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

Through an invocation to $\mathcal{F}^N_{\mathrm{OT}-\bar{1}}$, $P_\mathcal{R}$ receives $\mathbf{s}_y$, for $y \in \mathtt{GR}(2^k, d)\setminus\{\varDelta\}$[6]. Then, $P_\mathcal{R}$ locally computes $\mathbf{K} := \sum_{y \in \mathtt{GR}(2^k,d)\setminus\{\varDelta\}} \mathbf{s}_y \cdot (\varDelta - y)$. By $P_\mathcal{S}$ locally computing $\mathbf{M} := -\sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y \cdot y$ and $\mathbf{x} := \sum_{y \in \mathtt{GR}(2^k,d)} \mathbf{s}_y$, we have that $\mathbf{K} = \mathbf{M} + \varDelta \cdot \mathbf{x}$ (i.e., $P_\mathcal{S}$ and $P_\mathcal{R}$ obtain a VOLE correlation over $\mathtt{GR}(2^k, d)$). The main drawback

[6] We naturally view $y \in \mathtt{GR}(2^k, d)$ as an integer in $[N]$, and $\varDelta \in [N]$ as an element in $\mathtt{GR}(2^k, d)$, since $N = 2^{kd}$.

for implementation is that the computational costs (both for $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$) are high, since the above construction involves N heavy multiplication operations over $\mathtt{GR}(2^k,d)$ per party. To reduce the computation complexity, we utilize the algebraic structure of Galois rings that $\mathtt{GR}(2^k,d)$ contains $\mathbb{F}_{2^d}$ as a "subfield", and restrict Δ to be sampled from $\mathbb{F}_{2^d}$. More specifically, let $N = 2^d$, and we have:

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

We present the resulting VOLE protocol in Figure 9, and we have the following theorem with proof deferred to Appendix D.1.

Theorem 4. *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.*

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

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

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

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

1. Upon receiving $\{\mathbf{s}_y\}_{y\in\mathbb{F}_{2^d}\setminus\{\Delta\}}$ from $\mathcal{F}_{\mathrm{OT}-\bar{1}}^{N}$, 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 $\mathcal{F}_{\mathrm{OT}-\bar{1}}^{N}$, $P_{\mathcal{S}}$ 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 $\mathbf{K},\mathbf{M},\mathbf{x} \in \mathtt{GR}(2^k,d)^l$.

Fig. 9: Protocol for VOLE over $\mathtt{GR}(2^k,d)$ in the $\mathcal{F}_{\mathrm{OT}-\bar{1}}^{N}$-hybrid model.

We remark that $\Pi_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$ in fact realizes a variant of $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$, where $P_{\mathcal{R}}$ obtains random Δ in $\mathbb{F}_{2^d}$ instead of $\mathtt{GR}(2^k,d)$. We argue that our ZK protocols remain secure, if the underlying VOLE functionality is instantiated with $\Pi_{\mathrm{sfVOLE}}^{\mathtt{GR}(2^k,d)}$. Intuitively, the entropy of Δ (distributed uniformly at random in $\mathbb{F}_{2^d}$ rather than $\mathtt{GR}(2^k,d)$) is still sufficient for guaranteeing a negligible soundness error, as we originally require d to be linear in the security parameter. We refer to a more detailed discussion in Appendix D.1. As shown in [50], 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 (independent of the length parameter l). In addition, similar to that in [50,5], it is plausible that the efficiency can be further improved by using linear codes over $\mathtt{GR}(2^k,d)$ (e.g. Reed-Solomon codes defined over $\mathtt{GR}(2^k,d)$ [26]), which is beyond the scope of this paper and omitted.

Baum et al. [5] proposed the VOLE-in-the-head technique, where they constructed an compiler that transforms any *public coin*, interactive ZK protocol based on $(N-1)$-out-of-N OT to a *publicly verifiable* NIZK protocol. As shown in Section 3.2, our protocol $\Pi_{\text{ZK}}^{m,n,t}$ admits a public coin ZK in the VOLE-hybrid model. Therefore, we can apply their compiler to obtain the first publicly verifiable NIZK protocol over $\mathbb{Z}_{2^k}$, following the road map:

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

4.2 VOLE based on primal LPN

We also construct efficient VOLE protocols based on variants of LPN, following the established PCG-style VOLE paradigm (similar to OT extension) [7]. Basically, the paradigm involves two main steps. The first step is to construct a single point variant of VOLE (spVOLE for short), and the second step is to use LPN to locally convert multiple spVOLE instances (of the same length) into one (much longer) VOLE correlation. To begin with, we focus on constructing spVOLE over $\texttt{GR}(2^k, d)$, 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 [8] (see the ideal functionality $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$ in Figure 17).
Single-point VOLE. We adapt the spVOLE construction in Wolverine [53] from Galois fields to Galois rings. We briefly review the main ingredient, the puncturable pseudorandom function [14] (PPRF) based on GGM tree [37]. Informally, a PPRF allows to distribute n pseudorandom values among the two parties, where the receiver obtains all of them from a key, while the sender can compute them all except for one value on his choice from a punctured key. The GGM tree algorithms (presented in Figure 18) admit a semi-honestly secure protocol in the OT-hybrid model that achieves the above procedure.

In more detail, suppose the two parties $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ have already obtained random Galois ring elements $\{\mathbf{v}_j\}_{j\in[0,n)\setminus\{\alpha\}}$ and $\{\mathbf{v}_j\}_{j\in[0,n)}$ from PPRF, respectively, where $\alpha \in [0, n)$ is randomly picked by $P_{\mathcal{S}}$. Then $P_{\mathcal{S}}$ can define two vectors $\mathbf{w}, \mathbf{u} \in \texttt{GR}(2^k, d)^n$ such that $\mathbf{w}_j = \mathbf{v}_j, \mathbf{u}_j = 0$, for $j \neq \alpha$ (we do not care about $\mathbf{w}_\alpha, \mathbf{u}_\alpha$ for now). It can be directly observed that $\mathbf{v} = \mathbf{w} + \mathbf{u} \cdot \Delta$ holds for any Δ, if ignoring the entry indexed by α. To obtain the desired single-point VOLE correlation, we can consume a length one VOLE correlation $\gamma = \delta + \beta \cdot \Delta$ (intuitively, "inject" the correlation to the target entry). By $P_{\mathcal{R}}$ sending $g := \gamma - \sum_{j\in[0,n)} \mathbf{v}_j$ to $P_{\mathcal{S}}$, $P_{\mathcal{S}}$ can compute $\mathbf{w}_\alpha := \delta - (g + \sum_{i\neq\alpha} \mathbf{w}_i)$. And we can verify that $\mathbf{v}_\alpha = \mathbf{w}_\alpha + \beta \cdot \Delta$ holds. Thus, setting $\mathbf{u}_\alpha := \beta$, the two parties obtain the desired single-point VOLE 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 [53], to achieve malicious security, we only need to

[7] Note that instantiating the VOLE functionality of $\Pi_{\text{ZK}}^{m,n,t}$ with PCG-style constructions leads to a *designated verifier* NIZK, since the verifier is no longer *public coin.*

[8] We remark that we further require the non-zero entry to be a unit of the Galois ring, due to a modulo attack for LPN over Galois rings (Theorem 6).

guarantee the spVOLE correlation holds in each entry [9]. To this end, we can apply a random-linear combination check. Let $\chi_i \in \mathtt{GR}(2^k, d)$, $i \in [0, n)$ be uniformly random elements picked by $P_{\mathcal{S}}$. We have that

$$\sum_{i \in [0,n)} \chi_i \cdot \mathbf{v}_i = \sum_{i \in [0,n)} \chi_i \cdot \mathbf{w}_i + \chi_\alpha \cdot \beta \cdot \varDelta. \tag{3}$$

However, we cannot directly use Equation 3 for consistency check, since $\chi_\alpha \cdot \beta \cdot \varDelta$ can not be locally computed by any party and the linear combination would leak some information. These can be solved by masking with a VOLE correlation where $y = z + \chi_\alpha \cdot \beta \cdot \varDelta$. Namely, the two parties check the following equation

$$\sum_{i \in [0,n)} \chi_i \cdot \mathbf{v}_i - y = \sum_{i \in [0,n)} \chi_i \cdot \mathbf{w}_i - z. \tag{4}$$

We give the single-point VOLE protocol in Figure 10, and we have the following theorem (with proof deferred to Section D.2).

Theorem 5. *If G and G' are* PRG*s, then $\varPi_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$ functionality in the $(\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\text{OT}}, \mathcal{F}_{\text{EQ}})$-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 $1/2^d + \mathsf{negl}(\kappa)$.*

From spVOLE to VOLE. The extension procedure relies on an observation from LPN. Assume $A \in \mathtt{GR}(2^k, d)^{m \times n}$ is a generating matrix and $\mathcal{D}$ is a noise distribution over $\mathtt{GR}(2^k, d)^n$, for which the primal LPN assumption holds (we refer to Section B for LPN preliminaries). Suppose $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ have obtained a VOLE correlation of length $(m + n)$, written as $\mathbf{v} = \mathbf{w} + \mathbf{u} \cdot \varDelta$ and $\mathbf{b} = \mathbf{c} + \mathbf{e} \cdot \varDelta$, where $\mathbf{u} \in \mathtt{GR}(2^k, d)^m$ and $\mathbf{e} \leftarrow \mathcal{D}$. We have that

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

where $(\mathbf{M}, \mathbf{x}), \mathbf{K}$ 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 over $\mathtt{GR}(2^k, d)$. In the literature, $\mathcal{D}$ is usually instantiated with the so-called regular noise distribution, where the error vector of weight t can be divided into t blocks with each block having only one non-zero entry. Therefore, we can obtain the VOLE correlation $\mathbf{b} = \mathbf{c} + \mathbf{e} \cdot \varDelta$ from t repetitions of single-point VOLE. Recall that we require a VOLE correlation (of length two) to produce one single-point VOLE. Thus, the above construction essentially extends a VOLE correlation of length $m + 2t$ into length n. In addition, as showed in [53], a kind of "bootstrapping" technique can be used to further reduce the communication complexity, where a part of the outputs can be reserved as the base VOLE correlation for the next extension.

[9] 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.

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

Init: Both parties send `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 do 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 $\alpha \overset{\$}{\leftarrow} [0, n)$, 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 $\mathbf{u}_\alpha = \beta$ and $\mathbf{u}_i = 0$ 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 $s \leftarrow \mathtt{GGM.KeyGen}(1^\kappa)$, and runs $\mathtt{GGM.Gen}(1^n, s)$ to obtain $(\{\mathbf{v}_j\}_{j\in[0,n)}, \{(K_0^i, K_1^i)\}_{i\in[h]})$. 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 $\mathcal{F}_{\mathrm{OT}}$ while $P_{\mathcal{R}}$ sends (K_0^i, K_1^i) to $\mathcal{F}_{\mathrm{OT}}$, 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 $\{\mathbf{v}_j\}_{j\neq\alpha}$.
   (e) $P_{\mathcal{R}}$ sends $g := \gamma - \sum_{j\in[0,n)} \mathbf{v}_j \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ defines a vector $\mathbf{w} \in \mathtt{GR}(2^k, d)^n$ such that $\mathbf{w}_i := \mathbf{v}_i$ for $i \neq \alpha$ and $\mathbf{w}_\alpha := \delta - (g + \sum_{i\neq\alpha} \mathbf{w}_i)$. 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 $\chi_i \overset{\$}{\leftarrow} \mathtt{GR}(2^k, d)$ 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 $V_{P_{\mathcal{S}}} := \sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i - z$, and sends $V_{P_{\mathcal{S}}}$ to $\mathcal{F}_{\mathrm{EQ}}$ as $\mathcal{P}$. $P_{\mathcal{R}}$ computes $V_{P_{\mathcal{R}}} := \sum_{i\in[0,n)} \chi_i \cdot \mathbf{v}_i - y$, 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. 10: 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.

LPN security. We remark that similar to [53,4], our $\Pi_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$ 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 a corrupted $P_{\mathcal{S}}$ to guess Δ (we formulate such attacks in $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$). The former leakage would affect the LPN assumption. Namely, it allows the LPN adversary to guess a subset for each block of the noise vector, which would leak 1-bit information in general: the adversary learns whether his guesses are all right.

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

1. *Let $A \leftarrow \mathcal{G}(m,n) \in \texttt{GR}(2^k,d)^{m\times n}$ be a primal LPN matrix, $\mathbf{x} \leftarrow \texttt{GR}(2^k,d)^m$ be the secret, and $\mathbf{e} = (\mathbf{e}^{(1)},...,\mathbf{e}^{(t)}) \in \texttt{GR}(2^k,d)^n$ be the error vector, where each $\mathbf{e}^{(i)}, i \in [t]$ has only one entry invertible in $\texttt{GR}(2^k,d)$ 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.*
3. *Pick $b \leftarrow \{0,1\}$. If $b = 0$, send $\mathbf{y} := \mathbf{x}\cdot A + \mathbf{e}$ to $\mathcal{A}$, otherwise, send $\mathbf{y} \overset{\$}{\leftarrow} \texttt{GR}(2^k,d)^n$ 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 $(\texttt{RG},\mathcal{G},\mathcal{R}) - \text{LPN}(m,n,t)$ 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 at most ϵ, i.e.

$$|\Pr[G_{\text{LPN}} = 1] - 1/2| \le \epsilon.$$

We give the VOLE extension protocol in Figure 11, and we have the following theorem with proof deferred to Appendix D.2.

Theorem 6. *If the decisional $(\texttt{RG},\mathcal{G},\texttt{GR}(2^k,d))$-LPN$(m,n,t)$ with static leakage assumption holds, then $\Pi_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{\text{qVOLE}}^{\texttt{GR}(2^k,d)}$ (see Figure 16)* [10] *in the $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$-hybrid model.*

We adapt existing reduction results for LPN over $\mathbb{Z}_{2^k}$ [46] to Galois rings, obtaining the following theorems with proofs in Appendix B.2.

Theorem 7. *If decisional $(\mathcal{D},\mathcal{G},\texttt{GR}(2^k,d)) - \text{LPN}(m,n,w_1)$ is (T,ϵ)-hard, then decisional $(\mathcal{D},\mathcal{G},\mathbb{F}_{2^d}) - \text{LPN}(m,n,w_2)$ is $(T - \mathsf{poly}(m,n), \mathcal{O}(h\epsilon))$-hard, where $(\mathcal{D},w_1,w_2,h) \in \{(\texttt{HW},t,\frac{2^{d(k-1)}(2^d-1)}{2^{dk}-1}t,\sqrt{t}),(\texttt{Ber},\lambda,\lambda,1)\}$.*

Theorem 8. *If decisional $(\texttt{Ber},\mathbb{F}_{2^d}) - \text{LPN}(m,n,\frac{\lambda(2^d-1)}{(1-\lambda)2^{dk}+\lambda 2^d})$ is (T,ϵ)-hard, then decisional $(\texttt{Ber},\texttt{GR}(2^k,d)) - \text{LPN}(m,n,\lambda)$ is $(T - \mathsf{poly}(m,n), k\epsilon)$-hard.*

[10] $\mathcal{F}_{\text{qVOLE}}^{\texttt{GR}(2^k,d)}$ extends $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$, which additionally captures the leakage of Δ.

We remark that Theorem 7 introduces an attack for LPN over $\texttt{GR}(2^k, d)$, where one can reduce an LPN instance's noise weight by approximately $1/2^d$ fraction via modulo 2. We see this attack is much less efficient than that for LPN over $\mathbb{Z}_{2^k}$, since it may be tolerable if d is set sufficiently large. Alternatively, similar to [4,46], we can additionally require that each non-zero entry of the error vector $\mathbf{e}$ is invertible to avoid this reduction attack. We also review main attacks that may work on LPN over Galois rings in Appendix B.3, following the analysis of [32,14,15,4,46].

Protocol $\Pi_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$

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

Init:

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

Extend:

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

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

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 (2022)
6. Baum, C., Dittmer, S., Scholl, P., Wang, X.: Sok: Vector OLE-based zero-knowledge protocols. IACR Cryptol. ePrint Arch. p. 857 (2023), `https://eprint.iacr.org/2023/857`
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. 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`
9. 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)
10. 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)
11. 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)
12. 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`
13. 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)
14. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: CCS 2018. pp. 896–912. ACM (2018)
15. 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)
16. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudo-random correlation generators: Silent OT extension and more. In: CRYPTO 2019. LNCS, vol. 11694, pp. 489–518. Springer (2019)
17. 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)
18. 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)
19. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001. pp. 136–145. IEEE Computer Society (2001)

20. 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)
21. 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)
22. 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)
23. 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)
24. 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)
25. 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)
26. 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)
27. Damgård, I., Zakarias, S.: Constant-overhead secure computation of boolean circuits using preprocessing. In: TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer (2013)
28. Debris-Alazard, T., Tillich, J.: Statistical decoding. In: ISIT 2017. pp. 1798–1802. IEEE (2017)
29. 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)
30. 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)
31. 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)
32. Esser, A., Kübler, R., May, A.: LPN decoded. In: CRYPTO 2017. LNCS, vol. 10402, pp. 486–514. Springer (2017)
33. 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)
34. 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`
35. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: Snarks for ring arithmetic. IACR Cryptol. ePrint Arch. p. 322 (2021)
36. Giacomelli, I., Madsen, J., Orlandi, C.: Zkboo: Faster zero-knowledge for boolean circuits. In: USENIX Security 2016. pp. 1069–1083. USENIX Association (2016)
37. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
38. 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)

39. Groth, J.: On the size of pairing-based non-interactive arguments. In: EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer (2016)
40. 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)
41. Heath, D., Kolesnikov, V.: Stacked garbling for disjunctive zero-knowledge proofs. In: EUROCRYPT 2020. LNCS, vol. 12107, pp. 569–598. Springer (2020)
42. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC 2007. pp. 21–30. ACM (2007)
43. 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)
44. 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)
45. Lin, F., Xing, C., Yao, Y., Yuan, C.: Amortized NISC over $\mathbb{Z}_{2^k}$ from RMFE. IACR Cryptol. ePrint Arch. p. 1363 (2023), `https://eprint.iacr.org/2023/1363`
46. Liu, H., Wang, X., Yang, K., Yu, Y.: The hardness of LPN over any integer ring and field for PCG applications. IACR Cryptol. ePrint Arch. p. 712 (2022), `https://eprint.iacr.org/2022/712`
47. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. Commun. ACM **59**(2), 103–112 (2016)
48. Peters, C.: Information-set decoding for linear codes over $\mathbb{F}_q$. In: PQCrypto 2010. LNCS, vol. 6061, pp. 81–94. Springer (2010)
49. 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)
50. 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)
51. Setty, S.T.V.: Spartan: Efficient and general-purpose zksnarks without trusted setup. In: CRYPTO 2020. LNCS, vol. 12172, pp. 704–737. Springer (2020)
52. Wan, Z.X.: Lectures on finite fields and Galois rings. World Scientific Publishing Company (2003)
53. 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)
54. 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)
55. 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)
56. 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)
57. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: FOCS 1986. pp. 162–167. IEEE Computer Society (1986)
58. 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

Security Model. In UC framework [19], 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 the adversary $\mathcal{A}$ 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 the adversary $\mathcal{A}$ such that the environment $\mathcal{Z}$ can not distinguish $\mathcal{S}$ and the honest parties from the adversary $\mathcal{A}$'s view along with all parties' inputs and outputs. More formally, We say a protocol $\mathit{\Pi}$ 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, $\texttt{Commit}$ and $\texttt{Open}$. In the commit phase of the protocol, the sender $P_{\mathcal{S}}$ invokes $\texttt{Commit}$ to commit some value m, obtaining $(\texttt{com}, \texttt{unv}) \leftarrow \texttt{Commit}(m)$ as the result. Then he sends $\texttt{com}$ 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 $\texttt{unv}$ to $P_{\mathcal{R}}$ such that $P_{\mathcal{R}}$ can check whether $\texttt{Open}(\texttt{com}, \texttt{unv}, m) = 1$. Informally, there are two security properties that a commitment scheme should satisfy; 1.*Hiding*: $P_{\mathcal{R}}$ can not learn anything about m from $\texttt{com}$ in the commit phase, and 2.*Binding*: $P_{\mathcal{S}}$ can not provide a $m' \neq m$ and a $\texttt{unv}'$ such that $\texttt{Open}(\texttt{com}, \texttt{unv}', m') = 1$ 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, $\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}$. The $\texttt{KeyGen}$ algorithm outputs a secret key sk, which is determined by its random tape. The $\texttt{Enc}$ algorithm takes the secret key sk and message m as inputs, and outputs a ciphertext $\langle m \rangle$, while the $\texttt{Dec}$ algorithm decrypts the ciphertext via sk, denoted by $m := \texttt{Dec}(\langle m \rangle, sk)$. 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 $\mathit{\Pi}_{\mathrm{PAC}}$ (Figure 19), we additionally require the AHE scheme to satisfy circuit privacy and degree restriction, similar to that in [54]. Such AHE schemes exist as shown in [23].

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 $(V_{\mathcal{P}} \overset{?}{=} V_{\mathcal{V}})$ 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 $(V_{\mathcal{P}} \overset{?}{=} V_{\mathcal{V}})$ 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 $P_{\mathcal{S}}$ and m_0, m_1 from $P_{\mathcal{R}}$: Send m_b to $P_{\mathcal{S}}$.

Fig. 13: Ideal functionality for oblivious transfer.

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

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 $(N - 1)$-out-of-N oblivious transfer.

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

Init: Upon receiving ($\mathtt{Init}$) 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 ($\mathtt{Init}$) commands will be ignored.

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

1. If $P_{\mathcal{R}}$ is honest, sample $\mathbf{K} \xleftarrow{\$} \mathtt{GR}(2^k, d)^l$. Otherwise receive $\mathbf{K}$ from $\mathcal{A}$.
2. If $P_{\mathcal{S}}$ is honest, sample $\mathbf{x} \xleftarrow{\$} \mathtt{GR}(2^k, d)^l$ and compute $\mathbf{M} := \mathbf{K} - \Delta \cdot \mathbf{x} \in \mathtt{GR}(2^k, d)^l$. Otherwise, receive $\mathbf{x} \in \mathtt{GR}(2^k, d)^l$ and $\mathbf{M} \in \mathtt{GR}(2^k, d)^l$ 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 $\mathcal{F}_{\text{qVOLE}}^{\mathtt{GR}(2^k,d)}$

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

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 $\mathtt{success}$ to $P_{\mathcal{S}}$ and ignore any subsequent global-key queries. Otherwise, send $\mathtt{abort}$ to both parties and abort.

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

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

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

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

1. If $P_{\mathcal{R}}$ is honest, sample $\mathbf{v} \overset{\$}{\leftarrow} \mathtt{GR}(2^k,d)^n$. 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 $\mathtt{success}$ to $P_{\mathcal{R}}$ and continue. Otherwise, send $\mathtt{abort}$ 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 $\mathtt{success}$ to $P_{\mathcal{S}}$ and ignore any subsequent global-key queries. Otherwise, send $\mathtt{abort}$ 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 [10] has been studied for decades, and it was adapted to be defined over a finite field $\mathbb{F}_q$ [14] 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 $t \le n$, $\mathtt{Im}(\mathcal{D}_{t,n}(\mathcal{R})) \subset \mathcal{R}^n$. 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 κ. 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 $A \leftarrow \mathcal{G}(m,n,\mathcal{R}), \mathbf{x} \overset{\$}{\leftarrow} \mathcal{R}^m, \mathbf{u} \overset{\$}{\leftarrow} \mathcal{R}^n$, 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.

Algorithm GGM

Let $G : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$, and $G' : \{0,1\}^\kappa \to \mathtt{GR}(2^k, d)^2 \times \{0,1\}^\kappa$ 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 `GGM.KeyGen`, `GGM.Gen`, `GGM.Eval` are defined as follows:

```
– GGM.KeyGen: on input $1^\kappa$, output $s \xleftarrow{\$} \{0,1\}^\kappa$.
– GGM.Gen: on input $n$ and $s \in \{0,1\}^\kappa$, 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 $(\{v_j\}_{j\in[0,n)}, \{(K_{0,i}, K_{1,i})\}_{i\in[h]}), \{\Gamma_j\}_{j\in[0,2^{h-1})}$.
– 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 $(\{v_j\}_{j\in[0,n)\backslash\{\alpha\}})$.
– 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. 18: Algorithms for GGM tree based PPRF.

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

We mainly consider three kinds of noise distributions in this paper:
Bernoulli. Let $\mathtt{Ber}(\mathcal{R}) = \{\mathtt{Ber}_{\lambda,n}(\mathcal{R})\}_{\lambda,n}$ be the family of Bernoulli distributions. $\mathbf{e} \leftarrow \mathtt{Ber}_{\lambda,n}(\mathcal{R})$ indicates that each entry of $\mathbf{e}$ is a uniformly random element in $\mathcal{R}$ with probability λ and zero with probability $1 - \lambda$. Thus the expected Hamming weight of $\mathbf{e}$ is $\lambda N(|\mathcal{R}| - 1)/|\mathcal{R}|$. We remark that this definition is equivalent to sampling a uniformly non-zero element in $\mathcal{R}$ with probability $\lambda N(|\mathcal{R}| - 1)/|\mathcal{R}|$ 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 7, restated). *If decisional $(\mathcal{D}, \mathcal{G}, \mathtt{GR}(2^k, d))-LPN(m, n, w_1)$ is (T, ϵ)-hard, then decisional $(\mathcal{D}, \mathcal{G}, \mathbb{F}_{2^d})-LPN(m, n, w_2)$ is $(T-\mathsf{poly}(m, n), \mathcal{O}(h\epsilon))$-hard, where $(\mathcal{D}, w_1, w_2, h) \in \{(\mathtt{HW}, t, \frac{2^{d(k-1)}(2^d-1)}{2^{dk}-1}t, \sqrt{t}), (\mathtt{Ber}, \lambda, \lambda, 1)\}$.*

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 [46] 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 $\Omega(1/\sqrt{t})$. On the other hand, $(A^{(0)}, \mathbf{u}^{(0)})$ are uniformly random as well. Therefore, one can use a $(\mathtt{HW}, \mathcal{G}, \mathbb{F}_{2^d}) - LPN(m, n, t')$ distinguisher to distinguish $(A^{(0)}, \mathbf{b}^{(0)})$ from uniform samples. The proof for the second statement is similar, except that for $\mathbf{e} \leftarrow \mathtt{Ber}_{\lambda,n}(\mathtt{GR}(2^k, d))$, one can immediately gets that $\mathbf{e}^{(0)} \sim \mathtt{Ber}_{\lambda,n}(\mathbb{F}_{2^d})$. $\square$

Theorem 10 (Theorem 8, restated). *If decisional* $(\mathtt{Ber}, \mathbb{F}_{2^d})-\mathrm{LPN}(m, n, \frac{\lambda(2^d-1)}{(1-\lambda)2^{dk}+\lambda 2^d})$ *is* (T, ϵ)*-hard, then decisional* $(\mathtt{Ber}, \mathtt{GR}(2^k, d))-\mathrm{LPN}(m, n, \lambda)$ *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 $\mathbf{u}^{(i)} \overset{\$}{\leftarrow} \mathbb{F}_{2^d}^n$ *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* $(\mathbf{e}_j^{(0)}, \mathbf{e}_j^{(1)}, ..., \mathbf{e}_j^{(k-1)}) \leftarrow \mathtt{Decom}(\mathtt{Ber}_{\lambda,n}(\mathtt{GR}(2^k, d)))$*, where* $\mathbf{e}_j^{(i)}$ *is the* j*-th entry of* $\mathbf{e}^{(i)}$*. If there exists a non-zero component in* $(\mathbf{e}_j^{(0)}, \mathbf{e}_j^{(1)}, ..., \mathbf{e}_j^{(i-1)})$*,* $\mathbf{e}_j^{(i)}$ *must be uniformly random, which makes* $\mathbf{b}_j^{(i)}$ *uniformly random. Thus, we have that*

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

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

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 [32,14,15,4,46,17].

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 [14]. 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[14]	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 [14] simplified the complexity of the SD attack [28] by $\log(m+1) + t \cdot \log \frac{n}{n-m-1}$. A recent work [46] 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 [46], 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

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

Information Set Decoding (ISD) The authors of [46] analysed known ISD variants for different field size and show that the generalized SD-ISD attack [48] 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 [17] 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). *$\Pi^{\mathtt{GR}(2^k,d)}_{\text{embVOLE}}$ UC-realizes $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{embVOLE}}$ in the $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{VOLE}}$-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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{VOLE}}$.*

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

1. *$S_{\mathcal{S}}$ reads $\mathbf{M}, \mathbf{x} \in \mathtt{GR}(2^k, d)^{n+s}$ that $\mathcal{A}$ sends to $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{VOLE}}$.*
2. *Upon receiving $\boldsymbol{\eta} \in \mathtt{GR}(2^k, d)^{n+s}$ from $\mathcal{A}$, if $\boldsymbol{\eta} \notin \text{Ker}(\psi)^{n+s}$, $S_{\mathcal{S}}$ aborts. $S_{\mathcal{S}}$ sets $\mathbf{M}' := \mathbf{M}$.*
3. *$S_{\mathcal{S}}$ samples $\boldsymbol{\chi}^{(1)}, ..., \boldsymbol{\chi}^{(s)} \xleftarrow{\$} \mathbb{Z}^n_{2^k}$, and sends them to $\mathcal{A}$.*
4. *Upon receiving $\mathbf{a}, \mathbf{b} \in \mathtt{GR}(2^k, d)^s$ from $\mathcal{A}$. If $\mathbf{b}_i - \mathbf{a}_i \neq \eta_{n+i} + \sum_{j\in[n]} \chi^{(i)}_j \cdot \eta_j$, for some $i \in [s]$, or $\mathbf{b} \notin \mathtt{Im}(\phi)^s$, $S_{\mathcal{S}}$ aborts.*
5. *$S_{\mathcal{S}}$ receives $\hat{\mathbf{M}}_i, i \in [s]$ from $\mathcal{A}$. $S_{\mathcal{S}}$ computes $\hat{\mathbf{M}}'_i := \mathbf{M}_{n+i} + \sum_{j\in[n]} \chi^{(i)}_j \cdot \mathbf{M}_j$, for $i \in [s]$. If both $\boldsymbol{\eta} = (\tau(\mathbf{x}) - \mathbf{x})$ and $\hat{\mathbf{M}} = \hat{\mathbf{M}}'$ hold, $S_{\mathcal{S}}$ sends $(\mathbf{x}[1:n], \mathbf{M}[1:n])$ to $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{embVOLE}}$. 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 $\boldsymbol{\eta} \in \text{Ker}(\psi)^{n+s}$, but $\boldsymbol{\eta} \neq \tau(\boldsymbol{x}) - \boldsymbol{x}$. Let $\boldsymbol{\eta} = \tau(\mathbf{x}) - \mathbf{x} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon} \in \text{Ker}(\psi)^{n+s}$. We have that

$$\boldsymbol{\eta} = \tau(\mathbf{x}) - (\mathbf{x} - \boldsymbol{\varepsilon}) = \tau(\mathbf{x} - \boldsymbol{\varepsilon}) - (\mathbf{x} - \boldsymbol{\varepsilon}).$$

Therefore, $\mathcal{A}$ can set

$$\mathbf{a}^*_i := (\mathbf{x}_{n+i} - \varepsilon_{n+i}) + \sum_{j\in[n]} \chi^{(i)}_j \cdot (\mathbf{x}_j - \varepsilon_j),$$

and

$$\mathbf{b}_i^* := \tau(\mathbf{x}_{n+i} - \varepsilon_{n+i}) + \sum_{j\in[n]} \chi_j^{(i)} \cdot \tau(\mathbf{x}_j - \varepsilon_j) = \tau(\mathbf{x}_{n+i}) + \sum_{j\in[n]} \chi_j^{(i)} \cdot \tau(\mathbf{x}_j) = \mathbf{b}_i,$$

for $i \in [s]$. These $\mathbf{a}_i^*, \mathbf{b}_i^*$ would pass the check of honest $P_{\mathcal{R}}$. Now in real protocol, we have that

$$\begin{aligned}
\hat{\mathbf{M}}_i' &= \mathbf{K}_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \mathbf{K}_j - \varDelta \cdot \mathbf{a}_i^* \\
&= (\mathbf{M}_{n+i} + \varDelta\mathbf{x}_{n+i}) + \sum_{j\in[n]} \chi_j^{(i)}(\mathbf{M}_j + \varDelta\mathbf{x}_j) - \varDelta(\mathbf{x}_{n+i} - \varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)}(\mathbf{x}_j - \varepsilon_j)) \\
&= \mathbf{M}_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \mathbf{M}_j + \varDelta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \varepsilon_j) \\
&= \hat{\mathbf{M}}_i + \varDelta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \varepsilon_j).
\end{aligned}$$

Thus, the check $\hat{\mathbf{M}} = \hat{\mathbf{M}}'$ passes if and only if

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

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 $\boldsymbol{\eta} \neq \tau(\mathbf{x}) - \mathbf{x}$, 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 $\varDelta \in \mathtt{GR}(2^k, d)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ in the `Init` procedure. $S_{\mathcal{R}}$ then sends $\varDelta$ 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 $\mathbf{K} \in \mathtt{GR}(2^k, d)^{n+s}$ sent by $\mathcal{A}$. Upon receiving $\boldsymbol{\eta} \in \mathrm{Ker}(\psi)^n$ from $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, $S_{\mathcal{R}}$ samples $\boldsymbol{\eta}' \xleftarrow{\$} \mathrm{Ker}(\psi)^s$ and sends $\hat{\boldsymbol{\eta}} := (\boldsymbol{\eta}, \boldsymbol{\eta}') \in \mathrm{Ker}(\psi)^{n+s}$ to $\mathcal{A}$.*
2. *$S_{\mathcal{R}}$ sets $\mathbf{K}' := \mathbf{K} + \varDelta \cdot \hat{\boldsymbol{\eta}}$. Upon receiving $\boldsymbol{\chi}^{(i)} \in \mathbb{Z}_{2^k}^n, i \in [s]$ from $\mathcal{A}$, $S_{\mathcal{R}}$ samples $\mathbf{b} \xleftarrow{\$} \mathtt{Im}(\phi)^s$, and computes $\mathbf{a}_i := \mathbf{b}_i - \eta_{n+i} - \sum_{j\in[n]} \chi_j^{(i)} \cdot \eta_j$, for $i \in [s]$. Then, $S_{\mathcal{R}}$ sends $\mathbf{a}, \mathbf{b}$ to $\mathcal{A}$.*
3. *$S_{\mathcal{R}}$ computes $\hat{\mathbf{M}}_i := \mathbf{K}_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \mathbf{K}_j - \varDelta \cdot \mathbf{a}_i$, for $i \in [s]$. $S_{\mathcal{R}}$ sends $\hat{\mathbf{M}}$ to $\mathcal{A}$.*
4. *$S_{\mathcal{R}}$ sends $\mathbf{K}[1:n]$ 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 $\boldsymbol{\eta} \in \mathrm{Ker}(\psi)^{n+s}$. Since $\mathbf{x}$ is distributed uniformly at random in $\mathtt{GR}(2^k, d)^{n+s}$, $\boldsymbol{\eta}$ is distributed uniformly at random in $\mathrm{Ker}(\psi)^{n+s}$. While in simulation, $\hat{\boldsymbol{\eta}}$ are uniformly sampled from $\mathrm{Ker}(\psi)^{n+s}$ as well. As for $(\mathbf{a}, \mathbf{b} = \tau(\mathbf{a}))$ in real protocol, $\mathbf{a}$ is masked with $\mathbf{x}[n+1 : n+s]$, which makes $\mathbf{b}$ have the uniform distribution on $\mathrm{Im}(\phi)^s$. 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 $\hat{\mathbf{M}}$. It can be easily verified that

$$\hat{\mathbf{M}}'_i = \mathbf{K}_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \mathbf{K}_j - \Delta \cdot \mathbf{a}_i = \mathbf{M}_{n+i} + \sum_{j\in[n]} \chi_j^{(i)} \cdot \mathbf{M}_j = \hat{\mathbf{M}}_i,$$

Thus $\hat{\mathbf{M}}$ has the same distribution in both worlds. Finally, we turn to the output of the honest $P_{\mathcal{S}}$. In real protocol, $\mathbf{x}_i$ is conditioned on that $\tau(\mathbf{x}_i) - \mathbf{x}_i = \eta_i$, for all $i \in [n]$, while they have the same properties in ideal simulation. Therefore, no PPT *environment $\mathcal{Z}$ can distinguish the ideal simulation and the real execution. This completes the proof.* □

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

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 $(\pi, \mathbf{M}_\pi)$ 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 $\mathcal{A}$, $\mathcal{S}$ checks $\boldsymbol{\delta} - \tau(\boldsymbol{\delta}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$. If the check fails, aborts. $\mathcal{S}$ can locally compute $[\tau(\omega_i)] := [\tau(\mu_i)] + \tau(\delta_i)$, 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 $\boldsymbol{\omega} \in \mathtt{Im}(\phi)^n$, which can be one to one corresponded to m instances of $\mathbb{Z}_{2^k}^n$. The proof is direct since the check $\boldsymbol{\delta} - \tau(\boldsymbol{\delta}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ guarantees that

$$\boldsymbol{\omega} = \boldsymbol{\delta} + \boldsymbol{\mu} = \tau(\boldsymbol{\delta}) + \tau(\boldsymbol{\mu}) = \tau(\boldsymbol{\delta} + \boldsymbol{\mu}) = \tau(\boldsymbol{\omega}).$$

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 $\boldsymbol{d}$ 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*

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

and

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

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 $\mathbf{M}'_{\omega_h} := \mathbf{M}_{\omega_h} + e_{\omega_h}$ such that $\mathbf{K}_{\omega_h} = \mathbf{M}'_{\omega_h} + \Delta \cdot \phi(\mathbf{1})$ 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

$$\mathbf{K}_{\omega_h} = \mathbf{M}_{\omega_h} + \Delta \cdot \phi(\mathbf{r}).$$

Thus, honest verifier accepts if and only if

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

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 $\mathbf{K}_\pi$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$. Besides, $\mathcal{S}$ samples $\tau(\boldsymbol{\mu}) - \boldsymbol{\mu} \overset{\$}{\leftarrow} \mathrm{Ker}(\psi)^n$ and $\tau(\boldsymbol{\nu}) - \boldsymbol{\nu} \overset{\$}{\leftarrow} \mathrm{Ker}(\psi)^t$, and sends them to $\mathcal{A}$.*
2. *$\mathcal{S}$ samples $\tau(\boldsymbol{\delta}) \overset{\$}{\leftarrow} \mathrm{Im}(\phi)^n$ and sets $\boldsymbol{\delta} := \tau(\boldsymbol{\delta}) + \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$. $\mathcal{S}$ sends $\boldsymbol{\delta}$ to $\mathcal{A}$. $\mathcal{S}$ computes $\mathbf{K}_{\tau(\omega_i)} := \mathbf{K}'_{\mu_i} + \Delta \cdot \tau(\delta_i)$, for $i \in [n]$. Besides, $\mathcal{S}$ samples $\boldsymbol{\omega} \overset{\$}{\leftarrow} \mathrm{Im}(\phi)^n$.*
3. *For each gate $(\alpha, \beta, \gamma, T) \in \mathcal{C}$, in a topological order:*
   - *If T=*`Add`*, $\mathcal{S}$ computes $\mathbf{K}_{\omega_\gamma} := \mathbf{K}_{\omega_\alpha} + \mathbf{K}_{\omega_\beta}$ 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 $\mathbf{K}_{\omega_\gamma} := \mathbf{K}'_{\nu_i} + \Delta \cdot \tau(d_i)$, $\mathbf{K}_{\hat{\omega}_\gamma} := \mathbf{K}_{\nu_i} + \Delta \cdot d_i$ and $B_i := \mathbf{K}_{\omega_\alpha} \cdot \mathbf{K}_{\omega_\beta} - \Delta \cdot \mathbf{K}_{\hat{\omega}_\gamma}$ 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 $B^* := \mathbf{K}_\pi$. Then $\mathcal{S}$ samples $Y \overset{\$}{\leftarrow} \mathtt{GR}(2^k, d)$ 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 $\mathbf{K}_{\omega_h}$. $\mathcal{S}$ computes $\mathbf{M}_{\omega_h} := \mathbf{K}_{\omega_h} - \Delta \cdot \phi(\mathbf{1})$, and sends $\mathbf{M}_{\omega_h}$ to $\mathcal{A}$.*

Since $\mathtt{GR}(2^k, d) = \mathrm{Im}(\phi) \oplus \mathrm{Ker}(\psi)$, the distribution of $\boldsymbol{\delta}$ conditioned on $\boldsymbol{\delta} - \tau(\boldsymbol{\delta}) = \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ is equivalent to that of $\boldsymbol{\delta} := \tau(\boldsymbol{\delta}) + \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ where $\tau(\boldsymbol{\delta})$ are sampled uniformly at random in $\mathrm{Im}(\phi)^n$. Further, $\boldsymbol{\delta}$ is sufficient to perfectly hide 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 Our ZK protocol with sublinear communication

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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{embVOLE}}$, and $\mathcal{V}$ receives $\varDelta \in \mathtt{GR}(2^k, d)$.
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 $\varLambda \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and obtain ciphertexts $\langle \varLambda^i \rangle := \mathtt{Enc}(sk, \varLambda^i; r_i)$ 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 $\langle \varLambda^1 \rangle, ..., \langle \varLambda^m \rangle$ to $\mathcal{P}$.

Pre-Gen: On input $[\mathbf{u}]$:

1. For the input MACs $[\mathbf{u}]$, suppose $\mathcal{P}$ 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 $\mathbf{w} := \mathbf{v} - \varDelta \cdot \mathbf{u}$.
2. 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 $\langle \mathbf{b_j} \rangle := \sum_{i \in [m]} f_{j,i} \cdot \langle \varLambda^i \rangle + f_{j,0} - \mathbf{u}_j$.
3. $\mathcal{P}$ computes $(\mathtt{com}_2, \mathtt{unv}_2) \leftarrow \mathtt{Commit}(\langle \mathbf{b}_1 \rangle, ..., \langle \mathbf{b}_n \rangle)$, and sends $\mathtt{com}_2$ to $\mathcal{V}$.

Gen:

1. $\mathcal{V}$ sends $(\mathtt{unv}_1, \mathtt{seed})$ and $\varLambda$ to $\mathcal{P}$. $\mathcal{P}$ checks $\mathtt{Open}(\mathtt{com}_1, \mathtt{unv}_1, \mathtt{seed}) = 1$. If the check fails, $\mathcal{P}$ aborts. Then, $\mathcal{P}$ computes $(r_0, r_1, ..., r_m) \leftarrow G(\mathtt{seed})$ and obtains $sk \leftarrow \mathtt{KeyGen}(r_0)$. $\mathcal{P}$ checks that $\langle \varLambda^i \rangle = \mathtt{Enc}(sk, \varLambda^i; r_i)$ for all $i \in [m]$, and aborts if the check fails. For each $j \in [n]$, $\mathcal{P}$ sets $\mathbf{M}_j := \mathbf{w}_j$.
2. $\mathcal{P}$ sends $(\mathtt{unv}_2, \langle \mathbf{b}_1 \rangle, ..., \langle \mathbf{b}_n \rangle)$ to $\mathcal{V}$. $\mathcal{V}$ checks $\mathtt{Open}(\mathtt{com}_2, \mathtt{unv}_2, \langle \mathbf{b}_1 \rangle, ..., \langle \mathbf{b}_n \rangle) = 1$. If the check fails, $\mathcal{V}$ aborts. Then, for $j \in [n]$, $\mathcal{V}$ computes $b_j := \mathtt{Dec}(sk, \langle \mathbf{b}_j \rangle)$, and sets $\mathbf{K}_j := \mathbf{v}_j + \varDelta \cdot \mathbf{b}_j$. Thus, $\mathcal{P}$ and $\mathcal{V}$ obtain a PAC $[f_j(\cdot)]$, for each $j \in [n]$.

Fig. 19: Protocol for generating PACs over $\mathtt{GR}(2^k, d)$.

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

Proof. Since the proof has the similar structure to that in AntMan [54], hence we only explain the difference here. The readers may refer to [54] 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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{embVOLE}}$ which delivers MACs of random re-embedding pairs $(\boldsymbol{\mu}, \tau(\boldsymbol{\mu}))$ over Galois rings. Recall that we let $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{embVOLE}}$ 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 [54] by the* LPZK-*based approach, namely the multiplication check procedure in protocol $\Pi^{m,n,t}_{\mathrm{ZK}}$. 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 $\mathbb{F}_{2^d}$ is sufficient 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. $\Pi^{m,n,t}_{\mathrm{ZK}}$, 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 4, restated). *Protocol $\Pi^{\mathtt{GR}(2^k,d)}_{\mathrm{sfVOLE}}$ UC realizes $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{sfVOLE}}$ (Figure 15) in the $\mathcal{F}^{N}_{\mathrm{OT}-\bar{1}}$-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 $\mathbf{M}, \mathbf{x} \in \mathtt{GR}(2^k, d)^l$ to the ideal functionality $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{sfVOLE}}$. 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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{sfVOLE}}$. Then $S_{\mathcal{R}}$ computes $\mathbf{K} = \sum_{y \in \mathbb{F}_{2^d} \setminus \{\Delta\}} \mathbf{s}_y \cdot (\Delta - y)$ and sends*

$\mathbf{K} \in \mathtt{GR}(2^k,d)^l$ *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

Theorem 15 (Theorem 5, 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 $1/2^d + \mathsf{negl}(\kappa)$.*

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 $K^i \xleftarrow{\$} \{0,1\}^\kappa$, 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 $K^i_{\bar{\alpha}_i} := K^i$ 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 $\mathbf{u}_\alpha = \beta$ and $\mathbf{u}_i = 0$ for $i \neq \alpha$. Next, $S_{\mathcal{S}}$ runs $\mathtt{GGM.Eval}(\alpha, \{K^i_{\bar{\alpha}_i}\}_{i\in[h]})$ and obtains $\{\mathbf{v}_j\}_{j\neq\alpha}$.*
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 $\mathbf{w}_\alpha = \delta - (g + \sum_{i\neq\alpha} \mathbf{v}_i)$ and $\mathbf{w}_i = \mathbf{v}_i$ 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 $V'_{P_{\mathcal{S}}} := \sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i - z$ 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
      - *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 $\Delta' := \frac{\varepsilon}{2^{id_\eta}} \cdot (\frac{\eta}{2^{id_\eta}})^{-1} \bmod 2^{k-id_\eta}$, and sends a global-key query $(\mathtt{Guess}, \Delta', k - id_\eta)$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. If $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ returns* `success`*, $S_{\mathcal{S}}$ sends* `true` *to $\mathcal{A}$, and sends $\boldsymbol{u}, \boldsymbol{w}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. 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 $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Thus, we focus on analysing indistinguishability between the* ***`SP-Extend`*** *procedures.*

$S_{\mathcal{S}}$ can record a, c that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. 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 $\varDelta$ 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 $(\mathtt{Extend}, 1)$ *enables $S_{\mathcal{S}}$ to record x, z. In real protocol, honest verifier $P_{\mathcal{R}}$ computes*

$$\begin{aligned}
V_{P_{\mathcal{R}}} := & \sum_{i\in[0,n)} \chi_i \cdot \mathbf{v}_i - y \\
= & \sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i + \sum_{i\in[0,n)} \chi_i \cdot (\mathbf{v}_i - \mathbf{w}_i) - y \\
= & \sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i + \chi_\alpha \cdot (\mathbf{v}_\alpha - \mathbf{w}_\alpha) - (y^* + \varDelta \cdot x^*) \\
= & \sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i - (y^* - \varDelta \cdot x) - \varDelta \cdot (x^* + x - \chi_\alpha \cdot \beta) \\
= & V'_{P_{\mathcal{S}}} - \varDelta \cdot (x' - \chi_\alpha \cdot \beta).
\end{aligned}$$

If $\mathcal{A}$ behaves honestly, then $x' = \chi_\alpha \cdot \beta$ will hold and $\mathcal{F}_{\mathrm{EQ}}$ will return `true`. *Note that $S_{\mathcal{S}}$ can extract α from $\mathcal{A}$'s inputs to $\mathcal{F}_{\mathrm{OT}}$, thus $S_{\mathcal{S}}$ can check $x' = \chi_\alpha \cdot \beta$. If the equation holds, $\mathcal{F}_{\mathrm{EQ}}$ 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

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

where $\varDelta, \eta, \epsilon \in \mathtt{GR}(2^k, d)$. Although $S_{\mathcal{S}}$ does not hold $\varDelta$, he can query the global key to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Since the above equation is over $\mathtt{GR}(2^k, d)$, there may be no solutions for $\varDelta$, or more than one solutions for $\varDelta$. Thus the ***`Global-key Query`*** *is extended to allow queries of "lower bit" of $\varDelta$. 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 $\varDelta \in \mathtt{GR}(2^k, d)$ 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 $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ 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 $\mathcal{F}_{\text{OT}}$. 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 $\{\mathbf{w}_j^{(\alpha)}\}_{j \neq \alpha} := \mathtt{GGM.Eval}(\alpha, \{K_{\bar{\alpha}_i}^i\}_{i \in [h]})$ and $\mathbf{w}_\alpha^{(\alpha)} := \delta - (g + \sum_{i \neq \alpha} \mathbf{w}_i^{(\alpha)})$.*
3. *$S_{\mathcal{R}}$ reads the value y^* that $\mathcal{A}$ sends to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ 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 $\mathcal{F}_{\text{EQ}}$. Then $S_{\mathcal{R}}$ constructs a set $I \subseteq [0, n)$ as follows:*
   (a) For $\alpha \in [0, n)$, compute $V_{P_{\mathcal{S}}}^{\alpha} := \sum_{i \in [0,n)} \chi_i \cdot \mathbf{w}_i^{(\alpha)} + \Delta \cdot \chi_\alpha \cdot \beta - y$.
   (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 $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$. 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 $\mathcal{F}_{\text{EQ}}$), 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 $\mathbf{v}_i := \mathbf{w}_i^{(\alpha)}$, for $i \neq \alpha$ and $\mathbf{v}_\alpha := \gamma - g - \sum_{i \neq \alpha} \mathbf{v}_i$. $S_{\mathcal{R}}$ sends $\mathbf{v}$ to $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$.*

The Init *procedure can be called only once, and $S_{\mathcal{R}}$ learns Δ that $\mathcal{A}$ sends to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. 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 $\mathcal{F}_{\text{OT}}$, 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 $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. 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 $(\{\mathbf{v}_j\}_{j \in [0,n)}, \{(K_0^i, K_1^i)\}_{i \in [h]})$, (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

$$\Pr\Big[\mathbf{v}^{(\alpha)} \neq \mathbf{v}^{(\alpha')} \;\Big|\, V^{\alpha}_{P_{\mathcal{S}}} = V^{\alpha'}_{P_{\mathcal{S}}}\Big] \leq \frac{1}{2^d}.$$

We have that

$$V^{\alpha}_{P_{\mathcal{S}}} = V^{\alpha'}_{P_{\mathcal{S}}} \iff$$

$$\sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i^{(\alpha)} + \Delta \cdot \chi_\alpha \cdot \beta - y = \sum_{i\in[0,n)} \chi_i \cdot \mathbf{w}_i^{(\alpha')} + \Delta \cdot \chi_{\alpha'} \cdot \beta - y \iff$$

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

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

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

and

$$\Delta \cdot \beta = \mathbf{w}_\alpha^{(\alpha')} - \mathbf{w}_\alpha^{(\alpha)} = \mathbf{w}_{\alpha'}^{(\alpha)} - \mathbf{w}_{\alpha'}^{(\alpha')}$$

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 $V^{\alpha}_{P_{\mathcal{S}}} = V^{\alpha'}_{P_{\mathcal{S}}}$. 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 6, restated). *If the decisional $(\mathtt{RG}, \mathcal{G}, \mathtt{GR}(2^k,d))$-LPN$(m,n,t)$ with static leakage assumption holds, then $\Pi^{\mathtt{GR}(2^k,d)}_{\text{VOLE}}$ UC-realizes $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{q\text{VOLE}}$ in the $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{spVOLE}}$-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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\text{spVOLE}}$.*

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

3. *Whenever $\mathcal{A}$ sends a global-key query $(Guess, \Delta', s')$ to $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$, $S_{\mathcal{S}}$ sends $(Guess, \Delta')$ to $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$ and forwards the answer from $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$ 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 $\Delta \in \texttt{GR}(2^k,d)$ from $\mathcal{A}$ and reads $\mathbf{v} \in \texttt{GR}(2^k,d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$ in the* `Init` *procedure. $S_{\mathcal{R}}$ sends Δ to $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$. 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 $\mathbf{b}^{(i)} \in \texttt{GR}(2^k,d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$. Let $\mathbf{b} := (\mathbf{b}^{(1)}, ..., \mathbf{b}^{(t)}) \in \texttt{GR}(2^k,d)^n$.*
2. *$S_{\mathcal{R}}$ samples a random $\mathbf{e} := (\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)}) \in \texttt{GR}(2^k,d)^n$, 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 $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$. 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 $\mathbf{K} := \mathbf{v} \cdot A + \mathbf{b} \in \texttt{GR}(2^k,d)^n$, and updates $\mathbf{v} := \mathbf{K}[1:m]$. $S_{\mathcal{R}}$ sends $\mathbf{K}[m+1:n]$ to $\mathcal{F}_{\text{VOLE}}^{\texttt{GR}(2^k,d)}$.*

It is not hard to see that $\mathbf{e}$ sampled by $S_{\mathcal{R}}$ has the same distribution to that provided by $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$, therefore, the probability that $\mathcal{F}_{\text{spVOLE}}^{\texttt{GR}(2^k,d)}$ 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 $(\texttt{RG}, \mathcal{G}, \texttt{GR}(2^k,d))$-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 4.2 can not be made two-round, since the two parties need to obtain an additional VOLE correlation for the check. Following the approach of [15], we construct a two-round VOLE extension protocol that relies on the ideal functionality for generalized VOLE over Galois rings (see $\mathcal{F}_{\text{gVOLE}}^{\texttt{GR}(2^k,d)}$ 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 $\mathcal{F}_{\text{gVOLE}}^{\texttt{GR}(2^k,d)}$ can be naturally realized by OTs. To achieve malicious security, as shown in [45] we can apply a Galois ring analogue of the eVOLE technique [30].

We provide a multi-point VOLE construction in Figure 22, which can be viewed as a concatenation 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 construction 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 [40] 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 `GGM.Eval` algorithm and obtain `GGM.Eval`$'$ in Figure 18. To ensure all the right leaves fix a unique tree, we require that G' has the right half injective property [11].

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

Init: Upon receiving (`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 (`Init`) commands will be ignored.
Extend: Upon receiving (`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 $\mathtt{GR}(2^k, d)$.

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

[11] As noted in [15], 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.

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

Init: Upon receiving ($\mathtt{Init}$) 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 ($\mathtt{Init}$) 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 $\mathbf{e}^{(j)}_{\alpha_j} = y_j$ and $\mathbf{e}^{(j)}_i = 0, i \neq \alpha_j$, 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 $\mathtt{success}$ to $\mathcal{A}$ and continue. Otherwise, send $\mathtt{abort}$ 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 $\mathtt{success}$ to $\mathcal{A}$ and continue. Otherwise, send $\mathtt{abort}$ 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 $\mathtt{GR}(2^k, d)$.

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

Init: Both parties send `Init` to $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$, 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 $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$, 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 $s^{(j)} \leftarrow \mathtt{GGM.KeyGen}(1^{\kappa})$, and runs $\mathtt{GGM.Gen}(2^{h+1}, s^{(j)})$ to obtain $(\{(\mathbf{v}_{2i}^{(j)}, \mathbf{v}_{2i+1}^{(j)}, \Gamma_i^{(j)})\}_{i \in [0,2^h)}, \{(K_{0,i}^{(j)}, K_{1,i}^{(j)})\}_{i \in [h+1]})$. Recompute $K_{1,h+1}^{(j)} := \bigoplus_{i \in [0,2^h)} \Gamma_i^{(j)}$. Let $\alpha^{(j)} = \sum_{i \in [h]} 2^{h-i} \alpha_i^{(j)}$, where $\alpha_i^{(j)} \overset{\$}{\leftarrow} \{0,1\}$, and let $\bar{\alpha}_i^{(j)}$ denote the complement of $\alpha_i^{(j)}$. For $i \in [h]$, $P_{\mathcal{S}}$ sends $\bar{\alpha}_i^{(j)}$ to $\mathcal{F}_{\text{OT}}$ while $P_{\mathcal{R}}$ sends $(K_{0,i}^{(j)}, K_{1,i}^{(j)})$ to $\mathcal{F}_{\text{OT}}$, then $P_{\mathcal{S}}$ receives $K_{\bar{\alpha}_i^{(j)},i}^{(j)}$. $P_{\mathcal{R}}$ sends $K_{1,h+1}^{(j)}$ to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ runs $\mathtt{GGM.Eval}'(\alpha^{(j)}, \{K_{\bar{\alpha}_i^{(j)},i}^{(j)}\}_{i \in [h]}, K_{1,h+1}^{(j)})$ and gets $(\{(\mathbf{v}_{2i}^{(j)}, \mathbf{v}_{2i+1}^{(j)})\}_{i \neq \alpha^{(j)}}, \{\Gamma_i^{(j)}\}_{i \in [0,2^h)})$. Additionally, $P_{\mathcal{R}}$ sends $g_0^{(j)} := \mathbf{K}_j - \sum_{i \in [0,2^h)} \mathbf{v}_{2i}^{(j)}$ and $g_1^{(j)} := \mathbf{a}_j - \sum_{i \in [0,2^h)} \mathbf{v}_{2i+1}^{(j)} \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{S}}$.

(c) $P_{\mathcal{S}}$ defines vectors $\mathbf{W}_0^{(j)}, \mathbf{W}_1^{(j)} \in \mathtt{GR}(2^k, d)^{2^h}$ such that $\mathbf{W}_{0,i}^{(j)} = \mathbf{v}_{2i}^{(j)}$, $\mathbf{W}_{1,i}^{(j)} = \mathbf{v}_{2i+1}^{(j)}$ for $i \neq \alpha^{(j)}$ and $\mathbf{W}_{0,\alpha^{(j)}}^{(j)} := \mathbf{M}_j - (g_0^{(j)} + \sum_{i \neq \alpha^j} \mathbf{W}_{0,i}^{(j)})$, $\mathbf{W}_{1,\alpha^{(j)}}^{(j)} := \mathbf{c}_j - (g_1^{(j)} + \sum_{i \neq \alpha^{(j)}} \mathbf{W}_{1,i}^{(j)})$. $P_{\mathcal{R}}$ defines vectors $\mathbf{V}_0^{(j)} := (\mathbf{v}_0^{(j)}, \mathbf{v}_2^{(j)}, ..., \mathbf{v}_{2^{h+1}-2}^{(j)})$ and $\mathbf{V}_1^{(j)} := (\mathbf{v}_1^{(j)}, \mathbf{v}_3^{(j)}, ..., \mathbf{v}_{2^{h+1}-1}^{(j)})$. Note that $\mathbf{V}_0^{(j)} = \mathbf{W}_0^{(j)} + \mathbf{e}^{(j)} \cdot \Delta$ and $\mathbf{V}_1^{(j)} = \mathbf{W}_1^{(j)} + \mathbf{e}^{(j)} \cdot \delta$, 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. $e_{\alpha^j}^j = y_j \in \mathtt{GR}(2^k, d)^*$ and $e_i^j = 0$ for $i \neq \alpha^j$.

(d) $P_{\mathcal{R}}$ computes $\Gamma^{(j)} := \mathtt{Hash}(\Gamma_0^{(j)}, ..., \Gamma_{2^h-1}^{(j)})$ and sends it to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ computes $\hat{\Gamma}^{(j)} := \mathtt{Hash}(\Gamma_0^{(j)}, ..., \Gamma_{2^h-1}^{(j)})$ 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 $V_j := \sum_{i \in [0,2^h)} \chi_i (\mathbf{v}_{2i}^{(j)} + \chi \cdot \mathbf{v}_{2i+1}^{(j)})$. $P_{\mathcal{R}}$ 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 $W_j := \sum_{i \in [0,2^h)} \chi_i (\mathbf{W}_{0,i}^{(j)} + \chi \cdot \mathbf{W}_{1,i}^{(j)})$. $P_{\mathcal{S}}$ 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 $(\mathbf{W}_0^{(1)}, ..., \mathbf{W}_0^{(t)}) \in \mathtt{GR}(2^k, d)^n$, $(\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)}) \in \mathtt{GR}(2^k, d)^n$. $P_{\mathcal{R}}$ outputs $(\mathbf{V}_0^{(1)}, ..., \mathbf{V}_0^{(t)}) \in \mathtt{GR}(2^k, d)^n$.

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

positions twice, while the corrupted sender learns nothing about Δ. We have the following theorem.

Theorem 17. *If G and G' are PRGs with G' having the right half injective property, and $\mathtt{Hash} : \{0,1\}^{2^h \kappa} \to \{0,1\}^{\kappa}$ is a collision resistant hash function, $\Pi^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$ realizes the functionality $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$ in the $(\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}, \mathcal{F}_{\mathrm{OT}})$-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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}$ by $\mathcal{A}$.*
2. *For $j \in [t]$ and $i \in [h]$, $S_{\mathcal{S}}$ records the input $\bar{\alpha}^{(j)}_i$ 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 $s^{(j)} \leftarrow \mathtt{GGM.KeyGen}(1^{\kappa})$, and gets $(\{(\mathbf{v}^{(j)}_{2i}, \mathbf{v}^{(j)}_{2i+1}, \Gamma^{(j)}_i)\}_{i\in[0,2^h)}, \{(K^{(j)}_{0,i}, K^{(j)}_{1,i})\}_{i\in[h+1]}) \leftarrow \mathtt{GGM.Gen}(2^{h+1}, s^{(j)})$. $S_{\mathcal{S}}$ resets $K^{(j)}_{1,h+1} := \bigoplus_{i\in[0,2^h)} \Gamma^{(j)}_i$. $S_{\mathcal{S}}$ emulates $\mathcal{F}_{\mathrm{OT}}$ by sending $K^{(j)}_{\bar{\alpha}^{(j)}_i, i}$ to $\mathcal{A}$. $S_{\mathcal{S}}$ sends $g^{(j)}_0, g^{(j)}_1 \xleftarrow{\$} \mathtt{GR}(2^k,d)$ and $K^{(j)}_{1,h+1}$ to $\mathcal{A}$.*
3. *For $j \in [t]$, $S_{\mathcal{S}}$ computes $\Gamma^{(j)} := \mathtt{Hash}(\Gamma^{(j)}_0, ..., \Gamma^{(j)}_{2^h-1})$ 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 $\mathbf{W}^{(j)}_{0,\alpha^{(j)}} := \mathbf{M}_j - g^{(j)}_0 - \sum_{i\neq\alpha^{(j)}} \mathbf{v}^{(j)}_{2i}$ and $\mathbf{W}^{(j)}_{1,\alpha^{(j)}} := \mathbf{c}_j - g^{(j)}_1 - \sum_{i\neq\alpha^{(j)}} \mathbf{v}^{(j)}_{2i+1}$. Next, $S_{\mathcal{S}}$ computes $W_j := \sum_{i\neq\alpha^{(j)}} \chi_i(\mathbf{v}^{(j)}_{2i} + \chi \cdot \mathbf{v}^{(j)}_{2i+1}) + \chi_{\alpha^{(j)}}(\mathbf{W}^{(j)}_{0,\alpha^{(j)}} + \chi \cdot \mathbf{W}^{(j)}_{1,\alpha^{(j)}})$. Finally, $S_{\mathcal{S}}$ samples $X \xleftarrow{\$} \mathtt{GR}(2^k,d)$, and sends $X, \{V_j := W_j + X \cdot \chi_{\alpha^{(j)}} \cdot \mathbf{y}_j\}_{j\in[t]}$ to $\mathcal{A}$.*
5. *$S_{\mathcal{S}}$ sets $\mathbf{W}^{(j)}_{0,i} := \mathbf{v}^{(j)}_{2i}$, 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 $\mathbf{e}^{(j)}_{\alpha^{(j)}} := \mathbf{y}_j$, and $\mathbf{e}^{(j)}_i = 0$ for $i \neq \alpha^{(j)}$. $S_{\mathcal{S}}$ sends $(\mathbf{W}^{(1)}_0, ..., \mathbf{W}^{(t)}_0)$, $(\mathbf{e}^{(1)}, ..., \mathbf{e}^{(t)})$ to $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$.*

The messages produced by $S_{\mathcal{S}}$ are generated in the same way to that in the real execution except for $\{g^{(j)}_0, g^{(j)}_1\}_{j\in[t]}$ 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 $(\mathbf{W}^{(1)}_0, ..., \mathbf{W}^{(t)}_0)$, 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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}$ 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 $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}$ by $\mathcal{A}$.*
2. *For $j \in [t], i \in [0,2^h)$, $S_{\mathcal{R}}$ records the input $(K^{(j)}_{0,i}, K^{(j)}_{1,i})$ sent to $\mathcal{F}_{\mathrm{OT}}$ by $\mathcal{A}$. $S_{\mathcal{R}}$ sets $\alpha^{(l)} = l$, for $l \in [0,2^h)$. Upon receiving $\{K^{(j)}_{1,h+1}, \Gamma^{(j)}\}_{j\in[t]}$, $S_{\mathcal{R}}$ runs $\mathtt{GGM.Eval}'(\alpha^{(l)}, \{K^{(j)}_{\bar{\alpha}^{(l)}_i, i}\}_{i\in[h]}, K^{(j)}_{1,h+1})$ and gets $(\{(\mathbf{v}^{(j,l)}_{2i}, \mathbf{v}^{(j,l)}_{2i+1})\}_{i\neq\alpha^{(l)}}, \{\Gamma^{(j,l)}_i\}_{i\in[0,2^h)})$, for $l \in [0,2^h)$. $S_{\mathcal{R}}$ computes $\Gamma^{(j)}_l := \mathtt{Hash}(\Gamma^{(j,l)}_0, ..., \Gamma^{(j,l)}_{2^h-1})$.*

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 $\mathcal{F}_{\text{mpVOLE}}^{\mathtt{GR}(2^k,d)}$. 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 $\mathbf{v}_{2i}^{(j,l)} \neq \mathbf{v}_{2i}^{(j,l')}$, $S_{\mathcal{R}}$ aborts. Thus, for $j \in [t] \backslash J$, $S_{\mathcal{R}}$ can obtain consistent $\{(\mathbf{v}_{2i}^{(j)}, \mathbf{v}_{2i+1}^{(j)})\}_{i \in [0,2^h)}$. Besides, for $j \in J$, suppose $I^{(j)} = \{\alpha^{(j)}\}$, then $S_{\mathcal{R}}$ can set $\mathbf{v}_{2\alpha^{(j)}}^{(j)} = \mathbf{v}_{2\alpha^{(j)}+1}^{(j)} = 0$ 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 $\beta_0^{(j)} := g_0^{(j)} - (\mathbf{K}_j - \sum_{i \in [0,2^h)} \mathbf{v}_{2i}^{(j)})$ and $\beta_1^{(j)} := g_1^{(j)} - (\mathbf{a}_j - \sum_{i \in [0,2^h)} \mathbf{v}_{2i+1}^{(j)})$. 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 $\hat{V}_j := \sum_{i \in [0,2^h)} \chi_i (\mathbf{v}_{2i}^{(j)} + \chi \cdot \mathbf{v}_{2i+1}^{(j)})$.*
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 $\rho, \rho' \in \{0,1\}^\kappa$ such that $(\mathbf{v}_{2i}^{(j,l)}, \mathbf{v}_{2i+1}^{(j,l)}, \Gamma_i^{(j,l)}) = G'(\rho)$ and $(\mathbf{v}_{2i}^{(j,l')}, \mathbf{v}_{2i+1}^{(j,l')}, \Gamma_i^{(j,l')}) = G'(\rho')$. 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 $(t/2^d + 1/2^{d-h} + \mathsf{negl}(\kappa))$. This concludes the whole proof. □

$P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ can naturally obtain pseudorandom VOLE correlations 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 $\mathbf{y} \overset{\$}{\leftarrow} \mathtt{GR}(2^k,d)^{(n-m)}$ 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.

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

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.